石化产品价格管理数据库V1.0源代码

北京杰迅鸿翔信息技术有限公司

### Java:

package com.jxhs.asm.service.impl;  
  
import java.util.List;  
  
import com.jxhs.asm.domain.AsmModelDeviceInput;  
import com.jxhs.asm.mapper.AsmModelDeviceInputMapper;  
import com.jxhs.asm.service.IAsmModelDeviceInputService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 装置投入Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@Service  
public class AsmModelDeviceInputServiceImpl implements IAsmModelDeviceInputService  
{  
 @Autowired  
 private AsmModelDeviceInputMapper asmModelDeviceInputMapper;  
  
 */\*\*  
 \* 查询装置投入  
 \*  
 \* @param inputId 装置投入主键  
 \* @return 装置投入  
 \*/* @Override  
 public AsmModelDeviceInput selectAsmModelDeviceInputByInputId(Long inputId)  
 {  
 return asmModelDeviceInputMapper.selectAsmModelDeviceInputByInputId(inputId);  
 }  
  
 */\*\*  
 \* 查询装置投入列表  
 \*  
 \* @param asmModelDeviceInput 装置投入  
 \* @return 装置投入  
 \*/* @Override  
 public List<AsmModelDeviceInput> selectAsmModelDeviceInputList(AsmModelDeviceInput asmModelDeviceInput)  
 {  
 return asmModelDeviceInputMapper.selectAsmModelDeviceInputList(asmModelDeviceInput);  
 }  
  
 */\*\*  
 \* 新增装置投入  
 \*  
 \* @param asmModelDeviceInput 装置投入  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModelDeviceInput(AsmModelDeviceInput asmModelDeviceInput)  
 {  
 return asmModelDeviceInputMapper.insertAsmModelDeviceInput(asmModelDeviceInput);  
 }  
  
 */\*\*  
 \* 修改装置投入  
 \*  
 \* @param asmModelDeviceInput 装置投入  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModelDeviceInput(AsmModelDeviceInput asmModelDeviceInput)  
 {  
 return asmModelDeviceInputMapper.updateAsmModelDeviceInput(asmModelDeviceInput);  
 }  
  
 */\*\*  
 \* 批量删除装置投入  
 \*  
 \* @param inputIds 需要删除的装置投入主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceInputByInputIds(Long[] inputIds)  
 {  
 return asmModelDeviceInputMapper.deleteAsmModelDeviceInputByInputIds(inputIds);  
 }  
  
 */\*\*  
 \* 删除装置投入信息  
 \*  
 \* @param inputId 装置投入主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceInputByInputId(Long inputId)  
 {  
 return asmModelDeviceInputMapper.deleteAsmModelDeviceInputByInputId(inputId);  
 }  
  
 @Override  
 public int deleteAsmModelDeviceInputByModelIds(Long[] modelIds) {  
 return asmModelDeviceInputMapper.deleteAsmModelDeviceInputByModelIds(modelIds);  
 }  
}

package com.jxhs.asm.service.impl;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import com.jxhs.asm.domain.AsmModelDeviceOsbl;  
import com.jxhs.asm.mapper.AsmModelDeviceOsblMapper;  
import com.jxhs.asm.service.IAsmModelDeviceOsblService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 装置消耗Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@Service  
public class AsmModelDeviceOsblServiceImpl implements IAsmModelDeviceOsblService  
{  
 @Autowired  
 private AsmModelDeviceOsblMapper asmModelDeviceOsblMapper;  
  
 */\*\*  
 \* 查询装置消耗  
 \*  
 \* @param consumeId 装置消耗主键  
 \* @return 装置消耗  
 \*/* @Override  
 public AsmModelDeviceOsbl selectAsmModelDeviceOsblByConsumeId(Long consumeId)  
 {  
 return asmModelDeviceOsblMapper.selectAsmModelDeviceOsblByConsumeId(consumeId);  
 }  
  
 */\*\*  
 \* 查询装置消耗列表  
 \*  
 \* @param asmModelDeviceOsbl 装置消耗  
 \* @return 装置消耗  
 \*/* @Override  
 public List<AsmModelDeviceOsbl> selectAsmModelDeviceOsblList(AsmModelDeviceOsbl asmModelDeviceOsbl)  
 {  
 return asmModelDeviceOsblMapper.selectAsmModelDeviceOsblList(asmModelDeviceOsbl);  
 }  
  
 */\*\*  
 \* 新增装置消耗  
 \*  
 \* @param asmModelDeviceOsbl 装置消耗  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModelDeviceOsbl(AsmModelDeviceOsbl asmModelDeviceOsbl)  
 {  
 return asmModelDeviceOsblMapper.insertAsmModelDeviceOsbl(asmModelDeviceOsbl);  
 }  
  
 */\*\*  
 \* 修改装置消耗  
 \*  
 \* @param asmModelDeviceOsbl 装置消耗  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModelDeviceOsbl(AsmModelDeviceOsbl asmModelDeviceOsbl)  
 {  
 return asmModelDeviceOsblMapper.updateAsmModelDeviceOsbl(asmModelDeviceOsbl);  
 }  
  
 */\*\*  
 \* 批量删除装置消耗  
 \*  
 \* @param consumeIds 需要删除的装置消耗主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceOsblByConsumeIds(Long[] consumeIds)  
 {  
 return asmModelDeviceOsblMapper.deleteAsmModelDeviceOsblByConsumeIds(consumeIds);  
 }  
  
 */\*\*  
 \* 删除装置消耗信息  
 \*  
 \* @param consumeId 装置消耗主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceOsblByConsumeId(Long consumeId)  
 {  
 return asmModelDeviceOsblMapper.deleteAsmModelDeviceOsblByConsumeId(consumeId);  
 }  
  
 @Override  
 public ArrayList<AsmModelDeviceOsbl> selectobs(Long deviceId) {  
 ArrayList<AsmModelDeviceOsbl> list = asmModelDeviceOsblMapper.selectobs(deviceId);  
 return list;  
 }  
  
 @Override  
 public int deleteAsmModelDeviceOsblByModelIds(Long[] modelIds) {  
 return asmModelDeviceOsblMapper.deleteAsmModelDeviceOsblByModelIds(modelIds);  
 }  
}

package com.jxhs.asm.service.impl;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import com.jxhs.asm.domain.AsmModelDeviceOutput;  
import com.jxhs.asm.mapper.AsmModelDeviceOutputMapper;  
import com.jxhs.asm.service.IAsmModelDeviceOutputService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 装置产出Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@Service  
public class AsmModelDeviceOutputServiceImpl implements IAsmModelDeviceOutputService  
{  
 @Autowired  
 private AsmModelDeviceOutputMapper asmModelDeviceOutputMapper;  
  
 */\*\*  
 \* 查询装置产出  
 \*  
 \* @param outputId 装置产出主键  
 \* @return 装置产出  
 \*/* @Override  
 public AsmModelDeviceOutput selectAsmModelDeviceOutputByOutputId(Long outputId)  
 {  
 return asmModelDeviceOutputMapper.selectAsmModelDeviceOutputByOutputId(outputId);  
 }  
  
 */\*\*  
 \* 查询装置产出列表  
 \*  
 \* @param asmModelDeviceOutput 装置产出  
 \* @return 装置产出  
 \*/* @Override  
 public List<AsmModelDeviceOutput> selectAsmModelDeviceOutputList(AsmModelDeviceOutput asmModelDeviceOutput)  
 {  
 return asmModelDeviceOutputMapper.selectAsmModelDeviceOutputList(asmModelDeviceOutput);  
 }  
  
 */\*\*  
 \* 新增装置产出  
 \*  
 \* @param asmModelDeviceOutput 装置产出  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModelDeviceOutput(AsmModelDeviceOutput asmModelDeviceOutput)  
 {  
 return asmModelDeviceOutputMapper.insertAsmModelDeviceOutput(asmModelDeviceOutput);  
 }  
  
 */\*\*  
 \* 修改装置产出  
 \*  
 \* @param asmModelDeviceOutput 装置产出  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModelDeviceOutput(AsmModelDeviceOutput asmModelDeviceOutput)  
 {  
 return asmModelDeviceOutputMapper.updateAsmModelDeviceOutput(asmModelDeviceOutput);  
 }  
  
 */\*\*  
 \* 批量删除装置产出  
 \*  
 \* @param outputIds 需要删除的装置产出主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceOutputByOutputIds(Long[] outputIds)  
 {  
 return asmModelDeviceOutputMapper.deleteAsmModelDeviceOutputByOutputIds(outputIds);  
 }  
  
 */\*\*  
 \* 删除装置产出信息  
 \*  
 \* @param outputId 装置产出主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceOutputByOutputId(Long outputId)  
 {  
 return asmModelDeviceOutputMapper.deleteAsmModelDeviceOutputByOutputId(outputId);  
 }  
  
 @Override  
 public ArrayList<AsmModelDeviceOutput> selectAsmModelDeviceOutputByModelId(Long modelId) {  
 ArrayList<AsmModelDeviceOutput> asmModelDeviceOutputs = asmModelDeviceOutputMapper.selectAsmModelDeviceOutputByModelId(modelId);  
 return asmModelDeviceOutputs;  
 }  
  
 @Override  
 public int deleteAsmModelDeviceOutputByModelIds(Long[] modelIds) {  
 return asmModelDeviceOutputMapper.deleteAsmModelDeviceOutputByModelIds(modelIds);  
 }  
}

package com.jxhs.asm.service.impl;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import com.jxhs.asm.domain.AsmModelMaterialBalance;  
import com.jxhs.asm.mapper.AsmModelMaterialBalanceMapper;  
import com.jxhs.asm.service.IAsmModelMaterialBalanceService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 物料平衡Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*@Service  
public class AsmModelMaterialBalanceServiceImpl implements IAsmModelMaterialBalanceService  
{  
 @Autowired  
 private AsmModelMaterialBalanceMapper asmModelMaterialBalanceMapper;  
  
 */\*\*  
 \* 查询物料平衡  
 \*  
 \* @param materielId 物料平衡主键  
 \* @return 物料平衡  
 \*/* @Override  
 public AsmModelMaterialBalance selectAsmModelMaterialBalanceByMaterielId(Long materielId)  
 {  
 return asmModelMaterialBalanceMapper.selectAsmModelMaterialBalanceByMaterielId(materielId);  
 }  
  
 */\*\*  
 \* 查询物料平衡列表  
 \*  
 \* @param asmModelMaterialBalance 物料平衡  
 \* @return 物料平衡  
 \*/* @Override  
 public List<AsmModelMaterialBalance> selectAsmModelMaterialBalanceList(AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 return asmModelMaterialBalanceMapper.selectAsmModelMaterialBalanceList(asmModelMaterialBalance);  
 }  
  
 */\*\*  
 \* 新增物料平衡  
 \*  
 \* @param asmModelMaterialBalance 物料平衡  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModelMaterialBalance(AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 return asmModelMaterialBalanceMapper.insertAsmModelMaterialBalance(asmModelMaterialBalance);  
 }  
  
 */\*\*  
 \* 修改物料平衡  
 \*  
 \* @param asmModelMaterialBalance 物料平衡  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModelMaterialBalance(AsmModelMaterialBalance asmModelMaterialBalance)  
 {  
 return asmModelMaterialBalanceMapper.updateAsmModelMaterialBalance(asmModelMaterialBalance);  
 }  
  
 */\*\*  
 \* 批量删除物料平衡  
 \*  
 \* @param materielIds 需要删除的物料平衡主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelMaterialBalanceByMaterielIds(Long[] materielIds)  
 {  
 return asmModelMaterialBalanceMapper.deleteAsmModelMaterialBalanceByMaterielIds(materielIds);  
 }  
  
 */\*\*  
 \* 删除物料平衡信息  
 \*  
 \* @param materielId 物料平衡主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelMaterialBalanceByMaterielId(Long materielId)  
 {  
 return asmModelMaterialBalanceMapper.deleteAsmModelMaterialBalanceByMaterielId(materielId);  
 }  
  
 @Override  
 public ArrayList<AsmModelMaterialBalance> selectAsmModelBlendByModelId(Long modelId) {  
 ArrayList<AsmModelMaterialBalance> asmModelMaterialBalances = asmModelMaterialBalanceMapper.selectAsmModelBlendByModelId(modelId);  
 return asmModelMaterialBalances;  
 }  
  
 @Override  
 public int deleteAsmModelMaterialBalanceByModelIds(Long[] modelIds) {  
 return asmModelMaterialBalanceMapper.deleteAsmModelMaterialBalanceByModelIds(modelIds);  
 }  
}

package com.jxhs.asm.service.impl;  
  
import java.util.List;  
  
import com.jxhs.asm.domain.AsmShowTable;  
import com.jxhs.asm.mapper.AsmShowTableMapper;  
import com.jxhs.asm.service.IAsmShowTableService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 【请填写功能名称】Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-05-09  
 \*/*@Service  
public class AsmShowTableServiceImpl implements IAsmShowTableService  
{  
 @Autowired  
 private AsmShowTableMapper asmShowTableMapper;  
  
 */\*\*  
 \* 查询【请填写功能名称】  
 \*  
 \* @param userId 【请填写功能名称】主键  
 \* @return 【请填写功能名称】  
 \*/* @Override  
 public List<AsmShowTable> selectAsmShowTableByUserId(Long userId)  
 {  
 return asmShowTableMapper.selectAsmShowTableByUserId(userId);  
 }  
  
 */\*\*  
 \* 查询【请填写功能名称】列表  
 \*  
 \* @param asmShowTable 【请填写功能名称】  
 \* @return 【请填写功能名称】  
 \*/* @Override  
 public List<AsmShowTable> selectAsmShowTableList(AsmShowTable asmShowTable)  
 {  
 return asmShowTableMapper.selectAsmShowTableList(asmShowTable);  
 }  
  
 */\*\*  
 \* 新增【请填写功能名称】  
 \*  
 \* @param asmShowTable 【请填写功能名称】  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmShowTable(AsmShowTable asmShowTable)  
 {  
 return asmShowTableMapper.insertAsmShowTable(asmShowTable);  
 }  
  
 */\*\*  
 \* 修改【请填写功能名称】  
 \*  
 \* @param asmShowTable 【请填写功能名称】  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmShowTable(AsmShowTable asmShowTable)  
 {  
 return asmShowTableMapper.updateAsmShowTable(asmShowTable);  
 }  
  
 */\*\*  
 \* 批量删除【请填写功能名称】  
 \*  
 \* @param userIds 需要删除的【请填写功能名称】主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmShowTableByUserIds(Long[] userIds)  
 {  
 return asmShowTableMapper.deleteAsmShowTableByUserIds(userIds);  
 }  
  
 */\*\*  
 \* 删除【请填写功能名称】信息  
 \*  
 \* @param userId 【请填写功能名称】主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmShowTableByUserId(Long userId)  
 {  
 return asmShowTableMapper.deleteAsmShowTableByUserId(userId);  
 }  
}

package com.jxhs.framework.verificationkey;  
  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.ApplicationArguments;  
import org.springframework.boot.ApplicationRunner;  
import org.springframework.stereotype.Component;  
  
@Component  
@Slf4j  
public class InitApplicationRunner implements ApplicationRunner {  
 @Value("${password.pwd}")  
 private String pwd;  
 @Value("${Validationdata.data}")  
 private String data;  
 @Autowired  
 private com.jxhs.common.encryption.sm4Demo sm4Demo;  
 private final String mi = "若依数据密钥";  
 private final String keys = "0ED82895525FC643";  
 @Override  
 public void run(ApplicationArguments args) throws Exception {  
 try {  
 String key = sm4Demo.decryptSm4(keys, pwd);  
 String s = sm4Demo.decryptSm4(key, data);  
 if (mi.equals(s)){  
 *log*.info("验证成功");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 *log*.info("验证失败");  
 }  
  
 }  
  
}

package com.jxhs.asm.domain;  
  
import java.util.Date;  
import java.util.List;  
  
import com.fasterxml.jackson.annotation.JsonFormat;  
import org.apache.commons.lang3.builder.ToStringBuilder;  
import org.apache.commons.lang3.builder.ToStringStyle;  
import com.jxhs.common.annotation.Excel;  
import com.jxhs.common.core.domain.BaseEntity;  
  
*/\*\*  
 \* 模型对象 asm\_model  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-21  
 \*/*public class AsmModel extends BaseEntity  
{  
 private static final long *serialVersionUID* = 1L;  
  
 */\*\* 模型ID \*/* private Long modelId;  
  
 */\*\* 模型名称 \*/* @Excel(name = "模型名称")  
 private String modelName;  
  
 */\*\* 所属企业ID \*/* @Excel(name = "所属企业ID")  
 private Long modelCompanyId;  
  
 */\*\* 所属工厂 \*/* @Excel(name = "所属工厂")  
 private String modelFactory;  
  
 */\*\* 所属区域 \*/* @Excel(name = "所属区域")  
 private String modelRegion;  
  
 */\*\* 所属省份 \*/* @Excel(name = "所属省份")  
 private String modelProvince;  
  
 */\*\* 所属城市 \*/* @Excel(name = "所属城市")  
 private String modelCity;  
  
 */\*\* 所属公司 \*/* @Excel(name = "所属公司")  
 private String modelCompany;  
  
 */\*\* 所属区县 \*/* @Excel(name = "所属区县")  
 private String modelArea;  
  
 */\*\* 开始时间 \*/* @JsonFormat(pattern = "yyyy-MM-dd")  
 @Excel(name = "开始时间", width = 30, dateFormat = "yyyy-MM-dd")  
 private Date beginTime;  
  
