石化产品碳排放数据展示系统V1.0源代码

北京杰迅鸿翔信息技术有限公司

### Java:

package com.jxhs.asm.controller;  
  
import java.util.List;  
import javax.servlet.http.HttpServletResponse;  
  
import com.jxhs.asm.domain.AsmModelMaterialBalance;  
import com.jxhs.asm.service.IAsmModelMaterialBalanceService;  
import org.springframework.security.access.prepost.PreAuthorize;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import com.jxhs.common.annotation.Log;  
import com.jxhs.common.core.controller.BaseController;  
import com.jxhs.common.core.domain.AjaxResult;  
import com.jxhs.common.enums.BusinessType;  
import com.jxhs.common.utils.poi.ExcelUtil;  
import com.jxhs.common.core.page.TableDataInfo;  
  
*/\*\*  
 \* 物料平衡Controller  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@RestController  
@RequestMapping("/system/balance")  
public class AsmModelMaterialBalanceController extends BaseController  
{  
 @Autowired  
 private IAsmModelMaterialBalanceService asmModelMaterialBalanceService;  
  
 */\*\*  
 \* 查询物料平衡列表  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:list')")  
 @GetMapping("/list")  
 public TableDataInfo list(AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 startPage();  
 List<AsmModelMaterialBalance> list = asmModelMaterialBalanceService.selectAsmModelMaterialBalanceList(asmModelMaterialBalance);  
 return getDataTable(list);  
 }  
  
 */\*\*  
 \* 导出物料平衡列表  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:export')")  
 @Log(title = "物料平衡", businessType = BusinessType.*EXPORT*)  
 @PostMapping("/export")  
 public void export(HttpServletResponse response, AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 List<AsmModelMaterialBalance> list = asmModelMaterialBalanceService.selectAsmModelMaterialBalanceList(asmModelMaterialBalance);  
 ExcelUtil<AsmModelMaterialBalance> util = new ExcelUtil<AsmModelMaterialBalance>(AsmModelMaterialBalance.class);  
 util.exportExcel(response, list, "物料平衡数据");  
 }  
  
 */\*\*  
 \* 获取物料平衡详细信息  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:query')")  
 @GetMapping(value = "/{materielId}")  
 public AjaxResult getInfo(@PathVariable("materielId") Long materielId)  
 {  
 return AjaxResult.*success*(asmModelMaterialBalanceService.selectAsmModelMaterialBalanceByMaterielId(materielId));  
 }  
  
 */\*\*  
 \* 新增物料平衡  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:add')")  
 @Log(title = "物料平衡", businessType = BusinessType.*INSERT*)  
 @PostMapping  
 public AjaxResult add(@RequestBody AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 return toAjax(asmModelMaterialBalanceService.insertAsmModelMaterialBalance(asmModelMaterialBalance));  
 }  
  
 */\*\*  
 \* 修改物料平衡  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:edit')")  
 @Log(title = "物料平衡", businessType = BusinessType.*UPDATE*)  
 @PutMapping  
 public AjaxResult edit(@RequestBody AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 return toAjax(asmModelMaterialBalanceService.updateAsmModelMaterialBalance(asmModelMaterialBalance));  
 }  
  
 */\*\*  
 \* 删除物料平衡  
 \*/* @PreAuthorize("@ss.hasPermi('system:balance:remove')")  
 @Log(title = "物料平衡", businessType = BusinessType.*DELETE*)  
 @DeleteMapping("/{materielIds}")  
 public AjaxResult remove(@PathVariable Long[] materielIds)  
 {  
 return toAjax(asmModelMaterialBalanceService.deleteAsmModelMaterialBalanceByMaterielIds(materielIds));  
 }  
}

package com.jxhs.asm.service.impl;  
  
import com.jxhs.asm.domain.\*;  
import com.jxhs.asm.domain.mes.MbMtrlbalnsheetdataV;  
import com.jxhs.asm.domain.mes.PmMtrlMoveV;  
import com.jxhs.asm.domain.pricedb.OilCodeReference;  
import com.jxhs.asm.domain.pricedb.OilMaterial;  
import com.jxhs.asm.domain.pricedb.OilPrice;  
import com.jxhs.asm.domain.pricedb.OilSystemPrice;  
import com.jxhs.asm.mapper.AsmModelMapper;  
import com.jxhs.asm.mapper.AsmOsblDictMapper;  
import com.jxhs.asm.service.\*;  
import com.jxhs.common.config.RuoYiConfig;  
import com.jxhs.common.constant.Constants;  
import com.jxhs.common.core.domain.AjaxResult;  
import com.jxhs.common.core.domain.entity.SysDictData;  
import com.jxhs.common.core.domain.entity.SysParent;  
import com.jxhs.common.core.domain.entity.SysRole;  
import com.jxhs.common.core.domain.model.LoginUser;  
import com.jxhs.common.core.text.Convert;  
import com.jxhs.common.exception.ServiceException;  
import com.jxhs.common.exception.UtilException;  
import com.jxhs.common.utils.DateUtils;  
import com.jxhs.common.utils.SecurityUtils;  
import com.jxhs.common.utils.StringUtils;  
import com.jxhs.framework.web.service.ConfigService;  
import com.jxhs.system.service.ISysDictTypeService;  
import com.jxhs.system.service.ISysParentService;  
import com.jxhs.system.service.ISysRoleService;  
import com.jxhs.system.service.ISysUserService;  
import lombok.extern.slf4j.Slf4j;  
import org.apache.poi.hssf.usermodel.HSSFCellStyle;  
import org.apache.poi.hssf.usermodel.HSSFDataFormat;  
import org.apache.poi.hssf.usermodel.HSSFFormulaEvaluator;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
import org.apache.poi.hssf.util.HSSFColor;  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.ss.util.CellRangeAddress;  
import org.apache.poi.ss.util.CellReference;  
import org.apache.poi.util.IOUtils;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import org.springframework.ui.ModelMap;  
import org.springframework.web.client.RestTemplate;  
  
import java.io.\*;  
import java.math.BigDecimal;  
import java.util.\*;  
  
*/\*\*  
 \* 模型Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@Slf4j  
@Service  
public class AsmModelServiceImpl implements IAsmModelService {  
  
 @Autowired  
 private RestTemplate restTemplate;  
  
 @Autowired  
 private AsmModelMapper asmModelMapper;  
  
 @Autowired  
 private ConfigService configService;  
  
 @Autowired  
 private IAsmModelBlendService asmModelBlendService;  
  
 @Autowired  
 private ISysParentService sysParentService;  
  
 @Autowired  
 private IAsmModelDeviceService asmModelDeviceService;  
  
 @Autowired  
 private IAsmModelDeviceOsblService asmModelDeviceOsblService;  
  
 @Autowired  
 private IAsmModelMaterialBalanceService asmModelMaterialBalanceService;  
  
 @Autowired  
 private IAsmModelDeviceInputService asmModelDeviceInputService;  
  
 @Autowired  
 private IAsmModelDeviceOutputService asmModelDeviceOutputService;  
  
 @Autowired  
 private ISysDictTypeService sysDictTypeService;  
  
 @Autowired  
 private IMesModelDataService mesModelDataService;  
  
 @Autowired  
 private MesMaterialBalanceService mesMaterialBalanceService;  
  
 @Autowired  
 private MesDeviceMaterialBalanceService mesDeviceMaterialBalanceService;  
  
 @Autowired  
 private AsmOsblDictMapper asmOsblDictMapper;  
  
 public final List<String> rsList = Arrays.*asList*("ECC", "CCK", "CKK", "EOO", "UFL", "GAS", "DIO", "LPS", "LTG", "UGS", "GAG", "NAG");  
  
 public final List<String> zqList = Arrays.*asList*("10M", "05M", "3D5", "2D5", "1D5", "D7M", "D5M", "LD3");  
  
 public final List<String> hdList = Arrays.*asList*("KWS", "KWH");  
  
 public final List<String> qtList = Arrays.*asList*("WAT", "CWT", "SWT", "DAW", "DOW", "COT", "COH", "NNS");  
  
 @Autowired  
 private ISysUserService sysUserService;  
  
 @Autowired  
 private ISysRoleService sysRoleService;  
  
 */\*\*  
 \* 查询导出所需模型  
 \*  
 \* @param modelId 模型主键  
 \* @return 模型  
 \*/* @Override  
 public AsmModel selectAsmModelByModelId(Long modelId) {  
 AsmModel model = asmModelMapper.selectAsmModelByModelId(modelId);  
 List<AsmModelDevice> devices = getFullDevices(modelId);  
 List<AsmModelBlend> blends = getBlends(modelId);  
 List<AsmModelMaterialBalance> balances = getBalance(modelId);  
 model.setDevices(devices);  
 model.setBlends(blends);  
 model.setMaterialBalances(balances);  
 return model;  
 }  
  
 */\*\*  
 \* 查询计算所需模型  
 \*  
 \* @param modelId 模型主键  
 \* @return 模型  
 \*/* @Override  
 public AsmModel selectexportAsmModelByModelId(Long modelId) {  
 AsmModel model = asmModelMapper.selectAsmModelByModelId(modelId);  
 List<AsmModelDevice> devices = getexportFullDevices(modelId);  
 List<AsmModelBlend> blends = getBlends(modelId);  
 List<AsmModelMaterialBalance> balances = getBalance(modelId);  
 model.setDevices(devices);  
 model.setBlends(blends);  
 model.setMaterialBalances(balances);  
 return model;  
 }  
  
  
 */\*\*  
 \* 查询模型列表  
 \*  
 \* @param asmModel 模型  
 \* @return 模型  
 \*/* @Override  
 public List<AsmModel> selectAsmModelList(AsmModel asmModel) {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole = sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("common")) {  
 asmModel.setCreateId(user.getUserId());  
 return asmModelMapper.selectAsmModelList(asmModel);  
 }  
 return asmModelMapper.selectAsmModelList(asmModel);  
 }  
  
 */\*\*  
 \* 新增模型  
 \*  
 \* @param asmModel 模型  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModel(AsmModel asmModel) {  
 asmModel.setCreateTime(DateUtils.*getNowDate*());  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 asmModel.setCreateId(user.getUserId());  
 asmModel.setCreateBy(user.getUsername());  
 asmModel.setIsCalc("0");  
 if (asmModel.getPriceSystemId() == null) {  
 asmModel.setPriceSystemId(0L);  
 }  
*// System.out.println(asmModel);* int rows = asmModelMapper.insertAsmModel(asmModel);  
  
 if (StringUtils.*isNotEmpty*(asmModel.getFileUrl())) {  
 importExcel(asmModel);  
 }  
 return rows;  
 }  
  
  
 */\*\*  
 \* 修改模型  
 \*  
 \* @param asmModel 模型  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModel(AsmModel asmModel) {  
 asmModel.setUpdateTime(DateUtils.*getNowDate*());  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 asmModel.setUpdateId(user.getUserId());  
 asmModel.setUpdateBy(user.getUsername());  
 return asmModelMapper.updateAsmModel(asmModel);  
 }  
  
 */\*\*  
 \* 批量删除模型  
 \*  
 \* @param modelIds 需要删除的模型主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelByModelIds(Long[] modelIds) {  
 return asmModelMapper.deleteAsmModelByModelIds(modelIds);  
 }  
  
 */\*\*  
 \* 删除模型信息  
 \*  
 \* @param modelId 模型主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelByModelId(Long modelId) {  
 return asmModelMapper.deleteAsmModelByModelId(modelId);  
 }  
  
 @Override  
 public AjaxResult exportExcel(AsmModel model, boolean isPrint, Integer printnum) {  
 HSSFWorkbook wb = new HSSFWorkbook();  
 Sheet sheet = wb.createSheet("流程图");  
 List<AsmModelDevice> list = new ArrayList<>();  
 List<AsmModelDeviceInput> raw\_materials = new ArrayList<>();  
 List<AsmModelDeviceInput> oil\_materials = new ArrayList<>();  
 List<AsmModelDeviceOutput> products = new ArrayList<>();  
 bfs(model, list);  
 int start\_row\_index = 4;  
 int start\_column\_index = -10;  
 int current\_level = 0;  
 int level1start\_row\_index = 0;  
 int level2start\_row\_index = 0;  
 int level3start\_row\_index = 0;  
 int level4start\_row\_index = 0;  
 if (printnum == null) {  
*// printnum=300;* }  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 String rolekey = user.getUser().getRoles().get(0).getRoleKey();  
 String createby = "";  
 if (model.getModelName().contains("海南")) {  
 createby = rolekey;  
 } else {  
 createby = model.getModelName() + rolekey;  
 }  
 *//获取物料分类* Long priceSystemid = model.getPriceSystemId();  
 String url = "http://localhost:8082/oil/price/pricelist?SystemId=" + priceSystemid + "&&createBy=" + createby;  
 Map<String, Map<String, Map<String, Map>>> result = restTemplate.getForObject(url, Map.class);  
 List<OilMaterial> oilMaterialList = new ArrayList<>();  
 Set<String> keyset = result.keySet();  
 for (String key : keyset) {  
 OilMaterial oilMaterial = new OilMaterial();  
 Map map = result.get(key);  
 Object ob = map.get("oilMaterial");  
 oilMaterial.setMaterialName(String.*valueOf*(result.get(key).get("oilMaterial").get("materialName")));  
 oilMaterial.setMaterialCode(String.*valueOf*(result.get(key).get("oilMaterial").get("materialCode")));  
 if (result.get(key).get("oilMaterial").get("materialProp1") != null) {  
 oilMaterial.setMaterialProp1(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp1")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp2") != null) {  
 oilMaterial.setMaterialProp2(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp2")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp3") != null) {  
 oilMaterial.setMaterialProp3(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp3")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp4") != null) {  
 oilMaterial.setMaterialProp4(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp4")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp5") != null) {  
 oilMaterial.setMaterialProp5(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp5")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp6") != null) {  
 oilMaterial.setMaterialProp6(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp6")));  
 }  
 if (result.get(key).get("oilMaterial").get("oilCodeReference").get("factoryCode") != null) {  
 oilMaterial.setFactoryCode(String.*valueOf*(result.get(key).get("oilMaterial").get("oilCodeReference").get("factoryCode")));  
 }  
 oilMaterialList.add(oilMaterial);  
 }  
 for (AsmModelDevice device : list) {  
 int rows = device.getInputs() == null ? 0 : (device.getInputs().size() + 2);  
  
 if (device.getOutputs() != null && rows < (device.getOutputs().size() + 2)) {  
 rows = device.getOutputs().size() + 2;  
 }  
 if (rows < (device.getDeviceName().length() + 2)) {  
 rows = device.getDeviceName().length() + 2;  
 }  
 if (printnum != null) {  
 if (start\_row\_index + rows > printnum || (device.getLevel() == 1 && start\_row\_index <= 4)) {  
 start\_row\_index = 4;  
 start\_column\_index += 11;  
 }  
 }  
 if (isPrint) {  
 if (device.getLevel() != current\_level) {  
 start\_row\_index = 4;  
 start\_column\_index += 11;  
 */\*if (start\_column\_index>22){  
 start\_column\_index = 1;  
 }\*/* if (device.getLevel() == 3) {  
 start\_row\_index = level1start\_row\_index;  
 start\_column\_index = 1;  
 } else if (device.getLevel() == 4) {  
 start\_row\_index = level2start\_row\_index;  
 start\_column\_index = 12;  
 } else if (device.getLevel() == 5) {  
 start\_row\_index = level3start\_row\_index;  
 start\_column\_index = 1;  
 }  
 current\_level = device.getLevel();  
 }  
 } else {  
 if (printnum == null && device.getLevel() != current\_level) {  
 start\_row\_index = 4;  
 start\_column\_index += 11;  
 current\_level = device.getLevel();  
 }  
 }  
 start\_row\_index = export\_device(wb, sheet, raw\_materials, oil\_materials, products, start\_row\_index, start\_column\_index, device, printnum);  
 if (isPrint) {  
 if (device.getLevel() == 1) {  
 level1start\_row\_index = start\_row\_index;  
 } else if (device.getLevel() == 2) {  
 level2start\_row\_index = start\_row\_index;  
 } else if (device.getLevel() == 3) {  
 level3start\_row\_index = start\_row\_index;  
 } else if (device.getLevel() == 4) {  
 level4start\_row\_index = start\_row\_index;  
 }  
 }  
 }  
 if (isPrint) {  
 start\_row\_index = 4;  
 start\_column\_index += 22;  
 } else {  
 start\_row\_index = 4;  
 start\_column\_index += 11;  
 }  
 if (printnum == null) {  
 printnum = 200;  
 }  
 export\_material(wb, sheet, raw\_materials, oil\_materials, products, start\_row\_index, start\_column\_index, printnum, oilMaterialList);  
  
 while (true) {  
 start\_column\_index -= 11;  
 if (start\_column\_index < 0)  
 break;  
 sheet.setColumnWidth(start\_column\_index, 30 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 5, 30 \* 256);  
 }  
  
 List<AsmModelBlend> asmModelBlendList = model.getBlends();  
 export\_osbl(wb, list);  
 export\_all\_devices(wb, list);  
 export\_mass\_balance(wb, asmModelBlendList);  
  
  
 String filename = UUID.*randomUUID*().toString() + "\_" + model.getModelName() + ".xls";  
 String downloadPath = RuoYiConfig.*getUploadPath*() + filename;  
 File desc = new File(downloadPath);  
 if (!desc.getParentFile().exists()) {  
 desc.getParentFile().mkdirs();  
 }  
 OutputStream out = null;  
 try {  
 out = new FileOutputStream(downloadPath);  
 wb.write(out);  
 return AjaxResult.*success*(filename);  
 } catch (Exception e) {  
*// logger.error("导出Excel异常{}", e.getMessage());* throw new UtilException("导出Excel失败，请联系网站管理员！");  
 } finally {  
 IOUtils.*closeQuietly*(wb);  
 IOUtils.*closeQuietly*(out);  
 }  
 }  
  
 private void export\_mass\_balance(HSSFWorkbook wb, List<AsmModelBlend> list) {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
  
 Sheet sheet = wb.createSheet("油品调和表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("物料代码");  
 Cell cell1 = row.createCell(start\_col\_index + 1);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("物料名称");  
 Cell cell2 = row.createCell(start\_col\_index + 2);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("性质编码");  
 Cell cell3 = row.createCell(start\_col\_index + 3);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("性质名称");  
 Cell cell4 = row.createCell(start\_col\_index + 4);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("物性值");  
 Cell cell5 = row.createCell(start\_col\_index + 5);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("物性单位");  
 }  
 }  
 start\_row\_index++;  
 for (AsmModelBlend blend : list) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_point2);  
 cellTitle.setCellValue(blend.getMaterielCode());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_point2);  
 cell.setCellValue(blend.getMaterielName());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(blend.getPropertyCode());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(blend.getPropertyName());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(blend.getPropertyValue());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(blend.getPropertyUnit());  
 start\_row\_index++;  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 @Transactional  
 @Override  
 public void calc(AsmModel model) {  
 List<SysDictData> sysDictDataList = sysDictTypeService.selectDictDataByType("scorch");  
 int iterations = 100;  
 if (StringUtils.*isNotEmpty*(configService.getKey("co2e.calc.iteration"))) {  
 iterations = Integer.*parseInt*(configService.getKey("co2e.calc.iteration"));  
 }  
 double convergence = 0.0001;  
 if (StringUtils.*isNotEmpty*(configService.getKey("co2e.calc.convergence"))) {  
 convergence = Double.*parseDouble*(configService.getKey("co2e.calc.convergence"));  
 }  
  
 String ugsOsblCode = configService.getKey("co2e.osbl.ugs");  
 if (StringUtils.*isEmpty*(ugsOsblCode)) {  
 ugsOsblCode = "UGS";  
 }  
  
 *//查找调和出来的产品* Map<String, AsmModelBlend> outputBlendMap = new HashMap<>();  
 List<AsmModelBlend> outputBlends = asmModelBlendService.selectOutputBlendByModelId(model.getModelId());  
 if (outputBlends != null) {  
 for (AsmModelBlend blend : outputBlends) {  
 outputBlendMap.put(blend.getMaterielCode(), blend);  
 }  
 }  
 *///end* List<AsmModelDevice> devices = model.getDevices();  
 Map<Long, AsmModelDeviceOutput> outputMap = new HashMap<>();  
 List<AsmModelDeviceInput> inputs = new ArrayList<>();  
 for (AsmModelDevice device : devices) {  
  
 device.setDeviceCo2(0.0);  
 device.setDeviceFactor(0.0);  
 device.setDeviceProcess(0.0);  
 for (AsmModelDeviceInput input : device.getInputs()) {  
 device.setDeviceProcess(device.getDeviceProcess() + input.getInputQuantity());  
 input.setInputCo2(0.0);  
 inputs.add(input);  
 }  
  
 int zero\_count = 0;  
 double ugsCo2 = 0.0;  
 for (AsmModelDeviceOsbl osbl : device.getOsbls()) {  
 if (osbl.getCarbonEmission() == null || (Double.*doubleToLongBits*(osbl.getCarbonEmission()) == Double.*doubleToLongBits*(0.0))) {  
 zero\_count++;  
 continue;  
 }  
 if (osbl.getOsblCode().equals(ugsOsblCode)) {  
 ugsCo2 = osbl.getCarbonEmission();  
 }  
 device.setDeviceCo2(device.getDeviceCo2() + osbl.getCarbonEmission());  
 }  
  
 if ((zero\_count == (device.getOsbls().size() - 1)) && (Double.*doubleToLongBits*(ugsCo2) < Double.*doubleToLongBits*(0.0))) {  
 device.setDeviceCo2(0.0);  
 device.setDeviceFactor(0.0);  
 for (AsmModelDeviceOsbl osbl : device.getOsbls()) {  
 osbl.setCarbonEmission(0.0);  
 asmModelDeviceOsblService.updateAsmModelDeviceOsbl(osbl);  
 }  
 *log*.debug("device name = {}", device.getDeviceName());  
 } else {  
 device.setDeviceFactor(device.getDeviceCo2() / device.getDeviceProcess());  
 }  
  
  
 double total\_yield = 0.0;  
 for (AsmModelDeviceOutput output : device.getOutputs()) {  
 if (output.getOutputCode().equals(configService.getKey("co2e.calc.losscode")))  
 continue;  
 if (output.getOutputName().equals(configService.getKey("co2e.calc.IE"))) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 continue;  
 }  
 *//对干气置零* if (output.getIsCalc() != null && output.getIsCalc().equals("0")) {  
 for (AsmModelDevice device1 : devices) {  
 for (AsmModelDeviceInput input : device1.getInputs()) {  
 if (input.getIsCalc() != null && input.getIsCalc().equals("0")) {  
 if (input.getSuperOutputId().equals(output.getOutputId())) {  
 output.setOutputCo2(0.0);  
 output.setOutputCo2Factor(0.0);  
 continue;  
 }  
 }  
 }  
 }  
 continue;  
 }  
 for (SysDictData sys : sysDictDataList) {  
 if (output.getOutputName().equals(sys.getDictValue())) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 continue;  
 }  
 }  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 output.setYield(output.getOutputQuantity() / device.getDeviceProcess());  
 total\_yield += output.getYield();  
 }  
 device.setYield(total\_yield);  
 for (AsmModelDeviceOutput output : device.getOutputs()) {  
  
 if (output.getOutputQuantity() > 0.0) {  
 if (output.getOutputCode().equals(configService.getKey("co2e.calc.losscode"))) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 } else if (output.getOutputName().equals(configService.getKey("co2e.calc.IE"))) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 *//对干气置零* } else if (output.getIsCalc() != null && output.getIsCalc().equals("0")) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 } else if (outputBlendMap.get(output.getOutputCode()) != null) { *//调和产品使用其它算法* calc\_blend\_output(device, output);  
  
 } else {  
 output.setOutputCo2(device.getDeviceCo2() \* output.getYield() / device.getYield());  
 output.setOutputCo2Factor(output.getOutputCo2() / output.getOutputQuantity());  
 }  
 }  
 for (SysDictData sys : sysDictDataList) {  
 if (output.getOutputQuantity() > 0.0) {  
 if (output.getOutputName().equals(sys.getDictValue())) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 }  
 }  
 }*/\*else if (output.getOutputQuantity()==0.0){  
 if(output.getOutputCode().equals(configService.getKey("co2e.calc.losscode")))  
 {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 }  
 else if(output.getOutputName().equals(configService.getKey("co2e.calc.IE"))){  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 }  
 else {  
 output.setOutputCo2(device.getDeviceCo2() \* output.getYield() / device.getYield());  
 output.setOutputCo2Factor(0.0);  
 }  
 }\*/* outputMap.put(output.getOutputId(), output);  
 }  
 *//对devices里的output减去input所消耗的co2e  
/\* for(AsmModelDeviceInput input : inputs){  
 if (input.getIsCalc()!=null&&input.getIsCalc().equals("0")){  
 for (AsmModelDeviceOutput out:device.getOutputs()) {  
 if (input.getSuperOutputId()!=null&&input.getSuperOutputId().equals(out.getOutputId())){  
 Double outputco2=out.getOutputCo2()-input.getInputCo2();  
 Double outputco2factor=(out.getOutputCo2()-input.getInputCo2())/out.getOutputQuantity();  
 if (outputco2<10.0){  
 outputco2=0.0;  
 outputco2factor=0.0;  
 }  
 out.setOutputCo2(outputco2);  
 out.setOutputCo2Factor(outputco2factor);  
 }  
 }  
 }  
 }\*/* }  
 for (AsmModelDeviceInput input : inputs) {  
 AsmModelDeviceOutput output = outputMap.get(input.getSuperOutputId());  
 if (output != null) {  
 input.setInputCo2(output.getOutputCo2Factor() \* input.getInputQuantity());  
 }  
 }  
  
