**package** com.sdocean.common.model;

**import** java.util.List;

**public** **class** CombotreeModel {

**public** **static** **final** String *OPEN* = "open";

**public** **static** **final** String *CLOSED* = "closed";

**private** **int** id; //绑定节点的标示值

**private** String text; //显示的节点文本

**private** String state; //节点状态 open 或者是closed

**private** Boolean checked; //该节点是否被选中

**private** String iconCls; //显示的节点图标的CSS id

**private** String attributes; //绑定该节点的自定义属性

**private** String target; //目标节点的DOM对象

**private** List<CombotreeModel> children;

**public** List<CombotreeModel> getChildren() {

**return** children;

}

**public** **void** setChildren(List<CombotreeModel> children) {

**this**.children = children;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getText() {

**return** text;

}

**public** **void** setText(String text) {

**this**.text = text;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** Boolean getChecked() {

**return** checked;

}

**public** **void** setChecked(Boolean checked) {

**this**.checked = checked;

}

**public** String getIconCls() {

**return** iconCls;

}

**public** **void** setIconCls(String iconCls) {

**this**.iconCls = iconCls;

}

**public** String getAttributes() {

**return** attributes;

}

**public** **void** setAttributes(String attributes) {

**this**.attributes = attributes;

}

**public** String getTarget() {

**return** target;

}

**public** **void** setTarget(String target) {

**this**.target = target;

}

}

**package** com.sdocean.common.model;

**import** java.util.List;

**public** **class** Echarts {

**private** **static** **final** **long** *serialVersionUID* = 1L;

**public** List<Object> xAxis ;//多个折线图共有的横坐标

**public** List<YAxis> yAxis; //多个折线图的纵坐标

**public** List<Object> getxAxis() {

**return** xAxis;

}

**public** **void** setxAxis(List<Object> xAxis) {

**this**.xAxis = xAxis;

}

**public** List<YAxis> getyAxis() {

**return** yAxis;

}

**public** **void** setyAxis(List<YAxis> yAxis) {

**this**.yAxis = yAxis;

}

}

**package** com.sdocean.common.model;

**public** **class** Result {

**public** **static** **final** **int** *SUCCESS* = 1;

**public** **static** **final** **int** *FAILED* = 0;

/\*

\* 在公共代码中,parentcode = '0011'

\*/

**public** **static** **final** **int** *ADD* = 1;

**public** **static** **final** **int** *UPDATE* = 2;

**public** **static** **final** **int** *DELETE* = 3;

**private** **int** dotype;

**private** **int** result;

**private** String message;

**private** String model;

**private** Object res;

**public** Object getRes() {

**return** res;

}

**public** **void** setRes(Object res) {

**this**.res = res;

}

**public** String getModel() {

**return** model;

}

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** Result() {

**super**();

}

**public** Result(**int** dotype) {

**super**();

**this**.dotype = dotype;

}

**public** **int** getDotype() {

**return** dotype;

}

**public** **void** setDotype(**int** dotype) {

**this**.dotype = dotype;

}

**public** **int** getResult() {

**return** result;

}

**public** **void** setResult(**int** result) {

**this**.result = result;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

**package** com.sdocean.common.model;

**import** java.util.List;

**public** **class** SelectTree {

**private** String id;

**private** String name;

**private** Boolean selected;

**private** List<SelectTree> children;

**private** Boolean isExpanded;

**private** Boolean isActive;

**public** Boolean getIsActive() {

**return** isActive;

}

**public** **void** setIsActive(Boolean isActive) {

**this**.isActive = isActive;

}

**public** Boolean getIsExpanded() {

**return** isExpanded;

}

**public** **void** setIsExpanded(Boolean isExpanded) {

**this**.isExpanded = isExpanded;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Boolean getSelected() {

**return** selected;

}

**public** **void** setSelected(Boolean selected) {

**this**.selected = selected;

}

**public** List<SelectTree> getChildren() {

**return** children;

}

**public** **void** setChildren(List<SelectTree> children) {

**this**.children = children;

}

}

**package com.sdocean.common.model;**

**import java.util.LinkedList;**

**import java.util.List;**

**public class YAxis {**

**private static final long serialVersionUID = 1L;**

**public String fieldName;**

**public String unit;**

**public List yAxis2 = new LinkedList();//具体值**

**public String getFieldName() {**

**return fieldName;**

**}**

**public void setFieldName(String fieldName) {**

**this.fieldName = fieldName;**

**}**

**public String getUnit() {**

**return unit;**

**}**

**public void setUnit(String unit) {**

**this.unit = unit;**

**}**

**public List getyAxis2() {**

**return yAxis2;**

**}**

**public void setyAxis2(List yAxis2) {**

**this.yAxis2 = yAxis2;**

**}**

**}**

**package** com.sdocean.common.model;

**import** java.util.List;

**public** **class** ZTreeModel {

**private** String id;

**private** String pId;

**private** String name;

**private** String file;

**private** Boolean open;

**private** Integer ifNode; //代表是否节点

**private** String url;

**private** String target;

**private** Boolean checked; //初始化默认节点被勾选

**private** Boolean nocheck; //设置某节点不显示checkbox

**private** Boolean isParent;

**private** List<ZTreeModel> children;

**public** ZTreeModel() {

**super**();

}

**public** ZTreeModel(String id, String pId, String name, String file,

Boolean open, Integer ifNode, String url, String target,

Boolean checked, Boolean nocheck) {

**super**();

**this**.id = id;

**this**.pId = pId;

**this**.name = name;

**this**.file = file;

**this**.open = open;

**this**.ifNode = ifNode;

**this**.url = url;

**this**.target = target;

**this**.checked = checked;

**this**.nocheck = nocheck;

}

**public** Boolean getIsParent() {

**return** isParent;

}

**public** **void** setIsParent(Boolean isParent) {

**this**.isParent = isParent;

}

**public** List<ZTreeModel> getChildren() {

**return** children;

}

**public** **void** setChildren(List<ZTreeModel> children) {

**this**.children = children;

}

**public** Boolean getNocheck() {

**return** nocheck;

}

**public** **void** setNocheck(Boolean nocheck) {

**this**.nocheck = nocheck;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getpId() {

**return** pId;

}

**public** **void** setpId(String pId) {

**this**.pId = pId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getFile() {

**return** file;

}

**public** **void** setFile(String file) {

**this**.file = file;

}

**public** Boolean getOpen() {

**return** open;

}

**public** **void** setOpen(Boolean open) {

**this**.open = open;

}

**public** Integer getIfNode() {

**return** ifNode;

}

**public** **void** setIfNode(Integer ifNode) {

**this**.ifNode = ifNode;

}

**public** String getUrl() {

**return** url;

}

**public** **void** setUrl(String url) {

**this**.url = url;

}

**public** String getTarget() {

**return** target;

}

**public** **void** setTarget(String target) {

**this**.target = target;

}

**public** Boolean getChecked() {

**return** checked;

}

**public** **void** setChecked(Boolean checked) {

**this**.checked = checked;

}

}

**package com.sdocean.common.service;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpSession;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.dataQuery.dao.DataQueryDao;**

**import com.sdocean.dataQuery.dao.SynthQueryDao;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.dao.DeviceDao;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.dictionary.dao.PublicDao;**

**import com.sdocean.dictionary.model.PublicModel;**

**import com.sdocean.frame.model.ConfigInfo;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.dao.SysLoginLogDao;**

**import com.sdocean.log.model.SysLoginLogModel;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.position.dao.SysPositionDao;**

**import com.sdocean.position.model.SysPosition;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class CommonService {**

**@Autowired**

**private DeviceDao deviceDao;**

**/\***

**\* 将 deviceid#indicatorid格式转换为**

**\* list<deviceModel> 格式**

**\*/**

**public List<DeviceModel> indicatoridsToDevices(String indicatorIds){**

**String[] ids = indicatorIds.split(",");**

**Map<String, String> deids = new HashMap<String, String>();**

**for(String id:ids){**

**String indicatorid = id.substring(0, id.indexOf("#"));**

**String deviceid = id.substring(id.indexOf("#")+1,id.length());**

**if(deids.containsKey(deviceid)){**

**String indi = deids.get(deviceid)+","+indicatorid;**

**deids.remove(deviceid);**

**deids.put(deviceid, indi);**

**}else{**

**deids.put(deviceid, indicatorid);**

**}**

**}**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**for(String deviceId:deids.keySet()){**

**String indicatorid = deids.get(deviceId);**

**DeviceModel device = deviceDao.getDeviceByid(deviceId, indicatorid);**

**if(device!=null&&device.getIndicators()!=null&&device.getIndicators().size()>0){**

**list.add(device);**

**}**

**}**

**return list;**

**}**

**}**

**package com.sdocean.company.action;**

**import java.util.ArrayList;**

**import java.util.List;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.company.model.CompanyModel;**

**import com.sdocean.company.model.SysCompanyModel;**

**import com.sdocean.company.service.CompanyService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Controller**

**public class CompanyAction {**

**private static Logger log = Logger.getLogger(CompanyAction.class);**

**@Autowired**

**private CompanyService companyService;**

**@Autowired**

**private OperationLogService logService;**

**@RequestMapping(value="getCompanyList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getCompanyList(HttpServletRequest request,**