 */\*\* 结束时间 \*/* @JsonFormat(pattern = "yyyy-MM-dd")  
 @Excel(name = "结束时间", width = 30, dateFormat = "yyyy-MM-dd")  
 private Date endTime;  
  
 */\*\* 工厂碳排放 \*/* @Excel(name = "工厂碳排放")  
 private Double totalFactoryCo2;  
  
 */\*\* 产品碳排放 \*/* @Excel(name = "产品碳排放")  
 private Double totalProductCo2;  
  
 */\*\* 总加工量 \*/* @Excel(name = "总加工量")  
 private Double totalInput;  
  
 */\*\* 总产出量 \*/* @Excel(name = "总产出量")  
 private Double totalOutput;  
  
 private String fileUrl;  
  
 *//价格体系* private String priceSystem;  
  
 *//价格体系id* private Long priceSystemId;  
  
 *//是否计算过* private String isCalc;  
  
 private List<AsmModelDevice> devices;  
 private List<AsmModelMaterialBalance> materialBalances;  
 private List<AsmModelBlend> blends;  
  
 public void setModelId(Long modelId)  
 {  
 this.modelId = modelId;  
 }  
  
 public Long getModelId()  
 {  
 return modelId;  
 }  
 public void setModelName(String modelName)  
 {  
 this.modelName = modelName;  
 }  
  
 public String getModelName()  
 {  
 return modelName;  
 }  
  
 public void setModelRegion(String modelRegion)  
 {  
 this.modelRegion = modelRegion;  
 }  
  
 public String getModelRegion()  
 {  
 return modelRegion;  
 }  
 public void setModelProvince(String modelProvince)  
 {  
 this.modelProvince = modelProvince;  
 }  
  
 public String getModelProvince()  
 {  
 return modelProvince;  
 }  
 public void setModelCity(String modelCity)  
 {  
 this.modelCity = modelCity;  
 }  
  
 public String getModelCity()  
 {  
 return modelCity;  
 }  
 public void setModelCompany(String modelCompany)  
 {  
 this.modelCompany = modelCompany;  
 }  
  
 public String getModelCompany()  
 {  
 return modelCompany;  
 }  
 public void setModelArea(String modelArea)  
 {  
 this.modelArea = modelArea;  
 }  
  
 public String getModelArea()  
 {  
 return modelArea;  
 }  
 public void setBeginTime(Date beginTime)  
 {  
 this.beginTime = beginTime;  
 }  
  
 public Date getBeginTime()  
 {  
 return beginTime;  
 }  
 public void setEndTime(Date endTime)  
 {  
 this.endTime = endTime;  
 }  
  
 public Date getEndTime()  
 {  
 return endTime;  
 }  
  
 public Long getModelCompanyId() {  
 return modelCompanyId;  
 }  
  
 public void setModelCompanyId(Long modelCompanyId) {  
 this.modelCompanyId = modelCompanyId;  
 }  
  
 public Double getTotalFactoryCo2() {  
 return totalFactoryCo2;  
 }  
  
 public void setTotalFactoryCo2(Double totalFactoryCo2) {  
 this.totalFactoryCo2 = totalFactoryCo2;  
 }  
  
 public Double getTotalProductCo2() {  
 return totalProductCo2;  
 }  
  
 public void setTotalProductCo2(Double totalProductCo2) {  
 this.totalProductCo2 = totalProductCo2;  
 }  
  
 public Double getTotalInput() {  
 return totalInput;  
 }  
  
 public void setTotalInput(Double totalInput) {  
 this.totalInput = totalInput;  
 }  
  
 public Double getTotalOutput() {  
 return totalOutput;  
 }  
  
 public void setTotalOutput(Double totalOutput) {  
 this.totalOutput = totalOutput;  
 }  
  
 public String getModelFactory() {  
 return modelFactory;  
 }  
  
 public void setModelFactory(String modelFactory) {  
 this.modelFactory = modelFactory;  
 }  
  
 public String getFileUrl() {  
 return fileUrl;  
 }  
  
 public void setFileUrl(String fileUrl) {  
 this.fileUrl = fileUrl;  
 }  
  
 public List<AsmModelDevice> getDevices() {  
 return devices;  
 }  
  
 public void setDevices(List<AsmModelDevice> devices) {  
 this.devices = devices;  
 }  
  
 public List<AsmModelMaterialBalance> getMaterialBalances() {  
 return materialBalances;  
 }  
  
 public void setMaterialBalances(List<AsmModelMaterialBalance> materialBalances) {  
 this.materialBalances = materialBalances;  
 }  
  
 public List<AsmModelBlend> getBlends() {  
 return blends;  
 }  
  
 public void setBlends(List<AsmModelBlend> blends) {  
 this.blends = blends;  
 }  
  
 public String getPriceSystem() {  
 return priceSystem;  
 }  
  
 public void setPriceSystem(String priceSystem) {  
 this.priceSystem = priceSystem;  
 }  
  
 public Long getPriceSystemId() {  
 return priceSystemId;  
 }  
  
 public void setPriceSystemId(Long priceSystemId) {  
 this.priceSystemId = priceSystemId;  
 }  
  
 public String getIsCalc() {  
 return isCalc;  
 }  
  
 public void setIsCalc(String isCalc) {  
 this.isCalc = isCalc;  
 }  
  
 @Override  
 public String toString() {  
 return "AsmModel{" +  
 "modelId=" + modelId +  
 ", modelName='" + modelName + '\'' +  
 ", modelCompanyId=" + modelCompanyId +  
 ", modelFactory='" + modelFactory + '\'' +  
 ", modelRegion='" + modelRegion + '\'' +  
 ", modelProvince='" + modelProvince + '\'' +  
 ", modelCity='" + modelCity + '\'' +  
 ", modelCompany='" + modelCompany + '\'' +  
 ", modelArea='" + modelArea + '\'' +  
 ", beginTime=" + beginTime +  
 ", endTime=" + endTime +  
 ", totalFactoryCo2=" + totalFactoryCo2 +  
 ", totalProductCo2=" + totalProductCo2 +  
 ", totalInput=" + totalInput +  
 ", totalOutput=" + totalOutput +  
 ", fileUrl='" + fileUrl + '\'' +  
 ", priceSystem='" + priceSystem + '\'' +  
 ", priceSystemId=" + priceSystemId +  
 ", isCalc='" + isCalc + '\'' +  
 ", devices=" + devices +  
 ", materialBalances=" + materialBalances +  
 ", blends=" + blends +  
 '}';  
 }  
}

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);  
 }  
 }  
 }  
}

package com.jxhs.asm.service.impl;  
  
  
import java.math.BigDecimal;  
import java.text.DecimalFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.List;  
  
import com.fasterxml.jackson.annotation.JsonTypeInfo;  
import com.jxhs.asm.domain.\*;  
import com.jxhs.asm.domain.AsmAccessoryDetail;  
import com.jxhs.asm.domain.AsmCompany;  
import com.jxhs.asm.domain.AsmModel;  
import com.jxhs.asm.domain.AsmModelDeviceData;  
import com.jxhs.asm.mapper.AsmAccessoryDetailMapper;  
import com.jxhs.asm.mapper.AsmModelDeviceDataMapper;  
import com.jxhs.asm.service.IAsmCompanyService;  
import com.jxhs.asm.service.IAsmModelDeviceDataService;  
import com.jxhs.asm.vo.AsmModelDeviceDatasVo;  
import com.jxhs.common.core.domain.entity.SysRole;  
import com.jxhs.common.core.domain.model.LoginUser;  
import com.jxhs.common.encryption.sm4Demo;  
import com.jxhs.common.exception.ServiceException;  
import com.jxhs.common.utils.DateUtils;  
import com.jxhs.common.utils.SecurityUtils;  
import com.jxhs.common.utils.StringUtils;  
import com.jxhs.system.domain.vo.UnitQuantityCapacityVo;  
import com.jxhs.system.service.ISysRoleService;  
import lombok.extern.slf4j.Slf4j;  
import org.apache.commons.lang3.time.DateFormatUtils;  
import org.apache.poi.hssf.usermodel.HSSFCellStyle;  
import org.apache.poi.hssf.usermodel.HSSFDateUtil;  
import org.apache.poi.ss.usermodel.\*;  
import org.jodconverter.office.utils.Lo;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.stereotype.Service;  
  
*/\*\*  
 \* 【请填写功能名称】Service业务层处理  
 \*  
 \* @author ruoyi  
 \* @date 2022-04-27  
 \*/*@Slf4j  
@Service  
public class AsmModelDeviceDataServiceImpl implements IAsmModelDeviceDataService  
{  
 @Autowired  
 private AsmAccessoryDetailMapper asmAccessoryDetailMapper;  
  
  
 @Autowired  
 private AsmModelDeviceDataMapper asmModelDeviceDataMapper;  
  
  
 @Autowired  
 private IAsmCompanyService asmCompanyService;  
  
 @Autowired  
 private ISysRoleService sysRoleService;  
  
  
 @Value("${password.pwd}")  
 private String pwd;  
  
  
 @Autowired  
 private com.jxhs.common.encryption.sm4Demo sm4Demo;  
  
 */\*\*  
 \* 查询【请填写功能名称】  
 \*  
 \* @param detailId 【请填写功能名称】主键  
 \* @return 【请填写功能名称】  
 \*/* @Override  
 public AsmModelDeviceData selectAsmModelDeviceDataByPlantId(Long detailId)  
 {  
 AsmModelDeviceData asmModelDeviceData=asmModelDeviceDataMapper.selectAsmModelDeviceDataByPlantId(detailId);  
 AsmModelDeviceData amsdecrypt = this.amsdecrypt(asmModelDeviceData);  
 List<AsmAccessoryDetail> asmAccessoryDetailList=asmAccessoryDetailMapper.selectAsmAccessoryDetailByDetailId(detailId);  
 amsdecrypt.setAsmAccessoryDetails(asmAccessoryDetailList);  
 return amsdecrypt;  
 }  
  
 */\*\*  
 \* 查询【请填写功能名称】列表  
 \*  
 \* @param asmModelDeviceData 【请填写功能名称】  
 \* @return 【请填写功能名称】  
 \*/* @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDataList(AsmModelDeviceData asmModelDeviceData)  
 {  
 String key = sm4Demo.passwordSm4(pwd);  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 AsmModelDeviceData amsdecrypt = this.amsdecrypt(asmModelDeviceData);  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectTestAsmModelDeviceDataList(amsdecrypt);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(asmModelDeviceDataList);  
 for (AsmModelDeviceData amd:amsdecryptlist) {  
 List<AsmAccessoryDetail> asmAccessoryDetailList=asmAccessoryDetailMapper.selectAsmAccessoryDetailByDetailId(amd.getDetailId());  
 amd.setAsmAccessoryDetails(asmAccessoryDetailList);  
 }  
 return asmModelDeviceDataList;  
 }else if (sysRole.getRoleKey().equals("ripp")) {  
 AsmModelDeviceData amsdecrypt = this.amsdecrypt(asmModelDeviceData);  
 List<AsmModelDeviceData> asmModelDeviceDataList = asmModelDeviceDataMapper.selectCommonAsmModelDeviceDataList(amsdecrypt);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(asmModelDeviceDataList);  
 for (AsmModelDeviceData amd : amsdecryptlist) {  
 List<AsmAccessoryDetail> asmAccessoryDetailList = asmAccessoryDetailMapper.selectAsmAccessoryDetailByDetailId(amd.getDetailId());  
 amd.setAsmAccessoryDetails(asmAccessoryDetailList);  
 }  
 return asmModelDeviceDataList;  
 }else if (sysRole.getRoleKey().equals("factory")) {  
 asmModelDeviceData.setCreateId(user.getUserId());  
 AsmModelDeviceData amsdecrypt = this.amsdecrypt(asmModelDeviceData);  
 List<AsmModelDeviceData> asmModelDeviceDataList = asmModelDeviceDataMapper.selectFactoryAsmModelDeviceDataList(amsdecrypt);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(asmModelDeviceDataList);  
 for (AsmModelDeviceData amd : amsdecryptlist) {  
 List<AsmAccessoryDetail> asmAccessoryDetailList = asmAccessoryDetailMapper.selectAsmAccessoryDetailByDetailId(amd.getDetailId());  
 amd.setAsmAccessoryDetails(asmAccessoryDetailList);  
 }  
 return asmModelDeviceDataList;  
 }  
 AsmModelDeviceData amsencrypt = this.amsencrypt(asmModelDeviceData);  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDataList(amsencrypt);  
 for (AsmModelDeviceData modelDeviceData : asmModelDeviceDataList) {  
 modelDeviceData.setCompanyName(sm4Demo.decryptSm41(key,modelDeviceData.getCompanyName()));  
 modelDeviceData.setDeviceName(sm4Demo.decryptSm41(key,modelDeviceData.getDeviceName()));  
 modelDeviceData.setApprovalDepartment(sm4Demo.decryptSm41(key,modelDeviceData.getApprovalDepartment()));  
 modelDeviceData.setApprovalDetails(sm4Demo.decryptSm41(key,modelDeviceData.getApprovalDetails()));  
 }  
 for (AsmModelDeviceData amd:asmModelDeviceDataList) {  
 List<AsmAccessoryDetail> asmAccessoryDetailList=asmAccessoryDetailMapper.selectAsmAccessoryDetailByDetailId(amd.getDetailId());  
 amd.setAsmAccessoryDetails(asmAccessoryDetailList);  
 }  
 return asmModelDeviceDataList;  
 }  
  
 */\*\*  
 \* 新增【请填写功能名称】  
 \*  
 \* @param asmModelDeviceData 【请填写功能名称】  
 \* @return 结果  
 \*/* @Override  
 public int insertAsmModelDeviceData(AsmModelDeviceData asmModelDeviceData)  
 {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 asmModelDeviceData.setCreateId(user.getUserId());  
 asmModelDeviceData.setCreateBy(user.getUsername());  
 asmModelDeviceData.setCreateTime(DateUtils.*getNowDate*());  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 Long roleId=sysRole.getRoleId();  
 String roleKey=sysRole.getRoleKey();  
 Long aLong=Long.*valueOf*(3);  
 if (roleId==(aLong)&&roleKey.equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else {  
 asmModelDeviceData.setRoleKey("common");  
 }  
 AsmModelDeviceData asmModelDeviceData1=new AsmModelDeviceData();  
*// StringBuffer remark=new StringBuffer();* try {  
 if (asmModelDeviceData.getApprovalDepartment()!=null){  
 if (asmModelDeviceData.getApprovalDepartment().contains("国家")){  
 asmModelDeviceData.setApprovalDepartmentLevel("国家");  
 }else if (asmModelDeviceData.getApprovalDepartment().contains("省")||asmModelDeviceData.getApprovalDepartment().contains("内蒙古")  
 ||asmModelDeviceData.getApprovalDepartment().contains("新疆")||asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 ||asmModelDeviceData.getApprovalDepartment().contains("宁夏")||asmModelDeviceData.getApprovalDepartment().contains("广西")  
 ||asmModelDeviceData.getApprovalDepartment().contains("北京")||asmModelDeviceData.getApprovalDepartment().contains("天津")  
 ||asmModelDeviceData.getApprovalDepartment().contains("上海")||asmModelDeviceData.getApprovalDepartment().contains("重庆")){  
 asmModelDeviceData.setApprovalDepartmentLevel("省级");  
 }else {  
 asmModelDeviceData.setApprovalDepartmentLevel("");  
 }  
 }  
  
 if (asmModelDeviceData.getIsExpand().equals("是")){  
 asmModelDeviceData.setIsExpand("1");  
 }else {  
 asmModelDeviceData.setIsExpand("0");  
 }  
 if (asmModelDeviceData.getIsProject().equals("是")){  
 asmModelDeviceData.setIsProject("1");  
 }else {  
 asmModelDeviceData.setIsProject("0");  
 }  
 if (asmModelDeviceData.getIsClean().equals("是")){  
 asmModelDeviceData.setIsClean("1");  
 }else {  
 asmModelDeviceData.setIsClean("0");  
 }  
 if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("正常运行")){  
 asmModelDeviceData.setStatus("1");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("已改扩建")){  
 asmModelDeviceData.setStatus("2");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("停工")){  
 asmModelDeviceData.setStatus("3");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("已拆除")){  
 asmModelDeviceData.setStatus("4");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("在建")){  
 asmModelDeviceData.setStatus("5");  
 } else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("未投产")){  
 asmModelDeviceData.setStatus("6");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("拟建")){  
 asmModelDeviceData.setStatus("7");  
 }  
  
 if (asmModelDeviceData.getRealScale()!=null&&asmModelDeviceData.getRealScale()<200.0&&asmModelDeviceData.getStatus().equals("1")  
 &&!asmModelDeviceData.getCompanyAddress().contains("格尔木")&&!asmModelDeviceData.getCompanyAddress().contains("泽普")){  
*// remark.append("T");* asmModelDeviceData.setEliminationCapacity("T");  
 }  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
*// Date middleTime=new Date();* Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2005-01-01");  
 endTime = new SimpleDateFormat(format).parse("2011-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 if ((asmModelDeviceData.getApprovalDate().before(endTime)&&asmModelDeviceData.getApprovalDate().after(startTime)  
 &&asmModelDeviceData.getApprovalScale()<800.0)||(asmModelDeviceData.getApprovalDate().after(endTime)&&asmModelDeviceData.getApprovalScale()<1000.0)){  
*// remark.append("X");* asmModelDeviceData.setLimiteCapacity("X");  
 }  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2004-01-01");  
 endTime = new SimpleDateFormat(format).parse("2016-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 if ((asmModelDeviceData.getApprovalDate().before(startTime)&&asmModelDeviceData.getIsClean().equals("0"))  
 &&!(asmModelDeviceData.getApprovalDate().before(endTime)&&asmModelDeviceData.getApprovalDate().after(startTime)  
 &&asmModelDeviceData.getApprovalDepartment().contains("国")&&asmModelDeviceData.getApprovalDepartment().contains("省")  
 &&asmModelDeviceData.getApprovalDepartment().contains("新疆")&&asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("广西")&&asmModelDeviceData.getApprovalDepartment().contains("宁夏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("内蒙古")&&asmModelDeviceData.getIsProject().equals("1"))  
 &&!(asmModelDeviceData.getApprovalDate().after(endTime)&&asmModelDeviceData.getApprovalDepartment().contains("省")  
 &&asmModelDeviceData.getApprovalDepartment().contains("新疆")&&asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("广西")&&asmModelDeviceData.getApprovalDepartment().contains("宁夏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("内蒙古"))){  
*// remark.append("W");* asmModelDeviceData.setApprovalCompliance("W");  
 }  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 AsmModelDeviceData amsencrypt = this.amsencrypt(asmModelDeviceData);  
  
 int a=asmModelDeviceDataMapper.insertAsmModelDeviceData(amsencrypt);  
 if (asmModelDeviceData.getAsmAccessoryDetails()!=null&&asmModelDeviceData.getAsmAccessoryDetails().size()>0){  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDataListOrder(asmModelDeviceData1);  
 if (asmModelDeviceDataList!=null&&asmModelDeviceDataList.size()>0){  
 Long detailid=asmModelDeviceDataList.get(asmModelDeviceDataList.size()-1).getDetailId();  
 List<AsmAccessoryDetail> asmAccessoryDetails=asmModelDeviceData.getAsmAccessoryDetails();  
 AsmAccessoryDetail asmAccessoryDetail=null;  
 if (asmAccessoryDetails!=null){  
 for (AsmAccessoryDetail detail:asmAccessoryDetails) {  
 asmAccessoryDetail=new AsmAccessoryDetail();  
 asmAccessoryDetail.setDetailId(String.*valueOf*(detailid));  
 asmAccessoryDetail.setAccessory(detail.getAccessory());  
 asmAccessoryDetail.setFileName(detail.getFileName());  
 asmAccessoryDetailMapper.insertAsmAccessoryDetail(asmAccessoryDetail);  
 }  
 }  
 }  
  