 AsmModelBlend searchBlend = new AsmModelBlend();  
 searchBlend.setModelId(model.getModelId());  
 searchBlend.setInOutType("");  
  
 for (int i = 0; i < iterations; i++) {  
 for (AsmModelDevice device : devices) {  
 double total\_co2 = device.getDeviceCo2();  
 for (AsmModelDeviceInput input : device.getInputs()) {  
 total\_co2 += input.getInputCo2();  
 }  
 for (AsmModelDeviceOutput output : device.getOutputs()) {  
 if (output.getOutputCode().equals(configService.getKey("co2e.calc.losscode")))  
 continue;  
 if (output.getOutputName().equals(configService.getKey("co2e.calc.IE"))) {  
 output.setOutputCo2(0.0);  
 output.setOutputCo2Factor(0.0);  
 continue;  
 }  
 *//减去干气的量* if (output.getIsCalc() != null && output.getIsCalc().equals("0")) {  
 for (AsmModelDevice device1 : devices) {  
 for (AsmModelDeviceInput input : device1.getInputs()) {  
 if (input.getIsCalc() != null && input.getIsCalc().equals("0")) {  
 if (input.getSuperOutputId().equals(output.getOutputId())) {  
 output.setOutputCo2(0.0);  
 output.setOutputCo2Factor(0.0);  
 continue;  
 }  
 }  
 }  
 }  
 continue;  
 }  
 if (outputBlendMap.get(output.getOutputCode()) != null) { *///计算调和* calc\_blend\_output(device, output);  
 } else {  
 output.setOutputCo2(total\_co2 \* output.getYield() / device.getYield());  
  
 if (output.getOutputQuantity() > 0.0) {  
 output.setOutputCo2Factor(output.getOutputCo2() / output.getOutputQuantity());  
 }  
 }  
 for (SysDictData sys : sysDictDataList) {  
 if (output.getOutputName().equals(sys.getDictValue())) {  
 output.setOutputCo2Factor(0.0);  
 output.setOutputCo2(0.0);  
 continue;  
 }  
 }  
 }  
 }  
 boolean isBreak = false;  
 for (AsmModelDeviceInput input : inputs) {  
 if (input.getSuperOutputId() != null) {  
 AsmModelDeviceOutput output = outputMap.get(input.getSuperOutputId());  
 if (output != null) {  
 double input\_factor = input.getInputCo2() / input.getInputQuantity();  
 if (Math.*abs*(input\_factor - output.getOutputCo2Factor()) < convergence) {  
 isBreak = true;  
 } else {  
 isBreak = false;  
 break;  
 }  
 }  
 }  
 }  
 if (isBreak) {  
 *log*.debug("breaked. i = " + i);  
 break;  
 }  
 for (AsmModelDeviceInput input : inputs) {  
 AsmModelDeviceOutput output = outputMap.get(input.getSuperOutputId());  
 if (output != null) {  
 input.setInputCo2(output.getOutputCo2Factor() \* input.getInputQuantity());  
 }  
 }  
 }  
 *//对map里的output减去input消耗的co2e(针对投入物料有干气的)  
 /\*for(AsmModelDeviceInput input : inputs){  
  
 if (input.getIsCalc()!=null&&input.getIsCalc().equals("0")) {  
 int num=0;  
  
 for (AsmModelDeviceInput in : inputs) {  
 if (input.getSuperOutputId() != null && input.getSuperOutputId() == in.getSuperOutputId()) {  
 num++;  
 }  
 }  
  
 AsmModelDeviceOutput output = outputMap.get(input.getSuperOutputId());  
 if (output != null) {  
  
  
 for (AsmModelDevice device : devices) {  
  
 if (device.getDeviceId() .equals(output.getDeviceId()) ) {  
 Double lostpro = 0.0;  
 Double shoajiao=0.0;  
 Double ganqiout=0.0;  
 Double ganqiyield=0.0;  
 for (AsmModelDeviceOutput out : device.getOutputs()) {  
 if (out.getOutputName().equals("损失")) {  
 lostpro += out.getOutputQuantity()/device.getDeviceProcess();  
 }else if (out.getOutputName().equals("烧焦")){  
 shoajiao+=out.getOutputQuantity()/device.getDeviceProcess();  
 }else if (out.getOutputName().contains("干气")){  
 ganqiout+=out.getOutputCo2();  
 ganqiyield=out.getYield();  
 }else {  
 lostpro = 0.0;  
 shoajiao=0.0;  
 }  
 for (SysDictData sys:sysDictDataList) {  
 if(out.getOutputName().equals(sys.getDictValue())){  
 shoajiao+=out.getOutputQuantity()/device.getDeviceProcess();  
 continue;  
 }  
 }  
 }  
 Double ganqiinput=0.0;  
 //计算干气  
 for (AsmModelDeviceOutput out : device.getOutputs()) {  
 Double suminputco2=0.0;  
 double inputco2e=0.0;  
  
 for (AsmModelDeviceInput in : inputs) {  
  
  
 if (in.getSuperOutputId() != null && in.getSuperOutputId().equals(out.getOutputId()) ) {  
 suminputco2 =(suminputco2+in.getInputCo2())/num/num;  
 if (in.getInputName().contains("干气")){  
 ganqiinput+=in.getInputCo2();  
 }  
 }else if (in.getDeviceId().equals(device.getDeviceId())){  
 inputco2e+=in.getInputCo2();  
 }  
 }  
 if (!out.getOutputName().equals("损失") && !out.getOutputName().contains("干气")&&!out.getOutputName().contains("烧焦")) {  
// AsmModelDeviceOutput output1 = outputMap.get(out.getOutputId());  
 Double yield=out.getYield();  
 if (yield==null){  
 yield=0.0;  
 }  
 Double proco2e=0.0;  
 if (ganqiout==0.0){  
 proco2e=out.getOutputCo2();  
 }else {  
 proco2e=(yield / (1 - (lostpro +shoajiao+ ganqiinput /ganqiout\*ganqiyield))) \* (device.getDeviceCo2()+inputco2e);  
 }  
 Double proyield = proco2e / out.getOutputQuantity();  
 out.setOutputCo2(proco2e);  
 out.setOutputCo2Factor(proyield);  
 outputMap.put(out.getOutputId(), out);  
 }  
 }  
 }  
 }  
 Double outputco2e=output.getOutputCo2() - input.getInputCo2();  
 Double OutputCo2Factor=(output.getOutputCo2() - input.getInputCo2()) / output.getOutputQuantity();  
 if (outputco2e<10){  
 outputco2e=0.0;  
 OutputCo2Factor=0.0;  
 }  
 output.setOutputCo2(outputco2e);  
 output.setOutputCo2Factor(OutputCo2Factor);  
 outputMap.put(output.getOutputId(), output);  
 }  
 }  
 }\*/  
  
 //循环减去消耗的物料  
 log*.debug("calc complete. save to db");  
 double total\_input\_process = 0.0;  
 double total\_output\_co2 = 0.0;  
 double total\_device\_co2 = 0.0;  
 double total\_product = 0.0;  
 for (AsmModelDeviceInput input : inputs) {  
 if (StringUtils.*isNotEmpty*(input.getMaterialProperty())) {  
 total\_input\_process += input.getInputQuantity();  
 }  
 asmModelDeviceInputService.updateAsmModelDeviceInput(input);  
 }  
 Iterator<Map.Entry<Long, AsmModelDeviceOutput>> iterator = outputMap.entrySet().iterator();  
 while (iterator.hasNext()) {  
 Map.Entry<Long, AsmModelDeviceOutput> entry = iterator.next();  
 AsmModelDeviceOutput value = entry.getValue();  
 String dgas = configService.getKey("co2e.calc.drygas");  
 String[] strings = dgas.split(",");  
  
 for (int i = 0; i < strings.length; i++) {  
 if (value.getOutputName().equals(strings[i])  
 && StringUtils.*isNotEmpty*(value.getMaterialProperty())  
 && value.getMaterialProperty().equals(configService.getKey("co2e.product.property"))) {  
 value.setOutputCo2(0.0);  
 value.setOutputCo2Factor(0.0);  
 break;  
 }  
 }  
  
  
 asmModelDeviceOutputService.updateAsmModelDeviceOutput(value);  
 AsmModelDeviceInput asmModelDeviceInput = new AsmModelDeviceInput();  
 Double output\_quantity1 = 0.0;  
 Double output\_co2 = 0.0;  
 output\_quantity1 = value.getOutputQuantity();  
 output\_co2 = value.getOutputCo2();  
 asmModelDeviceInput.setSuperOutputId(value.getOutputId());  
 asmModelDeviceInput.setInputName(value.getOutputName());  
 asmModelDeviceInput.setInputCode(value.getOutputCode());  
 List<AsmModelDeviceInput> asmModelDeviceInputs = asmModelDeviceInputService.selectAsmModelDeviceInputList(asmModelDeviceInput);  
 for (AsmModelDeviceInput amd : asmModelDeviceInputs) {  
 output\_quantity1 = output\_quantity1 - amd.getInputQuantity();  
 output\_co2 = output\_co2 - amd.getInputCo2();  
 }  
 value.setOutputCo2(output\_co2);  
 value.setOutputQuantity(output\_quantity1);  
 if (StringUtils.*isNotEmpty*(value.getMaterialProperty()) && value.getMaterialProperty().equals(configService.getKey("co2e.product.property"))) {  
 total\_output\_co2 += value.getOutputCo2();  
 total\_product += value.getOutputQuantity();  
 }  
 if (value.getOutputCode().equals(configService.getKey("co2e.calc.losscode"))) {  
 total\_product += value.getOutputQuantity();  
 total\_output\_co2 += value.getOutputCo2();  
 }  
 }  
 for (AsmModelDevice device : devices) {  
 total\_device\_co2 += device.getDeviceCo2();  
 asmModelDeviceService.updateAsmModelDevice(device);  
 }  
  
 model.setTotalFactoryCo2(total\_device\_co2);  
 model.setTotalProductCo2(total\_output\_co2);  
 model.setTotalInput(total\_input\_process);  
 model.setTotalOutput(total\_product);  
 model.setIsCalc("1");  
 this.updateAsmModel(model);  
  
 }  
  
 private void calc\_blend\_output(AsmModelDevice device, AsmModelDeviceOutput output) {  
 String planCode = output.getPlanCode();  
 List<AsmModelDeviceInput> deviceInputs = device.getInputs();  
 if (deviceInputs == null)  
 deviceInputs = new ArrayList<>();  
 double co2 = 0.0;  
 for (AsmModelDeviceInput deviceInput : deviceInputs) {  
 if (deviceInput.getPlanCode().equals(planCode))  
 co2 += deviceInput.getInputCo2();  
 }  
 List<AsmModelDeviceOsbl> deviceOsbls = device.getOsbls();  
 if (deviceOsbls == null)  
 deviceOsbls = new ArrayList<>();  
 for (AsmModelDeviceOsbl deviceOsbl : deviceOsbls) {  
 if (deviceOsbl.getPlanCode().equals(planCode)) {  
 co2 += deviceOsbl.getCarbonEmission();  
 }  
 }  
 output.setOutputCo2(co2);  
 if(output.getOutputQuantity() > 0.0) {  
 output.setOutputCo2Factor(co2 / output.getOutputQuantity());  
 }  
 else  
 output.setOutputCo2Factor(0.0);  
 }  
  
 @Override  
 public ModelMap edit\_graph(AsmModel model, ModelMap mmap) {  
 List<CEdge> edges = new ArrayList<>();  
 List<AsmModelDevice> list = new ArrayList<>();  
  
 Map<Long, AsmModelDevice> deviceMap = bfs(model, list);  
  
 for (AsmModelDevice device : list) {  
 int inputCount = device.getInputs() == null ? 0 : device.getInputs().size();  
 if (device.getOutputs() == null)  
 continue;  
 for (int i = 0; i < device.getOutputs().size(); i++) {  
 AsmModelDeviceOutput output = device.getOutputs().get(i);  
 if (output.getInputs() == null)  
 continue;  
 for (int j = 0; j < output.getInputs().size(); j++) {  
 CEdge edge = new CEdge();  
 From from = new From();  
 List list1 = new ArrayList();  
 from.setFrom(device.getDeviceId());  
 AsmModelDeviceInput asmModelDeviceInput = new AsmModelDeviceInput();  
 asmModelDeviceInput.setDeviceId(device.getDeviceId());  
 asmModelDeviceInput.setModelId(model.getModelId());  
 List<AsmModelDeviceInput> asmModelDeviceInputs = asmModelDeviceInputService.selectAsmModelDeviceInputList(asmModelDeviceInput);  
 list1.add(asmModelDeviceInputs);  
 AsmModelDeviceOutput asmModelDeviceOutput = new AsmModelDeviceOutput();  
 asmModelDeviceOutput.setModelId(model.getModelId());  
 asmModelDeviceOutput.setDeviceId(device.getDeviceId());  
 List<AsmModelDeviceOutput> asmModelDeviceOutputs = asmModelDeviceOutputService.selectAsmModelDeviceOutputList(asmModelDeviceOutput);  
 list1.add(asmModelDeviceOutputs);  
 from.setFromlist(list1);  
 edge.setFrom(from);  
 edge.setFromIndex(inputCount + i);  
 AsmModelDeviceInput input = output.getInputs().get(j);  
 ;  
 AsmModelDevice obj = deviceMap.get(input.getDeviceId());  
 To to = new To();  
 List list2 = new ArrayList();  
 to.setTo(obj.getDeviceId());  
 asmModelDeviceInput.setDeviceId(obj.getDeviceId());  
 List<AsmModelDeviceInput> asmModelDeviceInputs1 = asmModelDeviceInputService.selectAsmModelDeviceInputList(asmModelDeviceInput);  
 list2.add(asmModelDeviceInputs1);  
 asmModelDeviceOutput.setDeviceId(obj.getDeviceId());  
 List<AsmModelDeviceOutput> asmModelDeviceOutputs1 = asmModelDeviceOutputService.selectAsmModelDeviceOutputList(asmModelDeviceOutput);  
 list2.add(asmModelDeviceOutputs1);  
 to.setTolist(list2);  
 edge.setTo(to);  
 for (int k = 0; k < obj.getInputs().size(); k++) {  
 if (obj.getInputs().get(k).getInputId() == input.getInputId()) {  
 edge.setToIndex(k);  
 edges.add(edge);  
 }  
 }  
 }  
 }  
 }  
 mmap.put("nodes", list);  
 mmap.put("edges", edges);  
 return mmap;  
 }  
  
 @Transactional  
 @Override  
 public void calcBatch(String ids) {  
 Long[] idArray = Convert.*toLongArray*(ids);  
 for (int i = 0; i < idArray.length; i++) {  
 AsmModel model = selectAsmModelByModelId(idArray[i]);  
 if (model == null)  
 continue;  
 calc(model);  
 }  
 }  
  
 @Override  
 public AsmModel selectgasbymodelId(Long modelId) {  
 AsmModel model = asmModelMapper.selectAsmModelByModelId(modelId);  
 List<AsmModelDevice> devices = getGasDevices(modelId);  
 model.setDevices(devices);  
 return model;  
 }  
  
 @Override  
 public AjaxResult getthousandoutputvalue(AsmModel asmModel) {  
 AsmModelMaterialBalance asmModelMaterialBalance = new AsmModelMaterialBalance();  
 asmModelMaterialBalance.setModelId(asmModel.getModelId());  
 asmModelMaterialBalance.setMaterielProperty("产品");  
 List<AsmModelMaterialBalance> asmModelMaterialBalanceList = asmModelMaterialBalanceService.selectAsmModelMaterialBalanceList(asmModelMaterialBalance);  
 AsmModel model = asmModelMapper.selectAsmModelByModelId(asmModel.getModelId());  
 Long priceSystemid = asmModel.getPriceSystemId();  
 String url = "http://localhost:8082/oil/price/pricelist?SystemId=" + priceSystemid;  
  
 Map<String, Map> result = restTemplate.getForObject(url, Map.class);  
 List<OilPrice> oilPrices = new ArrayList<>();  
 OilPrice oilPrice = new OilPrice();  
 Double sumoutputprice = 0.0;  
 for (AsmModelMaterialBalance balance : asmModelMaterialBalanceList) {  
 OilPrice oilPrice1 = new OilPrice();  
 String materialCode = balance.getMaterielCode();  
 Double price = 1.0;  
 if (result.get(materialCode) != null) {  
 price = (Double) result.get(materialCode).get("priceValue");  
 }  
 Double outputco2eQuantity = 0.0;  
 AsmModelDeviceOutput asmModelDeviceOutput = new AsmModelDeviceOutput();  
 asmModelDeviceOutput.setModelId(asmModel.getModelId());  
 asmModelDeviceOutput.setOutputName(balance.getMaterielName());  
 asmModelDeviceOutput.setOutputCode(balance.getMaterielCode());  
 List<AsmModelDeviceOutput> asmModelDeviceOutputList = asmModelDeviceOutputService.selectAsmModelDeviceOutputList(asmModelDeviceOutput);  
 for (AsmModelDeviceOutput out : asmModelDeviceOutputList) {  
 outputco2eQuantity += out.getOutputCo2();  
 }  
 AsmModelDeviceInput asmModelDeviceInput = new AsmModelDeviceInput();  
 asmModelDeviceInput.setModelId(asmModel.getModelId());  
 asmModelDeviceInput.setInputName(balance.getMaterielName());  
 asmModelDeviceInput.setInputCode(balance.getMaterielCode());  
 List<AsmModelDeviceInput> asmModelDeviceInputList = asmModelDeviceInputService.selectAsmModelDeviceInputList(asmModelDeviceInput);  
 if (asmModelDeviceInputList.size() > 0) {  
 for (AsmModelDeviceInput input : asmModelDeviceInputList) {  
 outputco2eQuantity = outputco2eQuantity - input.getInputQuantity();  
 }  
 }  
 Double outputQuantity = balance.getMaterielQuan();  
 if (price == null)  
 continue;  
 Double outputprice = price \* outputQuantity;  
 sumoutputprice += outputprice;  
 oilPrice1.setPriceValue(outputco2eQuantity / (outputprice / 10000));  
 oilPrice1.setMaterialName(balance.getMaterielName());  
 oilPrices.add(oilPrice1);  
 }  
 oilPrice.setMaterialName("quanchang");  
 oilPrice.setPriceValue(model.getTotalProductCo2() / sumoutputprice);  
 Map<String, Object> hashmap = new HashMap<>();  
 hashmap.put("wuliao", oilPrices);  
 hashmap.put("quanchang", oilPrice);  
 return AjaxResult.*success*(hashmap);  
 }  
  
 @Override  
 public AjaxResult getPriceSystem() {  
 String url = "http://localhost:8082/oil/oilSystem/list";  
 AjaxResult result = restTemplate.getForObject(url, AjaxResult.class);  
 return result;  
 }  
  
 @Override  
 public AjaxResult getfactorylist(AsmModel asmModel) {  
 System.*out*.println(asmModel);  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 asmModel.setCreateId(user.getUserId());  
 asmModel.setCreateBy(user.getUsername());  
 String rolekey = user.getUser().getRoles().get(0).getRoleKey();  
 String createBy = asmModel.getModelFactory() + rolekey;  
 String url = "http://localhost:8082/oil/material/factorylist?createBy=" + createBy;  
 AjaxResult result = restTemplate.getForObject(url, AjaxResult.class);  
 return result;  
 }  
  
 @Override  
 public AjaxResult addfactorylist(List<OilCodeReference> oilCodeReferenceList) {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 String rolekey = user.getUser().getRoles().get(0).getRoleKey();  
 for (OilCodeReference oil : oilCodeReferenceList) {  
 oil.setModelFactory(oil.getModelFactory() + rolekey);  
 }  
 String url = "http://localhost:8082/oil/reference/add";  
 Map<String, Object> requestMap = new HashMap<>();  
 requestMap.put("oilCodeReference", oilCodeReferenceList);  
 AjaxResult result = restTemplate.postForObject(url, requestMap, AjaxResult.class);  
 return result;  
 }  
  
 @Override  
 public AjaxResult getfactorypricelist(AsmModel asmModel) {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 String rolekey = user.getUser().getRoles().get(0).getRoleKey();  
 String createby = "";  
 */\*if(asmModel.getModelFactory().contains("海南")){  
 createby=rolekey;  
 }else {  
  
 }\*/* createby = asmModel.getModelFactory() + rolekey;  
 *//获取物料分类* Long priceSystemid = asmModel.getPriceSystemId();  
 String url = "http://localhost:8082/oil/price/pricelist?SystemId=" + priceSystemid + "&&createBy=" + createby;  
 Map<String, Map<String, Map<String, Map>>> result = restTemplate.getForObject(url, Map.class);  
 List<OilMaterial> oilMaterialList = new ArrayList<>();  
 Set<String> keyset = result.keySet();  
 for (String key : keyset) {  
 OilMaterial oilMaterial = new OilMaterial();  
 Map map = result.get(key);  
 Object ob = map.get("oilMaterial");  
 oilMaterial.setMaterialName(String.*valueOf*(result.get(key).get("oilMaterial").get("materialName")));  
 oilMaterial.setMaterialCode(String.*valueOf*(result.get(key).get("oilMaterial").get("materialCode")));  
 if (result.get(key).get("oilMaterial").get("materialProp1") != null) {  
 oilMaterial.setMaterialProp1(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp1")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp2") != null) {  
 oilMaterial.setMaterialProp2(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp2")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp3") != null) {  
 oilMaterial.setMaterialProp3(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp3")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp4") != null) {  
 oilMaterial.setMaterialProp4(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp4")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp5") != null) {  
 oilMaterial.setMaterialProp5(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp5")));  
 }  
 if (result.get(key).get("oilMaterial").get("materialProp6") != null) {  
 oilMaterial.setMaterialProp6(String.*valueOf*(result.get(key).get("oilMaterial").get("materialProp6")));  
 }  
 OilSystemPrice oilSystemPrice = new OilSystemPrice();  
 if (result.get(key).get("priceValue") != null) {  
 oilSystemPrice.setPriceValue(Double.*valueOf*(String.*valueOf*(result.get(key).get("priceValue"))));  
 oilMaterial.setOilSystemPrice(oilSystemPrice);  
 }  
 if (result.get(key).get("systemId") != null) {  
 oilSystemPrice.setSystemId(Long.*valueOf*(String.*valueOf*(result.get(key).get("systemId"))));  
 oilMaterial.setOilSystemPrice(oilSystemPrice);  
 }  
 if (result.get(key).get("priceId") != null) {  
 oilSystemPrice.setPriceId(Long.*valueOf*(String.*valueOf*(result.get(key).get("priceId"))));  
 oilMaterial.setOilSystemPrice(oilSystemPrice);  
 }  
 if (result.get(key).get("refernceId") != null) {  
 oilSystemPrice.setRefernceId(Long.*valueOf*(String.*valueOf*(result.get(key).get("refernceId"))));  
 oilMaterial.setOilSystemPrice(oilSystemPrice);  
 }  
 if (result.get(key).get("materialId") != null) {  
 oilSystemPrice.setMaterialId(Long.*valueOf*(String.*valueOf*(result.get(key).get("materialId"))));  
 oilMaterial.setOilSystemPrice(oilSystemPrice);  
 }  
 if (result.get(key).get("oilMaterial").get("oilCodeReference").get("factoryCode") != null) {  
 oilMaterial.setFactoryCode(String.*valueOf*(result.get(key).get("oilMaterial").get("oilCodeReference").get("factoryCode")));  
 }  
 oilMaterialList.add(oilMaterial);  
 }  
 return AjaxResult.*success*(oilMaterialList);  
 }  
  
 @Override  
 public AjaxResult updateFactoryPricelist(List<OilSystemPrice> oilSystemPriceList) {  
 String url = "http://localhost:8082/oil/price/update";  
 Map<String, Object> requestMap = new HashMap<>();  
 requestMap.put("OilSystemPrice", oilSystemPriceList);  
 AjaxResult result = restTemplate.postForObject(url, requestMap, AjaxResult.class);  
 return result;  
 }  
  
 @Override  
 public AjaxResult addMaterialList(List<OilCodeReference> oilCodeReferenceList) {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 String rolekey = user.getUser().getRoles().get(0).getRoleKey();  
 for (OilCodeReference oil : oilCodeReferenceList) {  
 oil.setModelFactory(oil.getModelFactory() + rolekey);  
 }  
 String url = "http://localhost:8082/oil/reference/addMaterialList";  
 Map<String, Object> requestMap = new HashMap<>();  
 requestMap.put("oilCodeReference", oilCodeReferenceList);  
 AjaxResult result = restTemplate.postForObject(url, requestMap, AjaxResult.class);  
 return result;  
 }  
  
 @Override  
 public AjaxResult exportCcplantExcel(AsmModel model) {  
 HSSFWorkbook wb = new HSSFWorkbook();  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
 List<AsmModelDevice> devices = model.getDevices();  
 List<AsmModelDeviceInput> inputs = new ArrayList<>();  
 List<AsmModelDeviceOutput> outputs = new ArrayList<>();  
 List<AsmModelBlend> modelBlends = model.getBlends();  
 List<AsmModelDeviceOsbl> osbls = new ArrayList<>();  
 List<AsmModelMaterialBalance> balances = model.getMaterialBalances();  
 for (AsmModelDevice device : devices) {  
 inputs.addAll(device.getInputs());  
 outputs.addAll(device.getOutputs());  
 osbls.addAll(device.getOsbls());  
 }  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
 export\_model(wb, model, cellStyle\_4side, cellStyle\_point2, cellStyle\_default);  
 export\_modelBlend(wb, modelBlends);  
 export\_modelDevice(wb, devices, cellStyle\_4side, cellStyle\_point2, start\_row\_index, start\_col\_index, cellStyle\_default);  
 export\_modelDeviceInput(wb, inputs, cellStyle\_4side, cellStyle\_point2, start\_row\_index, start\_col\_index, cellStyle\_default);  
 export\_modelDeviceOsbl(wb, osbls, cellStyle\_4side, cellStyle\_point2, start\_row\_index, start\_col\_index, cellStyle\_default);  
 export\_modelDeviceOutput(wb, outputs, cellStyle\_4side, cellStyle\_point2, start\_row\_index, start\_col\_index, cellStyle\_default);  
 export\_modelBalance(wb, balances, cellStyle\_4side, cellStyle\_point2, start\_row\_index, start\_col\_index, cellStyle\_default);  
  
 String filename = UUID.*randomUUID*().toString() + "\_" + model.getModelName() + ".xls";  
 String downloadPath = RuoYiConfig.*getUploadPath*() + filename;  
 File desc = new File(downloadPath);  
 if (!desc.getParentFile().exists()) {  
 desc.getParentFile().mkdirs();  
 }  
 OutputStream out = null;  
 try {  
 out = new FileOutputStream(downloadPath);  
 wb.write(out);  
 return AjaxResult.*success*(filename);  
 } catch (Exception e) {  
*// logger.error("导出Excel异常{}", e.getMessage());* throw new UtilException("导出Excel失败，请联系网站管理员！");  
 } finally {  
 IOUtils.*closeQuietly*(wb);  
 IOUtils.*closeQuietly*(out);  
 }  
 }  
  
 @Override  
 public AsmModel selectReconcileDeviceByModelId(Long modelId) {  
 AsmModel model = asmModelMapper.selectAsmModelByModelId(modelId);  
 List<AsmModelDevice> devices = getReconcileDevices(modelId);  
 model.setDevices(devices);  
 */\*getFullDevices(modelId);  
 asmModelDeviceInputService.selectAsmModelDeviceInputList();\*/* return model;  
 }  
  