**HttpServletResponse response){**

**List<SysCompanyModel> companys = companyService.getSysCompList();**

**return JsonUtil.toJson(companys);**

**}**

**/\***

**\* 跳转到公司管理页面**

**\*/**

**@RequestMapping("info\_company.do")**

**public ModelAndView info\_company(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/company/companyInfo");**

**return mav;**

**}**

**/\***

**\* 获得公司列表的查询结果**

**\*/**

**@RequestMapping(value="getgCompanyList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getgCompanyList(@ModelAttribute("model") CompanyModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//添加表头**

**List<UiColumn> cols = companyService.getCols4List();**

**//添加数据**

**List<CompanyModel> list = companyService.getCompanyList(model);**

**result.setCols(cols);**

**result.setRows(list);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 保存修改**

**\*/**

**@RequestMapping(value="saveCompanyChange.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveCompanyChange(@ModelAttribute("model") CompanyModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = companyService.saveCompanyChange(model);**

**logService.saveOperationLog(result, request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增**

**\*/**

**@RequestMapping(value="saveNewCompany.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveNewCompany(@ModelAttribute("model") CompanyModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = companyService.saveNewCompany(model);**

**logService.saveOperationLog(result, request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存删除**

**\*/**

**@RequestMapping(value="deleCompany.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String deleCompany(@ModelAttribute("model") CompanyModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = companyService.deleCompany(model);**

**logService.saveOperationLog(result, request);**

**return result.getMessage();**

**}**

**/\***

**\* 根据公司获得pcode树**

**\*/**

**@RequestMapping(value="showCompanyTree4Pcode.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showCompanyTree4Pcode(@ModelAttribute("model") CompanyModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**list = companyService.showCompanyTree4Pcode(model);**

**return JsonUtil.toJson(list);**

**}**

**/\***

**\* 根据人员获得公司树**

**\*/**

**@RequestMapping(value="showComList4Users.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showComList4Users(@ModelAttribute("model") SysUser model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**list = companyService.showComList4Users(model);**

**return JsonUtil.toJson(list);**

**}**

**/\***

**\* 根据站点获得公司树**

**\*/**

**@RequestMapping(value="geCompanyListByStation.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String geCompanyListByStation(@ModelAttribute("station") StationModel station,HttpServletRequest request,**

**HttpServletResponse response){**

**List<SelectTree> trees = new ArrayList<SelectTree>();**

**trees = companyService.geCompanyListByStation(station);**

**return JsonUtil.toJson(trees);**

**}**

**}**

**package com.sdocean.company.dao;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.common.model.ZTreeModel;**

**import com.sdocean.company.model.CompanyModel;**

**import com.sdocean.company.model.SysCompanyModel;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Component**

**public class CompanyDao extends OracleEngine {**

**/\***

**\* 获得所有的有效的公司列表**

**\***

**\*/**

**public List<SysCompanyModel> getSysCompList(){**

**List<SysCompanyModel> complist = new ArrayList<>();**

**//开始拼接SQL语句**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.id,a.code,a.name,a.shortname,a.region\_id,a.isactive,a.ordercode");**

**sql.append(" from sys\_company a");**

**complist = this.queryObjectList(sql.toString(), SysCompanyModel.class);**

**return complist;**

**}**

**/\***

**\* 获得所有有效的公司列表,按照等级排序编码排序**

**\*/**

**public List<CompanyModel> getCompanyList(CompanyModel model){**

**List<CompanyModel> list = new ArrayList<>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select id,code,pcode,name,shortname,level,");**

**sql.append(" isactive,case isactive when 0 then '禁用' else '启用' end as isactivename,ordercode");**

**sql.append(" from g\_company ");**

**sql.append(" where 1=1");**

**if(model!=null&&model.getCode()!=null&&model.getCode().length()>0){**

**sql.append(" and ( code like '%").append(model.getCode()).append("%' or");**

**sql.append(" name like '%").append(model.getCode()).append("%' or");**

**sql.append(" shortname like '%").append(model.getCode()).append("%' )");**

**}**

**//添加排序**

**sql.append(" order by orderCode");**

**list = this.queryObjectList(sql.toString(), CompanyModel.class);**

**return list;**

**}**

**/\***

**\* 修改更改**

**\*/**

**public Result saveCompanyChange(CompanyModel model){**

**//初始化返回结果**

**Result result = new Result();**

**result.setDotype(result.UPDATE);**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("修改成功");**

**//判断CODE是否重复**

**StringBuffer checkSql = new StringBuffer("");**

**checkSql.append(" select count(1) from g\_company where code = '").append(model.getCode()).append("' and id <> ").append(model.getId());**

**int check = 0;**

**try {**

**check=this.queryForInt(checkSql.toString(), null);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("检查Code唯一性时失败");**

**return result;**

**}**

**if(check>0){**

**result.setResult(result.FAILED);**

**result.setMessage("违反CODE唯一性原则");**

**return result;**

**}**

**//执行修改代码**

**StringBuffer sql = new StringBuffer("");**

**sql.append("update g\_company set code=?,pcode=?,name=?,shortname=?,level=?,isactive=?,ordercode=? where id=?");**

**Object[] params = new Object[]{**

**model.getCode(),model.getPcode(),model.getName(),model.getShortName(),**

**model.getLevel(),model.getIsactive(),model.getOrderCode(),model.getId()**

**};**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), params);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("修改失败");**

**return result;**

**}**

**return result;**

**}**

**/\***

**\* 新增保存**

**\*/**

**public Result saveNewCompany(CompanyModel model){**

**Result result = new Result();**

**result.setDotype(result.ADD);**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("保存成功");**

**//判断CODE是否重复**

**StringBuffer checkSql = new StringBuffer("");**

**checkSql.append(" select count(1) from g\_company where code = '").append(model.getCode()).append("' ");**

**int check = 0;**

**try {**

**check=this.queryForInt(checkSql.toString(), null);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("检查Code唯一性时失败");**

**return result;**

**}**

**if(check>0){**

**result.setResult(result.FAILED);**

**result.setMessage("违反CODE唯一性原则");**

**return result;**

**}**

**//执行修改代码**

**StringBuffer sql = new StringBuffer("");**

**sql.append("insert into g\_company(code,pcode,name,shortname,level,isactive,ordercode) values(?,?,?,?,?,?,?)");**

**Object[] params = new Object[]{**

**model.getCode(),model.getPcode(),model.getName(),model.getShortName(),**

**model.getLevel(),model.getIsactive(),model.getOrderCode()**

**};**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), params);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("保存失败");**

**return result;**

**}**

**return result;**

**}**

**/\***

**\* 保存删除功能**

**\*/**

**public Result deleCompany(CompanyModel model){**

**//初始化返回结果**

**Result result = new Result();**

**result.setDotype(result.DELETE);**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("删除成功");**

**//开始删除**

**StringBuffer sql = new StringBuffer();**

**sql.append(" delete from g\_company where id = ").append(model.getId());**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), null);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("删除失败,请重试");**

**}**

**return result;**

**}**

**/\***

**\* 根据当前的company获得所有有效的pcode树**

**\* 排除当前的company**

**\*/**

**public List<SelectTree> showCompanyTree4Pcode(CompanyModel model){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**//获得第一层数据**

**SelectTree first = new SelectTree();**

**StringBuffer firstSql = new StringBuffer("");**

**firstSql.append(" select code as id,shortName as name ,'true' as isExpanded,");**

**firstSql.append(" case when code = '").append(model.getPcode()).append("' then 'true' else 'false' end as selected from g\_company where code = '0001'");**

**firstSql.append(" and code <> '").append(model.getCode()).append("' limit 1");**

**first = this.queryObject(firstSql.toString(), SelectTree.class);**

**//根据第一层,获得他的子类**

**if(first!=null){**

**this.getChildTree4Pcode(first, model);**

**}else{**

**first = new SelectTree();**

**first.setName("当前已经是最高等级");**

**}**

**list.add(first);**

**return list;**

**}**

**public void getChildTree4Pcode(SelectTree pmodel,CompanyModel model){**

**List<SelectTree> children = new ArrayList<SelectTree>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select code as id,shortName as name ,'true' as isExpanded,");**

**sql.append(" case when code = '").append(model.getPcode()).append("' then 'true' else 'false' end as selected ");**

**sql.append(" from g\_company where isactive = 1 and pcode = '").append(pmodel.getId()).append("'");**

**sql.append(" and code <> '").append(model.getCode()).append("'");**

**children = this.queryObjectList(sql.toString(), SelectTree.class);**

**for(SelectTree child:children){**

**this.getChildTree4Pcode(child, model);**

**}**

**pmodel.setChildren(children);**

**}**

**/\***

**\* 根据人员获得公司数**

**\*/**

**public List<SelectTree> showComList4Users(SysUser model){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**//获得第一层数据**

**SelectTree first = new SelectTree();**

**StringBuffer firstSql = new StringBuffer("");**

**firstSql.append(" select code as id,shortName as name ,'true' as isExpanded,");**

**firstSql.append(" case when code = '").append(model.getCompanyId()).append("' then 'true' else 'false' end as selected");**

**firstSql.append(" from g\_company where code = '0001' and isactive = 1 limit 1");**

**first = this.queryObject(firstSql.toString(), SelectTree.class);**

**this.getChildCom4Pcom(first, model);**

**list.add(first);**

**return list;**

**}**

**//根据上层公司,以及人员信息,获得下层的公司树**

**public void getChildCom4Pcom(SelectTree parent,SysUser user){**

**List<SelectTree> children = new ArrayList<SelectTree>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select code as id,shortName as name ,'true' as isExpanded,");**