 }  
  
  
 return a;  
 }  
  
 */\*\*  
 \* 修改【请填写功能名称】  
 \*  
 \* @param asmModelDeviceData 【请填写功能名称】  
 \* @return 结果  
 \*/* @Override  
 public int updateAsmModelDeviceData(AsmModelDeviceData asmModelDeviceData)  
 {  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 asmModelDeviceData.setUpdateId(user.getUserId());  
 asmModelDeviceData.setUpdateBy(user.getUsername());  
 asmModelDeviceData.setUpdateTime(DateUtils.*getNowDate*());  
*// StringBuffer remark=new StringBuffer();* try {  
 if (asmModelDeviceData.getApprovalDepartment()!=null){  
 if (asmModelDeviceData.getApprovalDepartment().contains("国家")){  
 asmModelDeviceData.setApprovalDepartmentLevel("国家");  
 }else if (asmModelDeviceData.getApprovalDepartment().contains("省")||asmModelDeviceData.getApprovalDepartment().contains("内蒙古")  
 ||asmModelDeviceData.getApprovalDepartment().contains("新疆")||asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 ||asmModelDeviceData.getApprovalDepartment().contains("宁夏")||asmModelDeviceData.getApprovalDepartment().contains("广西")){  
 asmModelDeviceData.setApprovalDepartmentLevel("省级");  
 }else {  
 asmModelDeviceData.setApprovalDepartmentLevel("");  
 }  
 }  
  
 */\*if (asmModelDeviceData.getIsExpand().equals("是")){  
 asmModelDeviceData.setIsExpand("1");  
 }else {  
 asmModelDeviceData.setIsExpand("0");  
 }\*/  
 /\*if (asmModelDeviceData.getIsProject().equals("是")){  
 asmModelDeviceData.setIsProject("1");  
 }else {  
 asmModelDeviceData.setIsProject("0");  
 }\*/  
 /\*if (asmModelDeviceData.getIsClean().equals("是")){  
 asmModelDeviceData.setIsClean("1");  
 }else {  
 asmModelDeviceData.setIsClean("0");  
 }\*/* if (asmModelDeviceData.getStatus().equals("正常运行")){  
 asmModelDeviceData.setStatus("1");  
 }else if (asmModelDeviceData.getStatus().equals("已改扩建")){  
 asmModelDeviceData.setStatus("2");  
 }else if (asmModelDeviceData.getStatus().equals("停工")){  
 asmModelDeviceData.setStatus("3");  
 }else if (asmModelDeviceData.getStatus().equals("已拆除")){  
 asmModelDeviceData.setStatus("4");  
 }else if (asmModelDeviceData.getStatus().equals("在建")){  
 asmModelDeviceData.setStatus("5");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("未投产")){  
 asmModelDeviceData.setStatus("6");  
 }else if (asmModelDeviceData.getStatus()!=null&&asmModelDeviceData.getStatus().equals("拟建")){  
 asmModelDeviceData.setStatus("7");  
 }  
  
 if (asmModelDeviceData.getApprovalScale()<200.0&&asmModelDeviceData.getStatus().equals("1")  
 &&!asmModelDeviceData.getCompanyAddress().contains("格尔木")&&!asmModelDeviceData.getCompanyAddress().contains("泽普")){  
*// remark.append("T");* asmModelDeviceData.setEliminationCapacity("T");  
 }  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
*// Date middleTime=new Date();* Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2005-01-01");  
 endTime = new SimpleDateFormat(format).parse("2011-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 if ((asmModelDeviceData.getApprovalDate().before(endTime)&&asmModelDeviceData.getApprovalDate().after(startTime)  
 &&asmModelDeviceData.getApprovalScale()<800.0)||(asmModelDeviceData.getApprovalDate().after(endTime)&&asmModelDeviceData.getApprovalScale()<1000.0)){  
*// remark.append("X");* asmModelDeviceData.setLimiteCapacity("X");  
 }  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2004-01-01");  
 endTime = new SimpleDateFormat(format).parse("2016-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 if ((asmModelDeviceData.getApprovalDate().before(startTime)&&asmModelDeviceData.getIsClean().equals("0"))  
 &&!(asmModelDeviceData.getApprovalDate().before(endTime)&&asmModelDeviceData.getApprovalDate().after(startTime)  
 &&asmModelDeviceData.getApprovalDepartment().contains("国")&&asmModelDeviceData.getApprovalDepartment().contains("省")  
 &&asmModelDeviceData.getApprovalDepartment().contains("新疆")&&asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("广西")&&asmModelDeviceData.getApprovalDepartment().contains("宁夏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("内蒙古")&&asmModelDeviceData.getIsProject().equals("1"))  
 &&!(asmModelDeviceData.getApprovalDate().after(endTime)&&asmModelDeviceData.getApprovalDepartment().contains("省")  
 &&asmModelDeviceData.getApprovalDepartment().contains("新疆")&&asmModelDeviceData.getApprovalDepartment().contains("西藏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("广西")&&asmModelDeviceData.getApprovalDepartment().contains("宁夏")  
 &&asmModelDeviceData.getApprovalDepartment().contains("内蒙古"))){  
*// remark.append("W");* asmModelDeviceData.setApprovalCompliance("W");  
 }  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 AsmModelDeviceData amsencrypt = this.amsencrypt(asmModelDeviceData);  
*// asmModelDeviceData.setRemark(String.valueOf(remark));* int a=asmModelDeviceDataMapper.updateAsmModelDeviceData(amsencrypt);  
 int b=asmAccessoryDetailMapper.deleteAsmAccessoryDetailByDetailId(asmModelDeviceData.getDetailId());  
 List<AsmAccessoryDetail> asmAccessoryDetails=asmModelDeviceData.getAsmAccessoryDetails();  
 if (asmModelDeviceData.getAsmAccessoryDetails()!=null&&asmModelDeviceData.getAsmAccessoryDetails().size()>0){  
 for (AsmAccessoryDetail aad:asmAccessoryDetails){  
 aad.setDetailId(String.*valueOf*(asmModelDeviceData.getDetailId()));  
 asmAccessoryDetailMapper.insertAsmAccessoryDetail(aad);  
 }  
 }  
 return a;  
 }  
  
 */\*\*  
 \* 批量删除【请填写功能名称】  
 \*  
 \* @param detailIds 需要删除的【请填写功能名称】主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceDataByPlantIds(Long[] detailIds)  
 {  
 int a=asmModelDeviceDataMapper.deleteAsmModelDeviceDataByPlantIds(detailIds);  
 asmAccessoryDetailMapper.deleteAsmAccessoryDetailByDetailIds(detailIds);  
 return a;  
 }  
  
 */\*\*  
 \* 删除【请填写功能名称】信息  
 \*  
 \* @param detailId 【请填写功能名称】主键  
 \* @return 结果  
 \*/* @Override  
 public int deleteAsmModelDeviceDataByPlantId(Long detailId)  
 {  
 return asmModelDeviceDataMapper.deleteAsmModelDeviceDataByPlantId(detailId);  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDeviceSequenceList(AsmModelDeviceData asmModelDeviceData) {  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDeviceDeviceSequenceList(asmModelDeviceData);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public int deleteAll() {  
 return asmModelDeviceDataMapper.deleteAll();  
 }  
  
 @Override  
 public int insertlistAsm(List<AsmModelDeviceData> asmModelDeviceDataList) {  
 int i=0;  
 for (AsmModelDeviceData data:asmModelDeviceDataList) {  
 insertAsmModelDeviceData(data);  
 i++;  
 */\*if (data.getIsExpand().equals("是")){  
 data.setIsExpand("1");  
 }else {  
 data.setIsExpand("0");  
 }  
 if (data.getIsProject().equals("是")){  
 data.setIsProject("1");  
 }else {  
 data.setIsProject("0");  
 }  
 if (data.getIsClean().equals("是")){  
 data.setIsClean("1");  
 }else {  
 data.setIsClean("0");  
 }  
 if (data.getStatus().equals("正常运行")){  
 data.setStatus("1");  
 }else if (data.getStatus().equals("已改扩建")){  
 data.setStatus("2");  
 }else if (data.getStatus().equals("停工")){  
 data.setStatus("3");  
 }else if (data.getStatus().equals("已拆除")){  
 data.setStatus("4");  
 }else if (data.getStatus().equals("在建")){  
 data.setStatus("5");  
 }  
 StringBuffer remark=null;  
 if (data.getApprovalScale()<200.0&&data.getStatus().equals("1")  
 &&!data.getCompanyAddress().contentEquals("格尔木")&&!data.getCompanyAddress().contentEquals("泽普")){  
 remark.append("T");  
 }  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
// Date middleTime=new Date();  
 Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
// middleTime = new SimpleDateFormat(format).parse("2011-01");  
 startTime = new SimpleDateFormat(format).parse("2005-01-01");  
 endTime = new SimpleDateFormat(format).parse("2011-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 if ((data.getCompanyEstablishDate().before(endTime)&&data.getCompanyEstablishDate().after(startTime)  
 &&data.getApprovalScale()<800)&&(data.getCompanyEstablishDate().after(endTime)&&data.getApprovalScale()<1000)){  
 remark.append("X");  
 }  
 try {  
// middleTime = new SimpleDateFormat(format).parse("2011-01");  
 startTime = new SimpleDateFormat(format).parse("2004-01-01");  
 endTime = new SimpleDateFormat(format).parse("2016-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 if ((data.getApprovalDate().before(startTime)&&data.getIsClean().equals("0"))  
 &&!(data.getApprovalDate().before(endTime)&&data.getApprovalDate().after(startTime)  
 &&data.getApprovalDepartment().contains("国")&&data.getApprovalDepartment().contains("省")  
 &&data.getApprovalDepartment().contains("新疆")&&data.getApprovalDepartment().contains("西藏")  
 &&data.getApprovalDepartment().contains("广西")&&data.getApprovalDepartment().contains("宁夏")  
 &&data.getApprovalDepartment().contains("内蒙古")&&data.getIsProject().equals("1"))  
 &&!(data.getApprovalDate().after(endTime)&&data.getApprovalDepartment().contains("省")  
 &&data.getApprovalDepartment().contains("新疆")&&data.getApprovalDepartment().contains("西藏")  
 &&data.getApprovalDepartment().contains("广西")&&data.getApprovalDepartment().contains("宁夏")  
 &&data.getApprovalDepartment().contains("内蒙古"))){  
 remark.append("W");  
 }  
 data.setRemark(String.valueOf(remark));\*/* }  
 return i;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> importAsmData(Workbook workbook) {  
 {  
 List<AsmModelDeviceData> asmModelDeviceDataList = new ArrayList<>();  
 Sheet sheet = workbook.getSheetAt(0);  
 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 asmModelDeviceDataList;  
 }  
 AsmModelDeviceData asmModelDeviceData = new AsmModelDeviceData();  
 for (int i = 2; 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 deviceid1 = String.*valueOf*(propeterCell.getNumericCellValue());  
 String deviceid=deviceid1.substring(0,deviceid1.length()-2);  
 AsmModelDeviceData asmModelDeviceData1=new AsmModelDeviceData();  
  
  
 String companystate = row.getCell(1).getStringCellValue();  
 String companyzone = row.getCell(2).getStringCellValue();  
 String company\_parent = row.getCell(3).getStringCellValue();  
 String company\_name = row.getCell(4).getStringCellValue();  
 String format = "yyyy-MM-dd";  
 DataFormatter formatter = new DataFormatter();  
 SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");  
 String company\_establish\_date2 = String.*valueOf*(row.getCell(5).getNumericCellValue());  
 String a=company\_establish\_date2.substring(0,company\_establish\_date2.length()-2);  
 Date company\_establish\_date=null;  
 if (company\_establish\_date2.length()>=7){  
 company\_establish\_date= row.getCell(5).getDateCellValue();  
 }else {  
 try {  
 company\_establish\_date = new SimpleDateFormat(format).parse(a + "-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
 String company\_address = row.getCell(6).getStringCellValue();  
 String ownership = row.getCell(7).getStringCellValue();  
 String device\_name = row.getCell(8).getStringCellValue();  
 String device\_type = row.getCell(9).getStringCellValue();  
 Cell cell=row.getCell(10);  
 String value = "";  
  
 if (HSSFDateUtil.*isCellDateFormatted*(cell)) {  
 Date date = cell.getDateCellValue();  
 value = DateFormatUtils.*format*(date, "yyyy-MM-dd");  
 } else {  
 double v = cell.getNumericCellValue();  
 DecimalFormat df1 = new DecimalFormat("0");  
 value = df1.format(v);  
 }  
 *// double value = cell.getNumericCellValue();* String text = formatter.formatCellValue(row.getCell(10));  
 Date approval\_date=null;  
 int s1=text.length();  
 int s2=text.replace("/","").length();  
 int s=s1-s2;  
 if (s==0){  
 value=value.substring(0,4);  
 try {  
 approval\_date=new SimpleDateFormat(format).parse(value+"-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }else if (s==1){  
 value=value.substring(0,7);  
 try {  
 approval\_date=new SimpleDateFormat(format).parse(value+"-30");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }else {  
 try {  
 approval\_date=new SimpleDateFormat(format).parse(value);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*String approval\_date = String.valueOf(row.getCell(10).getNumericCellValue());  
 String approvaldate=approval\_date.substring(0,approval\_date.length()-2);  
 Date approval\_date1=null;  
 if (approval\_date.length()>=7){  
 approval\_date1= row.getCell(5).getDateCellValue();  
 }else {  
 try {  
 approval\_date1 = new SimpleDateFormat(format).parse(approvaldate + "-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }\*/* String approval\_department = row.getCell(11).getStringCellValue();  
 String approval\_details = row.getCell(12).getStringCellValue();  
 String approval\_property = row.getCell(13).getStringCellValue();  
 if (approval\_property==null){  
 approval\_property="新建";  
 }  
 Cell cell1=row.getCell(16);  
 String start\_time1 = "";  
  
 if (HSSFDateUtil.*isCellDateFormatted*(cell1)) {  
 Date date = cell1.getDateCellValue();  
 start\_time1 = DateFormatUtils.*format*(date, "yyyy-MM-dd");  
 } else {  
 double v = cell1.getNumericCellValue();  
 DecimalFormat df1 = new DecimalFormat("0");  
 start\_time1 = df1.format(v);  
 }  
 String start\_time3 = formatter.formatCellValue(row.getCell(16));  
 Date start\_time2=null;  
  
 int b1=start\_time3.length();  
 int b2=start\_time3.replace("/","").length();  
 int b=b1-b2;  
 if (b==0){  
 start\_time1=start\_time1.substring(0,4);  
 try {  
 start\_time2=new SimpleDateFormat(format).parse(start\_time1+"-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }else if (b==1){  
 start\_time1=start\_time1.substring(0,7);  
 try {  
 start\_time2=new SimpleDateFormat(format).parse(start\_time1+"-30");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }else {  
 try {  
 start\_time2=new SimpleDateFormat(format).parse(start\_time1);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
  
 String is\_expand = row.getCell(17).getStringCellValue();  
 String is\_project = row.getCell(18).getStringCellValue();  
 String is\_clean = row.getCell(19).getStringCellValue();  
 String status = row.getCell(20).getStringCellValue();  
 String remark = row.getCell(21).getStringCellValue();  
  