 @Override  
 public AjaxResult selectMesData() {  
 List<PmMtrlMoveV> pmMtrlMoveVS=mesDeviceMaterialBalanceService.selectMesDeviceMaterialBalanceList(new PmMtrlMoveV());  
 List<MbMtrlbalnsheetdataV> mbMtrlbalnsheetdataVList=mesMaterialBalanceService.selectMesMaterialBalanceList(new MbMtrlbalnsheetdataV());  
 Map map=new HashMap();  
 map.put("1",pmMtrlMoveVS);  
 map.put("2",mbMtrlbalnsheetdataVList);  
 return AjaxResult.*success*(map);  
 }  
  
 private List<AsmModelDevice> getReconcileDevices(Long modelId) {  
 AsmModelDevice device = new AsmModelDevice();  
 device.setModelId(modelId);  
 device.setDeviceName("调合");  
 List<AsmModelDevice> devices = asmModelDeviceService.selectAsmModelDeviceList(device);  
 List<AsmModelDevice> list = new ArrayList<>();  
 if (devices == null)  
 return list;  
 for (AsmModelDevice device1 : devices) {  
 AsmModelDevice obj = asmModelDeviceService.selectAsmModelDeviceByDeviceId(device1.getDeviceId());  
 list.add(obj);  
 }  
 return list;  
 }  
  
 private void export\_modelBlend(HSSFWorkbook wb, List<AsmModelBlend> modelBlends) {  
 {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
  
 Sheet sheet = wb.createSheet("油品调和表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("调和代码");  
 Cell cell1 = row.createCell(start\_col\_index + 1);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("模型代码");  
 Cell cell2 = row.createCell(start\_col\_index + 2);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("物料编码");  
 Cell cell3 = row.createCell(start\_col\_index + 3);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("物料名称");  
 Cell cell4 = row.createCell(start\_col\_index + 4);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("性质编码");  
 Cell cell5 = row.createCell(start\_col\_index + 5);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("性质名称");  
 Cell cell6 = row.createCell(start\_col\_index + 6);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("物性值");  
 Cell cell7 = row.createCell(start\_col\_index + 7);  
 cell7.setCellStyle(cellStyle\_4side);  
 cell7.setCellValue("物性单位");  
 }  
 }  
 start\_row\_index++;  
 for (AsmModelBlend blend : modelBlends) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(blend.getBlendId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(blend.getModelId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(blend.getMaterielCode());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(blend.getMaterielName());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(blend.getPropertyCode());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(blend.getPropertyName());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(blend.getPropertyValue());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_point2);  
 cell6.setCellValue(blend.getPropertyUnit());  
 start\_row\_index++;  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
 }  
  
 private void export\_modelBalance(HSSFWorkbook wb, List<AsmModelMaterialBalance> balances, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, int start\_row\_index, int start\_col\_index, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("物料平衡表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("物料代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("模型代码");  
 Cell cell12 = row.createCell(start\_col\_index + 2);  
 cell12.setCellStyle(cellStyle\_4side);  
 cell12.setCellValue("物料属性");  
 Cell cell1 = row.createCell(start\_col\_index + 3);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("物料编码");  
 Cell cell2 = row.createCell(start\_col\_index + 4);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("物料名称");  
 Cell cell3 = row.createCell(start\_col\_index + 5);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("量/万吨");  
 Cell cell4 = row.createCell(start\_col\_index + 6);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("碳排放");  
 }  
 start\_row\_index++;  
 for (AsmModelMaterialBalance balance : balances) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(balance.getMaterielId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(balance.getModelId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(balance.getMaterielProperty());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(balance.getMaterielCode());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(balance.getMaterielName());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(balance.getMaterielQuan());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 if (balance.getCo2z() == null) {  
  
 } else {  
 cell5.setCellValue(balance.getCo2z());  
 }  
 start\_row\_index++;  
 }  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 private void export\_modelDeviceOutput(HSSFWorkbook wb, List<AsmModelDeviceOutput> outputs, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, int start\_row\_index, int start\_col\_index, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("装置产出物料表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置产出代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("装置代码");  
 Cell cell12 = row.createCell(start\_col\_index + 2);  
 cell12.setCellStyle(cellStyle\_4side);  
 cell12.setCellValue("物料名称");  
 Cell cell1 = row.createCell(start\_col\_index + 3);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("物料编码");  
 Cell cell2 = row.createCell(start\_col\_index + 4);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("产出量");  
 Cell cell3 = row.createCell(start\_col\_index + 5);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("物料碳足迹");  
 Cell cell4 = row.createCell(start\_col\_index + 6);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("物料碳强度");  
 Cell cell5 = row.createCell(start\_col\_index + 7);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("模型代码");  
 Cell cell6 = row.createCell(start\_col\_index + 8);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("模型名称");  
 Cell cell7 = row.createCell(start\_col\_index + 9);  
 cell7.setCellStyle(cellStyle\_4side);  
 cell7.setCellValue("方案编码");  
 Cell cell8 = row.createCell(start\_col\_index + 10);  
 cell8.setCellStyle(cellStyle\_4side);  
 cell8.setCellValue("方案名称");  
 Cell cell9 = row.createCell(start\_col\_index + 11);  
 cell9.setCellStyle(cellStyle\_4side);  
 cell9.setCellValue("物料属性");  
 Cell cell10 = row.createCell(start\_col\_index + 12);  
 cell10.setCellStyle(cellStyle\_4side);  
 cell10.setCellValue("重点产品分类");  
 }  
 start\_row\_index++;  
 for (AsmModelDeviceOutput output : outputs) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(output.getOutputId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(output.getDeviceId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(output.getOutputName());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(output.getOutputCode());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(output.getOutputQuantity());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(output.getOutputCo2());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(output.getOutputCo2Factor());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_default);  
 cell6.setCellValue(output.getModelId());  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_point2);  
 cell7.setCellValue(output.getModelName());  
 Cell cell8 = row.createCell(start\_col\_index + 9);  
 cell8.setCellStyle(cellStyle\_point2);  
 cell8.setCellValue(output.getPlanCode());  
 Cell cell9 = row.createCell(start\_col\_index + 10);  
 cell9.setCellStyle(cellStyle\_point2);  
 cell9.setCellValue(output.getPlanName());  
 Cell cell10 = row.createCell(start\_col\_index + 11);  
 cell10.setCellStyle(cellStyle\_point2);  
 cell10.setCellValue(output.getMaterialProperty());  
 Cell cell11 = row.createCell(start\_col\_index + 12);  
 cell11.setCellStyle(cellStyle\_point2);  
 cell11.setCellValue(output.getKeyProudctClass());  
 start\_row\_index++;  
 }  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 private void export\_modelDeviceOsbl(HSSFWorkbook wb, List<AsmModelDeviceOsbl> osbls, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, int start\_row\_index, int start\_col\_index, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("装置消耗表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置消耗代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("装置代码");  
 Cell cell12 = row.createCell(start\_col\_index + 2);  
 cell12.setCellStyle(cellStyle\_4side);  
 cell12.setCellValue("装置名称");  
 Cell cell1 = row.createCell(start\_col\_index + 3);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("公用工程代码");  
 Cell cell2 = row.createCell(start\_col\_index + 4);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("公用工程名称");  
 Cell cell3 = row.createCell(start\_col\_index + 5);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("公用工程编码");  
 Cell cell4 = row.createCell(start\_col\_index + 6);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("折标系数");  
 Cell cell5 = row.createCell(start\_col\_index + 7);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("CO2排放系数");  
 Cell cell6 = row.createCell(start\_col\_index + 8);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("热值");  
 Cell cell7 = row.createCell(start\_col\_index + 9);  
 cell7.setCellStyle(cellStyle\_4side);  
 cell7.setCellValue("消耗量");  
 Cell cell8 = row.createCell(start\_col\_index + 10);  
 cell8.setCellStyle(cellStyle\_4side);  
 cell8.setCellValue("碳排放量");  
 Cell cell9 = row.createCell(start\_col\_index + 11);  
 cell9.setCellStyle(cellStyle\_4side);  
 cell9.setCellValue("模型代码");  
 Cell cell10 = row.createCell(start\_col\_index + 12);  
 cell10.setCellStyle(cellStyle\_4side);  
 cell10.setCellValue("模型名称");  
 Cell cell11 = row.createCell(start\_col\_index + 13);  
 cell11.setCellStyle(cellStyle\_4side);  
 cell11.setCellValue("方案编码");  
 Cell cell13 = row.createCell(start\_col\_index + 14);  
 cell13.setCellStyle(cellStyle\_4side);  
 cell13.setCellValue("方案名称");  
 }  
 start\_row\_index++;  
 for (AsmModelDeviceOsbl osbl : osbls) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(osbl.getConsumeId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(osbl.getDeviceId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(osbl.getDeviceName());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_default);  
 cell2.setCellValue(osbl.getOsblId());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(osbl.getOsblName());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(osbl.getOsblCode());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(osbl.getFoldCoefficient());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_point2);  
 cell6.setCellValue(osbl.getCo2EmissionCoefficient());  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_point2);  
 cell7.setCellValue(osbl.getCalorificValue());  
 Cell cell8 = row.createCell(start\_col\_index + 9);  
 cell8.setCellStyle(cellStyle\_point2);  
 cell8.setCellValue(osbl.getConsumption());  
 Cell cell9 = row.createCell(start\_col\_index + 10);  
 cell9.setCellStyle(cellStyle\_point2);  
 cell9.setCellValue(osbl.getCarbonEmission());  
 Cell cell10 = row.createCell(start\_col\_index + 11);  
 cell10.setCellStyle(cellStyle\_default);  
 cell10.setCellValue(osbl.getModelId());  
 Cell cell11 = row.createCell(start\_col\_index + 12);  
 cell11.setCellStyle(cellStyle\_point2);  
 cell11.setCellValue(osbl.getPlanCode());  
 Cell cell12 = row.createCell(start\_col\_index + 13);  
 cell12.setCellStyle(cellStyle\_point2);  
 cell12.setCellValue(osbl.getModelName());  
 Cell cell13 = row.createCell(start\_col\_index + 14);  
 cell13.setCellStyle(cellStyle\_point2);  
 cell13.setCellValue(osbl.getPlanName());  
 start\_row\_index++;  
 }  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 private void export\_modelDeviceInput(HSSFWorkbook wb, List<AsmModelDeviceInput> inputs, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, int start\_row\_index, int start\_col\_index, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("装置投入物料表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置投入代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("装置代码");  
 Cell cell12 = row.createCell(start\_col\_index + 2);  
 cell12.setCellStyle(cellStyle\_4side);  
 cell12.setCellValue("上一级产出代码");  
 Cell cell1 = row.createCell(start\_col\_index + 3);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("投入量");  
 Cell cell2 = row.createCell(start\_col\_index + 4);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("原料碳足迹");  
 Cell cell3 = row.createCell(start\_col\_index + 5);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("模型编码");  
 Cell cell4 = row.createCell(start\_col\_index + 6);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("所属项目名称");  
 Cell cell5 = row.createCell(start\_col\_index + 7);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("方案编码");  
 Cell cell6 = row.createCell(start\_col\_index + 8);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("方案名称");  
 Cell cell7 = row.createCell(start\_col\_index + 9);  
 cell7.setCellStyle(cellStyle\_4side);  
 cell7.setCellValue("物料编码");  
 Cell cell8 = row.createCell(start\_col\_index + 10);  
 cell8.setCellStyle(cellStyle\_4side);  
 cell8.setCellValue("物料名称");  
 Cell cell9 = row.createCell(start\_col\_index + 11);  
 cell9.setCellStyle(cellStyle\_4side);  
 cell9.setCellValue("物料属性");  
 Cell cell10 = row.createCell(start\_col\_index + 12);  
 cell10.setCellStyle(cellStyle\_4side);  
 cell10.setCellValue("是否计算碳");  
 }  
 start\_row\_index++;  
 for (AsmModelDeviceInput input : inputs) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(input.getInputId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(input.getDeviceId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_default);  
 if (input.getSuperOutputId() == null) {  
  
 } else {  
 cell1.setCellValue(input.getSuperOutputId());  
 }  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(input.getInputQuantity());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(input.getInputCo2());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_default);  
 cell4.setCellValue(input.getModelId());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(input.getModelName());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_point2);  
 cell6.setCellValue(input.getPlanCode());  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_point2);  
 cell7.setCellValue(input.getPlanName());  
 Cell cell8 = row.createCell(start\_col\_index + 9);  
 cell8.setCellStyle(cellStyle\_point2);  
 cell8.setCellValue(input.getInputCode());  
 Cell cell9 = row.createCell(start\_col\_index + 10);  
 cell9.setCellStyle(cellStyle\_point2);  
 cell9.setCellValue(input.getInputName());  
 Cell cell10 = row.createCell(start\_col\_index + 11);  
 cell10.setCellStyle(cellStyle\_point2);  
 cell10.setCellValue(input.getMaterialProperty());  
 Cell cell11 = row.createCell(start\_col\_index + 12);  
 cell11.setCellStyle(cellStyle\_point2);  
 cell11.setCellValue(input.getIsCalc());  
 start\_row\_index++;  
 }  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 private void export\_modelDevice(HSSFWorkbook wb, List<AsmModelDevice> devices, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, int start\_row\_index, int start\_col\_index, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("装置表");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("模型代码");  
 Cell cell12 = row.createCell(start\_col\_index + 2);  
 cell12.setCellStyle(cellStyle\_4side);  
 cell12.setCellValue("模型名称");  
 Cell cell1 = row.createCell(start\_col\_index + 3);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("装置编码");  
 Cell cell2 = row.createCell(start\_col\_index + 4);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("装置名称");  
 Cell cell3 = row.createCell(start\_col\_index + 5);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("装置加工量");  
 Cell cell4 = row.createCell(start\_col\_index + 6);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("装置碳足迹");  
 Cell cell5 = row.createCell(start\_col\_index + 7);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("吨原料碳排放因子");  
 Cell cell6 = row.createCell(start\_col\_index + 8);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("虚拟装置");  
 }  
 start\_row\_index++;  
 for (AsmModelDevice device : devices) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(device.getDeviceId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(device.getModelId());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_point2);  
 cell1.setCellValue(device.getModelName());  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(device.getDeviceCode());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(device.getDeviceName());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(device.getDeviceProcess());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(device.getDeviceCo2());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_point2);  
 cell6.setCellValue(device.getDeviceFactor());  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_default);  
 cell7.setCellValue(device.getIsVirtual());  
 start\_row\_index++;  
 }  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
 private void export\_model(HSSFWorkbook wb, AsmModel model, HSSFCellStyle cellStyle\_4side, HSSFCellStyle cellStyle\_point2, HSSFCellStyle cellStyle\_default) {  
 Sheet sheet = wb.createSheet("模型表");  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("模型代码");  
 Cell cell111 = row.createCell(start\_col\_index + 1);  
 cell111.setCellStyle(cellStyle\_4side);  
 cell111.setCellValue("模型名称");  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("所属企业代码");  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("所属工厂");  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("所属区域");  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("所属省份");  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_4side);  
 cell5.setCellValue("所属城市");  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_4side);  
 cell6.setCellValue("所属公司");  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_4side);  
 cell7.setCellValue("所属区县");  
 Cell cell8 = row.createCell(start\_col\_index + 9);  
 cell8.setCellStyle(cellStyle\_4side);  
 cell8.setCellValue("开始时间");  
 Cell cell9 = row.createCell(start\_col\_index + 10);  
 cell9.setCellStyle(cellStyle\_4side);  
 cell9.setCellValue("结束时间");  
 Cell cell17 = row.createCell(start\_col\_index + 11);  
 cell17.setCellStyle(cellStyle\_4side);  
 cell17.setCellValue("工厂碳排放");  
 Cell cell18 = row.createCell(start\_col\_index + 12);  
 cell18.setCellStyle(cellStyle\_4side);  
 cell18.setCellValue("产品碳排放");  
 Cell cell19 = row.createCell(start\_col\_index + 13);  
 cell19.setCellStyle(cellStyle\_4side);  
 cell19.setCellValue("总加工量");  
 Cell cell20 = row.createCell(start\_col\_index + 14);  
 cell20.setCellStyle(cellStyle\_4side);  
 cell20.setCellValue("总产出量");  
 Cell cell21 = row.createCell(start\_col\_index + 15);  
 cell21.setCellStyle(cellStyle\_4side);  
 cell21.setCellValue("价格体系");  
 Cell cell22 = row.createCell(start\_col\_index + 16);  
 cell22.setCellStyle(cellStyle\_4side);  
 cell22.setCellValue("价格体系id");  
 Cell cell23 = row.createCell(start\_col\_index + 17);  
 cell23.setCellStyle(cellStyle\_4side);  
 cell23.setCellValue("是否计算过");  
 }  
 start\_row\_index++;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_default);  
 cellTitle.setCellValue(model.getModelId());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_default);  
 cell.setCellValue(model.getModelName());  
 Cell cell1 = row.createCell(start\_col\_index + 2);  
 cell1.setCellStyle(cellStyle\_default);  
 if (model.getModelCompanyId() == null) {  
 } else {  
 cell1.setCellValue(model.getModelCompanyId());  
 }  
 Cell cell2 = row.createCell(start\_col\_index + 3);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(model.getModelFactory());  
 Cell cell3 = row.createCell(start\_col\_index + 4);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(model.getModelRegion());  
 Cell cell4 = row.createCell(start\_col\_index + 5);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(model.getModelProvince());  
 Cell cell5 = row.createCell(start\_col\_index + 6);  
 cell5.setCellStyle(cellStyle\_point2);  
 cell5.setCellValue(model.getModelCity());  
 Cell cell6 = row.createCell(start\_col\_index + 7);  
 cell6.setCellStyle(cellStyle\_point2);  
 cell6.setCellValue(model.getModelCompany());  
 Cell cell7 = row.createCell(start\_col\_index + 8);  
 cell7.setCellStyle(cellStyle\_point2);  
 cell7.setCellValue(model.getModelArea());  
 Cell cell8 = row.createCell(start\_col\_index + 9);  
 cell8.setCellStyle(cellStyle\_point2);  
 cell8.setCellValue(model.getBeginTime());  
 Cell cell9 = row.createCell(start\_col\_index + 10);  
 cell9.setCellStyle(cellStyle\_point2);  
 cell9.setCellValue(model.getEndTime());  
 Cell cell10 = row.createCell(start\_col\_index + 11);  
 cell10.setCellStyle(cellStyle\_point2);  
 cell10.setCellValue(model.getTotalFactoryCo2());  
 Cell cell11 = row.createCell(start\_col\_index + 12);  
 cell11.setCellStyle(cellStyle\_point2);  
 cell11.setCellValue(model.getTotalProductCo2());  
 Cell cell12 = row.createCell(start\_col\_index + 13);  
 cell12.setCellStyle(cellStyle\_point2);  
 cell12.setCellValue(model.getTotalInput());  
 Cell cell13 = row.createCell(start\_col\_index + 14);  
 cell13.setCellStyle(cellStyle\_point2);  
 cell13.setCellValue(model.getTotalOutput());  
 Cell cell14 = row.createCell(start\_col\_index + 15);  
 cell14.setCellStyle(cellStyle\_point2);  
 cell14.setCellValue(model.getPriceSystem());  
 Cell cell15 = row.createCell(start\_col\_index + 16);  
 cell15.setCellStyle(cellStyle\_default);  
 if (model.getPriceSystemId() == null) {  
  
 } else {  
 cell15.setCellValue(model.getPriceSystemId());  
 }  
 Cell cell16 = row.createCell(start\_col\_index + 17);  
 cell16.setCellStyle(cellStyle\_point2);  
 cell16.setCellValue(model.getIsCalc());  
 start\_row\_index++;  
 }  
 sheet.setColumnWidth(start\_col\_index, 10 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 3, 30 \* 256);  
 }  
  
  
 private List<AsmModelDevice> getGasDevices(Long modelId) {  
 AsmModelDevice device = new AsmModelDevice();  
 device.setModelId(modelId);  
 List<AsmModelDevice> devices = asmModelDeviceService.selectAsmModelDeviceList(device);  
  
  
 List<AsmModelDevice> list = new ArrayList<>();  
 if (devices == null)  
 return list;  
 for (AsmModelDevice device1 : devices) {  
 AsmModelDevice obj = asmModelDeviceService.selectGasAsmModelDeviceByDeviceId(device1.getDeviceId());  
 list.add(obj);  
 }  
 return list;  
 }  
  
 @Override  
 public ArrayList<DischargeVo> cmissionCalculation(AsmModel cModel) {  
 ArrayList<DischargeVo> dischargeVos = this.co2pf(cModel);  
 return dischargeVos;  
 }  
  
 @Override  
 public ArrayList<AsmModelMaterialBalance> selectCo2z(AsmModel cModel) {  
 ArrayList<AsmModelMaterialBalance> asmModelMaterialBalances = new ArrayList<>();  
 Double sum = 0.00;  
 ArrayList<AsmModelMaterialBalance> asmModelMaterialBalance = asmModelMaterialBalanceService.selectAsmModelBlendByModelId(cModel.getModelId());  
 ArrayList<AsmModelDeviceOutput> asmModelDeviceOutputs = asmModelDeviceOutputService.selectAsmModelDeviceOutputByModelId(cModel.getModelId());  
 for (AsmModelMaterialBalance modelMaterialBalance : asmModelMaterialBalance) {  
 for (AsmModelDeviceOutput asmModelDeviceOutput : asmModelDeviceOutputs) {  
 if (modelMaterialBalance.getMaterielCode().equals(asmModelDeviceOutput.getOutputCode())) {  
 sum += asmModelDeviceOutput.getOutputCo2();  
 }  
 }  
 modelMaterialBalance.setCo2z(sum);  
 sum = 0.00;  
 asmModelMaterialBalances.add(modelMaterialBalance);  
 }  
 return asmModelMaterialBalances;  
 }  
  
 private ArrayList<DischargeVo> co2pf(AsmModel cModel) {  
 ArrayList<DischargeVo> dischargeVos = new ArrayList<>();  
 Double rsSum = 0.0;  
 Double gySum = 0.0;  
 Double zqSum = 0.0;  
 Double hdSum = 0.0;  
 Double qtSum = 0.0;  
 Double co2Sum = 0.0;  
 Double zhSum = 0.0;  
 Double co2v = 0.0;  
 List<AsmModelDevice> cModelDevices = asmModelDeviceService.selectDevicesByModelId(cModel.getModelId());  
 for (AsmModelDevice cModelDevice : cModelDevices) {  
 ArrayList<Double> zhs = new ArrayList<>();  
 ArrayList<Double> co2s = new ArrayList<>();  
 DischargeVo dischargeVo = new DischargeVo();  
 dischargeVo.setDeviceName(cModelDevice.getDeviceName());  
 ArrayList<AsmModelDeviceOsbl> selectobs = asmModelDeviceOsblService.selectobs(cModelDevice.getDeviceId());  
 for (AsmModelDeviceOsbl selectob : selectobs) {  
 AsmOsblDict cOsblDict = asmOsblDictMapper.selectCOsblDictByOsblId(selectob.getOsblId());  
 BigDecimal bg1 = new BigDecimal(selectob.getConsumption());  
 BigDecimal bg2 = new BigDecimal(cOsblDict.getCo2EmissionCoefficient());  
 BigDecimal bg3 = new BigDecimal(cOsblDict.getFoldCoefficient());  
 if (rsList.contains(selectob.getOsblCode())) {  
 rsSum += bg1.multiply(bg2).doubleValue();  
 }  
 if ("CKE".equals(selectob.getOsblCode())) {  
 gySum += bg1.multiply(bg2).doubleValue();  
 }  
 if (zqList.contains(selectob.getOsblCode())) {  
 zqSum += bg1.multiply(bg2).doubleValue();  
 }  
 if (hdList.contains(selectob.getOsblCode())) {  
 hdSum += bg1.multiply(bg2).doubleValue();  
 }  
 if (qtList.contains(selectob.getOsblCode())) {  
 qtSum += bg1.multiply(bg2).doubleValue();  
 }  
 if (selectob.getConsumption() != null && selectob.getConsumption() != 0 && cOsblDict.getFoldCoefficient() != null && cOsblDict.getFoldCoefficient() != 0) {  
 double v1 = bg1.multiply(bg3).doubleValue();  
 zhs.add(v1);  
 }  
 if (selectob.getConsumption() != null && selectob.getConsumption() != 0 && cOsblDict.getCo2EmissionCoefficient() != null && cOsblDict.getCo2EmissionCoefficient() != 0) {  
 double v = bg1.multiply(bg2).doubleValue();  
 co2s.add(v);  
 }  
  
 dischargeVo = this.z(dischargeVo, selectob, cOsblDict);  
  
 }  
 for (int i = 0; i < zhs.size(); i++) {  
 zhSum += zhs.get(i);  
 }  
 for (int i = 0; i < co2s.size(); i++) {  
 co2Sum += co2s.get(i);  
 }  
 dischargeVo.setcO2Emissions(co2Sum);  
 dischargeVo.setComprehensiveEnergyConsumption(zhSum);  
 dischargeVo.setDeviceName(cModelDevice.getDeviceName());  
 dischargeVo.setCombustionEmissions(rsSum);  
 dischargeVo.setProcessDischarge(gySum);  
 dischargeVo.setSteamDischarge(zqSum);  
 dischargeVo.setPowerConsumptionEmission(hdSum);  
 dischargeVo.setOtherEmissions(qtSum);  
 dischargeVo.setDirectDischarge(rsSum + gySum);  
 dischargeVo.setIndirectEmissions(qtSum + hdSum + zqSum);  
 dischargeVos.add(dischargeVo);  
 rsSum = 0.00;  
 gySum = 0.00;  
 zqSum = 0.00;  
 hdSum = 0.00;  
 qtSum = 0.00;  
 co2Sum = 0.00;  
 zhSum = 0.00;  
 }  
 Double qzjsum = 0.0;  
 Double qjjsum = 0.0;  
 Double qrssum = 0.0;  
 Double qgysum = 0.0;  
 Double qzqsum = 0.0;  
 Double qhdsum = 0.0;  
 Double qqtsum = 0.0;  
 for (DischargeVo dischargeVo : dischargeVos) {  
 qzjsum += dischargeVo.getDirectDischarge();  
 qjjsum += dischargeVo.getIndirectEmissions();  
 qrssum += dischargeVo.getCombustionEmissions();  
 qgysum += dischargeVo.getProcessDischarge();  
 qzqsum += dischargeVo.getSteamDischarge();  
 qhdsum += dischargeVo.getPowerConsumptionEmission();  
 qqtsum += dischargeVo.getOtherEmissions();  
 }  
 for (DischargeVo dischargeVo : dischargeVos) {  
 dischargeVo.setQdirectDischarge(qzjsum);  
 dischargeVo.setQindirectEmissions(qjjsum);  
 dischargeVo.setQcombustionEmissions(qrssum);  
 dischargeVo.setQprocessDischarge(qgysum);  
 dischargeVo.setQsteamDischarge(qzqsum);  
 dischargeVo.setQpowerConsumptionEmission(qhdsum);  
 dischargeVo.setQotherEmissions(qqtsum);  
 }  
 for (DischargeVo dischargeVo : dischargeVos) {  
 co2v += dischargeVo.getcO2Emissions();  
  