**sql.append(" case when code ='").append(user.getCompanyId()).append("' then 'true' else 'false' end as selected ");**

**sql.append(" from g\_company where isactive = 1 and pcode = '").append(parent.getId()).append("'");**

**children = this.queryObjectList(sql.toString(), SelectTree.class);**

**for(SelectTree child:children){**

**this.getChildCom4Pcom(child, user);**

**}**

**parent.setChildren(children);**

**}**

**/\***

**\* 以ztree的形式获得公司列表**

**\***

**\*/**

**public List<ZTreeModel> getComList4ZTree(){**

**List<ZTreeModel> list = new ArrayList<ZTreeModel>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select concat('C',a.code) as id,concat('C',a.pcode) as pid,a.shortname as name,");**

**sql.append(" 'true' as open,'true' as nocheck");**

**sql.append(" from g\_company a where a.isactive = 1");**

**list = this.queryObjectList(sql.toString(), ZTreeModel.class);**

**return list;**

**}**

**/\***

**\* 在下拉框中展示站点树**

**\*/**

**public List<SelectTree> geCompanyListByStation(StationModel station){**

**List<SelectTree> trees = new ArrayList<SelectTree>();**

**//得到第一层的地区代码**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.code as id,a.name as name,'true' as isExpanded,'true' as isActive,");**

**sql.append(" case when a.code ='").append(station.getCompanyId()).append("' then 'true' else 'false' end as selected ");**

**sql.append(" from g\_company a where a.pcode = '0000'");**

**trees = this.queryObjectList(sql.toString(), SelectTree.class);**

**for(SelectTree child:trees){**

**this.setChildren4Tree(child,station);**

**}**

**return trees;**

**}**

**/\***

**\* 为当前的站点添加他的子元素**

**\*/**

**public void setChildren4Tree(SelectTree model,StationModel station){**

**List<SelectTree> child = new ArrayList<SelectTree>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append("select a.code as id,a.name as name,'true' as isExpanded,case when a.code = '").append(station.getCompanyId()).append("' then 'true' else 'false' end as selected");**

**sql.append(" from g\_company a where a.pcode = '").append(model.getId()).append("'");**

**child = this.queryObjectList(sql.toString(), SelectTree.class);**

**if(child!=null&&child.size()>0){**

**for(SelectTree children:child){**

**this.setChildren4Tree(children,station);**

**}**

**model.setChildren(child);**

**}**

**}**

**}**

**package** com.sdocean.company.model;

**public** **class** CompanyModel {

**private** **int** id;

**private** String code;

**private** String pcode;

**private** String name;

**private** String shortName;

**private** **int** level;

**private** **int** isactive;

**private** String isactiveName;

**private** String orderCode;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getPcode() {

**return** pcode;

}

**public** **void** setPcode(String pcode) {

**this**.pcode = pcode;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getShortName() {

**return** shortName;

}

**public** **void** setShortName(String shortName) {

**this**.shortName = shortName;

}

**public** **int** getLevel() {

**return** level;

}

**public** **void** setLevel(**int** level) {

**this**.level = level;

}

**public** **int** getIsactive() {

**return** isactive;

}

**public** **void** setIsactive(**int** isactive) {

**this**.isactive = isactive;

}

**public** String getIsactiveName() {

**return** isactiveName;

}

**public** **void** setIsactiveName(String isactiveName) {

**this**.isactiveName = isactiveName;

}

**public** String getOrderCode() {

**return** orderCode;

}

**public** **void** setOrderCode(String orderCode) {

**this**.orderCode = orderCode;

}

}

**package** com.sdocean.company.model;

**public** **class** SysCompanyModel {

**private** **int** id;

**private** String code;

**private** String name;

**private** String shortName;

**private** **int** regionId;

**private** String regionName;

**private** **int** isactive;

**private** String isactiveName;

**private** String orderCode;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getShortName() {

**return** shortName;

}

**public** **void** setShortName(String shortName) {

**this**.shortName = shortName;

}

**public** **int** getRegionId() {

**return** regionId;

}

**public** **void** setRegionId(**int** regionId) {

**this**.regionId = regionId;

}

**public** String getRegionName() {

**return** regionName;

}

**public** **void** setRegionName(String regionName) {

**this**.regionName = regionName;

}

**public** **int** getIsactive() {

**return** isactive;

}

**public** **void** setIsactive(**int** isactive) {

**this**.isactive = isactive;

}

**public** String getIsactiveName() {

**return** isactiveName;

}

**public** **void** setIsactiveName(String isactiveName) {

**this**.isactiveName = isactiveName;

}

**public** String getOrderCode() {

**return** orderCode;

}

**public** **void** setOrderCode(String orderCode) {

**this**.orderCode = orderCode;

}

}

**package com.sdocean.company.service;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.company.dao.CompanyDao;**

**import com.sdocean.company.model.CompanyModel;**

**import com.sdocean.company.model.SysCompanyModel;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class CompanyService {**

**@Autowired**

**private CompanyDao companyDao;**

**/\***

**\* 为公司展示列表添加表头**

**\*/**

**public List<UiColumn> getCols4List(){**

**List<UiColumn> cols = new ArrayList<UiColumn>();**

**UiColumn col1 = new UiColumn("id", "id", false, "\*");**

**UiColumn col2 = new UiColumn("code", "code", true, "\*");**

**UiColumn col3 = new UiColumn("父code", "pcode", true, "\*");**

**UiColumn col4 = new UiColumn("单位全称", "name", true, "\*");**

**UiColumn col5 = new UiColumn("单位简称", "shortName", true, "\*");**

**UiColumn col6 = new UiColumn("层级", "level", true, "\*");**

**UiColumn col19 = new UiColumn("isactive", "isactive", false, "\*");**

**UiColumn col10 = new UiColumn("状态", "isactiveName", true, "\*");**

**UiColumn col11 = new UiColumn("排序码", "orderCode", true, "\*");**

**cols.add(col1);**

**cols.add(col2);**

**cols.add(col3);**

**cols.add(col6);**

**cols.add(col4);**

**cols.add(col5);**

**cols.add(col19);**

**cols.add(col10);**

**cols.add(col11);**

**return cols;**

**}**

**/\***

**\* 获得所有有效的公司的列表**

**\*/**

**public List<SysCompanyModel> getSysCompList(){**

**return companyDao.getSysCompList();**

**}**

**/\***

**\* 获得所有有效的公司列表,按照等级排序编码排序**

**\*/**

**public List<CompanyModel> getCompanyList(CompanyModel model){**

**return companyDao.getCompanyList(model);**

**}**

**/\***

**\* 修改更改**

**\*/**

**public Result saveCompanyChange(CompanyModel model){**

**return companyDao.saveCompanyChange(model);**

**}**

**/\***

**\* 新增保存**

**\*/**

**public Result saveNewCompany(CompanyModel model){**

**return companyDao.saveNewCompany(model);**

**}**

**/\***

**\* 根据当前的company获得所有有效的pcode树**

**\* 排除当前的company**

**\*/**

**public List<SelectTree> showCompanyTree4Pcode(CompanyModel model){**

**return companyDao.showCompanyTree4Pcode(model);**

**}**

**/\***

**\* 根据人员获得公司数**

**\*/**

**public List<SelectTree> showComList4Users(SysUser model){**

**return companyDao.showComList4Users(model);**

**}**

**/\***

**\* 在下拉框中展示站点树**

**\*/**

**public List<SelectTree> geCompanyListByStation(StationModel station){**

**return companyDao.geCompanyListByStation(station);**

**}**

**/\***

**\* 保存删除功能**

**\*/**

**public Result deleCompany(CompanyModel model){**

**return companyDao.deleCompany(model);**

**}**

**}**

**package com.sdocean.dataQuery.action;**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**import java.util.Calendar;**

**import java.util.Date;**

**import java.util.List;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.dataQuery.service.DataQueryService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Controller**

**public class DataQueryAction {**

**private static Logger log = Logger.getLogger(DataQueryAction.class);**

**@Autowired**

**DataQueryService dataQueryService;**

**@RequestMapping("dataquery\_init.do")**

**public ModelAndView dataquery\_init(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dataquery/dataquery\_init");**

**return mav;**

**}**

**/\***

**\* 为查询条件初始化**

**\*/**

**@RequestMapping(value="dataquery\_info.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String dataquery\_info(HttpServletRequest request,**

**HttpServletResponse response){**

**DataQueryModel model = new DataQueryModel();**

**HttpSession session = request.getSession();**

**StationModel station = (StationModel) session.getAttribute("station");**

**model.setStationId(station.getId());**

**DateFormat beginDf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**Calendar calendar = Calendar.getInstance();**

**calendar.setTime(new Date());**

**//设置结束时间**

**String endDate = beginDf.format(calendar.getTime());**

**//设置开始时间**

**calendar.add(Calendar.MONTH, -1);**

**String beginDate = beginDf.format(calendar.getTime());**

**model.setBeginDate(beginDate);**

**model.setEndDate(endDate);**

**return JsonUtil.toJson(model);**

**}**

**/\***

**\* 为设备管理查询结果**

**\*/**

**@RequestMapping(value="showDataQueryInfo.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showDataQueryInfo(@ModelAttribute("model") DataQueryModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//为查询结果增加表头**