 Cell approval\_scale = row.getCell(14);  
 double approvalScale = 0.0;  
 if (approval\_scale.getCellType() == CellType.*STRING*)  
 approvalScale = Long.*parseLong*(approval\_scale.getStringCellValue());  
 else if (approval\_scale.getCellType() == CellType.*NUMERIC*)  
 approvalScale = approval\_scale.getNumericCellValue();  
 else {  
 approvalScale = 0.0;  
 }  
  
 Cell real\_scale = row.getCell(15);  
 double realScale = 0.0;  
 if (real\_scale.getCellType() == CellType.*STRING*)  
 realScale = Long.*parseLong*(real\_scale.getStringCellValue());  
 else if (real\_scale.getCellType() == CellType.*NUMERIC*)  
 realScale = real\_scale.getNumericCellValue();  
 else {  
 realScale = 0.0;  
 }  
  
 asmModelDeviceData.setDeviceId(deviceid);  
 asmModelDeviceData.setCompanyState(companystate);  
 asmModelDeviceData.setCompanyZone(companyzone);  
 asmModelDeviceData.setCompanyParent(company\_parent);  
 asmModelDeviceData.setCompanyName(company\_name);  
 asmModelDeviceData.setCompanyEstablishDate(company\_establish\_date);  
 asmModelDeviceData.setCompanyAddress(company\_address);  
 asmModelDeviceData.setOwnership(ownership);  
 asmModelDeviceData.setDeviceName(device\_name);  
 asmModelDeviceData.setDeviceType(device\_type);  
 asmModelDeviceData.setApprovalDate(approval\_date);  
 asmModelDeviceData.setApprovalDepartment(approval\_department);  
 asmModelDeviceData.setApprovalDetails(approval\_details);  
 asmModelDeviceData.setApprovalProperty(approval\_property);  
 asmModelDeviceData.setApprovalScale(approvalScale);  
 asmModelDeviceData.setRealScale(realScale);  
 asmModelDeviceData.setStartTime(start\_time2);  
 asmModelDeviceData.setIsExpand(is\_expand);  
 asmModelDeviceData.setIsProject(is\_project);  
 asmModelDeviceData.setIsClean(is\_clean);  
 asmModelDeviceData.setStatus(status);  
 asmModelDeviceData.setRemark(remark);  
 asmModelDeviceData.setApprovalCompliance(null);  
 asmModelDeviceData.setLimiteCapacity(null);  
 asmModelDeviceData.setEliminationCapacity(null);  
 if (asmModelDeviceData.getApprovalProperty()!=null&&asmModelDeviceData.getApprovalProperty().equals("新建")){  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData1.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData1.setRoleKey("common");  
 }  
 asmModelDeviceData1.setApprovalProperty("新建");  
 asmModelDeviceData1.setDeviceId(deviceid);  
  
 List<AsmModelDeviceData> asmModelDeviceDataList1=selectAsmModelDeviceDataList(asmModelDeviceData1);  
 if (asmModelDeviceDataList1.size()>=1){  
 throw new ServiceException("导入失败，装置id已存在"+deviceid);  
 }  
 if (StringUtils.*isEmpty*(deviceid)) {  
 *//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;  
 }  
 }else if (asmModelDeviceData.getApprovalProperty()!=null&&asmModelDeviceData.getApprovalProperty().equals("改扩建")){  
 AsmModelDeviceData asmModelDeviceData2=new AsmModelDeviceData();  
 asmModelDeviceData2.setApprovalProperty("改扩建");  
 asmModelDeviceData2.setDeviceId(deviceid);  
 asmModelDeviceData2.setDeviceName(device\_name);  
 asmModelDeviceData2.setCompanyName(company\_name);  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData2.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData2.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1=selectAsmModelDeviceDataList(asmModelDeviceData2);  
 if (asmModelDeviceDataList1.size()>=1){  
 throw new ServiceException("导入失败，装置id已存在"+deviceid+"公司全称已存在"+company\_name);  
 }  
 if (StringUtils.*isEmpty*(deviceid)) {  
 *//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;  
 }  
 }  
 if (0 == this.insertAsmModelDeviceData(asmModelDeviceData)) {  
 throw new ServiceException("处理全场数据表失败，无法新增全场数据。行号： " + i);  
 }  
 asmModelDeviceDataList.add(asmModelDeviceData);  
 }  
 AsmModelDeviceData asmModelDeviceData3=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData3.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData3.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> deviceIdList=this.selectAsmModelDeviceDeviceSequenceList(asmModelDeviceData3);  
 AsmModelDeviceData asmModelDeviceData1=new AsmModelDeviceData();  
 for (AsmModelDeviceData deviceid:deviceIdList) {  
 asmModelDeviceData1.setDeviceId(deviceid.getDeviceId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData1.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData1.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1=this.selectAsmModelDeviceDataListOrder(asmModelDeviceData1);  
 if (asmModelDeviceDataList1.size()>1){  
 String detail=String.*valueOf*(asmModelDeviceDataList1.get(0).getDetailId());  
 for (int i=1;i<asmModelDeviceDataList1.size();i++){  
 AsmModelDeviceData asmModelDeviceData2=asmModelDeviceDataList1.get(i);  
 asmModelDeviceData2.setParentId(detail);  
 this.updateAsmModelDeviceData(asmModelDeviceData2);  
 }  
 }  
  
 }  
 return asmModelDeviceDataList;  
 }  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDataListOrder(AsmModelDeviceData asmModelDeviceData1) {  
 *// 数据加密查询* AsmModelDeviceData amsencrypt = this.amsencrypt(asmModelDeviceData1);  
 List<AsmModelDeviceData> asmModelDeviceDatas = asmModelDeviceDataMapper.selectAsmModelDeviceDataListOrder(amsencrypt);  
 *// 查询出结果数据解密* List<AsmModelDeviceData> asmModelDeviceData = this.amsdecryptlist(asmModelDeviceDatas);  
 return asmModelDeviceData;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDatastate() {  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDatazone() {  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDeviceDatazone(asmModelDeviceData);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDataparent() {  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDeviceDataparent();  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDevicetypeparent() {  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDevicetypeparent();  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> checkasmModelDeviceDataUnique(AsmModelDeviceData asmModelDeviceData) {  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.checkasmModelDeviceDataUnique(asmModelDeviceData);  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
  
 @Override  
 public List<AsmModelDeviceData> selectAsmModelDeviceDataCompanyName() {  
 List<AsmModelDeviceData> list = asmModelDeviceDataMapper.selectAsmModelDeviceDataCompanyName();  
 List<AsmModelDeviceData> amsdecryptlist = this.amsdecryptlist(list);  
 return amsdecryptlist;  
 }  
  
 @Override  
 public List<AsmModelDeviceDatasVo> getstateeli(String eliminationCapacity) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 List<AsmModelDeviceDatasVo> asmModelDeviceDatasVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
  
 asmModelDeviceData.setEliminationCapacity(eliminationCapacity);  
 *//省份数量* for (AsmModelDeviceData amd:stateList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyState(amd.getCompanyState());  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 for (AsmModelDeviceData amdd:asmModelDeviceDataList1) {  
 StringBuffer a=new StringBuffer();  
 if ((amdd.getRealScale()!=null&&amdd.getRealScale()<200.0)){  
 a.append("实际建成常减压装置能力小于200万吨/年;");  
 }if(amdd.getStatus().equals("1")){  
 a.append("装置状态是正常运行;");  
 }if(!amdd.getCompanyAddress().contains("格尔木")){  
 a.append("装置位于青海格尔木;");  
 }if(!amdd.getCompanyAddress().contains("泽普")){  
 a.append("装置位于新疆泽普;");  
 }  
 amdd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList2 = this.amsdecryptlist(asmModelDeviceDataList1);  
 AsmModelDeviceDatasVo asmModelDeviceDatasVo=new AsmModelDeviceDatasVo();  
 asmModelDeviceDatasVo.setCompanyState(amd.getCompanyState());  
 asmModelDeviceDatasVo.setNum(asmModelDeviceDataList.size());  
 asmModelDeviceDatasVo.setAbility(ability);  
 asmModelDeviceDatasVo.setAsmModelDeviceDataList(asmModelDeviceDataList2);  
 asmModelDeviceDatasVoList.add(asmModelDeviceDatasVo);  
 }  
 }  
 return asmModelDeviceDatasVoList;  
 }  
  
 @Override  
 public List<AsmModelDeviceDatasVo> getstateast(String limite) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 List<AsmModelDeviceDatasVo> asmModelDeviceDatasVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
  
 asmModelDeviceData.setLimiteCapacity(limite);  
 *//省份数量* for (AsmModelDeviceData amd:stateList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyState(amd.getCompanyState());  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList2=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2005-01-01");  
 endTime = new SimpleDateFormat(format).parse("2011-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 for (AsmModelDeviceData amdd:asmModelDeviceDataList2) {  
 StringBuffer a=new StringBuffer();  
 if ((amdd.getApprovalDate().before(endTime)&&amdd.getApprovalDate().after(startTime)  
 &&amdd.getApprovalScale()<800.0)){  
 a.append("2005-2011年期间，新建常减压装置小于800万吨/年;");  
 }if((amdd.getApprovalDate().after(endTime)&&amdd.getApprovalScale()<1000.0)){  
 a.append("2011年以后，新建常减压装置小于1000万吨/年;");  
 }  
 amdd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);  
 AsmModelDeviceDatasVo asmModelDeviceDatasVo=new AsmModelDeviceDatasVo();  
 asmModelDeviceDatasVo.setCompanyState(amd.getCompanyState());  
 asmModelDeviceDatasVo.setNum(asmModelDeviceDataList.size());  
 asmModelDeviceDatasVo.setAbility(ability);  
 asmModelDeviceDatasVo.setAsmModelDeviceDataList(asmModelDeviceDataList1);  
 asmModelDeviceDatasVoList.add(asmModelDeviceDatasVo);  
 }  
 }  
 return asmModelDeviceDatasVoList;  
 }  
  
 @Override  
 public List<AsmModelDeviceDatasVo> getstateout(String approval) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 List<AsmModelDeviceDatasVo> asmModelDeviceDatasVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
  
 asmModelDeviceData.setApprovalCompliance(approval);  
 *//省份数量* for (AsmModelDeviceData amd:stateList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyState(amd.getCompanyState());  
  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList2=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2004-01-01");  
 endTime = new SimpleDateFormat(format).parse("2016-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 for (AsmModelDeviceData amdd:asmModelDeviceDataList2) {  
 StringBuffer a=new StringBuffer();  
 if ((amdd.getApprovalDate().before(startTime)&&amdd.getIsClean().equals("0"))){  
 a.append("2004年以前新建及改扩建，“是否2000年清理整顿保留的小炼油厂”中填“否”;");  
 }if (!(amdd.getApprovalDate().before(endTime)&&amdd.getApprovalDate().after(startTime)  
 &&amdd.getApprovalDepartment().contains("国"))&&amdd.getIsProject().equals("1")){  
 a.append("2004-2016年期间，由“国家”审批/核准,备案审批内容是否列入规划”中为“是”;");  
 }if (amdd.getApprovalDepartment().contains("省")  
 &&amdd.getApprovalDepartment().contains("新疆")&&amdd.getApprovalDepartment().contains("西藏")  
 &&amdd.getApprovalDepartment().contains("广西")&&amdd.getApprovalDepartment().contains("宁夏")  
 &&amdd.getApprovalDepartment().contains("内蒙古")&&amdd.getIsProject().equals("1")){  
 a.append("2004-2016年期间，由“省级”审批/核准,备案审批内容是否列入规划”中为“是”;");  
 }if (!(amdd.getApprovalDate().after(endTime)&&amdd.getApprovalDepartment().contains("省")  
 &&amdd.getApprovalDepartment().contains("新疆")&&amdd.getApprovalDepartment().contains("西藏")  
 &&amdd.getApprovalDepartment().contains("广西")&&amdd.getApprovalDepartment().contains("宁夏")  
 &&amdd.getApprovalDepartment().contains("内蒙古"))){  
 a.append("2016年以后，由“省级”审批/核准之外;");  
 }  
 amdd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);  
 AsmModelDeviceDatasVo asmModelDeviceDatasVo=new AsmModelDeviceDatasVo();  
 asmModelDeviceDatasVo.setCompanyState(amd.getCompanyState());  
 asmModelDeviceDatasVo.setNum(asmModelDeviceDataList.size());  
 asmModelDeviceDatasVo.setAbility(ability);  
 asmModelDeviceDatasVo.setAsmModelDeviceDataList(asmModelDeviceDataList1);  
 asmModelDeviceDatasVoList.add(asmModelDeviceDatasVo);  
 }  
 }  
 return asmModelDeviceDatasVoList;  
 }  
  
 @Override  
 public List<UnitQuantityCapacityVo> getstateatmospheric() {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
  
 *//先解密密钥* String key = sm4Demo.passwordSm4(pwd);  
 *//拿到解密的密钥 key 去加密 想要加密的字段 String  
// String deviceName=sm4Demo.encryptSm41(key,"常减压装置");* asmModelDeviceData.setDeviceType("常减压蒸馏");  
  
 *//省份数量* for (AsmModelDeviceData amd:stateList) {  
 Double ability=0.0;  
 int small200num=0;  
 Double small200ability=0.0;  
 int big200small500num=0;  
 Double big200small500ability=0.0;  
 int big500small1000num=0;  
 Double big500small1000ability=0.0;  
 int big1000num=0;  
 Double big1000ability=0.0;  
 asmModelDeviceData.setCompanyState(amd.getCompanyState());  
  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 if (add.getRealScale()<=200){  
 small200num++;  
 big500small1000ability+=add.getRealScale();  
 }else if (add.getRealScale()>200&&add.getRealScale()<=500){  
 big200small500num++;  
 big200small500ability+=add.getRealScale();  
 }else if (add.getRealScale()>500&&add.getRealScale()<=1000){  
 big500small1000num++;  
 big500small1000ability+=add.getRealScale();  
 }else if (add.getRealScale()>1000){  
 big1000num++;  
 big1000ability+=add.getRealScale();  
 }  
 }  
 UnitQuantityCapacityVo unitQuantityCapacityVo=new UnitQuantityCapacityVo();  
 unitQuantityCapacityVo.setSmall200num(small200num);  
 unitQuantityCapacityVo.setSmall200ability(small200ability);  
 unitQuantityCapacityVo.setBig200small500num(big200small500num);  
 unitQuantityCapacityVo.setBig200small500ability(big200small500ability);  
 unitQuantityCapacityVo.setBig500small1000num(big500small1000num);  
 unitQuantityCapacityVo.setBig500small1000ability(big500small1000ability);  
 unitQuantityCapacityVo.setBig1000num(big1000num);  
 unitQuantityCapacityVo.setBig1000ability(big1000ability);  
 unitQuantityCapacityVo.setCompanyState(amd.getCompanyState());  
 unitQuantityCapacityVo.setNum(asmModelDeviceDataList.size());  
 unitQuantityCapacityVo.setAbility(ability);  
 unitQuantityCapacityVoList.add(unitQuantityCapacityVo);  
 }  
 }  
 return unitQuantityCapacityVoList;  
 }  
  
 @Override  
 public List<UnitQuantityCapacityVo> getareaatmospheric() {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
 List<AsmModelDeviceData> zoneList=asmModelDeviceDataMapper.selectAsmModelDeviceDatazone(asmModelDeviceData);  
  
 *//先解密密钥* String key = sm4Demo.passwordSm4(pwd);  
 *//拿到解密的密钥 key 去加密 想要加密的字段 String  
/\* String deviceName=sm4Demo.encryptSm41(key,"常减压装置");  
 asmModelDeviceData.setDeviceName(deviceName);\*/* asmModelDeviceData.setDeviceType("常减压蒸馏");  
  
 *//大区数量* for (AsmModelDeviceData amd:zoneList) {  
 Double ability=0.0;  
 int small200num=0;  
 Double small200ability=0.0;  
 int big200small500num=0;  
 Double big200small500ability=0.0;  
 int big500small1000num=0;  
 Double big500small1000ability=0.0;  
 int big1000num=0;  
 Double big1000ability=0.0;  
 asmModelDeviceData.setCompanyZone(amd.getCompanyZone());  
  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
*// List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);* if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 if (add.getRealScale()<=200){  
 small200num++;  
 small200ability+=add.getRealScale();  
 }else if (add.getRealScale()>200&&add.getRealScale()<=500){  
 big200small500num++;  
 big200small500ability+=add.getRealScale();  
 }else if (add.getRealScale()>500&&add.getRealScale()<=1000){  
 big500small1000num++;  
 big500small1000ability+=add.getRealScale();  
 }else if (add.getRealScale()>1000){  
 big1000num++;  
 big1000ability+=add.getRealScale();  
 }  
 }  
 UnitQuantityCapacityVo unitQuantityCapacityVo=new UnitQuantityCapacityVo();  
 unitQuantityCapacityVo.setSmall200num(small200num);  
 unitQuantityCapacityVo.setSmall200ability(small200ability);  
 unitQuantityCapacityVo.setBig200small500num(big200small500num);  
 unitQuantityCapacityVo.setBig200small500ability(big200small500ability);  
 unitQuantityCapacityVo.setBig500small1000num(big500small1000num);  
 unitQuantityCapacityVo.setBig500small1000ability(big500small1000ability);  
 unitQuantityCapacityVo.setBig1000num(big1000num);  
 unitQuantityCapacityVo.setBig1000ability(big1000ability);  
 unitQuantityCapacityVo.setCompanyZone(amd.getCompanyZone());  
 unitQuantityCapacityVo.setNum(asmModelDeviceDataList.size());  
 unitQuantityCapacityVo.setAbility(ability);  
 unitQuantityCapacityVoList.add(unitQuantityCapacityVo);  
 }  
 }  
 return unitQuantityCapacityVoList;  
 }  
  