 }  
 for (DischargeVo dischargeVo : dischargeVos) {  
 dischargeVo.setCo2z(co2v);  
 BigDecimal bg1 = new BigDecimal(dischargeVo.getCo2z());  
 BigDecimal bg2 = new BigDecimal(dischargeVo.getcO2Emissions());  
 Double b = bg2.divide(bg1, 3, BigDecimal.*ROUND\_HALF\_UP*).doubleValue() \* 100;  
 dischargeVo.setPercentage(b);  
 }  
  
 return dischargeVos;  
 }  
  
 private DischargeVo z(DischargeVo dischargeVo, AsmModelDeviceOsbl selectob, AsmOsblDict cOsblDict) {  
 switch (cOsblDict.getDictCode()) {  
 case "CAT":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCatalyzer(selectob.getConsumption());  
 }  
 break;  
 case "CCC":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setAuxiliaryMaterials(selectob.getConsumption());  
 }  
 break;  
 case "WAT":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setFreshWater(selectob.getConsumption());  
 }  
 break;  
 case "CWT":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCirculatingWater(selectob.getConsumption());  
 }  
 break;  
 case "SWT":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setWaterSoftening(selectob.getConsumption());  
 }  
 break;  
 case "DAW":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setDemineralizedWater(selectob.getConsumption());  
 }  
 break;  
 case "DOW":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setDeaeratedWater(selectob.getConsumption());  
 }  
 break;  
 case "COT":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCondensate(selectob.getConsumption());  
 }  
 break;  
 case "COH":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCondensateHeating(selectob.getConsumption());  
 }  
 break;  
 case "KWS":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setSelfProduction(selectob.getConsumption());  
 }  
 break;  
 case "KWH":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setPowerGrid(selectob.getConsumption());  
 }  
 break;  
 case "10M":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setTenMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "05M":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setFiveMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "3D5":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setThreePointFiveMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "2D5":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setTwoPointFiveMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "1D5":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setOnePointFiveMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "1DM":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setOneMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "D7M":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setZeroPointSevenMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "D5M":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setZeroPointFiveMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "LD3":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setZeroPointThreeMPaSteam(selectob.getConsumption());  
 }  
 break;  
 case "ECC":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setStandardCoal(selectob.getConsumption());  
 }  
 break;  
 case "CCK":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCoke(selectob.getConsumption());  
 }  
 break;  
 case "CKK":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setPetroleumCoke(selectob.getConsumption());  
 }  
 break;  
 case "EOO":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setStandardOil(selectob.getConsumption());  
 }  
 break;  
 case "UFL":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setFuelOil(selectob.getConsumption());  
 }  
 break;  
 case "GAS":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setGasoline(selectob.getConsumption());  
 }  
 break;  
 case "DIO":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setDieselOil(selectob.getConsumption());  
 }  
 break;  
 case "LPS":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setLiquefiedPetroleumGas(selectob.getConsumption());  
 }  
 break;  
 case "LTG":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setLiquefiedNaturalGas(selectob.getConsumption());  
 }  
 break;  
 case "UGS":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setFuelGas(selectob.getConsumption());  
 }  
 break;  
 case "GAG":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setRetortGas(selectob.getConsumption());  
 }  
 break;  
 case "NAG":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setNaturalGas(selectob.getConsumption());  
 }  
 break;  
 case "CKE":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setCatalyticCoking(selectob.getConsumption());  
 }  
 break;  
 case "LLH":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setLowTemperatureHeat(selectob.getConsumption());  
 }  
 break;  
 case "NNS":  
 if (selectob.getConsumption() != null) {  
 dischargeVo.setNitrogen(selectob.getConsumption());  
 }  
 break;  
 }  
 return dischargeVo;  
 }  
  
 protected List<AsmModelDevice> getFullDevices(Long modelId) {  
 AsmModelDevice device = new AsmModelDevice();  
 device.setModelId(modelId);  
 List<AsmModelDevice> devices = asmModelDeviceService.selectAsmModelDeviceList(device);  
 List<AsmModelDevice> list = new ArrayList<>();  
 if (devices == null)  
 return list;  
 for (AsmModelDevice device1 : devices) {  
 AsmModelDevice obj = asmModelDeviceService.selectAsmModelDeviceByDeviceId(device1.getDeviceId());  
 list.add(obj);  
 }  
 return list;  
 }  
  
 protected List<AsmModelDevice> getexportFullDevices(Long modelId) {  
 AsmModelDevice device = new AsmModelDevice();  
 device.setModelId(modelId);  
 List<AsmModelDevice> devices = asmModelDeviceService.selectAsmModelDeviceList(device);  
 List<AsmModelDevice> list = new ArrayList<>();  
 if (devices == null)  
 return list;  
 for (AsmModelDevice device1 : devices) {  
 AsmModelDevice obj = asmModelDeviceService.selectexportAsmModelDeviceByDeviceId(device1.getDeviceId());  
 list.add(obj);  
 }  
 return list;  
 }  
  
 protected List<AsmModelBlend> getBlends(Long modelId) {  
 AsmModelBlend blend = new AsmModelBlend();  
 blend.setModelId(modelId);  
 List<AsmModelBlend> list = asmModelBlendService.selectAsmModelBlendList(blend);  
 if (list == null)  
 list = new ArrayList<>();  
 return list;  
 }  
  
 protected List<AsmModelMaterialBalance> getBalance(Long modelId) {  
 AsmModelMaterialBalance balance = new AsmModelMaterialBalance();  
 balance.setModelId(modelId);  
 List<AsmModelMaterialBalance> list = asmModelMaterialBalanceService.selectAsmModelMaterialBalanceList(balance);  
 if (list == null)  
 list = new ArrayList<>();  
 return list;  
 }  
  
 private void export\_all\_devices(HSSFWorkbook wb, List<AsmModelDevice> devices) {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
  
 Sheet sheet = wb.createSheet("装置碳排放量");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置名称");  
 Cell cell1 = row.createCell(start\_col\_index + 1);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("装置碳排放量");  
 }  
 {  
 Row row = createRow(sheet, start\_row\_index + 1);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("");  
 Cell cell1 = row.createCell(start\_col\_index + 1);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("万吨");  
 sheet.addMergedRegion(new CellRangeAddress(start\_row\_index + 0, start\_row\_index + 1, start\_col\_index, start\_col\_index));  
 }  
 }  
 start\_row\_index += 2;  
 for (AsmModelDevice device : devices) {  
 if (device.getIsVirtual() == 1)  
 continue;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_name);  
 cellTitle.setCellValue(device.getDeviceName());  
 Cell cell = row.createCell(start\_col\_index + 1);  
 cell.setCellStyle(cellStyle\_point2);  
 cell.setCellValue(device.getDeviceCo2() / 10000);  
 start\_row\_index++;  
 }  
 sheet.setColumnWidth(start\_col\_index, 30 \* 256);  
 sheet.setColumnWidth(start\_col\_index + 1, 10 \* 256);  
 }  
  
 private void export\_material(HSSFWorkbook wb, Sheet sheet, List<AsmModelDeviceInput> raw\_materials, List<AsmModelDeviceInput> oil\_materials, List<AsmModelDeviceOutput> products, int start\_row\_index, int start\_column\_index, int printnum, List<OilMaterial> oilMaterialList) {  
  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
  
 AsmModelDeviceOutput asmModelDeviceOutput = new AsmModelDeviceOutput();  
 Double output\_quantity = 0.0;  
 Double totalOutputQuality = 0.0;  
 Double totalCo2e = 0.0;  
 Double output\_quantity2 = 0.0;  
 Double output\_co21 = 0.0;  
 AsmModelDeviceInput asmModelDeviceInput1 = new AsmModelDeviceInput();  
 List<AsmModelDeviceInput> inputs1 = new ArrayList<>();  
 if (products.size() > 0) {  
 int i = 0;  
 *//计算合计的信息* for (i = 0; i < products.size(); i++) {  
 *//对产出的co2e减去投入的co2e* output\_quantity2 = products.get(i).getOutputQuantity();  
 output\_co21 = products.get(i).getOutputCo2();  
 asmModelDeviceInput1.setSuperOutputId(products.get(i).getOutputId());  
 asmModelDeviceInput1.setInputName(products.get(i).getOutputName());  
 asmModelDeviceInput1.setInputCode(products.get(i).getOutputCode());  
 inputs1 = asmModelDeviceInputService.selectAsmModelDeviceInputList(asmModelDeviceInput1);  
 for (AsmModelDeviceInput amd : inputs1) {  
 output\_quantity2 = output\_quantity2 - amd.getInputQuantity();  
 output\_co21 = output\_co21 - amd.getInputCo2();  
 }  
 products.get(i).setOutputCo2(output\_co21);  
 products.get(i).setOutputQuantity(output\_quantity2);  
 totalCo2e += products.get(i).getOutputCo2();  
 totalOutputQuality += products.get(i).getOutputQuantity();  
 }  
 for (i = 0; i < products.size(); i++) {  
 if (products.get(i).getOutputCode().equals("LOS")) {  
 output\_quantity = output\_quantity + products.get(i).getOutputQuantity();  
 products.remove(i);  
 i = -1;  
 }  
 }  
 AsmModelDeviceOutput asmModelDeviceOutput1 = new AsmModelDeviceOutput();  
 asmModelDeviceOutput1.setOutputCo2(totalCo2e);  
 asmModelDeviceOutput1.setOutputQuantity(totalOutputQuality);  
 asmModelDeviceOutput1.setOutputCo2Factor(totalCo2e / totalOutputQuality);  
 asmModelDeviceOutput1.setOutputName("合计");  
 products.add(asmModelDeviceOutput1);  
 asmModelDeviceOutput.setOutputQuantity(output\_quantity);  
 asmModelDeviceOutput.setOutputName("损失");  
 asmModelDeviceOutput.setOutputCode("LOS");  
 asmModelDeviceOutput.setOutputCo2(0.00);  
 asmModelDeviceOutput.setOutputCo2Factor(0.00);  
 products.add(asmModelDeviceOutput);  
 }  
  
 Double totalQuality = 0.0;  
 Double oilQuality = 0.0;  
 Double outoilQuality = 0.0;  
 *//导出投入原料油* Map<String, Integer> indexmap = export\_materialOil(wb, sheet, raw\_materials, oil\_materials, products, start\_row\_index, start\_column\_index, printnum, oilMaterialList, totalQuality, oilQuality, outoilQuality, 0);  
 sheet.setColumnWidth(start\_column\_index + 0, 20 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 1, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 2, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 3, 20 \* 256);  
 start\_row\_index = indexmap.get("row");  
 start\_column\_index = indexmap.get("column");  
 sheet.setColumnWidth(start\_column\_index + 0, 20 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 1, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 2, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 3, 20 \* 256);  
 *//对products里的数据分级* Set<String> MaterialProp3set = new HashSet<>();  
 Set<String> MaterialProp2set = new HashSet<>();  
 Set<String> MaterialProp1set = new HashSet<>();  
  
 for (OilMaterial oil : oilMaterialList) {  
 MaterialProp2set.add(oil.getMaterialProp2());  
 MaterialProp3set.add(oil.getMaterialProp3());  
 MaterialProp1set.add(oil.getMaterialProp1());  
 }  
  
 Integer num = 0;  
 *//一级分类* for (Iterator it = MaterialProp1set.iterator(); it.hasNext(); ) {  
 String s = (String) it.next();  
 if (s == null || s.equals("")) {  
 continue;  
 }  
 List<AsmModelDeviceOutput> asmModelDeviceOutputList = new ArrayList<>();  
 AsmModelDeviceOutput output = new AsmModelDeviceOutput();  
 Double outputquality = 0.0;  
 Double outputco2e = 0.0;  
  
 AsmModelDeviceInput asmModelDeviceInput = new AsmModelDeviceInput();  
 List<AsmModelDeviceInput> inputs = new ArrayList<>();  
 for (OilMaterial oil : oilMaterialList) {  
  
 if (oil.getMaterialProp1() != null && oil.getMaterialProp1().equals(s)) {  
 for (int i = 0; i < products.size(); i++) {  
 Double co2eFactor = 0.0;  
 Double output\_quantity1 = 0.0;  
 Double output\_co2 = 0.0;  
 if (oil.getFactoryCode() != null && oil.getFactoryCode().equals(products.get(i).getOutputCode())) {  
 asmModelDeviceOutputList.add(products.get(i));  
 output.setMaterialProp1List(asmModelDeviceOutputList);  
 output.setOutputName(oil.getMaterialProp1());  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 i = 0;  
 num++;  
 }  
 }  
 }  
 }  
 products.add(output);  
 }  
  
 *//二级分类* for (Iterator it = MaterialProp2set.iterator(); it.hasNext(); ) {  
 String s = (String) it.next();  
 if (s == null || s.equals("")) {  
 continue;  
 }  
 List<AsmModelDeviceOutput> asmModelDeviceOutputList = new ArrayList<>();  
 AsmModelDeviceOutput output = new AsmModelDeviceOutput();  
 Double outputquality = 0.0;  
 Double outputco2e = 0.0;  
 Double co2eFactor = 0.0;  
 for (OilMaterial oil : oilMaterialList) {  
 if (oil.getMaterialProp2() != null && oil.getMaterialProp2().equals(s)) {  
 for (int i = 0; i < products.size(); i++) {  
 if (products.get(i).getOutputCode() == null) {  
 if (products.get(i).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp1List = products.get(i).getMaterialProp1List();  
 if (MaterialProp1List.size() > 0) {  
 for (AsmModelDeviceOutput out : MaterialProp1List) {  
 if (oil.getFactoryCode().equals(out.getOutputCode()) && oil.getMaterialProp2() != null) {  
 asmModelDeviceOutputList.add(products.get(i));  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 num++;  
 break;  
 }  
 }  
 }  
 output.setOutputName(oil.getMaterialProp2());  
 output.setMaterialProp2List(asmModelDeviceOutputList);  
 }  
 } else {  
 asmModelDeviceOutputList.add(products.get(i));  
 output.setMaterialProp2List(asmModelDeviceOutputList);  
 output.setOutputName(oil.getMaterialProp2());  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / output\_quantity;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 num++;  
 i = 0;  
 }  
  
 }  
 }  
 }  
 products.add(output);  
 }  
 *//三级分类* for (Iterator it = MaterialProp3set.iterator(); it.hasNext(); ) {  
 String s = (String) it.next();  
 if (s == null || s.equals("")) {  
 continue;  
 }  
 List<AsmModelDeviceOutput> asmModelDeviceOutputList = new ArrayList<>();  
 AsmModelDeviceOutput output = new AsmModelDeviceOutput();  
 Double outputquality = 0.0;  
 Double outputco2e = 0.0;  
 Double co2eFactor = 0.0;  
 for (OilMaterial oil : oilMaterialList) {  
 if (oil.getMaterialProp3() != null && oil.getMaterialProp3().equals(s)) {  
 for (int i = 0; i < products.size(); i++) {  
 if (products.get(i).getOutputCode() == null) {  
 *//判断二级分类里的* if (products.get(i).getMaterialProp2List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp2List = products.get(i).getMaterialProp2List();  
 if (MaterialProp2List.size() > 0) {  
 for (AsmModelDeviceOutput out : MaterialProp2List) {  
 *//二级分类里的一级分类* if (out.getOutputCode() == null) {  
 if (out.getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp1List = out.getMaterialProp1List();  
 if (MaterialProp1List.size() > 0) {  
 for (AsmModelDeviceOutput out1 : MaterialProp1List) {  
 if (oil.getFactoryCode().equals(out1.getOutputCode()) && oil.getMaterialProp2() != " ") {  
 asmModelDeviceOutputList.add(products.get(i));  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 i = 0;  
 num++;  
 break;  
 }  
 }  
 }  
 output.setOutputName(oil.getMaterialProp3());  
 output.setMaterialProp3List(asmModelDeviceOutputList);  
 }  
 }  
  
  
 if (oil.getFactoryCode().equals(out.getOutputCode()) && oil.getMaterialProp2() != null) {  
 asmModelDeviceOutputList.add(products.get(i));  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 num++;  
 break;  
 }  
 }  
 }  
 output.setOutputName(oil.getMaterialProp3());  
 output.setMaterialProp3List(asmModelDeviceOutputList);  
 }  
 *//判断一级分类* if (products.get(i).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp1List = products.get(i).getMaterialProp1List();  
 if (MaterialProp1List.size() > 0) {  
 for (AsmModelDeviceOutput out : MaterialProp1List) {  
 if (oil.getFactoryCode().equals(out.getOutputCode()) && oil.getMaterialProp1() != "") {  
 asmModelDeviceOutputList.add(products.get(i));  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 num++;  
 break;  
 }  
 }  
 }  
 output.setOutputName(oil.getMaterialProp3());  
 output.setMaterialProp3List(asmModelDeviceOutputList);  
 }  
 } else {  
 asmModelDeviceOutputList.add(products.get(i));  
 output.setMaterialProp3List(asmModelDeviceOutputList);  
 output.setOutputName(oil.getMaterialProp3());  
 outputquality += products.get(i).getOutputQuantity();  
 outputco2e += products.get(i).getOutputCo2();  
 co2eFactor = outputco2e / outputquality;  
 output.setOutputCo2(outputco2e);  
 output.setOutputQuantity(outputquality);  
 output.setOutputCo2Factor(co2eFactor);  
 products.remove(products.get(i));  
 i = 0;  
 num++;  
 }  
  
 }  
 }  
 }  
 products.add(output);  
 }  
 if (printnum - 1 < start\_row\_index + num + 3) {  
 start\_row\_index = 3;  
 start\_column\_index = start\_column\_index + 5;  
 }  
 start\_row\_index++;  
 {  
 Row head\_row = createRow(sheet, start\_row\_index);  
 for (int i = 0; i < 4; i++) {  
 Cell cell = head\_row.createCell(start\_column\_index + i);  
 cell.setCellStyle(cellStyle\_name);  
 if (i == 0) {  
 cell.setCellValue("二、产品");  
 }  
 }  
 sheet.addMergedRegion(new CellRangeAddress(start\_row\_index, start\_row\_index, start\_column\_index, start\_column\_index + 3));  
 }  
 start\_row\_index++;  
 {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("产品名称");  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("生产量/万吨");  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("碳足迹/万吨");  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("碳强度/(吨/吨）");  
 }  
 start\_row\_index++;  
  
 {  
 for (int i = products.size() - 1; i >= 0; i--) {  
 if (products.get(i).getOutputName() != null && products.get(i).getOutputName().equals("合计")) {  
 setCellvalue(sheet, products, start\_row\_index, start\_column\_index, i, wb);  
 }  
 }  
 *//循环products* for (int a = products.size() - 1; a >= 0; a--) {  
 if (products.get(a).getOutputName().equals("合计")) {  
 continue;  
 }  
 if (products.get(a).getOutputCode() == null) {  
 start\_row\_index++;  
 setCellvalue(sheet, products, start\_row\_index, start\_column\_index, a, wb);  
 *//products里的三级分类* if (products.get(a).getMaterialProp3List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp3List = products.get(a).getMaterialProp3List();  
 *//循环三级分类* for (int a31 = 0; a31 < MaterialProp3List.size(); a31++) {  
 if (MaterialProp3List.get(a31).getOutputCode() == null) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp3List, start\_row\_index, start\_column\_index, a31, wb);  
 *//三级里的二级* if (MaterialProp3List.get(a31).getMaterialProp2List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp32List = MaterialProp3List.get(a31).getMaterialProp2List();  
 *//循环二级分类* for (int a32 = 0; a32 < MaterialProp32List.size(); a32++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp32List, start\_row\_index, start\_column\_index, a32, wb);  
 *//二级里的一级* if (MaterialProp32List.get(a32).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp31List = MaterialProp32List.get(a32).getMaterialProp1List();  
 *//循环一级* for (int a321 = 0; a321 < MaterialProp31List.size(); a321++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp31List, start\_row\_index, start\_column\_index, a321, wb);  
*/\*// products.get(a).getMaterialProp3List().get(a31).getMaterialProp2List().get(a32).getMaterialProp1List().get(a321).  
 MaterialProp31List.remove(a321);  
 a321--;\*/  
// MaterialProp32List.get(a32).getMaterialProp1List().remove(MaterialProp31List.get(a321));  
/\* if (MaterialProp31List.size()==0){  
 MaterialProp32List.remove(a32);  
 a32--;  
 }\*/* }  
 }  
*/\* if (MaterialProp32List.size()==0){  
 MaterialProp3List.remove(a31);  
 a31--;  
 }\*/* }  
*/\* if (MaterialProp3List.size()==0){  
 products.remove(a);  
 a--;  
 }\*/  
 //三级里的一级* } else if (MaterialProp3List.get(a31).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp31List = MaterialProp3List.get(a31).getMaterialProp1List();  
 for (int a311 = 0; a311 < MaterialProp31List.size(); a311++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp31List, start\_row\_index, start\_column\_index, a311, wb);  
*/\* MaterialProp31List.remove(a311);  
 a311--;\*/* }  
 }  
 }  
 }  
 *//产品里的二级分类* } else if (products.get(a).getMaterialProp2List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp1List = products.get(a).getMaterialProp2List();  
 for (int ap1 = 0; ap1 < MaterialProp1List.size(); ap1++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp1List, start\_row\_index, start\_column\_index, ap1, wb);  
*/\* MaterialProp1List.remove(ap1);  
 ap1--;\*/  
 //二级里的一级* if (MaterialProp1List.get(ap1).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp11List = MaterialProp1List.get(ap1).getMaterialProp1List();  
 for (int a11 = 0; a11 < MaterialProp11List.size(); a11++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp11List, start\_row\_index, start\_column\_index, a11, wb);  
*/\* MaterialProp11List.remove(a11);  
 a11--;\*/* }  
 }  
 }  
 *//产品里的一级分类* } else if (products.get(a).getMaterialProp1List() != null) {  
 List<AsmModelDeviceOutput> MaterialProp1List = products.get(a).getMaterialProp1List();  
 for (int ap1 = 0; ap1 < MaterialProp1List.size(); ap1++) {  
 start\_row\_index++;  
 setCellvalue(sheet, MaterialProp1List, start\_row\_index, start\_column\_index, ap1, wb);  
*/\* MaterialProp1List.remove(ap1);  
 ap1--;\*/* }  
 }  
 } else if (products.get(a).getOutputName() != "合计") {  
 start\_row\_index++;  
 setCellvalue(sheet, products, start\_row\_index, start\_column\_index, a, wb);  
 }  
 }  
  
 }  
 sheet.setColumnWidth(start\_column\_index + 0, 20 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 1, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 2, 15 \* 256);  
 sheet.setColumnWidth(start\_column\_index + 3, 20 \* 256);  
  
 }  
  
 Map<String, Integer> indexmap = new HashMap<>();  
  
 private Map<String, Integer> export\_materialOil(HSSFWorkbook wb, Sheet sheet, List<AsmModelDeviceInput> raw\_materials,  
 List<AsmModelDeviceInput> oil\_materials, List<AsmModelDeviceOutput> products, int start\_row\_index, int start\_column\_index,  
 int printnum, List<OilMaterial> oilMaterialList, Double totalQuality, Double oilQuality, Double outoilQuality, Integer isChange) {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
 cellStyle\_name.setVerticalAlignment(VerticalAlignment.*CENTER*);  
  
  
 if (raw\_materials.size() == 0) {  
 indexmap.put("row", 4);  
 indexmap.put("column", start\_column\_index);  
 }  
 Boolean isTitle = false;  
 if (oil\_materials.size() > 0) {  
 isTitle = true;  
 }  
 *//导出外购原料油* if (oil\_materials.size() > 0) {  
 {  
 Row head\_row = createRow(sheet, start\_row\_index);  
 for (int i = 0; i < 4; i++) {  
 Cell cell = head\_row.createCell(start\_column\_index + i);  
 cell.setCellStyle(cellStyle\_name);  
 if (i == 0) {  
 cell.setCellValue("一、外购原（料）油");  
 }  
 }  
 sheet.addMergedRegion(new CellRangeAddress(start\_row\_index, start\_row\_index, start\_column\_index, start\_column\_index + 3));  
 }  
  
 start\_row\_index++;  
 {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("原(料)油名称");  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("购买量/万吨");  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("碳足迹/万吨");  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("碳强度/(吨/吨）");  
 }  
 if (isChange != 1) {  
 for (int i = 0; i < oil\_materials.size(); i++) {  
 if (oil\_materials.get(i).getInputQuantity() != null) {  
 totalQuality += oil\_materials.get(i).getInputQuantity();  
 oilQuality = totalQuality;  
 }  
 }  
 AsmModelDeviceInput input = new AsmModelDeviceInput();  
 input.setInputName("原油小计");  
 input.setInputQuantity(oilQuality);  
 input.setInputCo2(0.0);  
 oil\_materials.add(input);  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputQuantity() != null) {  
 totalQuality += raw\_materials.get(i).getInputQuantity();  
 }  
 }  
 AsmModelDeviceInput input1 = new AsmModelDeviceInput();  
 input1.setInputName("合计");  
 input1.setInputQuantity(totalQuality);  
 input1.setInputCo2(0.0);  
 oil\_materials.add(input1);  
 start\_row\_index++;  
 *//放置合计和原油小计* for (int i = 0; i < oil\_materials.size(); i++) {  
 if (oil\_materials.get(i).getInputName().equals("合计")) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(oil\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(oil\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(oil\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (oil\_materials.get(i).getInputQuantity() != null && (oil\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(oil\_materials.get(i).getInputCo2() / oil\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 oil\_materials.remove(oil\_materials.get(i));  
 start\_row\_index++;  
 }  
 }  
 for (int i = 0; i < oil\_materials.size(); i++) {  
 if (oil\_materials.get(i).getInputName().equals("原油小计")) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(oil\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(oil\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(oil\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (oil\_materials.get(i).getInputQuantity() != null && (oil\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(oil\_materials.get(i).getInputCo2() / oil\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 oil\_materials.remove(oil\_materials.get(i));  
 }  
 }  
 }  
 {  
 for (int i = 0; i <= oil\_materials.size(); i++) {  
 i = 0;  
 start\_row\_index++;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(" " + oil\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(oil\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(oil\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (oil\_materials.get(i).getInputQuantity() != null && (oil\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(oil\_materials.get(i).getInputCo2() / oil\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 oil\_materials.remove(i);  
 indexmap.put("row", start\_row\_index);  
 indexmap.put("column", start\_column\_index);  
 if (printnum - 1 <= start\_row\_index) {  
 indexmap = export\_materialOil(wb, sheet, raw\_materials, oil\_materials, products, 4,  
 start\_column\_index + 5, printnum, oilMaterialList, totalQuality, oilQuality, outoilQuality, 1);  
 }  
 }  
 }  
 }  
  