**List<UiColumn> cols = dataQueryService.getCols4DataQuery(model);**

**result.setCols(cols);**

**List<Map<String, Object>> rows = dataQueryService.getRows4DataQuery(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**}**

**package** com.sdocean.dataQuery.action;

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** org.apache.log4j.Logger;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** org.springframework.web.servlet.ModelAndView;

**import** com.sdocean.common.model.Echarts;

**import** com.sdocean.common.model.SelectTree;

**import** com.sdocean.dataQuery.model.DataQueryModel;

**import** com.sdocean.dataQuery.model.GraphModel;

**import** com.sdocean.dataQuery.model.StatisModel;

**import** com.sdocean.dataQuery.service.GraphQueryService;

**import** com.sdocean.dataQuery.service.StatisQueryService;

**import** com.sdocean.frame.util.JsonUtil;

**import** com.sdocean.indicator.service.IndicatorService;

**import** com.sdocean.station.model.StationModel;

@Controller

**public** **class** GraphQueryAction {

**private** **static** Logger *log* = Logger.*getLogger*(GraphQueryAction.**class**);

@Autowired

IndicatorService indicatorService;

@Autowired

GraphQueryService graphService;

@RequestMapping("info\_graphquery.do")

**public** ModelAndView info\_graphquery(HttpServletRequest request,

HttpServletResponse response)**throws** Exception{

ModelAndView mav = **new** ModelAndView("/dataquery/graphquery\_init");

DateFormat beginDf = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

DateFormat df = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Calendar calendar = Calendar.*getInstance*();

calendar.setTime(**new** Date());

//默认开始时间为一个月以前

calendar.add(Calendar.*MONTH*, -2);

// 添加默认开始时间

String time1 = beginDf.format(calendar.getTime()) ;

// 添加默认结束时间

String time2 = df.format(**new** Date());

// 存放参数

mav.addObject("beginDate", time1);

mav.addObject("endDate", time2);

**return** mav;

}

/\*

\* 为查询条件初始化

\*/

@RequestMapping(value="graphquery\_init.do", method = RequestMethod.*POST*,produces = "application/json;charset=UTF-8")

@ResponseBody

**public** String graphquery\_init(HttpServletRequest request,

HttpServletResponse response){

DataQueryModel model = **new** DataQueryModel();

HttpSession session = request.getSession();

StationModel station = (StationModel) session.getAttribute("station");

model.setStationId(station.getId());

DateFormat beginDf = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Calendar calendar = Calendar.*getInstance*();

calendar.setTime(**new** Date());

//设置结束时间

String endDate = beginDf.format(calendar.getTime());

//设置开始时间

calendar.add(Calendar.*DATE*, -20);

String beginDate = beginDf.format(calendar.getTime());

model.setBeginDate(beginDate);

model.setEndDate(endDate);

//根据站点获得该站点下的参数列表

List<SelectTree> indicatorTree = indicatorService.getIndicators4StationDevice4Show(station);

model.setIndicatorTree(indicatorTree);

**return** JsonUtil.*toJson*(model);

}

/\*

\* 数据评价中综合趋势展示图标

\*/

@RequestMapping(value="graphShow4echarts.do", method = RequestMethod.*POST*,produces = "application/json;charset=UTF-8")

@ResponseBody

**public** String graphShow4echarts(@ModelAttribute("model") GraphModel model,HttpServletRequest request,

HttpServletResponse response) {

//对参数进行初始化

graphService.modelInfo(model);

HttpSession session = request.getSession();

StationModel station = (StationModel) session.getAttribute("station");

Echarts charts = **new** Echarts();

charts = graphService.getEcharts4Graph(station, model);

System.*out*.println(JsonUtil.*toJson*(charts));

**return** JsonUtil.*toJson*(charts);

}

}

**package com.sdocean.dataQuery.action;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dataQuery.model.MouldModel;**

**import com.sdocean.dataQuery.service.MouldService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.station.model.StationModel;**

**@Controller**

**public class MouldAction {**

**private static Logger log = Logger.getLogger(MouldAction.class);**

**@Autowired**

**MouldService mouldService;**

**@Autowired**

**OperationLogService logService;**

**@RequestMapping("info\_consetting.do")**

**public ModelAndView info\_consetting(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dataquery/consetting\_init");**

**return mav;**

**}**

**/\***

**\* 根据站点获得当前站点的水质评价模板**

**\*/**

**@RequestMapping(value="getMouldByStationId.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getMouldByStationId(@ModelAttribute("model") MouldModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**MouldModel result = mouldService.getMouldByStationId(model);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 保存新建或修改的水质评价模板**

**\*/**

**@RequestMapping(value="saveMouldSetting.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveMouldSetting(@ModelAttribute("model") MouldModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = mouldService.saveMouldSetting(model);**

**//保存此次操作信息**

**logService.saveOperationLog(result, request);**

**return result.getMessage();**

**}**

**}**

**package com.sdocean.dataQuery.action;**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**import java.util.Calendar;**

**import java.util.Date;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.dataQuery.model.StatisModel;**

**import com.sdocean.dataQuery.service.StatisQueryService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.station.model.StationModel;**

**@Controller**

**public class StatisQueryAction {**

**private static Logger log = Logger.getLogger(StatisQueryAction.class);**

**@Autowired**

**StatisQueryService statisService;**

**@RequestMapping("info\_statisquery.do")**

**public ModelAndView info\_statisquery(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dataquery/statisquery\_init");**

**DateFormat beginDf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**Calendar calendar = Calendar.getInstance();**

**calendar.setTime(new Date());**

**//默认开始时间为一个月以前**

**calendar.add(Calendar.MONTH, -2);**

**// 添加默认开始时间**

**String time1 = beginDf.format(calendar.getTime()) ;**

**// 添加默认结束时间**

**String time2 = df.format(new Date());**

**// 存放参数**

**mav.addObject("beginDate", time1);**

**mav.addObject("endDate", time2);**

**return mav;**

**}**

**/\***

**\* 水质统计中查询**

**\*/**

**@RequestMapping(value="/showStat.do", produces = "application/json; charset=utf-8")**

**@ResponseBody**

**public String showStat(@ModelAttribute("gmodel") StatisModel gmodel,**

**Map<String, Object> model, HttpServletRequest request,**

**HttpServletResponse response) {**

**HttpSession session = request.getSession();**

**StationModel station = (StationModel) session.getAttribute("station");**

**gmodel = statisService.getStatDataSearch(gmodel, station);**

**return JsonUtil.toJson(gmodel);**

**}**

**/\***

**\* 首页限制折线图**

**\*/**

**@RequestMapping(value="/showStat4First.do", produces = "application/json; charset=utf-8")**

**@ResponseBody**

**public String showStat4First(@ModelAttribute("gmodel") StatisModel gmodel,**

**Map<String, Object> model, HttpServletRequest request,**

**HttpServletResponse response) {**

**//获得USERID**

**Long userId = (Long) request.getSession().getAttribute("userId");**

**//获得站点信息**

**HttpSession session = request.getSession();**

**StationModel station = (StationModel) session.getAttribute("station");**

**gmodel = statisService.getStatDataSearch4First(userId, gmodel, station);**

**String result = JsonUtil.toJson(gmodel);**

**return result;**

**}**

**}**

**package com.sdocean.dataQuery.action;**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**import java.util.Calendar;**

**import java.util.Date;**

**import java.util.List;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.common.service.CommonService;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.dataQuery.service.SynthQueryService;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.indicator.service.IndicatorService;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Controller**

**public class SynthQueryAction {**

**private static Logger log = Logger.getLogger(SynthQueryAction.class);**

**@Autowired**

**SynthQueryService synthQueryService;**

**@Autowired**

**IndicatorService indicatorService;**

**@Autowired**

**CommonService commonService;**

**@RequestMapping("info\_synthquery.do")**

**public ModelAndView info\_synthquery(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dataquery/synthquery\_info");**

**return mav;**

**}**

**/\***

**\* 为查询条件初始化**

**\*/**

**@RequestMapping(value="synthquery\_init.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String synthquery\_init(HttpServletRequest request,**

**HttpServletResponse response){**

**DataQueryModel model = new DataQueryModel();**

**HttpSession session = request.getSession();**

**StationModel station = (StationModel) session.getAttribute("station");**

**model.setStationId(station.getId());**

**DateFormat beginDf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**Calendar calendar = Calendar.getInstance();**

**calendar.setTime(new Date());**

**//设置结束时间**

**String endDate = beginDf.format(calendar.getTime());**

**//设置开始时间**

**calendar.add(Calendar.MONTH, -1);**

**String beginDate = beginDf.format(calendar.getTime());**

**model.setBeginDate(beginDate);**

**model.setEndDate(endDate);**

**//根据站点获得该站点下的参数列表**

**List<SelectTree> indicatorTree = indicatorService.getIndicators4StationDevice4Show(station);**

**model.setIndicatorTree(indicatorTree);**

**return JsonUtil.toJson(model);**

**}**

**/\***

**\* 为综合查询提供结果**

**\*/**

**@RequestMapping(value="synthquery\_show.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String synthquery\_show(@ModelAttribute("model") DataQueryModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//对参数进行初始化**