 @Override  
 public ApplianceInformation selectapplianceInformation() {  
 ApplianceInformation applianceInformation = this.tb();  
 return applianceInformation;  
  
 }  
  
  
  
  
 @Override  
 public List<UnitQuantityCapacityVo> selectareaeli(String EliminationCapacity) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> stateList=asmModelDeviceDataMapper.selectAsmModelDeviceDatastate(asmModelDeviceData);  
 List<AsmModelDeviceData> zoneList=asmModelDeviceDataMapper.selectAsmModelDeviceDatazone(asmModelDeviceData);  
  
 asmModelDeviceData.setEliminationCapacity(EliminationCapacity);  
 asmModelDeviceData.setCompanyState(null);  
 *//大区数量* for (AsmModelDeviceData amd:zoneList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyZone(amd.getCompanyZone());  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 UnitQuantityCapacityVo unitQuantityCapacityVo=new UnitQuantityCapacityVo();  
 unitQuantityCapacityVo.setCompanyZone(amd.getCompanyZone());  
 unitQuantityCapacityVo.setNum(asmModelDeviceDataList.size());  
 unitQuantityCapacityVo.setAbility(ability);  
 unitQuantityCapacityVoList.add(unitQuantityCapacityVo);  
 }  
 }  
 return unitQuantityCapacityVoList;  
 }  
  
 @Override  
 public List<UnitQuantityCapacityVo> selectareaast(String limite) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> zoneList=asmModelDeviceDataMapper.selectAsmModelDeviceDatazone(asmModelDeviceData);  
  
 asmModelDeviceData.setLimiteCapacity(limite);  
 asmModelDeviceData.setCompanyState(null);  
 *//大区数量* for (AsmModelDeviceData amd:zoneList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyZone(amd.getCompanyZone());  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 UnitQuantityCapacityVo unitQuantityCapacityVo=new UnitQuantityCapacityVo();  
 unitQuantityCapacityVo.setCompanyZone(amd.getCompanyZone());  
 unitQuantityCapacityVo.setNum(asmModelDeviceDataList.size());  
 unitQuantityCapacityVo.setAbility(ability);  
 unitQuantityCapacityVoList.add(unitQuantityCapacityVo);  
 }  
 }  
 return unitQuantityCapacityVoList;  
 }  
  
 @Override  
 public List<UnitQuantityCapacityVo> selectareaout(String approval) {  
 List<UnitQuantityCapacityVo> unitQuantityCapacityVoList=new ArrayList<>();  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 List<AsmModelDeviceData> zoneList=asmModelDeviceDataMapper.selectAsmModelDeviceDatazone(asmModelDeviceData);  
  
 asmModelDeviceData.setApprovalCompliance(approval);  
 asmModelDeviceData.setCompanyState(null);  
 *//大区数量* for (AsmModelDeviceData amd:zoneList) {  
 Double ability=0.0;  
 asmModelDeviceData.setCompanyZone(amd.getCompanyZone());  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 if (asmModelDeviceDataList.size()>0){  
 for (AsmModelDeviceData add:asmModelDeviceDataList) {  
 ability=ability+add.getRealScale();  
 }  
 UnitQuantityCapacityVo unitQuantityCapacityVo=new UnitQuantityCapacityVo();  
 unitQuantityCapacityVo.setCompanyZone(amd.getCompanyZone());  
 unitQuantityCapacityVo.setNum(asmModelDeviceDataList.size());  
 unitQuantityCapacityVo.setAbility(ability);  
 unitQuantityCapacityVoList.add(unitQuantityCapacityVo);  
 }  
 }  
 return unitQuantityCapacityVoList;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectdiseli(String EliminationCapacity) {  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 asmModelDeviceData.setEliminationCapacity(EliminationCapacity);  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 for (AsmModelDeviceData amd:asmModelDeviceDataList) {  
 StringBuffer a=new StringBuffer();  
 if ((amd.getRealScale()!=null&&amd.getRealScale()<200.0)){  
 a.append("实际建成常减压装置能力小于200万吨/年;");  
 }if(amd.getStatus().equals("1")){  
 a.append("装置状态是正常运行;");  
 }if(!amd.getCompanyAddress().contains("格尔木")){  
 a.append("装置位于青海格尔木;");  
 }if(!amd.getCompanyAddress().contains("泽普")){  
 a.append("装置位于新疆泽普;");  
 }  
 amd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);  
 return asmModelDeviceDataList1;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectdisast(String limite) {  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 asmModelDeviceData.setLimiteCapacity(limite);  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2005-01-01");  
 endTime = new SimpleDateFormat(format).parse("2011-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 for (AsmModelDeviceData amd:asmModelDeviceDataList) {  
 StringBuffer a=new StringBuffer();  
 if ((amd.getApprovalDate().before(endTime)&&amd.getApprovalDate().after(startTime)  
 &&amd.getApprovalScale()<800.0)){  
 a.append("2005-2011年期间，新建常减压装置小于800万吨/年;");  
 }if((amd.getApprovalDate().after(endTime)&&amd.getApprovalScale()<1000.0)){  
 a.append("2011年以后，新建常减压装置小于1000万吨/年;");  
 }  
 amd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);  
 return asmModelDeviceDataList1;  
 }  
  
 @Override  
 public List<AsmModelDeviceData> selectdisout(String approval) {  
 AsmModelDeviceData asmModelDeviceData=new AsmModelDeviceData();  
 LoginUser user = SecurityUtils.*getLoginUser*();  
 SysRole sysRole=sysRoleService.selectRoleById(user.getUserId());  
 if (sysRole.getRoleKey().equals("test")){  
 asmModelDeviceData.setRoleKey("test");  
 }else if (sysRole.getRoleKey().equals("ripp")||sysRole.getRoleKey().equals("factory")){  
 asmModelDeviceData.setRoleKey("common");  
 }  
 asmModelDeviceData.setApprovalCompliance(approval);  
 List<AsmModelDeviceData> asmModelDeviceDataList=asmModelDeviceDataMapper.selectAsmModelDeviceDatadeweight(asmModelDeviceData);  
 Date startTime=new Date();  
 Date endTime=new Date();  
 String format = "yyyy-MM-dd";  
 try {  
*// middleTime = new SimpleDateFormat(format).parse("2011-01");* startTime = new SimpleDateFormat(format).parse("2004-01-01");  
 endTime = new SimpleDateFormat(format).parse("2016-12-31");  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 for (AsmModelDeviceData amd:asmModelDeviceDataList) {  
 StringBuffer a=new StringBuffer();  
 if ((amd.getApprovalDate().before(startTime)&&amd.getIsClean().equals("0"))){  
 a.append("2004年以前新建及改扩建，“是否2000年清理整顿保留的小炼油厂”中填“否”;");  
 }if (!(amd.getApprovalDate().before(endTime)&&amd.getApprovalDate().after(startTime)  
 &&amd.getApprovalDepartment().contains("国"))&&amd.getIsProject().equals("1")){  
 a.append("2004-2016年期间，由“国家”审批/核准,备案审批内容是否列入规划”中为“是”;");  
 }if (amd.getApprovalDepartment().contains("省")  
 &&amd.getApprovalDepartment().contains("新疆")&&amd.getApprovalDepartment().contains("西藏")  
 &&amd.getApprovalDepartment().contains("广西")&&amd.getApprovalDepartment().contains("宁夏")  
 &&amd.getApprovalDepartment().contains("内蒙古")&&amd.getIsProject().equals("1")){  
 a.append("2004-2016年期间，由“省级”审批/核准,备案审批内容是否列入规划”中为“是”;");  
 }if (!(amd.getApprovalDate().after(endTime)&&amd.getApprovalDepartment().contains("省")  
 &&amd.getApprovalDepartment().contains("新疆")&&amd.getApprovalDepartment().contains("西藏")  
 &&amd.getApprovalDepartment().contains("广西")&&amd.getApprovalDepartment().contains("宁夏")  
 &&amd.getApprovalDepartment().contains("内蒙古"))){  
 a.append("2016年以后，由“省级”审批/核准之外;");  
 }  
 amd.setMessage(a.toString());  
 }  
 List<AsmModelDeviceData> asmModelDeviceDataList1 = this.amsdecryptlist(asmModelDeviceDataList);  
 return asmModelDeviceDataList1;  
 }  
  
 public AsmModelDeviceData amsdecrypt(AsmModelDeviceData asmModelDeviceData){  
 String key = sm4Demo.passwordSm4(pwd);  
 if (asmModelDeviceData.getCompanyName()!=null) {  
 asmModelDeviceData.setCompanyName(sm4Demo.decryptSm41(key,asmModelDeviceData.getCompanyName()));  
 }  
 if (asmModelDeviceData.getDeviceName()!=null) {  
 asmModelDeviceData.setDeviceName(sm4Demo.decryptSm41(key,asmModelDeviceData.getDeviceName()));  
 }  
 if (asmModelDeviceData.getApprovalDepartment()!=null) {  
 asmModelDeviceData.setApprovalDepartment(sm4Demo.decryptSm41(key,asmModelDeviceData.getApprovalDepartment()));  
 }  
 if (asmModelDeviceData.getApprovalDetails()!=null) {  
 asmModelDeviceData.setApprovalDetails(sm4Demo.decryptSm41(key,asmModelDeviceData.getApprovalDetails()));  
 }  
 return asmModelDeviceData;  
 }  
 public AsmModelDeviceData amsencrypt(AsmModelDeviceData asmModelDeviceData){  
 String key = sm4Demo.passwordSm4(pwd);  
 if (asmModelDeviceData.getCompanyName()!=null) {  
 asmModelDeviceData.setCompanyName(sm4Demo.encryptSm41(key,asmModelDeviceData.getCompanyName()));  
 }  
 if (asmModelDeviceData.getDeviceName()!=null) {  
 asmModelDeviceData.setDeviceName(sm4Demo.encryptSm41(key,asmModelDeviceData.getDeviceName()));  
 }  
 if (asmModelDeviceData.getApprovalDepartment()!=null) {  
 asmModelDeviceData.setApprovalDepartment(sm4Demo.encryptSm41(key,asmModelDeviceData.getApprovalDepartment()));  
 }  
 if (asmModelDeviceData.getApprovalDetails()!=null) {  
 asmModelDeviceData.setApprovalDetails(sm4Demo.encryptSm41(key,asmModelDeviceData.getApprovalDetails()));  
 }  
 return asmModelDeviceData;  
 }  
 public List<AsmModelDeviceData> amsdecryptlist (List<AsmModelDeviceData> list){  
 String key = sm4Demo.passwordSm4(pwd);  
 for (AsmModelDeviceData asmModelDeviceData : list) {  
 if (asmModelDeviceData.getCompanyName()!=null) {  
 asmModelDeviceData.setCompanyName(sm4Demo.decryptSm41(key,asmModelDeviceData.getCompanyName()));  
 }  
 if (asmModelDeviceData.getDeviceName()!=null) {  
 asmModelDeviceData.setDeviceName(sm4Demo.decryptSm41(key,asmModelDeviceData.getDeviceName()));  
 }  
 if (asmModelDeviceData.getApprovalDepartment()!=null) {  
 asmModelDeviceData.setApprovalDepartment(sm4Demo.decryptSm41(key,asmModelDeviceData.getApprovalDepartment()));  
 }  
 if (asmModelDeviceData.getApprovalDetails()!=null) {  
 asmModelDeviceData.setApprovalDetails(sm4Demo.decryptSm41(key,asmModelDeviceData.getApprovalDetails()));  
 }  
 }  
 return list;  
 }  
 private ApplianceInformation tb() {  
 ApplianceInformation applianceInformation1 = new ApplianceInformation();  
 Integer totalQuantity = 0;  
 Double totalCapacity = 0.0;  
 Integer twoHundred = 0;  
 Double twoHundredN = 0.0;  
 Integer twoHundredToFiveHundred = 0;  
 Double twoHundredToFiveHundredN = 0.0;  
 Integer fiveHundredToOneThousand = 0;  
 Double fiveHundredToOneThousandN = 0.0;  
 Integer moreTanAThousand = 0;  
 Double moreTanAThousandN = 0.0;  
 ArrayList<AsmModelDeviceData> applianceInformations = asmModelDeviceDataMapper.selectapplianceInformation();  
 for (AsmModelDeviceData applianceInformation : applianceInformations) {  
 totalQuantity++;  
 totalCapacity+= applianceInformation.getRealScale();  
 if(applianceInformation.getRealScale()<=200){  
 twoHundred++;  
 twoHundredN+=applianceInformation.getRealScale();  
 }  
 if(applianceInformation.getRealScale()>200 && applianceInformation.getRealScale()<500){  
 twoHundredToFiveHundred++;  
 twoHundredToFiveHundredN+=applianceInformation.getRealScale();  
 }  
 if(applianceInformation.getRealScale()>500 && applianceInformation.getRealScale()<1000){  
 fiveHundredToOneThousand++;  
 fiveHundredToOneThousandN+=applianceInformation.getRealScale();  
 }  
 if(applianceInformation.getRealScale()>1000){  
 moreTanAThousand++;  
 moreTanAThousandN+=applianceInformation.getRealScale();  
 }  
 }  
 BigDecimal bg1 = new BigDecimal(twoHundred);  
 BigDecimal bg2 = new BigDecimal(twoHundredToFiveHundred);  
 BigDecimal bg3 = new BigDecimal(fiveHundredToOneThousand);  
 BigDecimal bg4 = new BigDecimal(moreTanAThousand);  
 BigDecimal bg5 = new BigDecimal(totalQuantity);  
 applianceInformation1.setTotalQuantity(totalQuantity);  
 applianceInformation1.setTotalCapacity(totalCapacity);  
 applianceInformation1.setTwoHundred(twoHundred);  
 applianceInformation1.setTwoHundredN(twoHundredN);  
 applianceInformation1.setTwoHundredToFiveHundred(twoHundredToFiveHundred);  
 applianceInformation1.setTwoHundredToFiveHundredN(twoHundredToFiveHundredN);  
 applianceInformation1.setFiveHundredToOneThousand(fiveHundredToOneThousand);  
 applianceInformation1.setFiveHundredToOneThousandN(fiveHundredToOneThousandN);  
 applianceInformation1.setMoreTanAThousand(moreTanAThousand);  
 applianceInformation1.setMoreTanAThousandN(moreTanAThousandN);  
 applianceInformation1.setTwoHundredZ(bg1.divide(bg5,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setTwoHundredToFiveHundredZ(bg2.divide(bg5,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setFiveHundredToOneThousandZ(bg3.divide(bg5,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setMoreTanAThousandZ(bg4.divide(bg5,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 Double qtotalCapacity = 0.0;  
 Integer qtotalQuantity = 0;  
 Integer qtwoHundred = 0;  
 Integer qtwoHundredToFiveHundred = 0;  
 Integer qfiveHundredToOneThousand = 0;  
 Integer qmoreTanAThousand = 0;  
 Double qtwoHundredN = 0.0;  
 Double qtwoHundredToFiveHundredN = 0.0;  
 Double qfiveHundredToOneThousandN = 0.0;  
 Double qmoreTanAThousandN = 0.0;  
 ArrayList<AsmModelDeviceData> asmModelDeviceDataArrayList1 = asmModelDeviceDataMapper.selectQ();  
 for (AsmModelDeviceData asmModelDeviceData : asmModelDeviceDataArrayList1) {  
 qtotalQuantity++;  
 qtotalCapacity+= asmModelDeviceData.getqDeviceQuantity();  
 if(asmModelDeviceData.getqDeviceQuantity()<=200){  
 qtwoHundred++;  
 qtwoHundredN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>200 && asmModelDeviceData.getqDeviceQuantity()<500){  
 qtwoHundredToFiveHundred++;  
 qtwoHundredToFiveHundredN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>500 && asmModelDeviceData.getqDeviceQuantity()<1000){  
 qfiveHundredToOneThousand++;  
 qfiveHundredToOneThousandN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>1000){  
 qmoreTanAThousand++;  
 qmoreTanAThousandN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
  
 }  
 BigDecimal bg1s = new BigDecimal(qtwoHundred);  
 BigDecimal bg2s = new BigDecimal(qtwoHundredToFiveHundred);  
 BigDecimal bg3s = new BigDecimal(qfiveHundredToOneThousand);  
 BigDecimal bg4s = new BigDecimal(qmoreTanAThousand);  
 BigDecimal bg5s = new BigDecimal(qtotalQuantity);  
 applianceInformation1.setQtwoHundredN(qtwoHundredN);  
 applianceInformation1.setQtwoHundredToFiveHundredN(qtwoHundredToFiveHundredN);  
 applianceInformation1.setQfiveHundredToOneThousandN(qfiveHundredToOneThousandN);  
 applianceInformation1.setQmoreTanAThousandN(qmoreTanAThousandN);  
 applianceInformation1.setqDeviceCapacity(qtotalCapacity);  
 applianceInformation1.setqDeviceQuantity(qtotalQuantity);  
 applianceInformation1.setQtwoHundredZ(bg1s.divide(bg5s,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setQtwoHundredToFiveHundredZ(bg2s.divide(bg5s,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setQfiveHundredToOneThousandZ(bg3s.divide(bg5s,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setQtwoHundredZ(bg4s.divide(bg5s,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
  