 *//判断是否换列* if (isChange == 1) {  
 if (isTitle) {  
 *//统计外购原料油小计* for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputQuantity() != null) {  
 outoilQuality += raw\_materials.get(i).getInputQuantity();  
 }  
 }  
 AsmModelDeviceInput input2 = new AsmModelDeviceInput();  
 input2.setInputName("外购原料油小计");  
 input2.setInputQuantity(outoilQuality);  
 input2.setInputCo2(0.0);  
 raw\_materials.add(input2);  
 *//导出外购原料油小计* for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputName().equals("外购原料油小计")) {  
 start\_row\_index++;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(raw\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 raw\_materials.remove(raw\_materials.get(i));  
 }  
 }  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 i = 0;  
 start\_row\_index++;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(" " + raw\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 raw\_materials.remove(i);  
 i--;  
  
 indexmap.put("row", start\_row\_index);  
 indexmap.put("column", start\_column\_index);  
 if (printnum - 1 <= start\_row\_index) {  
 indexmap = export\_materialOil(wb, sheet, raw\_materials, oil\_materials, products, 4,  
 start\_column\_index + 5, printnum, oilMaterialList, totalQuality, oilQuality, 0.0, 1);  
 }  
 }  
 } else {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_4side);  
 cell1.setCellValue("原(料)油名称");  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_4side);  
 cell2.setCellValue("购买量/万吨");  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_4side);  
 cell3.setCellValue("碳足迹/万吨");  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_4side);  
 cell4.setCellValue("碳强度/(吨/吨）");  
  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputName().equals("外购原料油小计")) {  
 start\_row\_index++;  
 Row row2 = createRow(sheet, start\_row\_index);  
 Cell cell12 = row2.createCell(start\_column\_index + 0);  
 cell12.setCellStyle(cellStyle\_name);  
 cell12.setCellValue(raw\_materials.get(i).getInputName());  
 Cell cell21 = row2.createCell(start\_column\_index + 1);  
 cell21.setCellStyle(cellStyle\_point2);  
 cell21.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell32 = row2.createCell(start\_column\_index + 2);  
 cell32.setCellStyle(cellStyle\_point2);  
 cell32.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell42 = row2.createCell(start\_column\_index + 3);  
 cell42.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell42.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell42.setCellValue(0.0);  
 raw\_materials.remove(raw\_materials.get(i));  
 }  
 }  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 i = 0;  
 start\_row\_index++;  
 Row row1 = createRow(sheet, start\_row\_index);  
 Cell cell11 = row1.createCell(start\_column\_index + 0);  
 cell11.setCellStyle(cellStyle\_name);  
 cell11.setCellValue(" " + raw\_materials.get(i).getInputName());  
 Cell cell22 = row1.createCell(start\_column\_index + 1);  
 cell22.setCellStyle(cellStyle\_point2);  
 cell22.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell33 = row1.createCell(start\_column\_index + 2);  
 cell33.setCellStyle(cellStyle\_point2);  
 cell33.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell44 = row1.createCell(start\_column\_index + 3);  
 cell44.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell44.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell44.setCellValue(0.0);  
 raw\_materials.remove(i);  
 i--;  
  
 indexmap.put("row", start\_row\_index);  
 indexmap.put("column", start\_column\_index);  
 if (printnum - 1 <= start\_row\_index) {  
 indexmap = export\_materialOil(wb, sheet, raw\_materials, oil\_materials, products, 4,  
 start\_column\_index + 5, printnum, oilMaterialList, totalQuality, oilQuality, 0.0, 1);  
 }  
 }  
 }  
 } else {  
 if (raw\_materials.size() > 0) {  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputQuantity() != null) {  
 outoilQuality += raw\_materials.get(i).getInputQuantity();  
 }  
 }  
 AsmModelDeviceInput input3 = new AsmModelDeviceInput();  
 input3.setInputName("外购原料油小计");  
 input3.setInputQuantity(outoilQuality);  
 input3.setInputCo2(0.0);  
 raw\_materials.add(input3);  
 }  
 *//导出外购原料油小计* start\_row\_index++;  
 int raw\_total\_row\_index = start\_row\_index;  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 if (raw\_materials.get(i).getInputName().equals("外购原料油小计")) {  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(raw\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 raw\_materials.remove(raw\_materials.get(i));  
 }  
 }  
 for (int i = 0; i < raw\_materials.size(); i++) {  
 i = 0;  
 start\_row\_index++;  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 cell1.setCellValue(" " + raw\_materials.get(i).getInputName());  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(raw\_materials.get(i).getInputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(raw\_materials.get(i).getInputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 if (raw\_materials.get(i).getInputQuantity() != null && (raw\_materials.get(i).getInputQuantity() > 0.0))  
 cell4.setCellValue(raw\_materials.get(i).getInputCo2() / raw\_materials.get(i).getInputQuantity());  
 else  
 cell4.setCellValue(0.0);  
 raw\_materials.remove(i);  
 i--;  
  
 indexmap.put("row", start\_row\_index);  
 indexmap.put("column", start\_column\_index);  
 if (printnum - 1 <= start\_row\_index) {  
 indexmap = export\_materialOil(wb, sheet, raw\_materials, oil\_materials, products, 4,  
 start\_column\_index + 5, printnum, oilMaterialList, totalQuality, oilQuality, 0.0, 1);  
 }  
 }  
 }  
  
 return indexmap;  
 }  
  
  
 private void setCellvalue(Sheet sheet, List<AsmModelDeviceOutput> products, int start\_row\_index, int start\_column\_index, int i, HSSFWorkbook wb) {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
 cellStyle\_name.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 Row row = createRow(sheet, start\_row\_index);  
 Cell cell1 = row.createCell(start\_column\_index + 0);  
 cell1.setCellStyle(cellStyle\_name);  
 if (products.get(i).getOutputCode() == null) {  
 cell1.setCellValue(products.get(i).getOutputName());  
 } else {  
 cell1.setCellValue(" " + products.get(i).getOutputName());  
 }  
 Cell cell2 = row.createCell(start\_column\_index + 1);  
 cell2.setCellStyle(cellStyle\_point2);  
 cell2.setCellValue(products.get(i).getOutputQuantity() / 10000);  
 Cell cell3 = row.createCell(start\_column\_index + 2);  
 cell3.setCellStyle(cellStyle\_point2);  
 cell3.setCellValue(products.get(i).getOutputCo2() / 10000);  
 Cell cell4 = row.createCell(start\_column\_index + 3);  
 cell4.setCellStyle(cellStyle\_point2);  
 cell4.setCellValue(products.get(i).getOutputCo2Factor());  
 }  
  
 private void create\_h\_divid\_formulas(HSSFWorkbook wb, Sheet sheet, int col\_index, int col\_index1, int row\_index, int row\_start, int row\_end) {  
 Row row = createRow(sheet, row\_index);  
 Cell cell = row.getCell(col\_index + 2);  
 String colName = CellReference.*convertNumToColString*(col\_index);  
 String colName1 = CellReference.*convertNumToColString*(col\_index1);  
 StringBuilder formual = new StringBuilder();  
 formual.append("SUM(");  
 formual.append(colName1);  
 formual.append(row\_start + 1);  
 formual.append(":");  
 formual.append(colName1);  
 formual.append(row\_end + 1);  
 formual.append(")/");  
 formual.append("SUM(");  
 formual.append(colName);  
 formual.append(row\_start + 1);  
 formual.append(":");  
 formual.append(colName);  
 formual.append(row\_end + 1);  
 formual.append(")");  
 cell.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell);  
  
 }  
  
 private int export\_device(HSSFWorkbook wb, Sheet sheet, List<AsmModelDeviceInput> raw\_materials,  
 List<AsmModelDeviceInput> oil\_materials, List<AsmModelDeviceOutput> products, int start\_row\_index,  
 int start\_column\_index, AsmModelDevice device, Integer printnum) {  
  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
  
 HSSFCellStyle cellStyle\_point3 = wb.createCellStyle(); *//小数点3位* cellStyle\_point3.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_point3.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.000"));  
 cellStyle\_point3.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point3.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point3.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point3.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_device = wb.createCellStyle();  
 cellStyle\_device.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_device.setWrapText(true);  
 cellStyle\_device.setRotation((short) 255);  
 cellStyle\_device.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_device.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_device.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_device.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_device.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_device.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_device.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_device.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
*// cellStyle\_device.setFillBackgroundColor(HSSFColor.HSSFColorPredefined.GREEN.getIndex());* cellStyle\_device.setFillForegroundColor(HSSFColor.HSSFColorPredefined.*LIGHT\_GREEN*.getIndex());  
 cellStyle\_device.setFillPattern(FillPatternType.*SOLID\_FOREGROUND*);  
  
  
 HSSFCellStyle cellStyle\_line = wb.createCellStyle();  
 cellStyle\_line.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_line.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_line.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_line\_red = wb.createCellStyle();  
 cellStyle\_line\_red.cloneStyleFrom(cellStyle\_line);  
 Font font\_red = wb.createFont();  
 font\_red.setFontName("宋体");  
 font\_red.setColor(Font.*COLOR\_RED*);  
 *//font.setBold(true);* font\_red.setFontHeightInPoints((short) 12);  
 cellStyle\_line\_red.setFont(font\_red);  
  
 HSSFCellStyle cellStyle\_point2\_red = wb.createCellStyle(); *//小数点2位* cellStyle\_point2\_red.cloneStyleFrom(cellStyle\_point2);  
 cellStyle\_point2\_red.setFont(font\_red);  
  
 create\_title(wb, sheet, start\_row\_index, start\_column\_index);  
 start\_row\_index++;  
 int rows = device.getInputs() == null ? 0 : (device.getInputs().size() + 2);  
  
 if (device.getOutputs() != null && rows < (device.getOutputs().size() + 2)) {  
 rows = device.getOutputs().size() + 2;  
 }  
 if (rows < (device.getDeviceName().length() + 2)) {  
 rows = device.getDeviceName().length() + 2;  
 }  
  
 {  
 for (int i = 0; i < rows; i++) {  
 Row row = createRow(sheet, start\_row\_index + i);  
 Cell cell = row.createCell(start\_column\_index + 4);  
 if (i == 0) {  
 cell.setCellValue(device.getDeviceName());  
 } else {  
 cell.setCellValue("");  
 }  
 cell.setCellStyle(cellStyle\_device);  
 }  
 sheet.addMergedRegion(new CellRangeAddress(start\_row\_index, start\_row\_index + rows - 2, start\_column\_index + 4, start\_column\_index + 4));  
 }  
 {  
 Row row\_total = createRow(sheet, start\_row\_index);  
 if (device.getOutputs() != null) {  
 for (int i = 0; i < device.getOutputs().size(); i++) {  
 AsmModelDeviceOutput output = device.getOutputs().get(i);  
 if (StringUtils.*isNotEmpty*(output.getMaterialProperty())) {  
 if (output.getMaterialProperty().equals(configService.getKey("co2e.product.property"))) {  
 products.add(output);  
 }  
 } else if (output.getOutputCode().equals("LOS") || output.getOutputCode() == "LOS") {  
 products.add(output);  
 }  
 Row row = createRow(sheet, start\_row\_index + i + 1);  
 Cell cell\_name = row.createCell(start\_column\_index + 5);  
 cell\_name.setCellValue(output.getOutputName());  
 cell\_name.setCellStyle(cellStyle\_line);  
  
 Cell cell\_yield = row.createCell(start\_column\_index + 6);  
 cell\_yield.setCellValue(output.getOutputQuantity() / device.getDeviceProcess() \* 100.0);  
 cell\_yield.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_process = row.createCell(start\_column\_index + 7);  
 cell\_process.setCellValue(output.getOutputQuantity() / 10000.0);  
 cell\_process.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_co2 = row.createCell(start\_column\_index + 8);  
 cell\_co2.setCellValue(output.getOutputCo2() / 10000.0);  
 cell\_co2.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_factor = row.createCell(start\_column\_index + 9);  
 cell\_factor.setCellValue(output.getOutputCo2Factor());  
 cell\_factor.setCellStyle(cellStyle\_point2);  
 }  
 }  
  
 Cell cell\_total\_name = row\_total.createCell(start\_column\_index + 5);  
 cell\_total\_name.setCellValue("合计");  
 cell\_total\_name.setCellStyle(cellStyle\_line);  
  
 Cell cell\_total\_yield = row\_total.createCell(start\_column\_index + 6);  
 cell\_total\_yield.setCellValue(100.0);  
 cell\_total\_yield.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_total\_process = row\_total.createCell(start\_column\_index + 7);  
 cell\_total\_process.setCellValue(device.getDeviceProcess() / 10000.00);  
 cell\_total\_process.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_total\_co2 = row\_total.createCell(start\_column\_index + 8);  
 String colName = CellReference.*convertNumToColString*(start\_column\_index + 8);  
 if (device.getOutputs() == null || device.getOutputs().size() == 0) {  
 cell\_total\_co2.setCellValue(0);  
 } else {  
 StringBuilder formual = new StringBuilder();  
 formual.append("SUM(");  
 formual.append(colName);  
 formual.append(start\_row\_index + 2);  
 formual.append(":");  
 formual.append(colName);  
 formual.append(start\_row\_index + 1 + device.getOutputs().size());  
 formual.append(")");  
 cell\_total\_co2.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell\_total\_co2);  
 }  
  
 cell\_total\_co2.setCellStyle(cellStyle\_point2);  
  
  
 Cell cell\_total\_factor = row\_total.createCell(start\_column\_index + 9);  
 cell\_total\_factor.setCellValue("");  
 cell\_total\_factor.setCellStyle(cellStyle\_line);  
 }  
 {  
 Row row\_total = createRow(sheet, start\_row\_index);  
  
 for (int i = 0; i < device.getInputs().size(); i++) {  
 AsmModelDeviceInput input = device.getInputs().get(i);  
  
 if (StringUtils.*isNotEmpty*(input.getMaterialProperty())) {  
 if (input.getMaterialProperty().equals(configService.getKey("co2e.input.rawmaterial"))) {  
 raw\_materials.add(input);  
 } else if (input.getMaterialProperty().equals(configService.getKey("co2e.input.oilmaterial"))) {  
 oil\_materials.add(input);  
 }  
 }  
  
 Row row = createRow(sheet, start\_row\_index + i + 1);  
 Cell cell\_name = row.createCell(start\_column\_index + 0);  
 cell\_name.setCellValue(input.getInputName());  
 cell\_name.setCellStyle(cellStyle\_line);  
  
  
 Cell cell\_process = row.createCell(start\_column\_index + 1);  
 cell\_process.setCellValue(input.getInputQuantity() / 10000.0);  
 cell\_process.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_co2 = row.createCell(start\_column\_index + 2);  
 cell\_co2.setCellValue(input.getInputCo2() / 10000.0);  
 cell\_co2.setCellStyle(cellStyle\_point2);  
  
 Cell cell\_factor = row.createCell(start\_column\_index + 3);  
 cell\_factor.setCellValue(input.getInputCo2() / input.getInputQuantity());  
 cell\_factor.setCellStyle(cellStyle\_point2);  
  
 }  
 {  
 Row row = createRow(sheet, start\_row\_index + device.getInputs().size() + 1);  
 Cell cell\_name = row.createCell(start\_column\_index + 0);  
 cell\_name.setCellValue("装置");  
 cell\_name.setCellStyle(cellStyle\_line\_red);  
  
  
 Cell cell\_process = row.createCell(start\_column\_index + 1);  
 cell\_process.setCellValue(device.getDeviceProcess() / 10000.0);  
 cell\_process.setCellStyle(cellStyle\_point2\_red);  
  
 Cell cell\_co2 = row.createCell(start\_column\_index + 2);  
 cell\_co2.setCellValue(device.getDeviceCo2() / 10000.0);  
 cell\_co2.setCellStyle(cellStyle\_point2\_red);  
  
 Cell cell\_factor = row.createCell(start\_column\_index + 3);  
 cell\_factor.setCellValue(device.getDeviceFactor());  
 cell\_factor.setCellStyle(cellStyle\_point2\_red);  
 }  
  
  
 Cell cell\_total\_name = row\_total.createCell(start\_column\_index + 0);  
 cell\_total\_name.setCellValue("合计");  
 cell\_total\_name.setCellStyle(cellStyle\_line);  
  
 {  
 Cell cell\_total\_process = row\_total.createCell(start\_column\_index + 1);  
 String colName = CellReference.*convertNumToColString*(start\_column\_index + 1);  
 StringBuilder formual = new StringBuilder();  
 formual.append("SUM(");  
 formual.append(colName);  
 formual.append(start\_row\_index + 2);  
 formual.append(":");  
 formual.append(colName);  
 formual.append(start\_row\_index + 1 + device.getInputs().size());  
 formual.append(")");  
 cell\_total\_process.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell\_total\_process);  
 cell\_total\_process.setCellStyle(cellStyle\_point2);  
 }  
  
 {  
 Cell cell\_total\_co2 = row\_total.createCell(start\_column\_index + 2);  
 String colName = CellReference.*convertNumToColString*(start\_column\_index + 2);  
 if (device.getInputs() == null || device.getInputs().size() == 0) {  
 cell\_total\_co2.setCellValue(0);  
 } else {  
 StringBuilder formual = new StringBuilder();  
 formual.append("SUM(");  
 formual.append(colName);  
 formual.append(start\_row\_index + 2);  
 formual.append(":");  
 formual.append(colName);  
 formual.append(start\_row\_index + 1 + device.getInputs().size());  
 formual.append(")");  
 cell\_total\_co2.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell\_total\_co2);  
 }  
  
 cell\_total\_co2.setCellStyle(cellStyle\_point2);  
 }  
  
 Cell cell\_total\_factor = row\_total.createCell(start\_column\_index + 3);  
 cell\_total\_factor.setCellValue("");  
 cell\_total\_factor.setCellStyle(cellStyle\_line);  
  
 }  
  
 start\_row\_index += rows + 2;  
 return start\_row\_index;  
 }  
  
 private void export\_osbl(HSSFWorkbook wb, List<AsmModelDevice> devices) {  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 *//font.setBold(true);* font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 HSSFCellStyle cellStyle\_4side = wb.createCellStyle();  
 cellStyle\_4side.cloneStyleFrom(cellStyle\_default);  
 cellStyle\_4side.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_4side.setBorderTop(BorderStyle.*THIN*);  
 cellStyle\_4side.setTopBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_4side.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderLeft(BorderStyle.*THIN*);  
 cellStyle\_4side.setLeftBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
 cellStyle\_4side.setBorderRight(BorderStyle.*THIN*);  
 cellStyle\_4side.setRightBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_point2 = wb.createCellStyle(); *//小数点2位* cellStyle\_point2.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_point2.setDataFormat(HSSFDataFormat.*getBuiltinFormat*("0.00"));  
 cellStyle\_point2.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_point2.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_point2.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle\_point2.setBottomBorderColor(HSSFColor.HSSFColorPredefined.*BLACK*.getIndex());  
  
 HSSFCellStyle cellStyle\_name = wb.createCellStyle(); *//小数点2位* cellStyle\_name.cloneStyleFrom(cellStyle\_4side);  
 cellStyle\_name.setAlignment(HorizontalAlignment.*LEFT*);  
  
 int start\_row\_index = 0;  
 int start\_col\_index = 0;  
  
 List<SysParent> dicts = sysParentService.selectchilerenList(new SysParent());  
  
 Sheet sheet = wb.createSheet("公用工程");  
 {  
 {  
 Row row = createRow(sheet, start\_row\_index + 0);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("装置名称");  
 for (int i = 0; i < dicts.size(); i++) {  
 Cell cell = row.createCell(start\_col\_index + 1 + i);  
 cell.setCellStyle(cellStyle\_4side);  
 cell.setCellValue(dicts.get(i).getDictName());  
 }  
 }  
 {  
 Row row = createRow(sheet, start\_row\_index + 1);  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_4side);  
 cellTitle.setCellValue("");  
 for (int i = 0; i < dicts.size(); i++) {  
 Cell cell = row.createCell(start\_col\_index + 1 + i);  
 cell.setCellStyle(cellStyle\_4side);  
 if (dicts.get(i).getDictCode().equals("LLH")) {  
 cell.setCellValue("MJ");  
 } else if (dicts.get(i).getDictCode().equals("KEO")) {  
 cell.setCellValue("toe");  
 } else if (dicts.get(i).getDictCode().equals("CAT") || dicts.get(i).getDictCode().equals("CCC")) {  
 cell.setCellValue("万元");  
 } else {  
 cell.setCellValue("万吨");  
 }  
 sheet.setColumnWidth(start\_col\_index + 1 + i, 10 \* 256);  
 }  
 sheet.addMergedRegion(new CellRangeAddress(start\_row\_index + 0, start\_row\_index + 1, start\_col\_index, start\_col\_index));  
 }  
 }  
 start\_row\_index += 2;  
 String ugsOsblCode = configService.getKey("co2e.osbl.ugs");  
 for (AsmModelDevice device : devices) {  
 if (device.getIsVirtual() == 1)  
 continue;  
 Row row = createRow(sheet, start\_row\_index);  
  
 Cell cellTitle = row.createCell(start\_col\_index + 0);  
 cellTitle.setCellStyle(cellStyle\_name);  
 cellTitle.setCellValue(device.getDeviceName());  
 for (int i = 0; i < dicts.size(); i++) {  
 Cell cell = row.createCell(i + 1);  
 cell.setCellStyle(cellStyle\_point2);  
 }  
 double keo = 0.0;  
 double ecd = 0.0;  
 double ecu = 0.0;  
 double ect = 0.0;  
 double ecs = 0.0;  
 double eck = 0.0;  
 double ecr = 0.0;  
 double ecz = 0.0;  
 double ecj = 0.0;  
 Map<String, AsmModelDeviceOsbl> osblMap = new HashMap<>();  
 for (AsmModelDeviceOsbl osbl : device.getOsbls()) {  
 osblMap.put(osbl.getOsblCode(), osbl);  
 keo += (osbl.getConsumption() == null ? 0 : osbl.getConsumption()) \* (osbl.getFoldCoefficient() == null ? 0 : osbl.getFoldCoefficient());  
 ecd += osbl.getCarbonEmission();  
 int col = getDictCol(osbl.getOsblCode(), dicts);  
 if (col == -1)  
 continue;  
 if ((Double.*doubleToLongBits*(osbl.getConsumption()) == Double.*doubleToLongBits*(0.0))) {  
 continue;  
 }  
 Cell cell = row.getCell(col + 1);  
  
 cell.setCellValue(osbl.getCarbonEmission());  
 *// cell.setCellValue(osbl.getOsblCo2());* cell.setCellValue(osbl.getConsumption() / 10000.0);  
 }  
 ecu = (osblMap.get("ECC") == null ? 0.0 : osblMap.get("ECC").getCarbonEmission())  
 + (osblMap.get("CCK") == null ? 0.0 : osblMap.get("CCK").getCarbonEmission())  
 + (osblMap.get("CKK") == null ? 0.0 : osblMap.get("CKK").getCarbonEmission())  
 + (osblMap.get("EOO") == null ? 0.0 : osblMap.get("EOO").getCarbonEmission())  
 + (osblMap.get("UFL") == null ? 0.0 : osblMap.get("UFL").getCarbonEmission())  
 + (osblMap.get("GAS") == null ? 0.0 : osblMap.get("GAS").getCarbonEmission())  
 + (osblMap.get("DIO") == null ? 0.0 : osblMap.get("DIO").getCarbonEmission())  
 + (osblMap.get("LPS") == null ? 0.0 : osblMap.get("LPS").getCarbonEmission())  
 + (osblMap.get("LTG") == null ? 0.0 : osblMap.get("LTG").getCarbonEmission())  
 + (osblMap.get("UGS") == null ? 0.0 : osblMap.get("UGS").getCarbonEmission())  
 + (osblMap.get("GAG") == null ? 0.0 : osblMap.get("GAG").getCarbonEmission())  
 + (osblMap.get("NAG") == null ? 0.0 : osblMap.get("NAG").getCarbonEmission());  
  
 ect = osblMap.get("CKE") == null ? 0.0 : osblMap.get("CKE").getCarbonEmission();  
  
 ecs = (osblMap.get("10M") == null ? 0.0 : osblMap.get("10M").getCarbonEmission())  
 + (osblMap.get("05M") == null ? 0.0 : osblMap.get("05M").getCarbonEmission())  
 + (osblMap.get("3D5") == null ? 0.0 : osblMap.get("3D5").getCarbonEmission())  
 + (osblMap.get("2D5") == null ? 0.0 : osblMap.get("2D5").getCarbonEmission())  
 + (osblMap.get("1D5") == null ? 0.0 : osblMap.get("1D5").getCarbonEmission())  
 + (osblMap.get("1DM") == null ? 0.0 : osblMap.get("1DM").getCarbonEmission())  
 + (osblMap.get("D7M") == null ? 0.0 : osblMap.get("D7M").getCarbonEmission())  
 + (osblMap.get("D5M") == null ? 0.0 : osblMap.get("D5M").getCarbonEmission())  
 + (osblMap.get("LD3") == null ? 0.0 : osblMap.get("LD3").getCarbonEmission());  
  
 eck = (osblMap.get("KWS") == null ? 0.0 : osblMap.get("KWS").getCarbonEmission())  
 + (osblMap.get("KWH") == null ? 0.0 : osblMap.get("KWH").getCarbonEmission());  
  
 ecr = (osblMap.get("WAT") == null ? 0.0 : osblMap.get("WAT").getCarbonEmission())  
 + (osblMap.get("CWT") == null ? 0.0 : osblMap.get("CWT").getCarbonEmission())  
 + (osblMap.get("SWT") == null ? 0.0 : osblMap.get("SWT").getCarbonEmission())  
 + (osblMap.get("DAW") == null ? 0.0 : osblMap.get("DAW").getCarbonEmission())  
 + (osblMap.get("DOW") == null ? 0.0 : osblMap.get("DOW").getCarbonEmission())  
 + (osblMap.get("COT") == null ? 0.0 : osblMap.get("COT").getCarbonEmission())  
 + (osblMap.get("COH") == null ? 0.0 : osblMap.get("COH").getCarbonEmission())  
 + (osblMap.get("NNS") == null ? 0.0 : osblMap.get("NNS").getCarbonEmission());  
  
 ecz = ecu + ect;  
 ecj = ecr + eck + ecs;  
  
 int col = getDictCol("KEO", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(keo / 10000.0);  
 }  
  
 col = getDictCol("ECD", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecd / 10000.0);  
 }  
 col = getDictCol("ECU", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecu / 10000.0);  
 }  
 col = getDictCol("ECT", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ect / 10000.0);  
 }  
  
 col = getDictCol("ECS", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecs / 10000.0);  
 }  
 col = getDictCol("ECK", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(eck / 10000.0);  
 }  
 col = getDictCol("ECR", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecr / 10000.0);  
 }  
 col = getDictCol("ECZ", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecz / 10000.0);  
 }  
 col = getDictCol("ECJ", dicts);  
 if (col >= 0) {  
 Cell cell = row.getCell(col + 1);  
 cell.setCellValue(ecj / 10000.0);  
 }  
  
 start\_row\_index++;  
 }  
 sheet.setColumnWidth(start\_col\_index, 30 \* 256);  
 }  
  
 private void create\_v\_sum\_formula(HSSFWorkbook wb, Sheet sheet, int col\_index, int row\_index, int row\_start, int row\_end) {  
 Row row = createRow(sheet, row\_index);  
 Cell cell = row.getCell(col\_index);  
 String colName = CellReference.*convertNumToColString*(col\_index);  
 StringBuilder formual = new StringBuilder();  
 formual.append("SUM(");  
 formual.append(colName);  
 formual.append(row\_start + 1);  
 formual.append(":");  
 formual.append(colName);  
 formual.append(row\_end + 1);  
 formual.append(")");  
 cell.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell);  
 }  
  
 private void create\_v\_add\_formula(HSSFWorkbook wb, Sheet sheet, int col\_index, int row\_index, int row\_index1, int row\_index\_2) {  
 Row row = createRow(sheet, row\_index);  
 Cell cell = row.getCell(col\_index);  
 String colName = CellReference.*convertNumToColString*(col\_index);  
  