**List<DeviceModel> devices = commonService.indicatoridsToDevices(model.getIndicatorIds());**

**model.setDevices(devices);**

**//为查询结果增加表头**

**List<UiColumn> cols = synthQueryService.getCols4SynQuery(model);**

**result.setCols(cols);**

**System.out.println(JsonUtil.toJson(cols));**

**//为查询结果增加结果集**

**List<Map<String, Object>> rows = synthQueryService.getRows4SynQuery(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**}** **package com.sdocean.dataQuery.dao;**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**import java.util.ArrayList;**

**import java.util.Calendar;**

**import java.util.Date;**

**import java.util.List;**

**import java.util.Map;**

**import javax.annotation.Resource;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.firstpage.model.Ddata;**

**import com.sdocean.firstpage.model.LastMetaData;**

**import com.sdocean.firstpage.model.MetaData4FirstPage;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.indicator.dao.IndicatorDao;**

**import com.sdocean.indicator.model.IndicatorModel;**

**import com.sdocean.metadata.dao.MetadataTableDao;**

**import com.sdocean.metadata.model.MetadataTable;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Component**

**public class DataQueryDao extends OracleEngine{**

**@Resource**

**MetadataTableDao tableDao;**

**@Resource**

**IndicatorDao indicatorDao;**

**/\***

**\* 为实时数据添加表头**

**\*/**

**public List<UiColumn> getCols4DataQuery(DataQueryModel model){**

**List<UiColumn> cols = new ArrayList<UiColumn>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select '时间' as displayName,'collect\_time' as field,'true' as visible,'\*' as width,'' as cellFilter");**

**sql.append(" union all ");**

**sql.append(" select distinct case when b.logo is null or length(b.logo) = 0 then a.title else concat(a.title,'(',b.logo,')') end as displayName,");**

**sql.append(" a.code as field,'true' as visible,'\*' as width,concat('number:',c.pointNum) as cellFilter");**

**sql.append(" from dm\_indicator a left join g\_unit b ");**

**sql.append(" on a.unitid = b.id and b.isactive = 1");**

**sql.append(" ,view\_stationid\_deviceid\_indicatorid c ");**

**sql.append(" where a.id = c.indicatorid");**

**sql.append(" and a.isactive = 1");**

**sql.append(" and c.deviceid =").append(model.getDeviceId());**

**sql.append(" and c.stationid =").append(model.getStationId());**

**cols = this.queryObjectList(sql.toString(), UiColumn.class);**

**return cols;**

**}**

**/\***

**\* 为实时数据查询提供结果**

**\*/**

**public List<Map<String, Object>> getRows4DataQuery(DataQueryModel model){**

**List<Map<String, Object>> rows = null;**

**StationModel station = new StationModel();**

**station.setId(model.getStationId());**

**//根据起始时间判断出需要查询的表名的集合**

**List<MetadataTable> tables = tableDao.getTables4Meta(station,model.getBeginDate(), model.getEndDate(), 1);**

**//查询出当前设备下的参数列表**

**DeviceModel device = new DeviceModel();**

**device.setId(model.getDeviceId());**

**List<IndicatorModel> indicators = indicatorDao.getIndicators4Deivce(device);**

**//定义总的SQL语句**

**StringBuffer sql = new StringBuffer("");**

**//定义查询参数**

**StringBuffer indicatorsql = new StringBuffer("");**

**StringBuffer indicatorGroupBySql = new StringBuffer("");**

**indicatorGroupBySql.append(" group by collect\_time having ( 0= 1");**

**for(IndicatorModel indicator:indicators){**

**indicatorsql.append(" ,sum(if(indicator\_code='").append(indicator.getCode()).append("',data,0)) as ").append(indicator.getCode());**

**indicatorGroupBySql.append(" ||sum(if(indicator\_code='").append(indicator.getCode()).append("',data,0)) <> 0");**

**}**

**indicatorGroupBySql.append(" )");**

**//定义where查询条件**

**StringBuffer wheresql = new StringBuffer(" where ");**

**wheresql.append("wpid = ").append(model.getStationId()).append(" and deviceid =").append(model.getDeviceId());**

**//增加时间参数.**

**if(model.getBeginDate()!=null){**

**wheresql.append(" and collect\_time >= '").append(model.getBeginDate()).append("'");**

**}**

**if(model.getEndDate()!=null){**

**wheresql.append(" and collect\_time <= '").append(model.getEndDate()).append("'");**

**}**

**//定义排序语句**

**StringBuffer ordersql = new StringBuffer(" order by collect\_time desc");**

**for(int i=0;i<tables.size();i++){**

**StringBuffer selectSql = new StringBuffer("select collect\_time");**

**MetadataTable table = tables.get(i);**

**String tableName = table.getTableName();**

**StringBuffer fromSql = new StringBuffer(" from ");**

**fromSql.append(tableName);**

**//生成每个表的查询语句**

**selectSql.append(indicatorsql).append(fromSql).append(wheresql).append(indicatorGroupBySql);**

**//将每个表的sql语句添加到总语句中**

**sql.append(selectSql);**

**if(i<tables.size()-1){**

**sql.append(" union all ");**

**}**

**}**

**sql.append(ordersql);**

**System.out.println(sql.toString());**

**rows = this.queryForList(sql.toString());**

**return rows;**

**}**

**/\***

**\* 查询出当前站点信息的最后实时数据**

**\* 为首页展示**

**\*/**

**public List<LastMetaData> getData4FirstPage(StationModel station){**

**//初始化返回结果**

**List<LastMetaData> list = new ArrayList<LastMetaData>();**

**//获得要展示的设备以及参数**

**StringBuffer devicesql = new StringBuffer("");**

**devicesql.append("select distinct b.id as deviceId ,b.name as deviceName,b.pointNum");**

**devicesql.append(" from aiot\_firstpage\_show a,device\_catalog b,device\_catalog\_indicator c,dm\_indicator d");**

**devicesql.append(" where a.deviceid = b.id and a.wpid = ").append(station.getId());**

**devicesql.append(" and b.id = c.catalogid and c.indicatorid = d.id and d.isactive = 1");**

**devicesql.append(" order by b.ordercode ");**

**list = this.queryObjectList(devicesql.toString(), LastMetaData.class);**

**//遍历设备列表,得到需要查询的参数列表**

**for(LastMetaData device:list){**

**StringBuffer indicatorSql = new StringBuffer("");**

**indicatorSql.append("select b.id as indicatorId,b.code as indicatorCode,b.title as indicatorTitle,c.logo as unitName,c.description as unitDescription,c.logo as unitLogo ");**

**indicatorSql.append(" from aiot\_firstpage\_show a,dm\_indicator b,g\_unit c ");**

**indicatorSql.append(" where a.indicatorid = b.id and b.isactive = 1 and c.id = b.unitid and a.wpid = ").append(station.getId());**

**indicatorSql.append(" and a.deviceid = ").append(device.getDeviceId());**

**indicatorSql.append(" order by b.orderCode");**

**List<MetaData4FirstPage> metaDatas = new ArrayList<MetaData4FirstPage>();**

**metaDatas = this.queryObjectList(indicatorSql.toString(), MetaData4FirstPage.class);**

**@SuppressWarnings("deprecation")**

**Date maxDate = new Date(1, 1, 1, 1, 1, 1);**

**//遍历metaDatas并得到最新的最大的数据**

**for(MetaData4FirstPage metaData:metaDatas){**

**//初始化参数,获得当前时间一天前的时间**

**DateFormat beginDf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**Calendar calendar = Calendar.getInstance();**

**calendar.setTime(new Date());**

**String nowDate = beginDf.format(calendar.getTime());**

**Ddata ddata = this.getDataByWDTDI(station, nowDate, 1, device, metaData);**

**if(ddata==null){**

**calendar.add(Calendar.MONTH, -1);**

**nowDate = beginDf.format(calendar.getTime());**

**ddata = this.getDataByWDTDI(station, nowDate, 1, device, metaData);**

**}**

**if(ddata!=null&&ddata.getLastTime()!=null&&ddata.getLastTime().after(maxDate)){**

**//获得采集时间**

**device.setLastTime(ddata.getLastTime());**

**}**

**if(ddata!=null&&ddata.getData()!=null){**

**//获得数据**

**metaData.setMdata(ddata.getData());**

**}**

**}**

**device.setMetaDatas(metaDatas);**

**}**

**return list;**

**}**

**//通过站点,起始时间,类型,设备,以及参数查询出实时数据**

**public Ddata getDataByWDTDI(StationModel station,String nowDate,int type,LastMetaData device,MetaData4FirstPage indicator){**

**Ddata ddata = null;**

**MetadataTable table = tableDao.getOneTable(station, nowDate, type);**

**String tableName = table.getTableName();**

**StringBuffer dsql = new StringBuffer("");**

**dsql.append("select collect\_time as lastTime, data from ").append(tableName);**

**dsql.append(" where wpid = ").append(station.getId()).append(" and indicator\_code ='").append(indicator.getIndicatorCode()).append("'");**