 Double stotalCapacity = 0.0;  
 Integer stotalQuantity = 0;  
 Integer stwoHundred = 0;  
 Integer stwoHundredToFiveHundred = 0;  
 Integer sfiveHundredToOneThousand = 0;  
 Integer smoreTanAThousand = 0;  
 Double stwoHundredN = 0.0;  
 Double stwoHundredToFiveHundredN = 0.0;  
 Double sfiveHundredToOneThousandN = 0.0;  
 Double smoreTanAThousandN = 0.0;  
 ArrayList<AsmModelDeviceData> asmModelDeviceDataArrayList2 = asmModelDeviceDataMapper.selectS();  
 for (AsmModelDeviceData asmModelDeviceData : asmModelDeviceDataArrayList2) {  
 stotalQuantity++;  
 stotalCapacity+= asmModelDeviceData.getqDeviceQuantity();  
 if(asmModelDeviceData.getqDeviceQuantity()<=200){  
 stwoHundred++;  
 stwoHundredN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>200 && asmModelDeviceData.getqDeviceQuantity()<500){  
 stwoHundredToFiveHundred++;  
 stwoHundredToFiveHundredN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>500 && asmModelDeviceData.getqDeviceQuantity()<1000){  
 sfiveHundredToOneThousand++;  
 sfiveHundredToOneThousandN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
 if(asmModelDeviceData.getqDeviceQuantity()>1000){  
 smoreTanAThousand++;  
 smoreTanAThousandN+=asmModelDeviceData.getqDeviceQuantity();  
 }  
  
 }  
 BigDecimal bg1q = new BigDecimal(stwoHundred);  
 BigDecimal bg2q = new BigDecimal(stwoHundredToFiveHundred);  
 BigDecimal bg3q = new BigDecimal(sfiveHundredToOneThousand);  
 BigDecimal bg4q = new BigDecimal(smoreTanAThousand);  
 BigDecimal bg5q = new BigDecimal(stotalQuantity);  
 applianceInformation1.setStwoHundredN(stwoHundredN);  
 applianceInformation1.setStwoHundredToFiveHundredN(stwoHundredToFiveHundredN);  
 applianceInformation1.setSfiveHundredToOneThousandN(sfiveHundredToOneThousandN);  
 applianceInformation1.setSmoreTanAThousandN(smoreTanAThousandN);  
 applianceInformation1.setSDeviceCapacity(stotalCapacity);  
 applianceInformation1.setSDeviceQuantity(stotalQuantity);  
 applianceInformation1.setStwoHundredZ(bg1q.divide(bg5q,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setStwoHundredToFiveHundredZ(bg2q.divide(bg5q,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setSfiveHundredToOneThousandZ(bg3q.divide(bg5q,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
 applianceInformation1.setStwoHundredZ(bg4q.divide(bg5q,3,BigDecimal.*ROUND\_HALF\_UP*).doubleValue()\*100);  
  
 Integer gslz = 0;  
 Double gnlz= 0.0;  
 ArrayList<AsmModelDeviceData> asmModelDeviceDataArrayList = asmModelDeviceDataMapper.selectB();  
 for (AsmModelDeviceData asmModelDeviceData : asmModelDeviceDataArrayList) {  
 gslz+=asmModelDeviceData.getBshuliang();  
 gnlz+=asmModelDeviceData.getBnengli();  
 }  
 ArrayList<Quantitay> quantitays = new ArrayList<>();  
 for (AsmModelDeviceData asmModelDeviceData : asmModelDeviceDataArrayList) {  
 BigDecimal bigDecimal = new BigDecimal(asmModelDeviceData.getBshuliang());  
 BigDecimal bigDecimal1 = new BigDecimal(asmModelDeviceData.getBnengli());  
 BigDecimal bigDecimal2 = new BigDecimal(gslz);  
 BigDecimal bigDecimal3 = new BigDecimal(gnlz);  
 Quantitay quantitay = new Quantitay();  
 quantitay.setgName(asmModelDeviceData.getOwnership());  
 quantitay.setgNL(asmModelDeviceData.getBnengli());  
 quantitay.setgSl(asmModelDeviceData.getBshuliang());  
 double v = bigDecimal.divide(bigDecimal2, 3, BigDecimal.*ROUND\_HALF\_UP*).doubleValue() \* 100;  
 quantitay.setgSlb(v);  
 double v1 = bigDecimal1.divide(bigDecimal3, 3, BigDecimal.*ROUND\_HALF\_UP*).doubleValue() \* 100;  
 quantitay.setgNlb(v1);  
 quantitay.setgSlZ(gslz);  
 quantitay.setgNlZ(gnlz);  
 quantitays.add(quantitay);  
 }  
 applianceInformation1.setQuantitays(quantitays);  
  
 ArrayList<AsmModelDeviceData> asmModelDeviceData1 = asmModelDeviceDataMapper.selectZz();  
 ArrayList<Provinces> provinces1 = new ArrayList<>();  
 for (AsmModelDeviceData modelDeviceData : asmModelDeviceData1) {  
 Provinces provinces = new Provinces();  
 AsmModelDeviceData asmModelDeviceData2 = asmModelDeviceDataMapper.selectZ(modelDeviceData.getStatus());  
 if("1".equals(modelDeviceData.getStatus())){  
 provinces.setZName("正常运行");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("2".equals(modelDeviceData.getStatus())){  
 provinces.setZName("已改扩建");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("3".equals(modelDeviceData.getStatus())){  
 provinces.setZName("停工");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("4".equals(modelDeviceData.getStatus())){  
 provinces.setZName("已拆除");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("5".equals(modelDeviceData.getStatus())){  
 provinces.setZName("在建");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("6".equals(modelDeviceData.getStatus())){  
 provinces.setZName("未投产");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("7".equals(modelDeviceData.getStatus())){  
 provinces.setZName("拟建");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 if("8".equals(modelDeviceData.getStatus())){  
 provinces.setZName("停产");  
 provinces.setZTotal(asmModelDeviceData2.getBshuliang());  
 provinces.setZNli(asmModelDeviceData2.getBnengli());  
 }  
 provinces1.add(provinces);  
 }  
 applianceInformation1.setProvinces(provinces1);  
 return applianceInformation1;  
 }  
}

### 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="参数名称" prop="configName">

<el-input

v-model="queryParams.configName"

placeholder="请输入参数名称"

clearable

style="width: 240px"

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="参数键名" prop="configKey">

<el-input

v-model="queryParams.configKey"

placeholder="请输入参数键名"

clearable

style="width: 240px"

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="系统内置" prop="configType">

<el-select v-model="queryParams.configType" placeholder="系统内置" clearable>

<el-option

v-for="dict in dict.type.sys\_yes\_no"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="创建时间">

<el-date-picker

v-model="dateRange"

style="width: 240px"

value-format="yyyy-MM-dd"

type="daterange"

range-separator="-"

start-placeholder="开始日期"

end-placeholder="结束日期"

></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 :gutter="10" class="mb8">

<el-col :span="1.5">

<el-button

type="primary"

plain

icon="el-icon-plus"

size="mini"

@click="handleAdd"

v-hasPermi="['system:config:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="success"

plain

icon="el-icon-edit"

size="mini"

:disabled="single"

@click="handleUpdate"

v-hasPermi="['system:config:edit']"

>修改</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-delete"

size="mini"

:disabled="multiple"

@click="handleDelete"

v-hasPermi="['system:config:remove']"

>删除</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="warning"

plain

icon="el-icon-download"

size="mini"

@click="handleExport"

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

>导出</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-refresh"

size="mini"

@click="handleRefreshCache"

v-hasPermi="['system:config:remove']"

>刷新缓存</el-button>

</el-col>

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

</el-row>

<el-table v-loading="loading" :data="configList" @selection-change="handleSelectionChange">

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

<el-table-column label="参数主键" align="center" prop="configId" />

<el-table-column label="参数名称" align="center" prop="configName" :show-overflow-tooltip="true" />

<el-table-column label="参数键名" align="center" prop="configKey" :show-overflow-tooltip="true" />

<el-table-column label="参数键值" align="center" prop="configValue" />

<el-table-column label="系统内置" align="center" prop="configType">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_yes\_no" :value="scope.row.configType"/>

</template>

</el-table-column>

<el-table-column label="备注" align="center" prop="remark" :show-overflow-tooltip="true" />

<el-table-column label="创建时间" align="center" prop="createTime" width="180">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:config:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:config:remove']"

>删除</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"

/>

<!-- 添加或修改参数配置对话框 -->

<el-dialog :title="title" :visible.sync="open" width="500px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="80px">

<el-form-item label="参数名称" prop="configName">

<el-input v-model="form.configName" placeholder="请输入参数名称" />

</el-form-item>

<el-form-item label="参数键名" prop="configKey">

<el-input v-model="form.configKey" placeholder="请输入参数键名" />

</el-form-item>

<el-form-item label="参数键值" prop="configValue">

<el-input v-model="form.configValue" placeholder="请输入参数键值" />

</el-form-item>

<el-form-item label="系统内置" prop="configType">

<el-radio-group v-model="form.configType">

<el-radio

v-for="dict in dict.type.sys\_yes\_no"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

<el-form-item label="备注" prop="remark">

<el-input v-model="form.remark" type="textarea" placeholder="请输入内容" />

</el-form-item>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listConfig, getConfig, delConfig, addConfig, updateConfig, refreshCache } from "@/api/system/config";

export default {

name: "Config",

dicts: ['sys\_yes\_no'],

data() {

return {

// 遮罩层

loading: 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,

// 参数表格数据

configList: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 日期范围

dateRange: [],

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

configName: undefined,

configKey: undefined,

configType: undefined

},

// 表单参数

form: {},

// 表单校验

rules: {

configName: [

{ required: true, message: "参数名称不能为空", trigger: "blur" }

],

configKey: [

{ required: true, message: "参数键名不能为空", trigger: "blur" }

],

configValue: [

{ required: true, message: "参数键值不能为空", trigger: "blur" }

]

}

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询参数列表 \*/

getList() {

this.loading = true;

listConfig(this.addDateRange(this.queryParams, this.dateRange)).then(response => {

this.configList = response.rows;

this.total = response.total;

this.loading = false;

}

);

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

configId: undefined,

configName: undefined,

configKey: undefined,

configValue: undefined,

configType: "Y",

remark: undefined

};

this.resetForm("form");

},

/\*\* 搜索按钮操作 \*/

handleQuery() {

this.queryParams.pageNum = 1;

this.getList();

},

/\*\* 重置按钮操作 \*/

resetQuery() {

this.dateRange = [];

this.resetForm("queryForm");

this.handleQuery();

},

/\*\* 新增按钮操作 \*/

handleAdd() {

this.reset();

this.open = true;

this.title = "添加参数";

},

// 多选框选中数据

handleSelectionChange(selection) {

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

this.single = selection.length!=1

this.multiple = !selection.length

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

const configId = row.configId || this.ids

getConfig(configId).then(response => {

this.form = response.data;

this.open = true;

this.title = "修改参数";

});

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.configId != undefined) {

updateConfig(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

addConfig(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

const configIds = row.configId || this.ids;

this.$modal.confirm('是否确认删除参数编号为"' + configIds + '"的数据项？').then(function() {

return delConfig(configIds);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

},

/\*\* 导出按钮操作 \*/

handleExport() {

this.download('system/config/export', {

...this.queryParams

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

},

/\*\* 刷新缓存按钮操作 \*/

handleRefreshCache() {

refreshCache().then(() => {

this.$modal.msgSuccess("刷新成功");

});

}

}

};

</script>

<template>

<div class="app-container">

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

<el-form-item label="部门名称" prop="deptName">

<el-input

v-model="queryParams.deptName"

placeholder="请输入部门名称"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="状态" prop="status">

<el-select v-model="queryParams.status" placeholder="部门状态" clearable>

<el-option

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</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"

v-hasPermi="['system:dept:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="info"

plain

icon="el-icon-sort"

size="mini"

@click="toggleExpandAll"

>展开/折叠</el-button>

</el-col>

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

</el-row>

<el-table

v-if="refreshTable"

v-loading="loading"

:data="deptList"

row-key="deptId"

:default-expand-all="isExpandAll"

:tree-props="{children: 'children', hasChildren: 'hasChildren'}"

>

<el-table-column prop="deptName" label="部门名称" width="260"></el-table-column>

<el-table-column prop="orderNum" label="排序" width="200"></el-table-column>

<el-table-column prop="status" label="状态" width="100">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_normal\_disable" :value="scope.row.status"/>

</template>

</el-table-column>

<el-table-column label="创建时间" align="center" prop="createTime" width="200">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:dept:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-plus"

@click="handleAdd(scope.row)"

v-hasPermi="['system:dept:add']"

>新增</el-button>

<el-button

v-if="scope.row.parentId != 0"

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:dept:remove']"

>删除</el-button>

</template>

</el-table-column>

</el-table>

<!-- 添加或修改部门对话框 -->

<el-dialog :title="title" :visible.sync="open" width="600px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="80px">

<el-row>

<el-col :span="24" v-if="form.parentId !== 0">

<el-form-item label="上级部门" prop="parentId">

<treeselect v-model="form.parentId" :options="deptOptions" :normalizer="normalizer" placeholder="选择上级部门" />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="12">

<el-form-item label="部门名称" prop="deptName">

<el-input v-model="form.deptName" placeholder="请输入部门名称" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序" prop="orderNum">

<el-input-number v-model="form.orderNum" controls-position="right" :min="0" />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="12">

<el-form-item label="负责人" prop="leader">

<el-input v-model="form.leader" placeholder="请输入负责人" maxlength="20" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="联系电话" prop="phone">

<el-input v-model="form.phone" placeholder="请输入联系电话" maxlength="11" />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="12">

<el-form-item label="邮箱" prop="email">

<el-input v-model="form.email" placeholder="请输入邮箱" maxlength="50" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="部门状态">

<el-radio-group v-model="form.status">

<el-radio

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listDept, getDept, delDept, addDept, updateDept, listDeptExcludeChild } from "@/api/system/dept";

import Treeselect from "@riophae/vue-treeselect";

import "@riophae/vue-treeselect/dist/vue-treeselect.css";

export default {

name: "Dept",

dicts: ['sys\_normal\_disable'],

components: { Treeselect },

data() {

return {

// 遮罩层

loading: true,

// 显示搜索条件

showSearch: true,

// 表格树数据

deptList: [],

// 部门树选项

deptOptions: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 是否展开，默认全部展开

isExpandAll: true,

// 重新渲染表格状态

refreshTable: true,

// 查询参数

queryParams: {

deptName: undefined,

status: undefined

},

// 表单参数

form: {},

// 表单校验

rules: {

parentId: [

{ required: true, message: "上级部门不能为空", trigger: "blur" }

],

deptName: [

{ required: true, message: "部门名称不能为空", trigger: "blur" }

],

orderNum: [

{ required: true, message: "显示排序不能为空", trigger: "blur" }

],

email: [

{

type: "email",

message: "请输入正确的邮箱地址",

trigger: ["blur", "change"]

}

],

phone: [

{

pattern: /^1[3|4|5|6|7|8|9][0-9]\d{8}$/,

message: "请输入正确的手机号码",

trigger: "blur"

}

]

}

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询部门列表 \*/

getList() {

this.loading = true;

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

this.deptList = this.handleTree(response.data, "deptId");

this.loading = false;

});

},

/\*\* 转换部门数据结构 \*/

normalizer(node) {

if (node.children && !node.children.length) {

delete node.children;

}

return {

id: node.deptId,

label: node.deptName,

children: node.children

};

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

deptId: undefined,

parentId: undefined,

deptName: undefined,

orderNum: undefined,

leader: undefined,

phone: undefined,

email: undefined,

status: "0"

};

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(row) {

this.reset();

if (row != undefined) {

this.form.parentId = row.deptId;

}

this.open = true;

this.title = "添加部门";

listDept().then(response => {

this.deptOptions = this.handleTree(response.data, "deptId");

});

},

/\*\* 展开/折叠操作 \*/

toggleExpandAll() {

this.refreshTable = false;

this.isExpandAll = !this.isExpandAll;

this.$nextTick(() => {

this.refreshTable = true;

});

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

getDept(row.deptId).then(response => {

this.form = response.data;

this.open = true;

this.title = "修改部门";

});

listDeptExcludeChild(row.deptId).then(response => {

this.deptOptions = this.handleTree(response.data, "deptId");

});

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.deptId != undefined) {

updateDept(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

addDept(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

this.$modal.confirm('是否确认删除名称为"' + row.deptName + '"的数据项？').then(function() {

return delDept(row.deptId);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

}

}

};

</script>

<template>

<div class="app-container">

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

<el-form-item label="菜单名称" prop="menuName">

<el-input

v-model="queryParams.menuName"

placeholder="请输入菜单名称"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="状态" prop="status">

<el-select v-model="queryParams.status" placeholder="菜单状态" clearable>

<el-option

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</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"

v-hasPermi="['system:menu:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="info"

plain

icon="el-icon-sort"

size="mini"

@click="toggleExpandAll"

>展开/折叠</el-button>

</el-col>

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

</el-row>

<el-table

v-if="refreshTable"

v-loading="loading"

:data="menuList"

row-key="menuId"

:default-expand-all="isExpandAll"

:tree-props="{children: 'children', hasChildren: 'hasChildren'}"

>

<el-table-column prop="menuName" label="菜单名称" :show-overflow-tooltip="true" width="160"></el-table-column>

<el-table-column prop="icon" label="图标" align="center" width="100">

<template slot-scope="scope">

<svg-icon :icon-class="scope.row.icon" />

</template>

</el-table-column>

<el-table-column prop="orderNum" label="排序" width="60"></el-table-column>

<el-table-column prop="perms" label="权限标识" :show-overflow-tooltip="true"></el-table-column>

<el-table-column prop="component" label="组件路径" :show-overflow-tooltip="true"></el-table-column>