 StringBuilder formual = new StringBuilder();  
  
 formual.append(colName);  
 formual.append(row\_index1 + 1);  
 formual.append("+");  
 formual.append(colName);  
 formual.append(row\_index\_2 + 1);  
 cell.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell);  
 }  
  
 private void create\_h\_divid\_formula(HSSFWorkbook wb, Sheet sheet, int col\_index, int noun\_col\_index, int deno\_col\_index, int row\_index) {  
 Row row = createRow(sheet, row\_index);  
 Cell cell = row.getCell(col\_index);  
 String nounColName = CellReference.*convertNumToColString*(noun\_col\_index);  
*// Double a=Double.parseDouble(nounColName);* String denoColName = CellReference.*convertNumToColString*(deno\_col\_index);  
 StringBuilder formual = new StringBuilder();  
  
 formual.append(nounColName);  
 formual.append(row\_index + 1);  
 formual.append("/");  
 formual.append(denoColName);  
 formual.append(row\_index + 1);  
 cell.setCellFormula(formual.toString());  
 HSSFFormulaEvaluator formulaEvaluator = wb.getCreationHelper().createFormulaEvaluator();  
 formulaEvaluator.evaluateFormulaCell(cell);  
 }  
  
 private void create\_title(HSSFWorkbook wb, Sheet sheet, int start\_row\_index, int start\_column\_index) {  
  
 Font font = wb.createFont();  
 font.setFontName("宋体");  
 font.setBold(true);  
 font.setFontHeightInPoints((short) 12);  
  
 HSSFCellStyle cellStyle\_default = wb.createCellStyle();  
 cellStyle\_default.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle\_default.setAlignment(HorizontalAlignment.*CENTER*);  
 cellStyle\_default.setFont(font);  
  
 Row title\_row = sheet.getRow(start\_row\_index);  
 if (title\_row == null)  
 title\_row = sheet.createRow(start\_row\_index);  
 Cell cell1 = title\_row.createCell(start\_column\_index + 0);  
 cell1.setCellValue("进料");  
 cell1.setCellStyle(cellStyle\_default);  
  
 autoColumnSize(start\_column\_index + 0, "进料", sheet);  
 Cell cell2 = title\_row.createCell(start\_column\_index + 1);  
 cell2.setCellValue("数量/万吨");  
 cell2.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 1, "数量/万吨", sheet);  
 Cell cell3 = title\_row.createCell(start\_column\_index + 2);  
 cell3.setCellValue("碳足迹/万吨");  
 cell3.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 2, "碳足迹/万吨", sheet);  
 Cell cell4 = title\_row.createCell(start\_column\_index + 3);  
 cell4.setCellValue("碳强度/(吨/吨）");  
 cell4.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 3, "碳强度/(吨/吨）", sheet);  
 Cell cell5 = title\_row.createCell(start\_column\_index + 5);  
 cell5.setCellValue("产品");  
 cell5.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 5, "产品", sheet);  
 Cell cell6 = title\_row.createCell(start\_column\_index + 6);  
 cell6.setCellValue("收率/%");  
 cell6.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 6, "收率/%", sheet);  
 Cell cell7 = title\_row.createCell(start\_column\_index + 7);  
 cell7.setCellValue("产量/万吨");  
 cell7.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 7, "产量/万吨", sheet);  
 Cell cell8 = title\_row.createCell(start\_column\_index + 8);  
 cell8.setCellValue("碳足迹/万吨");  
 cell8.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 8, "碳足迹/万吨", sheet);  
 Cell cell9 = title\_row.createCell(start\_column\_index + 9);  
 cell9.setCellValue("碳足迹/万吨");  
 cell9.setCellValue("碳强度/(吨/吨）");  
 cell9.setCellStyle(cellStyle\_default);  
 autoColumnSize(start\_column\_index + 9, "碳强度/(吨/吨）", sheet);  
 }  
  
 private Row createRow(Sheet sheet, int index) {  
 Row row = sheet.getRow(index);  
 if (row == null)  
 row = sheet.createRow(index);  
 return row;  
 }  
  
 private void autoColumnSize(int columnIndex, String value, Sheet sheet) {  
*// int columnwidth = sheet.getColumnWidth(columnIndex) / 256;  
// int length = value.getBytes().length;  
// if(columnwidth < length)  
// columnwidth = length;  
// sheet.setColumnWidth(columnIndex, columnwidth \* 17 / 10 \* 256);* sheet.autoSizeColumn(columnIndex);  
 sheet.setColumnWidth(columnIndex, sheet.getColumnWidth(columnIndex) \* 12 / 10);  
 }  
  
 private Map<Long, AsmModelDevice> bfs(AsmModel model, List<AsmModelDevice> list) {  
 Map<Long, AsmModelDevice> deviceMap = new HashMap<>();  
 Map<Long, AsmModelDeviceInput> inputMap = new HashMap<>();  
 Map<Long, AsmModelDeviceOutput> outputMap = new HashMap<>();  
 List<AsmModelDevice> firstDevices = new ArrayList<>();  
  
 for (AsmModelDevice device : model.getDevices()) {  
 deviceMap.put(device.getDeviceId(), device);  
 for (AsmModelDeviceInput input : device.getInputs()) {  
 inputMap.put(input.getInputId(), input);  
 }  
 for (AsmModelDeviceOutput output : device.getOutputs()) {  
 outputMap.put(output.getOutputId(), output);  
 }  
 }  
  
 for (Map.Entry<Long, AsmModelDeviceInput> entry : inputMap.entrySet()) {  
 AsmModelDeviceOutput output = outputMap.get(entry.getValue().getSuperOutputId());  
 if (output != null) {  
 if (output.getInputs() == null) {  
 output.setInputs(new ArrayList<>());  
 }  
 output.getInputs().add(entry.getValue());  
 }  
 }  
  
  
 int lastLevel = 1;  
 List<Long> idList = new ArrayList<>();  
 if (firstDevices.size() == 0) {  
 for (AsmModelDevice device : model.getDevices()) {  
 boolean isFirst = true;  
  
 for (AsmModelDeviceInput input : device.getInputs()) {  
 if (input.getSuperOutputId() != null)  
 isFirst = false;  
 }  
 if (isFirst) {  
 device.setLevel(1);  
 firstDevices.add(device);  
 }  
 }  
  
 *log*.debug("first devices size = {}", firstDevices.size());  
 if (firstDevices.size() == 0) {  
  
 AsmModelDevice device = model.getDevices().get(0);  
 device.setLevel(1);  
 firstDevices.add(device);  
 }  
  
  
 }  
  
 list.addAll(firstDevices);  
 for (AsmModelDevice device : firstDevices) {  
 idList.add(device.getDeviceId());  
 }  
  
 while (true) {  
 AsmModelDevice item = null;  
 for (AsmModelDevice device : list) {  
 if (device.isVisted() == false) {  
 item = device;  
 break;  
 }  
 }  
 if (item == null) {  
 break;  
 }  
 item.setVisted(true);  
  
 for (AsmModelDeviceOutput output : item.getOutputs()) {  
 if (output.getInputs() != null) {  
 for (AsmModelDeviceInput input : output.getInputs()) {  
 AsmModelDevice device = deviceMap.get(input.getDeviceId());  
 if (device != null && (!idList.contains(device.getDeviceId()))) {  
 *// device.setVisted(false);* device.setLevel(item.getLevel() + 1);  
 list.add(device);  
 idList.add(device.getDeviceId());  
 if(lastLevel < device.getLevel()){  
 lastLevel = device.getLevel();  
 }  
 }  
 }  
 }  
 }  
 }  
  
 final int level = lastLevel;  
 list.forEach( (value) -> {  
 boolean isLast = true;  
 for (AsmModelDeviceOutput output : value.getOutputs()){  
 if(output.getInputs() != null && output.getInputs().size() > 0)  
 isLast = false;  
 }  
 if(isLast)  
 value.setLevel(level + 1);  
 });  
  
  
 return deviceMap;  
 }  
  
 private Map<Long, AsmModelDevice> Picturebfs(AsmModel model, List<AsmModelDevice> list) {  
 Map<Long, AsmModelDevice> deviceMap = new HashMap<>();  
 Map<Long, AsmModelDeviceInput> inputMap = new HashMap<>();  
 Map<Long, AsmModelDeviceOutput> outputMap = new HashMap<>();  
 List<AsmModelDevice> firstDevices = new ArrayList<>();  
 for (AsmModelDevice device : model.getDevices()) {  
 deviceMap.put(device.getDeviceId(), device);  
*// if ((device.getDeviceName().startsWith("常减压")) || (device.getDeviceName().startsWith("烷基化"))) {  
// device.setLevel(1);  
// firstDevices.add(device);  
// }* for (AsmModelDeviceInput input : device.getInputs()) {  
 inputMap.put(input.getInputId(), input);  
 }  
 for (AsmModelDeviceOutput output : device.getOutputs()) {  
 outputMap.put(output.getOutputId(), output);  
 }  
 }  
  
 for (Map.Entry<Long, AsmModelDeviceInput> entry : inputMap.entrySet()) {  
 AsmModelDeviceOutput output = outputMap.get(entry.getValue().getSuperOutputId());  
 if (output != null) {  
 if (output.getInputs() == null) {  
 output.setInputs(new ArrayList<>());  
 }  
 output.getInputs().add(entry.getValue());  
 }  
 }  
  
  
 List<Long> idList = new ArrayList<>();  
 if (firstDevices.size() == 0) {  
 for (AsmModelDevice device : model.getDevices()) {  
 boolean isFirst = true;  
 for (AsmModelDeviceInput input : device.getInputs()) {  
 if (input.getSuperOutputId() != null)  
 isFirst = false;  
 }  
 if (isFirst) {  
 device.setLevel(1);  
 firstDevices.add(device);  
 }  
  
 }  
 if (firstDevices.size() == 0) {  
  
 AsmModelDevice device = model.getDevices().get(0);  
 device.setLevel(1);  
 firstDevices.add(device);  
 }  
  
  
 }  
  
 list.addAll(firstDevices);  
 for (AsmModelDevice device : firstDevices) {  
 idList.add(device.getDeviceId());  
 }  
  
 while (true) {  
 AsmModelDevice item = null;  
 for (AsmModelDevice device : list) {  
 if (device.isVisted() == false) {  
 item = device;  
 break;  
 }  
 }  
 if (item == null) {  
 break;  
 }  
 item.setVisted(true);  
  
 for (AsmModelDeviceOutput output : item.getOutputs()) {  
 if (output.getInputs() != null) {  
 for (AsmModelDeviceInput input : output.getInputs()) {  
 AsmModelDevice device = deviceMap.get(input.getDeviceId());  
 if (device != null && (!idList.contains(device.getDeviceId()))) {  
 *// device.setVisted(false);* device.setLevel(item.getLevel() + 1);  
 list.add(device);  
 idList.add(device.getDeviceId());  
 }  
 }  
 }  
 }  
 }  
 *//循环将产出物料对应的装置的level+1* for (int a = 2; a < 10; a++) {  
 for (int b = 0; b < list.size(); b++) {  
 if (list.get(b).isVisted() == false) {  
 break;  
 } else if (list.get(b).getLevel() == a) {  
 for (AsmModelDeviceOutput output : list.get(b).getOutputs()) {  
 for (int i = 0; i < list.size(); i++) {  
 for (AsmModelDeviceInput input : list.get(i).getInputs()) {  
 if (input.getSuperOutputId() != null && input.getSuperOutputId().equals(output.getOutputId())) {  
 AsmModelDevice device = deviceMap.get(input.getDeviceId());  
 if (device.getLevel() <= list.get(b).getLevel()) {  
 list.remove(device);  
 device.setLevel(list.get(b).getLevel() + 1);  
 list.add(device);  
 i--;  
 b--;  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 return deviceMap;  
 }  
  
 private void importExcel(AsmModel asmModel) {  
 String filepath = RuoYiConfig.*getProfile*() + File.*separator* + asmModel.getFileUrl().substring(Constants.*RESOURCE\_PREFIX*.length());  
 Workbook workbook = null;  
 try {  
 if (asmModel.getFileUrl().contains("xlsx")) {  
 workbook = new XSSFWorkbook(new FileInputStream(new File(filepath)));  
 } else {  
 workbook = new HSSFWorkbook(new FileInputStream(new File(filepath)));  
 }  
 Map<String, AsmModelDevice> deviceMap = process\_osbl(asmModel, workbook);  
 Map<String, AsmModelMaterialBalance> balanceMap = process\_balance(asmModel, workbook);  
 process\_device(asmModel, deviceMap, balanceMap, workbook);  
 process\_blen(asmModel, workbook);  
 } catch (FileNotFoundException e) {  
 *log*.error("导入文件失败", e.getCause());  
 throw new ServiceException("导入文件失败，文件不存在");  
 } catch (IOException e) {  
 *log*.error("导入文件失败", e.getCause());  
 throw new ServiceException("导入文件失败，文件读取失败");  
 }  
 *//去重查询油品调和表并设置装置为调和装置  
// int a=0;  
// AsmModelBlend asmModelBlend=new AsmModelBlend();  
// asmModelBlend.setModelId(asmModel.getModelId());  
// asmModelBlend.setInOutType("产出");  
// List<AsmModelBlend> asmModelBlendList=asmModelBlendService.selectAsmModelBlendListDeweightByModelId(asmModelBlend);  
// for (AsmModelBlend blend:asmModelBlendList) {  
// AsmModelDeviceOutput asmModelDeviceOutput=new AsmModelDeviceOutput();  
// asmModelDeviceOutput.setModelId(asmModel.getModelId());  
// asmModelDeviceOutput.setOutputCode(blend.getMaterielCode());  
// asmModelDeviceOutput.setOutputName(blend.getMaterielName());  
// List<AsmModelDeviceOutput> asmModelDeviceOutputList=asmModelDeviceOutputService.selectAsmModelDeviceOutputList(asmModelDeviceOutput);  
// for (AsmModelDeviceOutput out:asmModelDeviceOutputList) {  
// AsmModelDevice asmModelDevice=asmModelDeviceService.selectAsmModelDeviceByDeviceId(out.getDeviceId());  
// asmModelDevice.setIsVirtual(2);  
// asmModelDeviceService.updateAsmModelDevice(asmModelDevice);  
// a++;  
// }  
// }* }  
  
 private void process\_device(AsmModel asmModel, Map<String, AsmModelDevice> deviceMap, Map<String, AsmModelMaterialBalance> balanceMap, Workbook workbook) {  
 {  
 Sheet sheet = workbook.getSheetAt(0);  
 if (sheet == null) {  
 throw new ServiceException("处理装置表失败，不存在装置表");  
 }  
  
 List<AsmModelDeviceInput> inputs = new ArrayList<>();  
 Map<String, AsmModelDeviceOutput> outputs = new HashMap<>();  
 for (int i = 1; i <= sheet.getLastRowNum(); i++) {  
 Row row = sheet.getRow(i);  
 Cell codeCell = row.getCell(0);  
 if (codeCell == null) {  
 *log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 String code = codeCell.getStringCellValue();  
 if (code == null || code.length() == 0) {  
 *//throw new ServiceException("处理装置表失败，无法读取装置编码。行号： " + i);  
 log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 AsmModelDevice device = deviceMap.get(code);  
 if (device == null) {  
 device = new AsmModelDevice();  
 device.setModelId(asmModel.getModelId());  
 device.setModelName(asmModel.getModelName());  
 device.setDeviceCode(code);  
 Cell nameCell = row.getCell(1);  
 if (nameCell == null) {  
 throw new ServiceException("处理装置表失败，无法读取装置名称。行号： " + i);  
 }  
 String deviceName = nameCell.getStringCellValue();  
 if (deviceName == null) {  
 throw new ServiceException("处理装置表失败，无法读取装置名称。行号： " + i);  
 }  
 device.setDeviceName(deviceName);  
 device.setIsVirtual(1);  
 device.setDeviceProcess(0.0);  
 device.setDeviceCo2(0.0);  
 device.setDeviceFactor(0.0);  
 if (0 == asmModelDeviceService.insertAsmModelDevice(device)) {  
 throw new ServiceException("处理装置表失败，无法新增装置。行号： " + i);  
 }  
 deviceMap.put(code, device);  
 }  
 Cell planCodeCell = row.getCell(2);  
 Cell planNameCell = row.getCell(3);  
 if (planCodeCell == null || planNameCell == null) {  
 throw new ServiceException("处理装置表失败，无法读取方案信息。行号： " + i);  
 }  
  
 String planCode = planCodeCell.getStringCellValue();  
 String planName = planNameCell.getStringCellValue();  
 if (planCode == null || planName == null) {  
 throw new ServiceException("处理装置表失败，无法读取方案信息。行号： " + i);  
 }  
 Cell materielCodeCell = row.getCell(4);  
 Cell materielNaneCell = row.getCell(5);  
 if (materielCodeCell == null || materielNaneCell == null) {  
 throw new ServiceException("处理装置表失败，无法读取物料信息。行号： " + i);  
 }  
  
 String materielCode = materielCodeCell.getStringCellValue();  
 String materielName = materielNaneCell.getStringCellValue();  
 if (materielCode == null || materielName == null) {  
 throw new ServiceException("处理装置表失败，无法读取物料信息。行号： " + i);  
 }  
  
 Cell valueCell = row.getCell(7);  
 if (valueCell == null) {  
 throw new ServiceException("处理装置表失败，无法读取产出量信息。行号： " + i);  
 }  
 double value = 0.0;  
  
 if (valueCell.getCellType() == CellType.*STRING*)  
 value = Long.*parseLong*(valueCell.getStringCellValue());  
 else if (valueCell.getCellType() == CellType.*NUMERIC*)  
 value = valueCell.getNumericCellValue();  
 else {  
 value = 0.0;  
 }  
 value = value \* 10000.0;  
  
 Cell io = row.getCell(6);  
 if (io.getStringCellValue().equals("投入")) {  
 AsmModelDeviceInput input = new AsmModelDeviceInput();  
 input.setDeviceId(device.getDeviceId());  
 input.setModelId(asmModel.getModelId());  
 input.setModelName(asmModel.getModelName());  
 input.setPlanCode(planCode);  
 input.setPlanName(planName);  
 input.setInputCode(materielCode);  
 input.setInputName(materielName);  
 input.setInputQuantity(value);  
 input.setInputCo2(0.0);  
 AsmModelMaterialBalance balance = balanceMap.get(materielCode);  
 if (balance != null && (!balance.getMaterielProperty().equals("产品"))) {  
 input.setMaterialProperty(balance.getMaterielProperty());  
 }  
 asmModelDeviceInputService.insertAsmModelDeviceInput(input);  
 inputs.add(input);  
 } else {  
 AsmModelDeviceOutput output = new AsmModelDeviceOutput();  
 output.setDeviceId(device.getDeviceId());  
 output.setModelId(asmModel.getModelId());  
 output.setModelName(asmModel.getModelName());  
 output.setPlanCode(planCode);  
 output.setPlanName(planName);  
 output.setOutputCode(materielCode);  
 output.setOutputName(materielName);  
 output.setOutputQuantity(value);  
 output.setOutputCo2(0.0);  
 output.setOutputCo2Factor(0.0);  
 AsmModelMaterialBalance balance = balanceMap.get(materielCode);  
 if (balance != null) {  
 output.setMaterialProperty(balance.getMaterielProperty());  
 }  
 asmModelDeviceOutputService.insertAsmModelDeviceOutput(output);  
 outputs.put(output.getOutputCode(), output);  
 }  
 }  
 for (AsmModelDeviceInput input : inputs) {  
 AsmModelDeviceOutput output = outputs.get(input.getInputCode());  
 if (output != null) {  
 input.setSuperOutputId(output.getOutputId());  
 if (input.getInputName().contains("干气")) {  
 input.setIsCalc("0");  
 output.setIsCalc("0");  
 }  
 asmModelDeviceOutputService.updateAsmModelDeviceOutput(output);  
 asmModelDeviceInputService.updateAsmModelDeviceInput(input);  
 }  
 }  
 }  
 }  
  
 private Map<String, AsmModelMaterialBalance> process\_balance(AsmModel asmModel, Workbook workbook) {  
 Map<String, AsmModelMaterialBalance> map = new HashMap<>();  
 if (workbook.getNumberOfSheets() < 3)  
 return map;  
  
 Sheet sheet = workbook.getSheetAt(2);  
 if (!sheet.getSheetName().equals("全厂物料平衡"))  
 return map;  
 if (sheet == null) {  
 *log*.warn("处理全厂物料平衡表失败，不存在全厂物料平衡表");  
 return map;  
 }  
 for (int i = 1; i <= sheet.getLastRowNum(); i++) {  
 Row row = sheet.getRow(i);  
 Cell propeterCell = row.getCell(0);  
 if (propeterCell == null) {  
 *log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 String property = propeterCell.getStringCellValue();  
 if (StringUtils.*isEmpty*(property)) {  
 *//throw new ServiceException("处理全厂物料平衡表失败，无法读取装置编码。行号： " + i);  
 log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 String materCode = row.getCell(1).getStringCellValue();  
 String materName = row.getCell(2).getStringCellValue();  
  
 Cell valueCell = row.getCell(3);  
 double value = 0.0;  
 if (valueCell.getCellType() == CellType.*STRING*)  
 value = Long.*parseLong*(valueCell.getStringCellValue());  
 else if (valueCell.getCellType() == CellType.*NUMERIC*)  
 value = valueCell.getNumericCellValue();  
 else {  
 value = 0.0;  
 }  
 value = value \* 10000.0;  
  
 AsmModelMaterialBalance balance = new AsmModelMaterialBalance();  
 balance.setMaterielCode(materCode);  
 balance.setMaterielName(materName);  
 balance.setMaterielProperty(property);  
 balance.setMaterielQuan(value);  
 balance.setModelId(asmModel.getModelId());  
 if (0 == asmModelMaterialBalanceService.insertAsmModelMaterialBalance(balance)) {  
 throw new ServiceException("处理全厂物料平衡表失败，无法新增平衡数据s。行号： " + i);  
 }  
 map.put(materCode, balance);  
 }  
 return map;  
 }  
  
 private Map<String, AsmModelDevice> process\_osbl(AsmModel asmModel, Workbook workbook) {  
 Sheet osblSheet = workbook.getSheetAt(1);  
 if (osblSheet == null) {  
 throw new ServiceException("处理公用工程失败，不存在公用工程表");  
 }  
 Map<String, AsmModelDevice> deviceMap = new HashMap<>();  
  
 Map<String, SysParent> dicts = sysParentService.getDicts();  
 Map<Integer, SysParent> map = new HashMap<>();  
  
 Row osblNameRow = osblSheet.getRow(1);  
 if (osblNameRow == null) {  
 *log*.error("公用工程表解析错误");  
 throw new ServiceException("公用工程表解析错误");  
 }  
 StringBuilder sb = new StringBuilder();  
 int errNum = 0;  
 int lastCellNum = 0;  
 sb.append("不存在的公用工程： ");  
 for (int i = 4; i < osblNameRow.getLastCellNum(); i++) {  
 lastCellNum = i;  
 Cell cell = osblNameRow.getCell(i);  
 if (cell.getCellType() == CellType.*BLANK* || cell.getCellType() == CellType.*\_NONE*) {  
 *log*.warn("处理到最后1列： " + i);  
 break;  
 }  
 String osblName = cell.getStringCellValue();  
 if (StringUtils.*isEmpty*(osblName)) {  
 *log*.warn("处理到最后1列： " + i);  
 break;  
 }  
 SysParent dict = dicts.get(osblName);  
 if (dict == null) {  
 sb.append(osblName);  
 sb.append(",");  
 errNum++;  
 continue;  
 }  
 map.put(i, dict);  
 }  
 if (errNum > 0) {  
 *log*.error(sb.toString());  
 throw new ServiceException(sb.toString());  
 }  
  
  
 for (int i = 3; i <= osblSheet.getLastRowNum(); i++) {  
 Row row = osblSheet.getRow(i);  
 Cell codeCell = row.getCell(0);  
 if (codeCell == null) {  
 *log*.warn("处理得到最后一行： " + i);  
 }  
 String code = codeCell.getStringCellValue();  
 if (code == null || code.length() == 0) {  
 *//throw new ServiceException("处理装置表失败，无法读取装置编码。行号： " + i);  
 log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 AsmModelDevice device = deviceMap.get(code);  
 if (device == null) {  
 device = new AsmModelDevice();  
 device.setModelId(asmModel.getModelId());  
 device.setModelName(asmModel.getModelName());  
 device.setDeviceCode(code);  
 Cell nameCell = row.getCell(1);  
 if (nameCell == null) {  
 throw new ServiceException("公用工程表格格式错误，无法读取装置名称。行号： " + i);  
 }  
 String deviceName = nameCell.getStringCellValue();  
 if (deviceName == null) {  
 throw new ServiceException("公用工程表格格式错误，无法读取装置名称。行号： " + i);  
 }  
 device.setDeviceName(deviceName);  
 device.setIsVirtual(0);  
 device.setDeviceProcess(0.0);  
 device.setDeviceCo2(0.0);  
 device.setDeviceFactor(0.0);  
 if (0 == asmModelDeviceService.insertAsmModelDevice(device)) {  
 throw new ServiceException("公用工程表格格式错误，无法新增装置。行号： " + i);  
 }  
 deviceMap.put(code, device);  
 }  
 Cell planCodeCell = row.getCell(2);  
 Cell planNameCell = row.getCell(3);  
 if (planCodeCell == null || planNameCell == null) {  
 throw new ServiceException("公用工程表格格式错误，无法读取方案信息。行号： " + i);  
 }  
  