**dsql.append(" and data is not null and data <> 88888");**

**dsql.append(" and deviceid = ").append(device.getDeviceId());**

**dsql.append(" order by collect\_time desc limit 1");**

**ddata = this.queryObject(dsql.toString(), Ddata.class);**

**return ddata;**

**}**

**}**

**package com.sdocean.dataQuery.dao;**

**import java.util.ArrayList;**

**import java.util.HashMap;**

**import java.util.LinkedList;**

**import java.util.List;**

**import java.util.Map;**

**import javax.annotation.Resource;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.common.model.Echarts;**

**import com.sdocean.common.model.YAxis;**

**import com.sdocean.dataQuery.model.GraphModel;**

**import com.sdocean.device.dao.DeviceDao;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.indicator.dao.IndicatorDao;**

**import com.sdocean.indicator.model.IndicatorModel;**

**import com.sdocean.metadata.dao.MetadataTableDao;**

**import com.sdocean.metadata.model.MetadataTable;**

**import com.sdocean.station.model.StationModel;**

**@Component**

**public class GraphQueryDao extends OracleEngine{**

**@Resource**

**MetadataTableDao tableDao;**

**@Resource**

**IndicatorDao indicatorDao;**

**@Resource**

**DeviceDao deviceDao;**

**public Echarts getEcharts4Graph(StationModel station,GraphModel model){**

**Map<String, Object> indicatorNameMap = new HashMap<String, Object>();**

**indicatorNameMap = indicatorDao.getIndicatorTitleMap();**

**Map<String, Object> indicatorUnitMap = new HashMap<String, Object>();**

**indicatorUnitMap = indicatorDao.getIndicatorUnitMap();**

**Echarts echarts = new Echarts();**

**//根据起始时间判断出需要查询的表名的集合**

**List<MetadataTable> tables = tableDao.getTables4Meta(station,model.getBeginDate(), model.getEndDate(), 2);**

**List<DeviceModel> devices = model.getDevices();**

**//遍历设备列表,得到模板设备**

**//初始化模板设备**

**DeviceModel device = new DeviceModel();**

**Integer interal = 0;**

**for(DeviceModel dev:devices){**

**dev = deviceDao.getInterval4device(station.getId(), dev);**

**int last = dev.getInterval();**

**if(last >= interal){**

**interal = last;**

**device = dev;**

**}**

**}**

**//定义总的sql语句**

**StringBuffer sql = new StringBuffer("");**

**//根据表名遍历**

**for(int i=0;i<tables.size();i++){**

**//得到需要查询的表名**

**String tableName = tables.get(i).getTableName();**

**//得到别名**

**String deviceName = "d"+device.getId();**

**//定义最外层的select部分**

**StringBuffer allseleSql = new StringBuffer("");**

**//定义最外层的from部分**

**StringBuffer allfromsql = new StringBuffer("");**

**//定义最外层的group部分**

**StringBuffer allGroupSql = new StringBuffer("");**

**allGroupSql.append(" group by ").append(deviceName).append(".collect\_time");**

**allfromsql.append(" from (");**

**allseleSql.append("select ").append(deviceName).append(".collect\_time");**

**//定义最外层的select中的其他部分**

**StringBuffer selectSql = new StringBuffer(""); //定义select部分**

**StringBuffer fromSql = new StringBuffer(""); //定义from部分**

**StringBuffer whereSql = new StringBuffer(""); //定义where部分**

**StringBuffer groupSql = new StringBuffer(""); //定义group部分**

**StringBuffer orderSql = new StringBuffer(""); //定义排序部分**

**selectSql.append(" select collect\_time");**

**fromSql.append(" from ").append(tableName);**

**whereSql.append(" where 1 =1");**

**if(model!=null&&model.getBeginDate()!=null&&model.getBeginDate().length()>0){**

**whereSql.append(" and collect\_time >= '").append(model.getBeginDate()).append("'");**

**}**

**if(model!=null&&model.getEndDate()!=null&&model.getEndDate().length()>0){**

**whereSql.append(" and collect\_time <= '").append(model.getEndDate()).append("'");**

**}**

**whereSql.append(" and data <> 0 and data <> 88888 and data <> 9999");**

**whereSql.append(" and wpid = ").append(station.getId());**

**//设备条件**

**StringBuffer whereDeviceSql = new StringBuffer("");**

**whereDeviceSql.append(" and deviceid =").append(device.getId());**

**groupSql.append(" group by collect\_time");**

**orderSql.append(" order by collect\_time desc");**

**//**

**//遍历模板设备下的参数**

**List<IndicatorModel> indicators = device.getIndicators();**

**//**

**//定义select条件查询**

**StringBuffer sumsql = new StringBuffer("");**

**//定义indicatorcode in 语句**

**StringBuffer insql = new StringBuffer(" and indicator\_code in ('0'");**

**for(IndicatorModel indicator:indicators){**

**sumsql.append(",sum(if(indicator\_code='").append(indicator.getCode()).append("',data,0)) as ").append(indicator.getCode());**

**insql.append(",'").append(indicator.getCode()).append("'");**

**allseleSql.append(",").append(deviceName).append(".").append(indicator.getCode());**

**}**

**insql.append(")");**

**//整合第一个子查询**

**StringBuffer csql = new StringBuffer("");**

**csql.append(selectSql).append(sumsql).append(fromSql).append(whereSql).append(whereDeviceSql).append(insql).append(groupSql);**

**//将第一个子查询整合到allfrom中**

**allfromsql.append(csql).append(") ").append(deviceName);**

**//遍历设备列表中的其他列表**

**for(DeviceModel dev:devices){**

**//去除当做模板的设备**

**if(dev.getId()!=device.getId()&&dev.getIndicators().size()>0){**

**//得到该设备的别名**

**String devname ="d"+dev.getId();**

**allseleSql.append(",max(").append(devname).append(".collect\_time) as ").append(devname).append("time");**

**//设备条件**

**StringBuffer whereDevSql = new StringBuffer("");**

**whereDevSql.append(" and deviceid = ").append(dev.getId());**

**//定义select条件查询**

**StringBuffer csumsql = new StringBuffer("");**

**//定义indicatorcode in 语句**

**StringBuffer cinsql = new StringBuffer(" and indicator\_code in ('0'");**

**//遍历该设备下的参数**

**List<IndicatorModel> cindicators = dev.getIndicators();**

**for(IndicatorModel indicator:cindicators){**

**csumsql.append(",sum(if(indicator\_code='").append(indicator.getCode()).append("',data,0)) as ").append(indicator.getCode());**

**cinsql.append(",'").append(indicator.getCode()).append("'");**

**allseleSql.append(", case when ").append(devname).append(".").append(indicator.getCode()).append(" is null then 0 else ").append(devname).append(".").append(indicator.getCode());**

**allseleSql.append(" end as ").append(indicator.getCode());**

**}**

**cinsql.append(")");**

**//定义该设备的子查询部分**

**StringBuffer cdevSql = new StringBuffer("");**

**cdevSql.append(" left join (");**

**cdevSql.append(selectSql).append(csumsql).append(fromSql).append(whereSql).append(whereDevSql).append(cinsql).append(groupSql);**

**cdevSql.append(orderSql).append(") ").append(devname).append(" on ").append(deviceName).append(".collect\_time >=").append(devname).append(".collect\_time");**

**//将该设备下的子查询部分整合到allfrom中**

**allfromsql.append(cdevSql);**

**}**

**}**

**sql.append(allseleSql).append(allfromsql).append(allGroupSql);**

**if(i<tables.size()-1){**

**sql.append(" union all ");**

**}**

**}**

**sql.append(" order by collect\_time");**

**List<Map<String, Object>> rows = new ArrayList<Map<String, Object>>();**

**rows = this.queryForList(sql.toString());**

**//定义四个YAxis备用**

**List<Object> xAxis = new ArrayList<Object>();**

**YAxis YAxis1 = new YAxis();**

**YAxis YAxis2 = new YAxis();**

**YAxis YAxis3 = new YAxis();**

**YAxis YAxis4 = new YAxis();**

**//循环遍历list**

**for(Map<String,Object> resmap:rows){**

**int i=0;**

**for(String key:resmap.keySet()){**

**if(i==0){**

**xAxis.add(resmap.get(key));**

**i++;**

**}else{**

**String fieldName = (String) indicatorNameMap.get(key);**

**if(fieldName!=null&&fieldName.length()>0){**

**if(i==1){ //将数据插入到YAxis1中**

**YAxis1.fieldName=(String) indicatorNameMap.get(key);**

**YAxis1.unit=(String) indicatorUnitMap.get(key);**

**YAxis1.yAxis2.add(resmap.get(key));**

**}else if(i==2){ //将数据插入到YAxis2中**

**YAxis2.fieldName=(String) indicatorNameMap.get(key);**

**YAxis2.unit=(String) indicatorUnitMap.get(key);**

**YAxis2.yAxis2.add(resmap.get(key));**

**}else if(i==3){ //将数据插入到YAxis3中**

**YAxis3.fieldName=(String) indicatorNameMap.get(key);**

**YAxis3.unit=(String) indicatorUnitMap.get(key);**

**YAxis3.yAxis2.add(resmap.get(key));**

**}else if(i==4){ //将数据插入到YAxis4中**

**YAxis4.fieldName=(String) indicatorNameMap.get(key);**

**YAxis4.unit=(String) indicatorUnitMap.get(key);**

**YAxis4.yAxis2.add(resmap.get(key));**

**}**

**i++;**

**}**

**}**

**}**

**}**

**echarts.setxAxis(xAxis);**

**List<YAxis> ya = new ArrayList<>();**

**if(YAxis1.fieldName!=null&&YAxis1.fieldName.length()>0){**

**ya.add(YAxis1);**

**}**

**if(YAxis2.fieldName!=null&&YAxis2.fieldName.length()>0){**

**ya.add(YAxis2);**

**}**

**if(YAxis3.fieldName!=null&&YAxis3.fieldName.length()>0){**

**ya.add(YAxis3);**

**}**

**if(YAxis4.fieldName!=null&&YAxis4.fieldName.length()>0){**

**ya.add(YAxis4);**

**}**

**echarts.setyAxis(ya);**

**return echarts;**

**}**

**}**

**package com.sdocean.dataQuery.dao;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dataQuery.model.MouldModel;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.station.model.StationModel;**