<el-table-column prop="status" label="状态" width="80">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_normal\_disable" :value="scope.row.status"/>

</template>

</el-table-column>

<el-table-column label="创建时间" align="center" prop="createTime">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:menu:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-plus"

@click="handleAdd(scope.row)"

v-hasPermi="['system:menu:add']"

>新增</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:menu:remove']"

>删除</el-button>

</template>

</el-table-column>

</el-table>

<!-- 添加或修改菜单对话框 -->

<el-dialog :title="title" :visible.sync="open" width="680px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="100px">

<el-row>

<el-col :span="24">

<el-form-item label="上级菜单">

<treeselect

v-model="form.parentId"

:options="menuOptions"

:normalizer="normalizer"

:show-count="true"

placeholder="选择上级菜单"

/>

</el-form-item>

</el-col>

<el-col :span="24">

<el-form-item label="菜单类型" prop="menuType">

<el-radio-group v-model="form.menuType">

<el-radio label="M">目录</el-radio>

<el-radio label="C">菜单</el-radio>

<el-radio label="F">按钮</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="24" v-if="form.menuType != 'F'">

<el-form-item label="菜单图标" prop="icon">

<el-popover

placement="bottom-start"

width="460"

trigger="click"

@show="$refs['iconSelect'].reset()"

>

<IconSelect ref="iconSelect" @selected="selected" />

<el-input slot="reference" v-model="form.icon" placeholder="点击选择图标" readonly>

<svg-icon

v-if="form.icon"

slot="prefix"

:icon-class="form.icon"

class="el-input\_\_icon"

style="height: 32px;width: 16px;"

/>

<i v-else slot="prefix" class="el-icon-search el-input\_\_icon" />

</el-input>

</el-popover>

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="菜单名称" prop="menuName">

<el-input v-model="form.menuName" placeholder="请输入菜单名称" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序" prop="orderNum">

<el-input-number v-model="form.orderNum" controls-position="right" :min="0" />

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType != 'F'">

<el-form-item>

<span slot="label">

<el-tooltip content="选择是外链则路由地址需要以`http(s)://`开头" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

是否外链

</span>

<el-radio-group v-model="form.isFrame">

<el-radio label="0">是</el-radio>

<el-radio label="1">否</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType != 'F'">

<el-form-item prop="path">

<span slot="label">

<el-tooltip content="访问的路由地址，如：`user`，如外网地址需内链访问则以`http(s)://`开头" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

路由地址

</span>

<el-input v-model="form.path" placeholder="请输入路由地址" />

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType == 'C'">

<el-form-item prop="component">

<span slot="label">

<el-tooltip content="访问的组件路径，如：`system/user/index`，默认在`views`目录下" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

组件路径

</span>

<el-input v-model="form.component" placeholder="请输入组件路径" />

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType != 'M'">

<el-form-item>

<el-input v-model="form.perms" placeholder="请输入权限标识" maxlength="100" />

<span slot="label">

<el-tooltip content="控制器中定义的权限字符，如：@PreAuthorize(`@ss.hasPermi('system:user:list')`)" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

权限字符

</span>

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType == 'C'">

<el-form-item>

<el-input v-model="form.query" placeholder="请输入路由参数" maxlength="255" />

<span slot="label">

<el-tooltip content='访问路由的默认传递参数，如：`{"id": 1, "name": "ry"}`' placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

路由参数

</span>

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType == 'C'">

<el-form-item>

<span slot="label">

<el-tooltip content="选择是则会被`keep-alive`缓存，需要匹配组件的`name`和地址保持一致" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

是否缓存

</span>

<el-radio-group v-model="form.isCache">

<el-radio label="0">缓存</el-radio>

<el-radio label="1">不缓存</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType != 'F'">

<el-form-item>

<span slot="label">

<el-tooltip content="选择隐藏则路由将不会出现在侧边栏，但仍然可以访问" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

显示状态

</span>

<el-radio-group v-model="form.visible">

<el-radio

v-for="dict in dict.type.sys\_show\_hide"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12" v-if="form.menuType != 'F'">

<el-form-item>

<span slot="label">

<el-tooltip content="选择停用则路由将不会出现在侧边栏，也不能被访问" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

菜单状态

</span>

<el-radio-group v-model="form.status">

<el-radio

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listMenu, getMenu, delMenu, addMenu, updateMenu } from "@/api/system/menu";

import Treeselect from "@riophae/vue-treeselect";

import "@riophae/vue-treeselect/dist/vue-treeselect.css";

import IconSelect from "@/components/IconSelect";

export default {

name: "Menu",

dicts: ['sys\_show\_hide', 'sys\_normal\_disable'],

components: { Treeselect, IconSelect },

data() {

return {

// 遮罩层

loading: true,

// 显示搜索条件

showSearch: true,

// 菜单表格树数据

menuList: [],

// 菜单树选项

menuOptions: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 是否展开，默认全部折叠

isExpandAll: false,

// 重新渲染表格状态

refreshTable: true,

// 查询参数

queryParams: {

menuName: undefined,

visible: undefined

},

// 表单参数

form: {},

// 表单校验

rules: {

menuName: [

{ required: true, message: "菜单名称不能为空", trigger: "blur" }

],

orderNum: [

{ required: true, message: "菜单顺序不能为空", trigger: "blur" }

],

path: [

{ required: true, message: "路由地址不能为空", trigger: "blur" }

]

}

};

},

created() {

this.getList();

},

methods: {

// 选择图标

selected(name) {

this.form.icon = name;

},

/\*\* 查询菜单列表 \*/

getList() {

this.loading = true;

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

this.menuList = this.handleTree(response.data, "menuId");

this.loading = false;

});

},

/\*\* 转换菜单数据结构 \*/

normalizer(node) {

if (node.children && !node.children.length) {

delete node.children;

}

return {

id: node.menuId,

label: node.menuName,

children: node.children

};

},

/\*\* 查询菜单下拉树结构 \*/

getTreeselect() {

listMenu().then(response => {

this.menuOptions = [];

const menu = { menuId: 0, menuName: '主类目', children: [] };

menu.children = this.handleTree(response.data, "menuId");

this.menuOptions.push(menu);

});

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

menuId: undefined,

parentId: 0,

menuName: undefined,

icon: undefined,

menuType: "M",

orderNum: undefined,

isFrame: "1",

isCache: "0",

visible: "0",

status: "0"

};

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(row) {

this.reset();

this.getTreeselect();

if (row != null && row.menuId) {

this.form.parentId = row.menuId;

} else {

this.form.parentId = 0;

}

this.open = true;

this.title = "添加菜单";

},

/\*\* 展开/折叠操作 \*/

toggleExpandAll() {

this.refreshTable = false;

this.isExpandAll = !this.isExpandAll;

this.$nextTick(() => {

this.refreshTable = true;

});

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

this.getTreeselect();

getMenu(row.menuId).then(response => {

this.form = response.data;

this.open = true;

this.title = "修改菜单";

});

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.menuId != undefined) {

updateMenu(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

addMenu(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

this.$modal.confirm('是否确认删除名称为"' + row.menuName + '"的数据项？').then(function() {

return delMenu(row.menuId);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

}

}

};

</script>

<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="公告标题" prop="noticeTitle">

<el-input

v-model="queryParams.noticeTitle"

placeholder="请输入公告标题"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="操作人员" prop="createBy">

<el-input

v-model="queryParams.createBy"

placeholder="请输入操作人员"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="类型" prop="noticeType">

<el-select v-model="queryParams.noticeType" placeholder="公告类型" clearable>

<el-option

v-for="dict in dict.type.sys\_notice\_type"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</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"

v-hasPermi="['system:notice:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="success"

plain

icon="el-icon-edit"

size="mini"

:disabled="single"

@click="handleUpdate"

v-hasPermi="['system:notice:edit']"

>修改</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-delete"

size="mini"

:disabled="multiple"

@click="handleDelete"

v-hasPermi="['system:notice:remove']"

>删除</el-button>

</el-col>

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

</el-row>

<el-table v-loading="loading" :data="noticeList" @selection-change="handleSelectionChange">

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

<el-table-column label="序号" align="center" prop="noticeId" width="100" />

<el-table-column

label="公告标题"

align="center"

prop="noticeTitle"

:show-overflow-tooltip="true"

/>

<el-table-column label="公告类型" align="center" prop="noticeType" width="100">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_notice\_type" :value="scope.row.noticeType"/>

</template>

</el-table-column>

<el-table-column label="状态" align="center" prop="status" width="100">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_notice\_status" :value="scope.row.status"/>

</template>

</el-table-column>

<el-table-column label="创建者" align="center" prop="createBy" width="100" />

<el-table-column label="创建时间" align="center" prop="createTime" width="100">

<template slot-scope="scope">

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

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:notice:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:notice:remove']"

>删除</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"

/>

<!-- 添加或修改公告对话框 -->

<el-dialog :title="title" :visible.sync="open" width="780px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="80px">

<el-row>

<el-col :span="12">

<el-form-item label="公告标题" prop="noticeTitle">

<el-input v-model="form.noticeTitle" placeholder="请输入公告标题" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="公告类型" prop="noticeType">

<el-select v-model="form.noticeType" placeholder="请选择公告类型">

<el-option

v-for="dict in dict.type.sys\_notice\_type"

:key="dict.value"

:label="dict.label"

:value="dict.value"

></el-option>

</el-select>

</el-form-item>

</el-col>

<el-col :span="24">

<el-form-item label="状态">

<el-radio-group v-model="form.status">

<el-radio

v-for="dict in dict.type.sys\_notice\_status"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="24">

<el-form-item label="内容">

<editor v-model="form.noticeContent" :min-height="192"/>

</el-form-item>

</el-col>

</el-row>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listNotice, getNotice, delNotice, addNotice, updateNotice } from "@/api/system/notice";

export default {

name: "Notice",

dicts: ['sys\_notice\_status', 'sys\_notice\_type'],

data() {

return {

// 遮罩层

loading: 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,

// 公告表格数据

noticeList: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

noticeTitle: undefined,

createBy: undefined,

status: undefined

},

// 表单参数

form: {},

// 表单校验

rules: {

noticeTitle: [

{ required: true, message: "公告标题不能为空", trigger: "blur" }

],

noticeType: [

{ required: true, message: "公告类型不能为空", trigger: "change" }

]

}

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询公告列表 \*/

getList() {

this.loading = true;

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

this.noticeList = response.rows;

this.total = response.total;

this.loading = false;

});

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

noticeId: undefined,

noticeTitle: undefined,

noticeType: undefined,

noticeContent: undefined,

status: "0"

};

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();

},

// 多选框选中数据

handleSelectionChange(selection) {

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

this.single = selection.length!=1

this.multiple = !selection.length

},

/\*\* 新增按钮操作 \*/

handleAdd() {

this.reset();

this.open = true;

this.title = "添加公告";

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

const noticeId = row.noticeId || this.ids

getNotice(noticeId).then(response => {

this.form = response.data;

this.open = true;

this.title = "修改公告";

});

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.noticeId != undefined) {

updateNotice(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

addNotice(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

const noticeIds = row.noticeId || this.ids

this.$modal.confirm('是否确认删除公告编号为"' + noticeIds + '"的数据项？').then(function() {

return delNotice(noticeIds);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

}

}

};

</script>

<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="岗位编码" prop="postCode">

<el-input

v-model="queryParams.postCode"

placeholder="请输入岗位编码"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="岗位名称" prop="postName">

<el-input

v-model="queryParams.postName"

placeholder="请输入岗位名称"

clearable

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="状态" prop="status">

<el-select v-model="queryParams.status" placeholder="岗位状态" clearable>

<el-option

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</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"

v-hasPermi="['system:post:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="success"

plain

icon="el-icon-edit"

size="mini"

:disabled="single"

@click="handleUpdate"

v-hasPermi="['system:post:edit']"

>修改</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-delete"

size="mini"

:disabled="multiple"

@click="handleDelete"

v-hasPermi="['system:post:remove']"

>删除</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="warning"

plain

icon="el-icon-download"

size="mini"

@click="handleExport"

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

>导出</el-button>

</el-col>

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

</el-row>

<el-table v-loading="loading" :data="postList" @selection-change="handleSelectionChange">

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

<el-table-column label="岗位编号" align="center" prop="postId" />

<el-table-column label="岗位编码" align="center" prop="postCode" />

<el-table-column label="岗位名称" align="center" prop="postName" />

<el-table-column label="岗位排序" align="center" prop="postSort" />

<el-table-column label="状态" align="center" prop="status">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_normal\_disable" :value="scope.row.status"/>

</template>

</el-table-column>

<el-table-column label="创建时间" align="center" prop="createTime" width="180">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:post:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:post:remove']"

>删除</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"

/>

<!-- 添加或修改岗位对话框 -->

<el-dialog :title="title" :visible.sync="open" width="500px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="80px">

<el-form-item label="岗位名称" prop="postName">

<el-input v-model="form.postName" placeholder="请输入岗位名称" />

</el-form-item>

<el-form-item label="岗位编码" prop="postCode">

<el-input v-model="form.postCode" placeholder="请输入编码名称" />

</el-form-item>

<el-form-item label="岗位顺序" prop="postSort">

<el-input-number v-model="form.postSort" controls-position="right" :min="0" />

</el-form-item>

<el-form-item label="岗位状态" prop="status">

<el-radio-group v-model="form.status">

<el-radio

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

<el-form-item label="备注" prop="remark">

<el-input v-model="form.remark" type="textarea" placeholder="请输入内容" />

</el-form-item>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listPost, getPost, delPost, addPost, updatePost } from "@/api/system/post";

export default {

name: "Post",

dicts: ['sys\_normal\_disable'],

data() {

return {

// 遮罩层

loading: 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,

// 岗位表格数据

postList: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

postCode: undefined,

postName: undefined,

status: undefined

},

// 表单参数

form: {},

// 表单校验

rules: {

postName: [

{ required: true, message: "岗位名称不能为空", trigger: "blur" }

],

postCode: [

{ required: true, message: "岗位编码不能为空", trigger: "blur" }

],

postSort: [

{ required: true, message: "岗位顺序不能为空", trigger: "blur" }

]

}

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询岗位列表 \*/

getList() {

this.loading = true;

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

this.postList = response.rows;

this.total = response.total;

this.loading = false;

});

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 表单重置

reset() {

this.form = {

postId: undefined,

postCode: undefined,

postName: undefined,

postSort: 0,

status: "0",

remark: undefined

};

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();

},

// 多选框选中数据

handleSelectionChange(selection) {

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

this.single = selection.length!=1

this.multiple = !selection.length

},

/\*\* 新增按钮操作 \*/

handleAdd() {

this.reset();

this.open = true;

this.title = "添加岗位";

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

const postId = row.postId || this.ids

getPost(postId).then(response => {

this.form = response.data;

this.open = true;

this.title = "修改岗位";

});

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.postId != undefined) {

updatePost(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

addPost(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

const postIds = row.postId || this.ids;

this.$modal.confirm('是否确认删除岗位编号为"' + postIds + '"的数据项？').then(function() {

return delPost(postIds);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

},

/\*\* 导出按钮操作 \*/

handleExport() {

this.download('system/post/export', {

...this.queryParams

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

}

}

};

</script>

<template>

<div class="app-container">

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

<el-form-item label="用户名称" prop="userName">

<el-input

v-model="queryParams.userName"

placeholder="请输入用户名称"

clearable

style="width: 240px"

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="手机号码" prop="phonenumber">

<el-input

v-model="queryParams.phonenumber"

placeholder="请输入手机号码"

clearable

style="width: 240px"

@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="openSelectUser"

v-hasPermi="['system:role:add']"

>添加用户</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-circle-close"

size="mini"

:disabled="multiple"

@click="cancelAuthUserAll"

v-hasPermi="['system:role:remove']"

>批量取消授权</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="warning"

plain

icon="el-icon-close"

size="mini"

@click="handleClose"

>关闭</el-button>

</el-col>

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

</el-row>

<el-table v-loading="loading" :data="userList" @selection-change="handleSelectionChange">

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

<el-table-column label="用户名称" prop="userName" :show-overflow-tooltip="true" />

<el-table-column label="用户昵称" prop="nickName" :show-overflow-tooltip="true" />

<el-table-column label="邮箱" prop="email" :show-overflow-tooltip="true" />

<el-table-column label="手机" prop="phonenumber" :show-overflow-tooltip="true" />

<el-table-column label="状态" align="center" prop="status">

<template slot-scope="scope">

<dict-tag :options="dict.type.sys\_normal\_disable" :value="scope.row.status"/>

</template>

</el-table-column>

<el-table-column label="创建时间" align="center" prop="createTime" width="180">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope">

<el-button

size="mini"

type="text"

icon="el-icon-circle-close"

@click="cancelAuthUser(scope.row)"

v-hasPermi="['system:role:remove']"

>取消授权</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"

/>

<select-user ref="select" :roleId="queryParams.roleId" @ok="handleQuery" />

</div>

</template>

<script>

import { allocatedUserList, authUserCancel, authUserCancelAll } from "@/api/system/role";

import selectUser from "./selectUser";

export default {

name: "AuthUser",

dicts: ['sys\_normal\_disable'],

components: { selectUser },

data() {

return {

// 遮罩层

loading: true,

// 选中用户组

userIds: [],

// 非多个禁用

multiple: true,

// 显示搜索条件

showSearch: true,

// 总条数

total: 0,

// 用户表格数据

userList: [],

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

roleId: undefined,

userName: undefined,

phonenumber: undefined

}

};

},

created() {

const roleId = this.$route.params && this.$route.params.roleId;

if (roleId) {

this.queryParams.roleId = roleId;

this.getList();