 String planCode = planCodeCell.getStringCellValue();  
 String planName = planNameCell.getStringCellValue();  
 if (planCode == null || planName == null) {  
 throw new ServiceException("公用工程表格格式错误，无法读取方案信息。行号： " + i);  
 }  
  
 for (int j = 4; j < lastCellNum; j++) {  
 Cell cell = row.getCell(j);  
 if (cell == null) {  
 throw new ServiceException("公用工程表格格式错误，无法读取公用工程值。行号： " + i + " 列号: " + j);  
 }  
 AsmModelDeviceOsbl osbl = new AsmModelDeviceOsbl();  
 osbl.setDeviceId(device.getDeviceId());  
 osbl.setDeviceName(device.getDeviceName());  
 osbl.setModelId(asmModel.getModelId());  
 osbl.setModelName(asmModel.getModelName());  
 osbl.setPlanCode(planCode);  
 osbl.setPlanName(planName);  
 osbl.setOsblId(Long.*valueOf*(map.get(j).getDictId()));  
 osbl.setOsblName(map.get(j).getDictName());  
 osbl.setOsblCode(map.get(j).getDictCode());  
 if (cell.getCellType() == CellType.*BLANK* || cell.getCellType() == CellType.*\_NONE* || cell.getCellType() == CellType.*STRING*)  
 osbl.setConsumption(0.0);  
 else  
 osbl.setConsumption(cell.getNumericCellValue() \* 10000.0);  
 osbl.setCarbonEmission(osbl.getConsumption() \* (map.get(j).getCo2EmissionCoefficient() == null ? 0.0 : map.get(j).getCo2EmissionCoefficient()));  
 if (0 == asmModelDeviceOsblService.insertAsmModelDeviceOsbl(osbl)) {  
 throw new ServiceException("公用工程表格格式错误，无法新增装置公用工程数据。行号： " + i + " 列号: " + j);  
 }  
 }  
 }  
 return deviceMap;  
 }  
  
 private int getDictCol(String code, List<SysParent> dicts) {  
  
 for (int i = 0; i < dicts.size(); i++) {  
 if (code.equals(dicts.get(i).getDictCode())) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 private void process\_blen(AsmModel asmModel, Workbook workbook) {  
 if (workbook.getNumberOfSheets() < 4)  
 return;  
 Sheet sheet = workbook.getSheetAt(3);  
 if (!sheet.getSheetName().equals("油品调合"))  
 return;  
 if (sheet == null) {  
 *log*.warn("处理油品调和表失败，不存在油品调和表");  
 return;  
 }  
 for (int i = 1; i <= sheet.getLastRowNum(); i++) {  
 Row row = sheet.getRow(i);  
 Cell materCodeCell = row.getCell(0);  
 String oilCode = null;  
 if (materCodeCell.getCellType() == CellType.*BLANK* || materCodeCell.getCellType() == CellType.*\_NONE*)  
 oilCode = null;  
 else  
 oilCode = materCodeCell.getStringCellValue();  
  
  
 */\*\* 油品代码 \*/* String blendProduct = row.getCell(1).getStringCellValue();  
 if (blendProduct == null) {  
 *log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 if (StringUtils.*isEmpty*(blendProduct)) {  
 *//throw new ServiceException("处理全厂物料平衡表失败，无法读取装置编码。行号： " + i);  
 log*.warn("处理得到最后一行： " + i);  
 break;  
 }  
 */\*\* 调和产品 \*/* String materCode = row.getCell(2).getStringCellValue();  
 String materName = row.getCell(3).getStringCellValue();  
 String inOutType = row.getCell(4).getStringCellValue();  
  
 String proCode = null;  
 if (row.getCell(5).getCellType() == CellType.*BLANK* || row.getCell(5).getCellType() == CellType.*\_NONE*)  
 proCode = null;  
 else  
 proCode = row.getCell(5).getStringCellValue();  
  
 String proName = row.getCell(6).getStringCellValue();  
 Cell valueCell = row.getCell(7);  
 double value = 0.0;  
 String stringvalue = null;  
 if (valueCell.getCellType() == CellType.*STRING*) {  
 stringvalue = valueCell.getStringCellValue();  
 if (StringUtils.*isNotEmpty*(stringvalue))  
 value = Long.*parseLong*(valueCell.getStringCellValue());  
 } else if (valueCell.getCellType() == CellType.*NUMERIC*)  
 value = valueCell.getNumericCellValue();  
 else {  
 value = 0.0;  
 }  
 Cell proUnitCell = row.getCell(8);  
 String proUnit;  
 if (proUnitCell.getCellType() == CellType.*BLANK* || proUnitCell.getCellType() == CellType.*\_NONE*)  
 proUnit = null;  
 else  
 proUnit = proUnitCell.getStringCellValue();  
  
  
 AsmModelBlend blend = new AsmModelBlend();  
 blend.setModelId(asmModel.getModelId());  
 blend.setOilCode(oilCode);  
 blend.setBlendProduct(blendProduct);  
 blend.setMaterielCode(materCode);  
 blend.setMaterielName(materName);  
 blend.setInOutType(inOutType);  
 blend.setPropertyCode(proCode);  
 blend.setPropertyName(proName);  
 blend.setPropertyValue(value);  
 blend.setPropertyUnit(proUnit);  
 if (0 == asmModelBlendService.insertAsmModelBlend(blend)) {  
 throw new ServiceException("处理油品调和表失败，无法新增数据。行号： " + i);  
 }  
 }  
 }  
}

### Vue:

<template>

<div class="app-container">

<el-form :model="queryParams" ref="queryForm" size="small" :inline="true" v-show="showSearch" label-width="68px">

<el-form-item label="模型名称" prop="modelName">

<el-input

v-model="queryParams.modelName"

placeholder="请输入模型名称"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="所属工厂" prop="modelFactory">

<el-input

v-model="queryParams.modelFactory"

placeholder="请输入所属工厂"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="所属区域" prop="modelRegion">

<el-input

v-model="queryParams.modelRegion"

placeholder="请输入所属区域"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="所属省份" prop="modelProvince">

<el-input

v-model="queryParams.modelProvince"

placeholder="请输入所属省份"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="所属城市" prop="modelCity">

<el-input

v-model="queryParams.modelCity"

placeholder="请输入所属城市"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="所属公司" prop="modelCompany">

<el-input

v-model="queryParams.modelCompany"

placeholder="请输入所属公司"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="开始时间" prop="beginTime">

<el-date-picker clearable

v-model="queryParams.beginTime"

type="date"

value-format="yyyy-MM-dd"

placeholder="请选择开始时间">

</el-date-picker>

</el-form-item>

<el-form-item label="结束时间" prop="endTime">

<el-date-picker clearable

v-model="queryParams.endTime"

type="date"

value-format="yyyy-MM-dd"

placeholder="请选择结束时间">

</el-date-picker>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" size="mini" @click="handleQuery">搜索</el-button>

<el-button icon="el-icon-refresh" size="mini" @click="resetQuery">重置</el-button>

</el-form-item>

</el-form>

<el-row class="mb8">

<el-col>

<el-button

type="warning"

plain

icon="el-icon-download"

size="mini"

@click="handleExport"

v-hasPermi="['system:model:export']"

>导出</el-button>

</el-col>

<right-toolbar :showSearch.sync="showSearch" @queryTable="getList"></right-toolbar>

</el-row>

<el-table v-loading="loading" :data="modelList" @selection-change="handleSelectionChange" style="100%">

<el-table-column type="selection" width="55" align="center"/>

<el-table-column label="模型名称" align="center" prop="modelName" min-width="8%"/>

<el-table-column label="所属工厂" align="center" prop="modelFactory" min-width="12%"/>

<el-table-column label="所属区域" align="center" prop="modelRegion" min-width="6%"/>

<el-table-column label="所属省份" align="center" prop="modelProvince" min-width="6%"/>

<el-table-column label="所属城市" align="center" prop="modelCity" min-width="6%"/>

<el-table-column label="所属公司" align="center" prop="modelCompany" min-width="6%"/>

<el-table-column label="开始时间" align="center" prop="beginTime" min-width="8%">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.beginTime, '{y}-{m}-{d}') }}</span>

</template>

</el-table-column>

<el-table-column label="结束时间" align="center" prop="endTime" min-width="8%">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.endTime, '{y}-{m}-{d}') }}</span>

</template>

</el-table-column>

<el-table-column label="工厂碳排放" align="center" prop="totalFactoryCo2" min-width="7%">

<template slot-scope="scope">

{{scope.row.totalFactoryCo2 | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="产品碳排放" align="center" prop="totalProductCo2" min-width="7%">

<template slot-scope="scope">

{{scope.row.totalFactoryCo2 | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="总加工量" align="center" prop="totalInput" min-width="6%">

<template slot-scope="scope">

{{scope.row.totalInput | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="总产出量" align="center" prop="totalOutput" min-width="6%">

<template slot-scope="scope">

{{scope.row.totalOutput | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="操作" align="center" class-name="small-padding fixed-width" min-width="10%">

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit-outline"

@click="handleUpLoad(scope.row)"

v-hasPermi="['asm:model:cMissionCalculation']"

>打开</el-button>

</template>

</el-table-column>

</el-table>

<pagination

v-show="total>0"

:total="total"

:page.sync="queryParams.pageNum"

:limit.sync="queryParams.pageSize"

@pagination="getList"

/>

<!-- 添加或修改模型对话框 -->

<el-dialog :title="title" :visible.sync="open" width="1200px" append-to-body :close-on-click-modal=false>

<el-tabs v-model="activeName" @tab-click="handleClick">

<el-tab-pane label="装置管理" name="first">

<el-table v-loading="Dloading" :data="deviceList" height="550px">

<el-table-column type="selection" width="55" align="center"/>

<el-table-column align="center" prop="deviceCode" label="装置编码"></el-table-column>

<el-table-column align="center" prop="deviceName" label="装置名称"></el-table-column>

<el-table-column align="center" prop="deviceProcess" label="加工量（万吨）">

<template slot-scope="scope">

{{scope.row.deviceProcess | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column align="center" prop="deviceCo2" label="碳足迹（万吨CO2）">

<template slot-scope="scope">

{{scope.row.deviceCo2 | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column align="center" prop="deviceFactor" label="碳强度（吨CO2/吨）">

<template slot-scope="scope">

{{scope.row.deviceFactor | numFilter}}

</template>

</el-table-column>

</el-table>

</el-tab-pane>

<el-tab-pane label="装置消耗信息" name="second">

<el-table v-loading="Dictloading" :data="calList" height="550px">

<el-table-column label="装置名称" align="center" prop="deviceName" width="120px" fixed="left"/>

<el-table-column label="催化剂（万元）" align="center" prop="catalyzer" width="120px">

<template slot-scope="scope">

{{scope.row.catalyzer | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="辅材（万元）" align="center" prop="auxiliaryMaterials" width="80px">

<template slot-scope="scope">

{{scope.row.auxiliaryMaterials | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="新鲜水（万吨）" align="center" prop="freshWater" width="80px">

<template slot-scope="scope">

{{scope.row.freshWater | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="循环水（万吨）" align="center" prop="circulatingWater" width="80px">

<template slot-scope="scope">

{{scope.row.circulatingWater | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="软化水（万吨）" align="center" prop="waterSoftening" width="80px">

<template slot-scope="scope">

{{scope.row.waterSoftening | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="除盐水（万吨）" align="center" prop="demineralizedWater" width="80px">

<template slot-scope="scope">

{{scope.row.demineralizedWater | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="除氧水（万吨）" align="center" prop="deaeratedWater" width="80px">

<template slot-scope="scope">

{{scope.row.deaeratedWater | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="凝结水(透平)（万吨）" align="center" prop="condensate" width="100px">

<template slot-scope="scope">

{{scope.row.condensate | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="凝结水(加热)（万吨）" align="center" prop="condensateHeating" width="100px">

<template slot-scope="scope">

{{scope.row.condensateHeating | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="自产（万吨）" align="center" prop="selfProduction" width="80px">

<template slot-scope="scope">

{{scope.row.selfProduction | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="电网（万吨）" align="center" prop="powerGrid" width="80px">

<template slot-scope="scope">

{{scope.row.powerGrid | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="10Mpa蒸汽（万吨）" align="center" prop="tenMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.tenMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="5Mpa蒸汽（万吨）" align="center" prop="fiveMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.fiveMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="3.5MPa蒸汽（万吨）" align="center" prop="threePointFiveMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.threePointFiveMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="2.5MPa蒸汽（万吨）" align="center" prop="twoPointFiveMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.twoPointFiveMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="1.5MPa蒸汽（万吨）" align="center" prop="onePointFiveMPaSteam;" width="100px">

<template slot-scope="scope">

{{scope.row.onePointFiveMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="1.0MPa蒸汽（万吨）" align="center" prop="oneMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.oneMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="0.7MPa蒸汽（万吨）" align="center" prop="zeroPointSevenMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.zeroPointSevenMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="0.5MPa蒸汽（万吨）" align="center" prop="zeroPointFiveMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.zeroPointFiveMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="小于0.3MPa蒸汽（万吨）" align="center" prop="zeroPointThreeMPaSteam" width="100px">

<template slot-scope="scope">

{{scope.row.zeroPointThreeMPaSteam | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="标准煤（万吨）" align="center" prop="standardCoal" width="80px">

<template slot-scope="scope">

{{scope.row.standardCoal | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="焦炭（万吨）" align="center" prop="coke" width="80px">

<template slot-scope="scope">

{{scope.row.coke | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="石油焦（万吨）" align="center" prop="petroleumCoke" width="80px">

<template slot-scope="scope">

{{scope.row.petroleumCoke | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="标准油（万吨）" align="center" prop="standardOil" width="80px">

<template slot-scope="scope">

{{scope.row.standardOil | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="燃料油（万吨）" align="center" prop="fuelOil" width="80px">

<template slot-scope="scope">

{{scope.row.fuelOil | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="汽油（万吨）" align="center" prop="gasoline" width="80px">

<template slot-scope="scope">

{{scope.row.gasoline | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="柴油（万吨）" align="center" prop="dieselOil" width="80px">

<template slot-scope="scope">

{{scope.row.dieselOil | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="液化石油气（万吨）" align="center" prop="liquefiedPetroleumGas" width="90px">

<template slot-scope="scope">

{{scope.row.liquefiedPetroleumGas | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="液化天然气（万吨）" align="center" prop="liquefiedNaturalGas" width="90px">

<template slot-scope="scope">

{{scope.row.liquefiedNaturalGas | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="燃料气（万吨）" align="center" prop="fuelGas" width="80px">

<template slot-scope="scope">

{{scope.row.fuelGas | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="干馏瓦斯（万吨）" align="center" prop="retortGas" width="80px">

<template slot-scope="scope">

{{scope.row.retortGas | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="天然气（万吨）" align="center" prop="naturalGas" width="80px">

<template slot-scope="scope">

{{scope.row.naturalGas | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="催化烧焦（万吨）" align="center" prop="catalyticCoking" width="80px">

<template slot-scope="scope">

{{scope.row.catalyticCoking | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="低温热（MJ）" align="center" prop="lowTemperatureHeat" width="80px">

<template slot-scope="scope">

{{scope.row.lowTemperatureHeat | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="氮气（万吨）" align="center" prop="nitrogen" width="80px">

<template slot-scope="scope">

{{scope.row.nitrogen | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="综合能耗（toe）" align="center" prop="comprehensiveEnergyConsumption" width="80px">

<template slot-scope="scope">

{{scope.row.comprehensiveEnergyConsumption | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="CO2排放（万吨）" align="center" prop="cO2Emissions" width="80px">

<template slot-scope="scope">

{{scope.row.cO2Emissions | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="燃烧排放（万吨）" align="center" prop="combustionEmissions" width="80px">

<template slot-scope="scope">

{{scope.row.combustionEmissions | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="工艺排放（万吨）" align="center" prop="processDischarge" width="80px">

<template slot-scope="scope">

{{scope.row.processDischarge | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="蒸汽排放（万吨）" align="center" prop="steamDischarge" width="80px">

<template slot-scope="scope">

{{scope.row.steamDischarge | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="耗电排放（万吨）" align="center" prop="powerConsumptionEmission" width="80px">

<template slot-scope="scope">

{{scope.row.powerConsumptionEmission | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="其他排放（万吨）" align="center" prop="otherEmissions" width="80px">

<template slot-scope="scope">

{{scope.row.otherEmissions | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="直接排放（万吨）" align="center" prop="directDischarge" width="80px">

<template slot-scope="scope">

{{scope.row.directDischarge | ChangeDecimalToPercentage}}

</template>

</el-table-column>

<el-table-column label="间接排放（万吨）" align="center" prop="indirectEmissions" width="80px">

<template slot-scope="scope">

{{scope.row.indirectEmissions | ChangeDecimalToPercentage}}

</template>

</el-table-column>

</el-table>

</el-tab-pane>

<el-tab-pane label="装置碳排放" name="third" style="background: #ebedf2;">

<dv-loading style="height:600px;width:100%;" v-if="showLoading" >Loading...</dv-loading>

<!-- <dv-border-box-1 style="height:700px"> -->

<div style="height:1000px;width:100%;" id="init" v-else>

<div class="flex\_spaceAround" style="height: 50%;padding: 20px">

<div class="view" ref="discharge"></div>

<div class="view" ref="type"></div>

</div>

<div class="flex\_spaceAround" style="height: 50%;padding: 20px">

<div class="view" ref="carbon"></div>

<div class="view" ref="chart"></div>

</div>

</div>

<!-- </dv-border-box-1> -->

</el-tab-pane>

</el-tabs>

<div slot="footer" class="dialog-footer">

<el-button type="warning" @click="exportExcel">导 出</el-button>

<el-button @click="cancel">取 消</el-button>

</div>

</el-dialog>

</div>

</template>

<script>

import { listModel,listData,editModel} from "@/api/asm/model";

import { ElMapExportTable } from "table-excel";

//datav可视化

import Vue from 'vue';

import DataV from '@jiaminghi/data-view';

Vue.use(DataV);

const echarts = require('echarts');

require("echarts/theme/vintage");

import html2Canvas from 'html2canvas';

export default {

name: "Model",

data() {

return {

// 遮罩层

loading: true,

Dloading:true,

Dictloading:true,

// 选中数组

ids: [],

// 非单个禁用

single: true,

// 非多个禁用

multiple: true,

// 显示搜索条件

showSearch: true,

// 总条数

total: 0,

// 模型表格数据

modelList: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

modelName: null,

modelCompanyId: null,

modelFactory: null,

modelRegion: null,

modelProvince: null,

modelCity: null,

modelCompany: null,

modelArea: null,

beginTime: null,

endTime: null,

createId: null,

updataId: null,

totalFactoryCo2: null,

totalProductCo2: null,

totalInput: null,

totalOutput: null

},

// 表单参数

form: {},

activeName: 'first',

index:0,

//模型id

modelId:"",

//装置管理

deviceList:[],

//装置消耗

consumptionList:[],

calList:[],

showLoading:true

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询模型列表 \*/

getList() {

this.loading = true;

listModel(this.queryParams).then(response => {

this.modelList = response.rows;

this.total = response.total;

this.loading = false;

});

},

handleClick(tab, event) {

this.Dloading=true;

this.Dictloading=true;

this.showLoading=true;

listData(this.modelId).then(response=>{

// console.log(response);

this.calList=response;

if(tab.index==0){

this.index=tab.index;

this.Dloading=false

}else if(tab.index==1){

this.index=tab.index;

this.Dictloading=false

}else if(tab.index==2){

this.index=tab.index;

this.showLoading=false

this.$nextTick(() => {

this.setTest();

});

}

})

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

modelId: null,

modelName: null,

modelCompanyId: null,

modelFactory: null,

modelRegion: null,

modelProvince: null,

modelCity: null,

modelCompany: null,

modelArea: null,

beginTime: null,

endTime: null,

createId: null,

createBy: null,

createTime: null,

updataId: null,

updateBy: null,

updateTime: null,

totalFactoryCo2: null,

totalProductCo2: null,

totalInput: null,

totalOutput: null

};

this.resetForm("form");

},

/\*\* 搜索按钮操作 \*/

handleQuery() {

this.queryParams.pageNum = 1;

this.getList();

},

/\*\* 重置按钮操作 \*/

resetQuery() {

this.resetForm("queryForm");

this.handleQuery();

},

handleUpLoad(row){

this.open=true;

this.modelId=row.modelId;

this.Dloading=true;

this.Dictloading=true;

this.showLoading=true;

if(this.index==0){

editModel(row.modelId).then((response)=>{

this.deviceList=response.data.nodes;

this.Dloading=false;

})

}else if(this.index==1){

listData(this.modelId).then(response=>{

this.calList=response;

this.Dictloading=false;

})

}else if(this.index==2){

listData(this.modelId).then(response=>{

this.calList=response;

this.showLoading=false

this.$nextTick(() => {

this.setTest();

});

})

}

},

setTest(){

var uiData = [];

let deviceList=this.deviceList;

for(let item in deviceList){

uiData.push(parseFloat(deviceList[item].deviceFactor).toFixed(2));

}

//全厂碳排放总量饼图

let discharge = echarts.init(this.$refs.discharge);

let qdirectDischarge = 0;

let qindirectEmissions = 0;

let combustionEmissions = 0;

let processDischarge = 0;

let steamDischarge = 0;

let powerConsumptionEmission = 0;

let otherEmissions = 0;

this.calList.map((item) => {qdirectDischarge += item.qdirectDischarge})

this.calList.map((item) => {qindirectEmissions += item.qindirectEmissions})

this.calList.map((item) => {combustionEmissions += item.combustionEmissions})

this.calList.map((item) => {processDischarge += item.processDischarge})

this.calList.map((item) => {steamDischarge += item.steamDischarge})

this.calList.map((item) => {powerConsumptionEmission += item.powerConsumptionEmission})

this.calList.map((item) => {otherEmissions += item.otherEmissions})

discharge.setOption({

title: {

text: '全厂碳排放总量',

left: 'center',

textStyle: {

color: "#282828", // 主标题文字的颜色。

fontSize: 14,

// textShadowColor: "#C7E1CE", // 文字本身的阴影颜色。

// textShadowBlur: 5, // 文字本身的阴影长度。

},

},

tooltip: {

trigger: 'item',

formatter: '{b}: {c}万吨CO2（{d}%）'

},

color:['#F8DE74','#ED777A','#7DD6F5','#5a6fc0','#F09054','#fdcd00','#63b745'],

// legend: {

// data: [

// '间接排放',

// '直接排放',

// '燃烧排放',

// '工艺排放',

// '蒸汽排放',

// '耗电排放',

// '其他排放',

// ],

// orient: 'vertical',

// left:"0%",

// textStyle:{

// fontSize: 12,//字体大小

// color: '#ffffff'//字体颜色

// },

// },

series: [

{

type: 'pie',

radius: [0, '40%'],

label: {

position: 'inner',

fontSize: 11

},

itemStyle: {

borderColor: '#fff',

borderWidth: 2

},

selectedMode: 'single',

data: [

{ value:(qdirectDischarge/10000).toFixed(2), name: '直接排放' },

{ value:(qindirectEmissions/10000).toFixed(2), name: '间接排放'}

]

},

{

type: 'pie',

radius: ['45%', '60%'],

labelLine: {

length: 30,

lineStyle: {

color: '#c0bfbf',

},

},

itemStyle: {

borderColor: '#fff',

borderWidth: 2

},

label: {

formatter: '{b|{b}：}{c}万吨CO2',

fontSize: 12,

textStyle: {

color: '#666666'

},

rich: {}

},

data: [

{ value: (combustionEmissions/10000).toFixed(2), name: '燃烧排放' },

{ value: (processDischarge/10000).toFixed(2), name: '工艺排放' },

{ value: (steamDischarge/10000).toFixed(2), name: '蒸汽排放' },

{ value: (powerConsumptionEmission/10000).toFixed(2), name: '耗电排放' },

{ value: (otherEmissions/10000).toFixed(2), name: '其他排放' },

]

}

]

});

//装置排放类型占比柱状图

let data1 = [];

let data2 = [];

let data3 = [];

let data4 = [];

let data5 = [];

let px = this.paixu(this.calList, 'cO2Emissions');

let sortDeviceName=[]

for(let item in px){

sortDeviceName.push(px[item].deviceName)

}

data1=this.calList.map(item=>(item.combustionEmissions/10000).toFixed(2));

data2=this.calList.map(item=>(item.processDischarge/10000).toFixed(2));

data3=this.calList.map(item=>(item.steamDischarge/10000).toFixed(2));

data4=this.calList.map(item=>(item.powerConsumptionEmission/10000).toFixed(2));

data5=this.calList.map(item=>(item.otherEmissions/10000).toFixed(2));

var emphasisStyle = {

itemStyle: {

shadowBlur: 10,

shadowColor: 'rgba(0,0,0,0.3)'

}

};

var emphasisStyle = {

itemStyle: {

shadowBlur: 10,

shadowColor: 'rgba(0,0,0,0.3)'

}

};

let type = echarts.init(this.$refs.type,'light');

type.setOption({

title: {

text: '装置排放类型占比',

left: 'center',

textStyle: {

color: "#282828", // 主标题文字的颜色。

fontSize: 14

},

},

color:['#F8DE74','#ED777A','#7DD6F5','#57BD94','#F09054'],

legend: {

orient: 'vertical',

data: ['燃烧排放', '工艺排放', '蒸汽排放', '耗电排放','其他排放'],

left: '85%',

textStyle:{

fontSize: 10,//字体大小

color: '#666666'//字体颜色

},

},

tooltip: {

trigger: 'axis',

axisPointer: {

type: 'shadow'