**@Component**

**public class MouldDao extends OracleEngine{**

**/\***

**\* 获得当前站点的水质评价模板**

**\*/**

**public MouldModel getMouldByStationId(MouldModel model){**

**MouldModel result = new MouldModel();**

**StringBuffer sql = new StringBuffer("");**

**sql.append("select id,stationid,mould from sys\_mouldsetting where stationid = ").append(model.getStationId());**

**result = this.queryObject(sql.toString(), MouldModel.class);**

**if(result==null){**

**result = new MouldModel();**

**result.setStationId(model.getStationId());**

**}**

**return result;**

**}**

**/\***

**\* 保存当前站点的水质评价模板**

**\*/**

**public Result saveMouldSetting(MouldModel model){**

**//初始化返回结果**

**Result result = new Result();**

**if(model.getId()>0){**

**result.setDotype(result.UPDATE);**

**}else{**

**result.setDotype(result.ADD);**

**}**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("保存成功");**

**//开始拼接插入/修改语句**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" insert into sys\_mouldsetting(stationid,mould) values(?,?) on duplicate key update mould=values(mould)");**

**Object[] params = new Object[]{**

**model.getStationId(),model.getMould()**

**};**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), params);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("保存失败");**

**}**

**return result;**

**}**

**/\***

**\* 根据站点,获得水质评价模板**

**\*/**

**public MouldModel getMouldByStation(StationModel model){**

**MouldModel result = new MouldModel();**

**StringBuffer sql = new StringBuffer("");**

**sql.append("select id,stationid,mould from sys\_mouldsetting where stationid = ").append(model.getId());**

**result = this.queryObject(sql.toString(), MouldModel.class);**

**if(result==null){**

**result = new MouldModel();**

**result.setStationId(model.getId());**

**}**

**return result;**

**}**

**}**

**package com.sdocean.dataQuery.dao;**

**import java.text.SimpleDateFormat;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Date;**

**import java.util.List;**

**import javax.annotation.Resource;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.dataQuery.model.StatData;**

**import com.sdocean.dataQuery.model.StatisModel;**

**import com.sdocean.firstpage.model.WaterStandard;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.indicator.dao.IndicatorDao;**

**import com.sdocean.indicator.model.IndicatorModel;**

**import com.sdocean.metadata.dao.MetadataTableDao;**

**import com.sdocean.metadata.model.MetadataTable;**

**import com.sdocean.station.model.StationModel;**

**@Component**

**public class StatisQueryDao extends OracleEngine{**

**@Resource**

**MetadataTableDao tableDao;**

**@Resource**

**IndicatorDao indicatorDao;**

**/\***

**\* 为水质统计提供饼形图以及统计列表数据**

**\*/**

**public List<StatData> getData4pie(StatisModel model,StationModel station,int type){**

**List<StatData> list = new ArrayList<StatData>();**

**//获得起止时间**

**String bdate = model.getBeginDate();**

**String edate = model.getEndDate();**

**String groupSql = "";**

**//按照统计口径**

**String statType = model.getStatType()+"";**

**if(statType.equals("1")){//按照每月口径统计**

**groupSql = "date\_format(a.collect\_time,'%Y-%m')";**

**}else if(statType.equals("2")){//按照每周口径统计**

**groupSql = "concat(year(a.collect\_time),week(a.collect\_time))";**

**}else if(statType.equals("3")){**

**groupSql = "date\_format(a.collect\_time,'%Y-%m-%d')";**

**}**

**//通过观测点以及时间得到需要查询的TABLENAME列表**

**List<MetadataTable> tableList = tableDao.getTables4Meta(station,bdate, edate, type);**

**//查询每个表中的数据**

**StringBuffer sql = new StringBuffer("");**

**for(int i=0;i<tableList.size();i++){**

**MetadataTable table = tableList.get(i);**

**String tableName = table.getTableName();**

**sql.append(" select m.time,m.fieldName,max(m.standard\_grade) as standard\_grade");**

**sql.append(" from");**

**sql.append(" (");**

**sql.append(" select ").append(groupSql).append(" as time,c.title as fieldName,max(b.standard\_grade) as standard\_grade");**

**sql.append(" from ").append(tableName).append(" a,waterqualitystandard b,dm\_indicator c");**

**sql.append(" where date\_format(a.collect\_time,'%Y-%m-%d') between '").append(bdate).append("' and '").append(edate).append("'");**

**sql.append(" and a.indicator\_code = b.item and a.indicator\_code = c.code");**

**sql.append(" and a.wpid = ").append(station.getId());**

**sql.append(" and a.data >= b.min\_value and a.data < b.max\_value");**

**sql.append(" and b.water\_type = ").append(station.getWaterType());**

**sql.append(" group by ").append(groupSql).append(",a.indicator\_code");**

**sql.append(" order by max(b.standard\_grade) desc ");**

**sql.append(" ) m group by m.time");**

**if(tableList.size()>1&&i<tableList.size()-1){**

**sql.append(" union all ");**

**}**

**}**

**//拼接SQL语句,获得当前每个统计口径的最高等级参数**

**//获得总共的统计口径的个数**

**StringBuffer sqlCount = new StringBuffer("");**

**sqlCount.append("select count(1) from (").append(sql).append(") n");**

**int statCount = 0;**

**try {**

**statCount = this.queryForInt(sqlCount.toString(), null);**

**} catch (Exception e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**//获得每个水质等级的个数以及首要污染源是什么**

**StringBuffer sqlList = new StringBuffer("");**

**sqlList.append(" select count(n.time) as y,group\_concat(distinct(fieldName)) as firstThing,n.standard\_grade,s.value as name");**

**sqlList.append(" from (");**

**sqlList.append(sql);**

**sqlList.append(" ) n , sys\_public s where s.parentcode = '0007' and n.standard\_grade = s.classid group by n.standard\_grade");**

**System.out.println(sqlList.toString());**

**try {**

**list = this.queryObjectList(sqlList.toString(), StatData.class);**

**} catch (Exception e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**//将结果总数融合到list中**

**if(list!=null&&list.size()>0){**

**for(StatData statData:list){**

**statData.setStatcount(statCount);**

**}**

**}**

**return list;**

**}**

**/\***

**\*为水质统计查询折线图**

**\*/**

**public List<StatData> getStatDataSearchList(StatisModel gmodel,StationModel station,int type){**

**List<StatData> statDatas = new ArrayList<StatData>();**

**//获得起止时间**

**String bdate = gmodel.getBeginDate();**

**String edate = gmodel.getEndDate();**

**String groupSql = "";**

**//按照统计口径**

**String statType = gmodel.getStatType()+"";**

**if(statType.equals("1")){//按照每月口径统计**

**groupSql = "date\_format(a.collect\_time,'%Y-%m')";**

**}else if(statType.equals("2")){//按照每周口径统计**

**groupSql = "concat(year(a.collect\_time),week(a.collect\_time))";**

**}else if(statType.equals("3")){**

**groupSql = "date\_format(a.collect\_time,'%Y-%m-%d')";**

**}**

**//通过观测点以及时间得到需要查询的TABLENAME列表**

**List<MetadataTable> tableList = tableDao.getTables4Meta(station,bdate, edate, type);**

**//拼接SQL语句,获得当前每个统计口径的最高等级参数**

**StringBuffer sql = new StringBuffer("");**

**//读取配置文件,得到有水质标准的参数信息**

**List<IndicatorModel> indicators = new ArrayList<IndicatorModel>();**

**StringBuffer standardSql = new StringBuffer("");**

**standardSql.append(" select distinct b.id as indicatorid,b.code,b.title");**

**standardSql.append(" from waterqualitystandard a,dm\_indicator b");**

**standardSql.append(" where a.water\_type = ").append(station.getWaterType());**

**standardSql.append(" and a.item = b.code and b.isactive = 1");**

**standardSql.append(" order by a.id desc");**

**try {**

**indicators = this.queryObjectList(standardSql.toString(), IndicatorModel.class);**

**} catch (Exception e1) {**

**// TODO Auto-generated catch block**

**e1.printStackTrace();**

**}**

**if(indicators==null||indicators.size()<1){**

**return null;**

**}**

**//遍历参数信息,依次查询出在规定范围内的每个统计口径的最高水质标准**

**for(IndicatorModel indicator:indicators){**

**for(MetadataTable tableName:tableList){**

**String tablename = tableName.getTableName();**

**StringBuffer standSql = new StringBuffer("");**

**standSql.append(" select ").append(groupSql).append(" as xtime,'");**

**standSql.append(indicator.getTitle()).append("' as name,max(b.standard\_grade) as ydata");**

**standSql.append(" from ").append(tablename).append(" a ,waterqualitystandard b");**

**standSql.append(" where date\_format(a.collect\_time,'%Y-%m-%d') between '").append(bdate).append("' and '").append(edate).append("'");**