}

},

methods: {

/\*\* 查询授权用户列表 \*/

getList() {

this.loading = true;

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

this.userList = response.rows;

this.total = response.total;

this.loading = false;

}

);

},

// 返回按钮

handleClose() {

const obj = { path: "/system/role" };

this.$tab.closeOpenPage(obj);

},

/\*\* 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();

},

// 多选框选中数据

handleSelectionChange(selection) {

this.userIds = selection.map(item => item.userId)

this.multiple = !selection.length

},

/\*\* 打开授权用户表弹窗 \*/

openSelectUser() {

this.$refs.select.show();

},

/\*\* 取消授权按钮操作 \*/

cancelAuthUser(row) {

const roleId = this.queryParams.roleId;

this.$modal.confirm('确认要取消该用户"' + row.userName + '"角色吗？').then(function() {

return authUserCancel({ userId: row.userId, roleId: roleId });

}).then(() => {

this.getList();

this.$modal.msgSuccess("取消授权成功");

}).catch(() => {});

},

/\*\* 批量取消授权按钮操作 \*/

cancelAuthUserAll(row) {

const roleId = this.queryParams.roleId;

const userIds = this.userIds.join(",");

this.$modal.confirm('是否取消选中用户授权数据项？').then(function() {

return authUserCancelAll({ roleId: roleId, userIds: userIds });

}).then(() => {

this.getList();

this.$modal.msgSuccess("取消授权成功");

}).catch(() => {});

}

}

};

</script>

<template>

<div class="app-container">

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

<el-form-item label="角色名称" prop="roleName">

<el-input

v-model="queryParams.roleName"

placeholder="请输入角色名称"

clearable

style="width: 240px"

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="权限字符" prop="roleKey">

<el-input

v-model="queryParams.roleKey"

placeholder="请输入权限字符"

clearable

style="width: 240px"

@keyup.enter.native="handleQuery"

/>

</el-form-item>

<el-form-item label="状态" prop="status">

<el-select

v-model="queryParams.status"

placeholder="角色状态"

clearable

style="width: 240px"

>

<el-option

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.label"

:value="dict.value"

/>

</el-select>

</el-form-item>

<el-form-item label="创建时间">

<el-date-picker

v-model="dateRange"

style="width: 240px"

value-format="yyyy-MM-dd"

type="daterange"

range-separator="-"

start-placeholder="开始日期"

end-placeholder="结束日期"

></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 :gutter="10" class="mb8">

<el-col :span="1.5">

<el-button

type="primary"

plain

icon="el-icon-plus"

size="mini"

@click="handleAdd"

v-hasPermi="['system:role:add']"

>新增</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="success"

plain

icon="el-icon-edit"

size="mini"

:disabled="single"

@click="handleUpdate"

v-hasPermi="['system:role:edit']"

>修改</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="danger"

plain

icon="el-icon-delete"

size="mini"

:disabled="multiple"

@click="handleDelete"

v-hasPermi="['system:role:remove']"

>删除</el-button>

</el-col>

<el-col :span="1.5">

<el-button

type="warning"

plain

icon="el-icon-download"

size="mini"

@click="handleExport"

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

>导出</el-button>

</el-col>

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

</el-row>

<el-table v-loading="loading" :data="roleList" @selection-change="handleSelectionChange">

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

<el-table-column label="角色编号" prop="roleId" width="120" />

<el-table-column label="角色名称" prop="roleName" :show-overflow-tooltip="true" width="150" />

<el-table-column label="权限字符" prop="roleKey" :show-overflow-tooltip="true" width="150" />

<el-table-column label="显示顺序" prop="roleSort" width="100" />

<el-table-column label="状态" align="center" width="100">

<template slot-scope="scope">

<el-switch

v-model="scope.row.status"

active-value="0"

inactive-value="1"

@change="handleStatusChange(scope.row)"

></el-switch>

</template>

</el-table-column>

<el-table-column label="创建时间" align="center" prop="createTime" width="180">

<template slot-scope="scope">

<span>{{ parseTime(scope.row.createTime) }}</span>

</template>

</el-table-column>

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

<template slot-scope="scope" v-if="scope.row.roleId !== 1">

<el-button

size="mini"

type="text"

icon="el-icon-edit"

@click="handleUpdate(scope.row)"

v-hasPermi="['system:role:edit']"

>修改</el-button>

<el-button

size="mini"

type="text"

icon="el-icon-delete"

@click="handleDelete(scope.row)"

v-hasPermi="['system:role:remove']"

>删除</el-button>

<el-dropdown size="mini" @command="(command) => handleCommand(command, scope.row)" v-hasPermi="['system:role:edit']">

<span class="el-dropdown-link">

<i class="el-icon-d-arrow-right el-icon--right"></i>更多

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item command="handleDataScope" icon="el-icon-circle-check"

v-hasPermi="['system:role:edit']">数据权限</el-dropdown-item>

<el-dropdown-item command="handleAuthUser" icon="el-icon-user"

v-hasPermi="['system:role:edit']">分配用户</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

<pagination

v-show="total>0"

:total="total"

:page.sync="queryParams.pageNum"

:limit.sync="queryParams.pageSize"

@pagination="getList"

/>

<!-- 添加或修改角色配置对话框 -->

<el-dialog :title="title" :visible.sync="open" width="500px" append-to-body>

<el-form ref="form" :model="form" :rules="rules" label-width="100px">

<el-form-item label="角色名称" prop="roleName">

<el-input v-model="form.roleName" placeholder="请输入角色名称" />

</el-form-item>

<el-form-item prop="roleKey">

<span slot="label">

<el-tooltip content="控制器中定义的权限字符，如：@PreAuthorize(`@ss.hasRole('admin')`)" placement="top">

<i class="el-icon-question"></i>

</el-tooltip>

权限字符

</span>

<el-input v-model="form.roleKey" placeholder="请输入权限字符" />

</el-form-item>

<el-form-item label="角色顺序" prop="roleSort">

<el-input-number v-model="form.roleSort" controls-position="right" :min="0" />

</el-form-item>

<el-form-item label="状态">

<el-radio-group v-model="form.status">

<el-radio

v-for="dict in dict.type.sys\_normal\_disable"

:key="dict.value"

:label="dict.value"

>{{dict.label}}</el-radio>

</el-radio-group>

</el-form-item>

<el-form-item label="菜单权限">

<el-checkbox v-model="menuExpand" @change="handleCheckedTreeExpand($event, 'menu')">展开/折叠</el-checkbox>

<el-checkbox v-model="menuNodeAll" @change="handleCheckedTreeNodeAll($event, 'menu')">全选/全不选</el-checkbox>

<el-checkbox v-model="form.menuCheckStrictly" @change="handleCheckedTreeConnect($event, 'menu')">父子联动</el-checkbox>

<el-tree

class="tree-border"

:data="menuOptions"

show-checkbox

ref="menu"

node-key="id"

:check-strictly="!form.menuCheckStrictly"

empty-text="加载中，请稍候"

:props="defaultProps"

></el-tree>

</el-form-item>

<el-form-item label="备注">

<el-input v-model="form.remark" type="textarea" placeholder="请输入内容"></el-input>

</el-form-item>

</el-form>

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

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

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

</div>

</el-dialog>

<!-- 分配角色数据权限对话框 -->

<el-dialog :title="title" :visible.sync="openDataScope" width="500px" append-to-body>

<el-form :model="form" label-width="80px">

<el-form-item label="角色名称">

<el-input v-model="form.roleName" :disabled="true" />

</el-form-item>

<el-form-item label="权限字符">

<el-input v-model="form.roleKey" :disabled="true" />

</el-form-item>

<el-form-item label="权限范围">

<el-select v-model="form.dataScope" @change="dataScopeSelectChange">

<el-option

v-for="item in dataScopeOptions"

:key="item.value"

:label="item.label"

:value="item.value"

></el-option>

</el-select>

</el-form-item>

<el-form-item label="数据权限" v-show="form.dataScope == 2">

<el-checkbox v-model="deptExpand" @change="handleCheckedTreeExpand($event, 'dept')">展开/折叠</el-checkbox>

<el-checkbox v-model="deptNodeAll" @change="handleCheckedTreeNodeAll($event, 'dept')">全选/全不选</el-checkbox>

<el-checkbox v-model="form.deptCheckStrictly" @change="handleCheckedTreeConnect($event, 'dept')">父子联动</el-checkbox>

<el-tree

class="tree-border"

:data="deptOptions"

show-checkbox

default-expand-all

ref="dept"

node-key="id"

:check-strictly="!form.deptCheckStrictly"

empty-text="加载中，请稍候"

:props="defaultProps"

></el-tree>

</el-form-item>

</el-form>

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

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

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

</div>

</el-dialog>

</div>

</template>

<script>

import { listRole, getRole, delRole, addRole, updateRole, dataScope, changeRoleStatus } from "@/api/system/role";

import { treeselect as menuTreeselect, roleMenuTreeselect } from "@/api/system/menu";

import { treeselect as deptTreeselect, roleDeptTreeselect } from "@/api/system/dept";

export default {

name: "Role",

dicts: ['sys\_normal\_disable'],

data() {

return {

// 遮罩层

loading: 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,

// 角色表格数据

roleList: [],

// 弹出层标题

title: "",

// 是否显示弹出层

open: false,

// 是否显示弹出层（数据权限）

openDataScope: false,

menuExpand: false,

menuNodeAll: false,

deptExpand: true,

deptNodeAll: false,

// 日期范围

dateRange: [],

// 数据范围选项

dataScopeOptions: [

{

value: "1",

label: "全部数据权限"

},

{

value: "2",

label: "自定数据权限"

},

{

value: "3",

label: "本部门数据权限"

},

{

value: "4",

label: "本部门及以下数据权限"

},

{

value: "5",

label: "仅本人数据权限"

}

],

// 菜单列表

menuOptions: [],

// 部门列表

deptOptions: [],

// 查询参数

queryParams: {

pageNum: 1,

pageSize: 10,

roleName: undefined,

roleKey: undefined,

status: undefined

},

// 表单参数

form: {},

defaultProps: {

children: "children",

label: "label"

},

// 表单校验

rules: {

roleName: [

{ required: true, message: "角色名称不能为空", trigger: "blur" }

],

roleKey: [

{ required: true, message: "权限字符不能为空", trigger: "blur" }

],

roleSort: [

{ required: true, message: "角色顺序不能为空", trigger: "blur" }

]

}

};

},

created() {

this.getList();

},

methods: {

/\*\* 查询角色列表 \*/

getList() {

this.loading = true;

listRole(this.addDateRange(this.queryParams, this.dateRange)).then(response => {

this.roleList = response.rows;

this.total = response.total;

this.loading = false;

}

);

},

/\*\* 查询菜单树结构 \*/

getMenuTreeselect() {

menuTreeselect().then(response => {

this.menuOptions = response.data;

});

},

/\*\* 查询部门树结构 \*/

getDeptTreeselect() {

deptTreeselect().then(response => {

this.deptOptions = response.data;

});

},

// 所有菜单节点数据

getMenuAllCheckedKeys() {

// 目前被选中的菜单节点

let checkedKeys = this.$refs.menu.getCheckedKeys();

// 半选中的菜单节点

let halfCheckedKeys = this.$refs.menu.getHalfCheckedKeys();

checkedKeys.unshift.apply(checkedKeys, halfCheckedKeys);

return checkedKeys;

},

// 所有部门节点数据

getDeptAllCheckedKeys() {

// 目前被选中的部门节点

let checkedKeys = this.$refs.dept.getCheckedKeys();

// 半选中的部门节点

let halfCheckedKeys = this.$refs.dept.getHalfCheckedKeys();

checkedKeys.unshift.apply(checkedKeys, halfCheckedKeys);

return checkedKeys;

},

/\*\* 根据角色ID查询菜单树结构 \*/

getRoleMenuTreeselect(roleId) {

return roleMenuTreeselect(roleId).then(response => {

this.menuOptions = response.menus;

return response;

});

},

/\*\* 根据角色ID查询部门树结构 \*/

getRoleDeptTreeselect(roleId) {

return roleDeptTreeselect(roleId).then(response => {

this.deptOptions = response.depts;

return response;

});

},

// 角色状态修改

handleStatusChange(row) {

let text = row.status === "0" ? "启用" : "停用";

this.$modal.confirm('确认要"' + text + '""' + row.roleName + '"角色吗？').then(function() {

return changeRoleStatus(row.roleId, row.status);

}).then(() => {

this.$modal.msgSuccess(text + "成功");

}).catch(function() {

row.status = row.status === "0" ? "1" : "0";

});

},

// 取消按钮

cancel() {

this.open = false;

this.reset();

},

// 取消按钮（数据权限）

cancelDataScope() {

this.openDataScope = false;

this.reset();

},

// 表单重置

reset() {

if (this.$refs.menu != undefined) {

this.$refs.menu.setCheckedKeys([]);

}

this.menuExpand = false,

this.menuNodeAll = false,

this.deptExpand = true,

this.deptNodeAll = false,

this.form = {

roleId: undefined,

roleName: undefined,

roleKey: undefined,

roleSort: 0,

status: "0",

menuIds: [],

deptIds: [],

menuCheckStrictly: true,

deptCheckStrictly: true,

remark: undefined

};

this.resetForm("form");

},

/\*\* 搜索按钮操作 \*/

handleQuery() {

this.queryParams.pageNum = 1;

this.getList();

},

/\*\* 重置按钮操作 \*/

resetQuery() {

this.dateRange = [];

this.resetForm("queryForm");

this.handleQuery();

},

// 多选框选中数据

handleSelectionChange(selection) {

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

this.single = selection.length!=1

this.multiple = !selection.length

},

// 更多操作触发

handleCommand(command, row) {

switch (command) {

case "handleDataScope":

this.handleDataScope(row);

break;

case "handleAuthUser":

this.handleAuthUser(row);

break;

default:

break;

}

},

// 树权限（展开/折叠）

handleCheckedTreeExpand(value, type) {

if (type == 'menu') {

let treeList = this.menuOptions;

for (let i = 0; i < treeList.length; i++) {

this.$refs.menu.store.nodesMap[treeList[i].id].expanded = value;

}

} else if (type == 'dept') {

let treeList = this.deptOptions;

for (let i = 0; i < treeList.length; i++) {

this.$refs.dept.store.nodesMap[treeList[i].id].expanded = value;

}

}

},

// 树权限（全选/全不选）

handleCheckedTreeNodeAll(value, type) {

if (type == 'menu') {

this.$refs.menu.setCheckedNodes(value ? this.menuOptions: []);

} else if (type == 'dept') {

this.$refs.dept.setCheckedNodes(value ? this.deptOptions: []);

}

},

// 树权限（父子联动）

handleCheckedTreeConnect(value, type) {

if (type == 'menu') {

this.form.menuCheckStrictly = value ? true: false;

} else if (type == 'dept') {

this.form.deptCheckStrictly = value ? true: false;

}

},

/\*\* 新增按钮操作 \*/

handleAdd() {

this.reset();

this.getMenuTreeselect();

this.open = true;

this.title = "添加角色";

},

/\*\* 修改按钮操作 \*/

handleUpdate(row) {

this.reset();

const roleId = row.roleId || this.ids

const roleMenu = this.getRoleMenuTreeselect(roleId);

getRole(roleId).then(response => {

this.form = response.data;

this.open = true;

this.$nextTick(() => {

roleMenu.then(res => {

let checkedKeys = res.checkedKeys

checkedKeys.forEach((v) => {

this.$nextTick(()=>{

this.$refs.menu.setChecked(v, true ,false);

})

})

});

});

this.title = "修改角色";

});

},

/\*\* 选择角色权限范围触发 \*/

dataScopeSelectChange(value) {

if(value !== '2') {

this.$refs.dept.setCheckedKeys([]);

}

},

/\*\* 分配数据权限操作 \*/

handleDataScope(row) {

this.reset();

const roleDeptTreeselect = this.getRoleDeptTreeselect(row.roleId);

getRole(row.roleId).then(response => {

this.form = response.data;

this.openDataScope = true;

this.$nextTick(() => {

roleDeptTreeselect.then(res => {

this.$refs.dept.setCheckedKeys(res.checkedKeys);

});

});

this.title = "分配数据权限";

});

},

/\*\* 分配用户操作 \*/

handleAuthUser: function(row) {

const roleId = row.roleId;

this.$router.push("/system/role-auth/user/" + roleId);

},

/\*\* 提交按钮 \*/

submitForm: function() {

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

if (valid) {

if (this.form.roleId != undefined) {

this.form.menuIds = this.getMenuAllCheckedKeys();

updateRole(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.open = false;

this.getList();

});

} else {

this.form.menuIds = this.getMenuAllCheckedKeys();

addRole(this.form).then(response => {

this.$modal.msgSuccess("新增成功");

this.open = false;

this.getList();

});

}

}

});

},

/\*\* 提交按钮（数据权限） \*/

submitDataScope: function() {

if (this.form.roleId != undefined) {

this.form.deptIds = this.getDeptAllCheckedKeys();

dataScope(this.form).then(response => {

this.$modal.msgSuccess("修改成功");

this.openDataScope = false;

this.getList();

});

}

},

/\*\* 删除按钮操作 \*/

handleDelete(row) {

const roleIds = row.roleId || this.ids;

this.$modal.confirm('是否确认删除角色编号为"' + roleIds + '"的数据项？').then(function() {

return delRole(roleIds);

}).then(() => {

this.getList();

this.$modal.msgSuccess("删除成功");

}).catch(() => {});

},

/\*\* 导出按钮操作 \*/

handleExport() {

this.download('system/role/export', {

...this.queryParams

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

}

}

};

</script>