},

formatter: function (params) {

let str = params[0].name + '<br />';

params.forEach((item) => {

if (item.value) {

str += `<span style="display:inline-block;margin-right:5px;border-radius:50%;width:10px;height:10px;left:5px;background-color: ${item.color}"></span>${item.seriesName}:${item.value}万吨CO2<br/>`;

}

});

return str;

}

},

xAxis: {

data: sortDeviceName,

axisLabel: {

rotate : 40,

show: true,

textStyle: {

color: '#666666'

}

},

axisLine: { onZero: true },

splitLine: { show: false },

splitArea: { show: false },

axisLine:false,

axisTick:false

},

yAxis: [

{

type: 'value',

name: '万吨CO2',

nameTextStyle:{

color:"#666666",

},

axisLabel: {

show: true,

textStyle: {

color: '#666666'

}

},

splitLine:{

show:true,

lineStyle:{

color:"#f5f5f5"

}

},

axisLine: {

show: false,

},

axisTick:false

}

],

grid: {

bottom: 100

},

series: [

{

name: '燃烧排放',

type: 'bar',

stack: 'one',

emphasis: emphasisStyle,

barCategoryGap:'40%',

data: data1

},

{

name: '工艺排放',

type: 'bar',

stack: 'one',

emphasis: emphasisStyle,

data: data2

},

{

name: '蒸汽排放',

type: 'bar',

stack: 'one',

emphasis: emphasisStyle,

data: data3

},

{

name: '耗电排放',

type: 'bar',

stack: 'one',

emphasis: emphasisStyle,

data: data4

},

{

name: '其他排放',

type: 'bar',

stack: 'one',

emphasis: emphasisStyle,

data: data5

},

]

});

//装置碳排放总量及对比

var co2List=this.calList.map(item=>(item.cO2Emissions/10000).toFixed(2));

var percentList=this.calList.map(item=>(item.percentage).toFixed(2))

co2List.sort((a,b) => {return b-a})

percentList.sort((a,b) => {return b-a})

var maxa = Math.max.apply(null, co2List);//拿到左边y轴数组的最大值

var maxb = Math.max.apply(null, percentList);//拿到右边y轴数组的最大值

var mina = Math.min.apply(null, co2List);//拿到左边y轴数组的最小值

var minb = Math.min.apply(null, percentList);//拿到右边y轴数组的最小值

var cha= Math.max(maxa, maxb);//两个最大值之中的最大值

var low= Math.min(mina, minb);//两个最大值之中的最小值

if((cha%10)!=0){

cha=cha+(10-(cha%10));

}

if((low%10)!=0){

low=low+(10-(low%10));

}

let carbon = echarts.init(this.$refs.carbon);

carbon.setOption({

title: {

text: '装置碳排放总量对比',

left: 'center',

textStyle: {

color: "#282828", // 主标题文字的颜色。

fontSize: 14

},

},

tooltip: {

trigger: 'axis',

formatter: function (params) {

let str = params[0].name + '<br />';

params.forEach((item) => {

if (item.value) {

if (item.seriesName.indexOf('总量') != -1) {

str += `<span style="display:inline-block;margin-right:5px;border-radius:50%;width:10px;height:10px;left:5px;background-color: ${item.color}"></span>${item.seriesName}:${item.value}万吨CO2<br/>`;

}else{

str += `<span style="display:inline-block;margin-right:5px;border-radius:50%;width:10px;height:10px;left:5px;background-color: ${item.color}"></span>${item.seriesName}:${item.value}%<br/>`;

}

}

});

return str;

}

},

xAxis: [

{

type: 'category',

data: sortDeviceName,

axisLabel: {

rotate : 40,

show: true,

textStyle: {

color: '#999999'

}

},

axisPointer: {

type: 'shadow'

},

axisLine:false,

axisTick:false

}

],

yAxis: [

{

type: 'value',

name: '单位：万吨CO2',

nameTextStyle:{

color:"#666666",

},

min:low,

max: cha,//拿到最大值

axisLabel: {

show: true,

textStyle: {

color: '#666666'

}

},

splitLine:{

show:true,

lineStyle:{

color:"#f5f5f5"

}

},

axisLine:false,

axisTick:false

},

{

type: 'value',

name: '单位：%',

nameTextStyle:{

color:"#666666",

},

min:low,

max: cha,//拿到最大值

axisLabel: {

show: true,

textStyle: {

color: '#666666'

},

formatter: '{value} %'

},

splitLine:{

show:true,

lineStyle:{

color:"#f5f5f5"

}

},

axisLine:false,

axisTick:false

}

],

series: [

{

name: '装置碳排放总量',

type: 'bar',

barWidth: '40%',

itemStyle:{

color:"#f5cd7b"

},

data:co2List

},

{

name: '装置碳排放占比',

type: 'line',

barWidth: '40%',

yAxisIndex: 1,

lineStyle:{

color:"#dd4645"

},

data:percentList

}

]

});

//装置碳排放强度柱状图

let myChart = echarts.init(this.$refs.chart,'light');

myChart.setOption({

title: {

text: '装置碳排放强度对比',

left: 'center',

textStyle: {

color: "#282828", // 主标题文字的颜色。

fontSize: 14

},

},

tooltip: {

trigger: 'axis',

axisPointer: {

type: 'shadow',

},

formatter: function (params) {

let str = params[0].name + '<br />';

params.forEach((item) => {

if (item.value) {

str += `<span style="display:inline-block;margin-right:5px;border-radius:50%;width:10px;height:10px;left:5px;background-color: ${item.color}"></span>${item.seriesName}:${item.value}吨CO2/吨<br/>`;

}

});

return str;

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true,

},

xAxis: {

type: 'category',

data: sortDeviceName,

axisLabel: {

rotate : 40,

show: true,

textStyle: {

color: '#666666'

},

},

axisLine:false,

axisTick:false

},

yAxis: [

{

type: 'value',

name: '吨CO2/吨',

nameTextStyle:{

color:"#666666",

},

axisLabel: {

show: true,

textStyle: {

color: '#666666'

},

},

splitLine:{

show:true,

lineStyle:{

color:"#f5f5f5"

}

},

axisLine:false,

axisTick:false

}

],

series:

[

{

name: '装置碳排放强度',

type: 'bar',

barWidth: '40%',

itemStyle:{

color:"#f5cd7b"

},

data: uiData,

},

],

});

},

paixu(ary, key) {

return ary.sort(function (a, b) {

let x = a[key]

let y = b[key]

return ((x > y) ? -1 : (x < y) ? 1 : 0)

})

},

// 多选框选中数据

handleSelectionChange(selection) {

this.ids = selection.map(item => item.modelId)

this.single = selection.length!==1

this.multiple = !selection.length

},

/\*\* 导出按钮操作 \*/

handleExport() {

this.download('asm/model/export', {

...this.queryParams

}, `model\_${new Date().getTime()}.xlsx`)

},

exportExcel(){

var calList = this.calList;

if(this.activeName=="first"){

var deviceList = this.deviceList;

for(let a in deviceList){

deviceList[a].deviceProcess=(deviceList[a].deviceProcess/10000).toFixed(2);

deviceList[a].deviceCo2=(deviceList[a].deviceCo2/10000).toFixed(2);

deviceList[a].deviceFactor=parseFloat(deviceList[a].deviceFactor).toFixed(2);

}

var data=deviceList;

const column = [

{ title: "装置编码", dataIndex: "deviceCode" },

{ title: "装置名称", dataIndex: "deviceName" },

{ title: "加工量（万吨）", dataIndex: "deviceProcess" },

{ title: "碳足迹（万吨CO2）", dataIndex: "deviceCo2" },

{ title: "碳强度（吨CO2/吨）", dataIndex: "deviceFactor" }

];

const instance = new ElMapExportTable(

{ column, data },

);

instance.download(`device\_${new Date().getTime()}.xlsx`); // 导出的文件名

}else if(this.activeName=="second"){

for(let a in calList){

calList[a].deviceName=calList[a].deviceName;

calList[a].catalyzer=(calList[a].catalyzer/10000).toFixed(2);

calList[a].auxiliaryMaterials=(calList[a].auxiliaryMaterials/10000).toFixed(2);

calList[a].freshWater=(calList[a].freshWater/10000).toFixed(2);

calList[a].circulatingWater=(calList[a].circulatingWater/10000).toFixed(2);

calList[a].waterSoftening=(calList[a].waterSoftening/10000).toFixed(2);

calList[a].demineralizedWater=(calList[a].demineralizedWater/10000).toFixed(2);

calList[a].deaeratedWater=(calList[a].deaeratedWater/10000).toFixed(2);

calList[a].condensate=(calList[a].condensate/10000).toFixed(2);

calList[a].condensateHeating=(calList[a].condensateHeating/10000).toFixed(2);

calList[a].selfProduction=(calList[a].selfProduction/10000).toFixed(2);

calList[a].powerGrid=(calList[a].powerGrid/10000).toFixed(2);

calList[a].tenMPaSteam=(calList[a].tenMPaSteam/10000).toFixed(2);

calList[a].fiveMPaSteam=(calList[a].fiveMPaSteam/10000).toFixed(2);

calList[a].threePointFiveMPaSteam=(calList[a].threePointFiveMPaSteam/10000).toFixed(2);

calList[a].twoPointFiveMPaSteam=(calList[a].twoPointFiveMPaSteam/10000).toFixed(2);

calList[a].onePointFiveMPaSteam=(calList[a].onePointFiveMPaSteam/10000).toFixed(2);

calList[a].oneMPaSteam=(calList[a].oneMPaSteam/10000).toFixed(2);

calList[a].zeroPointSevenMPaSteam=(calList[a].zeroPointSevenMPaSteam/10000).toFixed(2);

calList[a].zeroPointFiveMPaSteam=(calList[a].zeroPointFiveMPaSteam/10000).toFixed(2);

calList[a].zeroPointThreeMPaSteam=(calList[a].zeroPointThreeMPaSteam/10000).toFixed(2);

calList[a].standardCoal=(calList[a].standardCoal/10000).toFixed(2);

calList[a].coke=(calList[a].coke/10000).toFixed(2);

calList[a].petroleumCoke=(calList[a].petroleumCoke/10000).toFixed(2);

calList[a].standardOil=(calList[a].standardOil/10000).toFixed(2);

calList[a].fuelOil=(calList[a].fuelOil/10000).toFixed(2);

calList[a].gasoline=(calList[a].gasoline/10000).toFixed(2);

calList[a].dieselOil=(calList[a].dieselOil/10000).toFixed(2);

calList[a].liquefiedPetroleumGas=(calList[a].liquefiedPetroleumGas/10000).toFixed(2);

calList[a].liquefiedNaturalGas=(calList[a].liquefiedNaturalGas/10000).toFixed(2);

calList[a].fuelGas=(calList[a].fuelGas/10000).toFixed(2);

calList[a].retortGas=(calList[a].retortGas/10000).toFixed(2);

calList[a].naturalGas=(calList[a].naturalGas/10000).toFixed(2);

calList[a].catalyticCoking=(calList[a].catalyticCoking/10000).toFixed(2);

calList[a].lowTemperatureHeat=(calList[a].lowTemperatureHeat/10000).toFixed(2);

calList[a].nitrogen=(calList[a].nitrogen/10000).toFixed(2);

calList[a].comprehensiveEnergyConsumption=(calList[a].comprehensiveEnergyConsumption/10000).toFixed(2);

calList[a].cO2Emissions=(calList[a].cO2Emissions/10000).toFixed(2);

calList[a].combustionEmissions=(calList[a].combustionEmissions/10000).toFixed(2);

calList[a].processDischarge=(calList[a].processDischarge/10000).toFixed(2);

calList[a].steamDischarge=(calList[a].steamDischarge/10000).toFixed(2);

calList[a].powerConsumptionEmission=(calList[a].powerConsumptionEmission/10000).toFixed(2);

calList[a].otherEmissions=(calList[a].otherEmissions/10000).toFixed(2);

calList[a].directDischarge=(calList[a].directDischarge/10000).toFixed(2);

calList[a].indirectEmissions=(calList[a].indirectEmissions/10000).toFixed(2);

}

var data=calList;

const column = [

{ title: "装置名称", dataIndex: "deviceName" },

{ title: "催化剂（万元）", dataIndex: "catalyzer" },

{ title: "辅材（万元）", dataIndex: "auxiliaryMaterials" },

{ title: "新鲜水（万吨）", dataIndex: "freshWater" },

{ title: "循环水（万吨）", dataIndex: "circulatingWater" },

{ title: "软化水（万吨）", dataIndex: "waterSoftening" },

{ title: "除盐水（万吨）", dataIndex: "demineralizedWater" },

{ title: "除氧水（万吨）", dataIndex: "deaeratedWater" },

{ title: "凝结水(透平)（万吨）", dataIndex: "condensate" },

{ title: "凝结水(加热)（万吨）", dataIndex: "condensateHeating" },

{ title: "自产（万吨）", dataIndex: "selfProduction" },

{ title: "电网（万吨）", dataIndex: "powerGrid" },

{ title: "10Mpa蒸汽（万吨）", dataIndex: "tenMPaSteam" },

{ title: "5Mpa蒸汽（万吨）", dataIndex: "fiveMPaSteam" },

{ title: "3.5MPa蒸汽（万吨）", dataIndex: "threePointFiveMPaSteam" },

{ title: "2.5MPa蒸汽（万吨）", dataIndex: "twoPointFiveMPaSteam" },

{ title: "1.5MPa蒸汽（万吨）", dataIndex: "onePointFiveMPaSteam" },

{ title: "1.0MPa蒸汽（万吨）", dataIndex: "oneMPaSteam" },

{ title: "0.7MPa蒸汽（万吨）", dataIndex: "zeroPointSevenMPaSteam" },

{ title: "0.5MPa蒸汽（万吨）", dataIndex: "zeroPointFiveMPaSteam" },

{ title: "小于0.3MPa蒸汽（万吨）", dataIndex: "zeroPointThreeMPaSteam" },

{ title: "标准煤（万吨）", dataIndex: "standardCoal" },

{ title: "焦炭（万吨）", dataIndex: "coke" },

{ title: "石油焦（万吨）", dataIndex: "petroleumCoke" },

{ title: "标准油（万吨）", dataIndex: "standardOil" },

{ title: "燃料油（万吨）", dataIndex: "fuelOil" },

{ title: "汽油（万吨）", dataIndex: "gasoline" },

{ title: "柴油（万吨）", dataIndex: "dieselOil" },

{ title: "液化石油气（万吨）", dataIndex: "liquefiedPetroleumGas" },

{ title: "液化天然气（万吨）", dataIndex: "liquefiedNaturalGas" },

{ title: "燃料气（万吨）", dataIndex: "fuelGas" },

{ title: "干馏瓦斯（万吨）", dataIndex: "retortGas" },

{ title: "天然气（万吨）", dataIndex: "naturalGas" },

{ title: "催化烧焦（万吨）", dataIndex: "catalyticCoking" },

{ title: "低温热（MJ）", dataIndex: "lowTemperatureHeat" },

{ title: "氮气（万吨）", dataIndex: "nitrogen" },

{ title: "综合能耗（toe）", dataIndex: "comprehensiveEnergyConsumption" },

{ title: "CO2排放（万吨）", dataIndex: "cO2Emissions" },

{ title: "燃烧排放（万吨）", dataIndex: "combustionEmissions" },

{ title: "工艺排放（万吨）", dataIndex: "processDischarge" },

{ title: "蒸汽排放（万吨）", dataIndex: "steamDischarge" },

{ title: "耗电排放（万吨）", dataIndex: "powerConsumptionEmission" },

{ title: "其他排放（万吨）", dataIndex: "otherEmissions" },

{ title: "直接排放（万吨）", dataIndex: "directDischarge" },

{ title: "间接排放（万吨）", dataIndex: "indirectEmissions" }

];

const instance = new ElMapExportTable(

{ column, data },

);

instance.download(`consume\_${new Date().getTime()}.xlsx`); // 导出的文件名

}else{

html2Canvas(document.querySelector("#init")).then(canvas => {

let dataURL = canvas.toDataURL("image/png");

if (dataURL !== "") {

var a = document.createElement('a')

a.download = "碳排放统计大屏展示";

a.href = dataURL;

a.click()

}

})

}

}

},

filters: {

numFilter (value) {

// 截取当前数据到小数点后两位

let realVal = parseFloat(value).toFixed(2)

return realVal

},

ChangeDecimalToPercentage(value) {

let data = (value/10000).toFixed(2)

return data

},

}

};

</script>

<style>

.view{

width:48%;height:100%;background: #fff;padding: 20px;

}

</style>

<template>

<div class="app-container">

<!-- <el-form :model="queryParams" ref="queryForm" size="small" :inline="true" v-show="showSearch" label-width="120px">

<el-form-item label="标准物料名称" prop="materialName">

<el-input

v-model="queryParams.materialName"

placeholder="请输入标准物料名称"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="标准编码" prop="materialCode">

<el-input

v-model="queryParams.materialCode"

placeholder="请输入标准编码"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item>

<el-button

type="primary"

icon="el-icon-search"

size="mini"

@click="handleQuery"

>搜索</el-button

>

<el-button icon="el-icon-refresh" size="mini" @click="resetQuery"

>重置</el-button>

</el-form-item>

</el-form> -->

<el-row :gutter="10" class="mb8">

<el-col :span="1.5">

<el-button

type="primary"

plain

icon="el-icon-plus"

size="mini"

@click="handleAdd"

>新增</el-button

>

</el-col>

<right-toolbar :showSearch.sync="showSearch" @queryTable="getList"></right-toolbar>

</el-row>

<el-table

row-key="materialId"

v-loading="loading"

:data="materialList"

:height="tableHight">

<el-table-column label="工厂物料名称" align="center" prop="oilCodeReference.factoryName" min-width="20%">

<template slot-scope="scope">

<el-select v-model="scope.row.oilCodeReference.factoryName" clearable filterable allow-create @change="materieChoose(scope.row.oilCodeReference.factoryName,scope.row,scope.$index)" >

<el-option

v-for="item in materialSelectList"

:key="item.materielId"

:label="item.materielName"

:value="item.materielName">

</el-option>

</el-select>

</template>

</el-table-column>

<el-table-column label="工厂物料编码" align="center" prop="oilCodeReference.factoryCode" min-width="20%">

<template slot-scope="scope">

<el-select v-model="scope.row.oilCodeReference.factoryCode" clearable filterable allow-create @change="materieCodeChoose(scope.row.oilCodeReference.factoryCode,scope.row,scope.$index)">

<el-option

v-for="item in materialSelectList"

:key="item.materielId"

:label="item.materielCode"

:value="item.materielCode">

</el-option>

</el-select>

</template>

</el-table-column>

<el-table-column label="标准物料名称" align="center" prop="materialName" min-width="10%"/>

<el-table-column label="标准编码" align="center" prop="materialCode" min-width="10%"/>

<el-table-column label="物料类型" align="center" prop="materialType" min-width="10%">

<template slot-scope="scope">

<dict-tag :options="dict.type.material\_type" :value="scope.row.materialType"/>

</template>

</el-table-column>

<el-table-column label="物料属性1" align="center" prop="materialProp1" min-width="10%">

<template slot-scope="scope">

<dict-tag :options="dict.type.material\_prop1" :value="scope.row.materialProp1"/>

</template>

</el-table-column>

<el-table-column label="物料属性2" align="center" prop="materialProp2" min-width="10%">

<template slot-scope="scope">

<dict-tag :options="dict.type.material\_prop2" :value="scope.row.materialProp2"/>

</template>

</el-table-column>

<el-table-column label="物料属性3" align="center" prop="materialProp3" min-width="10%">

<template slot-scope="scope">

<dict-tag :options="dict.type.material\_prop3" :value="scope.row.materialProp3"/>

</template>

</el-table-column>

</el-table>

<!-- <pagination

v-show="materialList.length>0"

:total="materialList.length"

:page.sync="queryParams.pageNum"

:limit.sync="queryParams.pageSize"

@pagination="getList"

/> -->

<el-dialog title="新增物料" :visible.sync="addDialog" append-to-body :close-on-click-modal=false>

<el-form label-position="left" ref="form" :model="form" :rules="rules" label-width="120px">

<el-form-item label="标准物料名称" prop="materialName">

<el-input v-model="form.materialName" placeholder="请输入标准物料名称" />

</el-form-item>

<el-form-item label="标准物料编码" prop="materialCode">

<el-input v-model="form.materialCode" placeholder="请输入标准物料编码" />

</el-form-item>

<el-form-item label="物料类型" prop="materialType">

<el-select v-model="form.materialType" placeholder="请选择物料类型" clearable style="width:100%">

<el-option

v-for="dict in dict.type.material\_type"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="物料属性1" prop="materialProp1">

<el-select v-model="form.materialProp1" placeholder="请选择物料属性1" clearable style="width:100%">

<el-option

v-for="dict in dict.type.material\_prop1"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="物料属性2" prop="materialProp2">

<el-select v-model="form.materialProp2" placeholder="请选择物料属性2" clearable style="width:100%">

<el-option

v-for="dict in dict.type.material\_prop2"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="物料属性3" prop="materialProp3">

<el-select v-model="form.materialProp3" placeholder="请选择物料属性3" clearable style="width:100%">

<el-option

v-for="dict in dict.type.material\_prop3"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="工厂物料名称" prop="factoryName">

<el-select v-model="form.factoryName" placeholder="请选择工厂物料名称" clearable filterable allow-create style="width:100%">

<el-option

v-for="item in materialSelectList"

:key="item.materielId"

:label="item.materielName"

:value="item.materielName">

</el-option>

</el-select>

</el-form-item>

<el-form-item label="工厂物料编码" prop="factoryCode">

<el-select v-model="form.factoryCode" placeholder="请选择工厂物料编码" clearable filterable allow-create style="width:100%">

<el-option

v-for="item in materialSelectList"

:key="item.materielId"

:label="item.materielCode"

:value="item.materielCode">

</el-option>

</el-select>

</el-form-item>

</el-form>

<span slot="footer" class="dialog-footer">

<el-button @click="addDialog = false">取 消</el-button>

<el-button type="primary" @click="addSubmit">确 定</el-button>

</span>

</el-dialog>

</div>

</template>

<script>

import {getMaterial,getSelectMaterial,addfactorylist,addMaterialList} from "@/api/asm/model";

export default {

name: "Configure",

dicts: ['material\_type','material\_prop1', 'material\_prop2', 'material\_prop3'],

data() {

return {

// 遮罩层

loading: true,

//表格的高度

tableHight:document.documentElement.clientHeight-200,

// 选中数组

ids: [],

// 非单个禁用

single: true,

// 非多个禁用

multiple: true,

// 显示搜索条件

showSearch: true,

// 总条数

total: 0,

// 查询参数

queryParams: {

materialName:null,

materialCode:null,

},

// 石化产品管理表格数据

materialList: [],

modelObj:{},

visibleConfig:false,

//新增物料弹框

addDialog:false,

//新增物料

form:{},

rules:{

materialName: [

{ required: true, message: "标准物料名称不能为空", trigger: "blur" },

],

materialCode: [

{ required: true, message: "标准物料编码不能为空", trigger: "blur" },

],

materialType: [

{ required: true, message: "物料类型不能为空", trigger: "change" },

],

factoryName: [

{ required: true, message: "工厂物料名称不能为空", trigger: 'change' },

],

factoryCode: [

{ required: true, message: "工厂物料编码不能为空", trigger: 'change' },

],

},

materialSelectList:[],

materielName:''

};

},

created() {

this.modelObj=JSON.parse(localStorage.getItem('modelObj'))

this.getList();

window.onresize = function () {

this.tableHight=document.documentElement.clientHeight-200;

};

},

methods: {

/\*\* 查询石化产品管理列表 \*/

getList() {

this.loading = true;

getMaterial(this.queryParams,this.modelObj.name,this.modelObj.id).then(response => {

// console.log(response);

this.materialList=response.rows

this.total = response.total;

this.loading = false;

});

getSelectMaterial(this.modelObj.id).then(response=>{

// console.log(response);

this.materialSelectList=response.rows

})

},

// 取消按钮

cancel() {

this.reset();

},

// 表单重置

reset() {

this.form = {

materialName: null,

materialCode:null,

materialType:null,

materialProp1:null,

materialProp2:null,

materialProp3:null,

factoryName:null,

factoryCode:null

};

this.resetForm("form");

},

/\*\* 搜索按钮操作 \*/

handleQuery() {

this.getList();

},

/\*\* 重置按钮操作 \*/

resetQuery() {

this.resetForm("queryForm");

this.handleQuery();

},

//新增按钮操作

handleAdd(){

this.reset();

this.addDialog = true;

this.form.modelFactory=this.modelObj.name;

this.title = "添加模型管理";

},

addSubmit(){

const list=[];

list.push(this.form);

this.$refs["form"].validate((valid)=>{

if(valid){

addMaterialList(list).then(response=>{

this.addDialog=false;

if(response.code==200){

this.$modal.msgSuccess("操作成功");

this.getList()

}else{

this.$modal.msgError("添加失败")

}

})

}

})

},

//配置物料名称下拉框选择

materieChoose(name,val){

let index=this.materialList.indexOf(val);

const materialSelectList=this.materialSelectList;

for(let i in materialSelectList){

if(name==materialSelectList[i].materielName){

this.materialList[index].oilCodeReference.factoryCode=materialSelectList[i].materielCode

}

}

const obj={

modelFactory:this.modelObj.name,

materialId:val.materialId,

materialCode:val.materialCode,

refernceId:val.oilCodeReference.refernceId?val.oilCodeReference.refernceId:'',

factoryName:val.oilCodeReference.factoryName?val.oilCodeReference.factoryName:name,

factoryCode:val.oilCodeReference.factoryCode

}

const list=[];

list.push(obj)

addfactorylist(list).then(response=>{

if(response.code!=200){

this.$modal.msgError("操作失败")

}

})

},

//配置物料编码下拉框选择

materieCodeChoose(name,val){

let index=this.materialList.indexOf(val);

const materialSelectList=this.materialSelectList;

for(let i in materialSelectList){

if(name==materialSelectList[i].materielCode){

this.materialList[index].oilCodeReference.factoryName=materialSelectList[i].materielName

}

}

const obj={

modelFactory:this.modelObj.name,

materialId:val.materialId,

materialCode:val.materialCode,

refernceId:val.oilCodeReference.refernceId?val.oilCodeReference.refernceId:'',

factoryName:val.oilCodeReference.factoryName?val.oilCodeReference.factoryName:name,

factoryCode:val.oilCodeReference.factoryCode

}

const list=[];

list.push(obj)

addfactorylist(list).then(response=>{

if(response.code!=200){

this.$modal.msgError("操作失败")

}

})

},

},

};

</script>

<style scoped>

/deep/.el-scrollbar {

overflow:auto;

}

</style>