**standSql.append(" and a.wpid = ").append(station.getId()).append(" and a.indicator\_code = '").append(indicator.getCode()).append("' and a.collect\_type = '1'");**

**standSql.append(" and b.item = '").append(indicator.getCode()).append("'");**

**standSql.append(" and a.data >= b.min\_value and a.data < b.max\_value");**

**standSql.append(" and a.data <> 88888");**

**standSql.append(" and collect\_type = 1");**

**standSql.append(" group by ").append(groupSql);**

**standSql.append(" order by ").append(groupSql).append(" ");**

**List<StatData> statlist = new ArrayList<StatData>();**

**try {**

**statlist = this.queryObjectList(standSql.toString(), StatData.class);**

**} catch (Exception e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**for(int i=0;i<statlist.size();i++){**

**StatData stat = statlist.get(i);**

**//定义是否有匹配 初始化没有匹配**

**Boolean ifnothave = true;**

**for(StatData statdata:statDatas){**

**if(stat.getXtime().equals(statdata.getXtime())){**

**if(Integer.parseInt(stat.getYdata())>Integer.parseInt(statdata.getYdata())){**

**Collections.replaceAll(statDatas, statdata, stat);**

**}**

**//**

**ifnothave = false;**

**}**

**}**

**//如果没有匹配,则加入数据 并生成正确的顺序**

**if(ifnothave){**

**//是否有符合条件**

**Boolean ifcheck = false;**

**//初始化要插入数据的索引**

**int index = 0;**

**//获得要插入的数据的时间**

**String inData = stat.getXtime();**

**for(int j=0;j<statDatas.size();j++){**

**//获得原来数据的时间**

**String stData = statDatas.get(j).getXtime();**

**int inst = inData.compareTo(stData);**

**if(inst<0&&!ifcheck){**

**ifcheck = true;**

**index = j;**

**}**

**}**

**if(ifcheck){//如果有匹配的,则根据索引插入数据**

**statDatas.add(index,stat);**

**}else{ //如果没有匹配的数据,则在最后插入数据**

**statDatas.add(stat);**

**}**

**}**

**}**

**}**

**}**

**return statDatas;**

**}**

**/\***

**\* 为首页提供水质**

**\*/**

**public WaterStandard getWaterStandard(StationModel station){**

**WaterStandard waterStandard = new WaterStandard();**

**//获得当前的时间,并判断应该查询的表**

**SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd 00:00:00");**

**String nowDate = df.format(new Date());**

**MetadataTable table = tableDao.getOneTable(station, nowDate, 1);**

**String tableName = table.getTableName();**

**//开始拼接SQL**

**StringBuffer wsql = new StringBuffer("");**

**wsql.append(" select m.standard\_grade,m.standard\_name as standardName,m.sdata,m.indicator\_code as indicatorcode,m.indicator\_name as indicatorName,m.unit\_name as unit ");**

**wsql.append(" from ( ");**

**wsql.append(" select max(k.collect\_time),k.indicator\_code,k.standard\_grade,k.standard\_name,");**

**wsql.append(" k.sdata,k.indicator\_name,k.unit\_name");**

**wsql.append(" from (");**

**wsql.append(" select a.collect\_time ,b.standard\_grade,s.value as standard\_name,");**

**wsql.append(" a.data as sdata,d.code as indicator\_code,d.title as indicator\_name,e.logo as unit\_name");**

**wsql.append(" from ").append(tableName).append(" a,waterqualitystandard b,dm\_indicator d,g\_unit e,sys\_public s");**

**wsql.append(" where a.indicator\_code = b.item and b.item = d.code and d.unitid = e.id");**

**wsql.append(" and a.data <> 88888");**

**wsql.append(" and a.wpid =").append(station.getId()).append(" and s.parentcode = '0007'");**

**wsql.append(" and b.standard\_grade = s.classid ");**

**wsql.append(" and a.data >= b.min\_value and a.data < b.max\_value");**

**wsql.append(" and a.collect\_time >= '").append(nowDate).append("'");**

**wsql.append(" and b.water\_type = ").append(station.getWaterType()).append(" and d.isactive = 1");**

**wsql.append(" order by a.collect\_time desc");**

**wsql.append(" ) k group by k.indicator\_code");**

**wsql.append(" ) m order by m.standard\_grade desc limit 1");**

**/\* wsql.append(" select m.standard\_grade,m.standard\_name as standardName,m.sdata,m.indicator\_code as indicatorcode,m.indicator\_name as indicatorName,m.unit\_name as unit from");**

**wsql.append(" (");**

**wsql.append(" select max(a.collect\_time) as collect\_time,b.standard\_grade,s.value as standard\_name,a.data as sdata,d.code as indicator\_code,d.title as indicator\_name,e.logo as unit\_name");**

**wsql.append(" from ").append(tableName).append(" a,waterqualitystandard b,dm\_indicator d,g\_unit e,sys\_public s");**

**wsql.append(" where a.indicator\_code = b.item and b.item = d.code and d.unitid = e.id");**

**wsql.append(" and a.wpid =").append(station.getId());**

**wsql.append(" and s.parentcode = '0007' and b.standard\_grade = s.classid");**

**wsql.append(" and a.data>= b.min\_value and a.data< b.max\_value");**

**wsql.append(" and b.water\_type = ").append(station.getWaterType());**

**wsql.append(" and d.isactive = 1");**

**wsql.append(" group by d.code");**

**wsql.append(" ) m order by m.standard\_grade desc limit 1");\*/**

**waterStandard = this.queryObject(wsql.toString(), WaterStandard.class);**

**return waterStandard;**

**}**

**}**

**package com.sdocean.dataQuery.dao;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Map;**

**import javax.annotation.Resource;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.indicator.dao.IndicatorDao;**

**import com.sdocean.indicator.model.IndicatorModel;**

**import com.sdocean.metadata.dao.MetadataTableDao;**

**import com.sdocean.metadata.model.MetadataTable;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Component**

**public class SynthQueryDao extends OracleEngine{**

**@Resource**

**MetadataTableDao tableDao;**

**@Resource**

**IndicatorDao indicatorDao;**

**/\***

**\* 为综合查询添加表头**

**\*/**

**public List<UiColumn> getCols4SynQuery(DataQueryModel model){**

**List<DeviceModel> devices = model.getDevices();**

**List<UiColumn> cols = new ArrayList<UiColumn>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select '时间' as displayName,'collect\_time' as field,'true' as visible,'\*' as width,'' as cellFilter");**

**for(DeviceModel device:devices){**

**//遍历设备内的参数**

**StringBuffer indicatorSql = new StringBuffer("(0");**

**for(IndicatorModel indicator:device.getIndicators()){**

**indicatorSql.append(",").append(indicator.getId());**

**}**

**indicatorSql.append(")");**

**sql.append(" union all ");**

**sql.append(" select case when b.logo is null or length(b.logo)=0 then a.title else concat(a.title,'(',b.logo,')') end as displayName,");**

**sql.append(" concat(a.code,c.catalogid) as field,'true' as visible,'\*' as width,concat('number:',d.pointNum) as cellFilter");**

**sql.append(" from dm\_indicator a left join g\_unit b ");**

**sql.append(" on a.unitid = b.id and b.isactive = 1");**

**sql.append(" ,device\_catalog\_indicator c,device\_catalog d");**

**sql.append(" where a.id = c.indicatorid");**

**sql.append(" and c.catalogid = d.id");**

**sql.append(" and a.isactive = 1");**

**sql.append(" and c.catalogid =").append(device.getId());**

**sql.append(" and a.id in ").append(indicatorSql);**

**}**

**System.out.println(sql.toString());**

**cols = this.queryObjectList(sql.toString(), UiColumn.class);**

**return cols;**

**}**

**/\***

**\* 为综合查询查询结果**

**\*/**

**public List<Map<String, Object>> getRows4SynQuery(DataQueryModel model){**

**List<Map<String, Object>> rows = null;**

**StationModel station = new StationModel();**

**station.setId(model.getStationId());**

**//根据起始时间判断出需要查询的表名的集合**

**List<MetadataTable> tables = tableDao.getTables4Meta(station,model.getBeginDate(), model.getEndDate(), 2);**

**//定义总的查询语句**

**StringBuffer sql = new StringBuffer("");**

**//定义select部分**

**StringBuffer selectSql = new StringBuffer(" select collect\_time ");**

**//定义where部分**

**StringBuffer whereSql = new StringBuffer(" where ");**

**whereSql.append(" wpid = ").append(model.getStationId()).append(" and (1=0 ");**

**//定义groupby部分**

**StringBuffer groupSql = new StringBuffer(" group by collect\_time");**

**//定义having部分**

**StringBuffer havingSql = new StringBuffer(" having 1=0 ");**

**//定义排序部分**

**StringBuffer orderSql = new StringBuffer(" order by collect\_time desc");**

**List<DeviceModel> devices = model.getDevices();**

**for(DeviceModel device:devices){**

**List<IndicatorModel> indicators = device.getIndicators();**

**StringBuffer indicatorCodes = new StringBuffer("'0'");**

**for(IndicatorModel indicator:indicators){**

**selectSql.append(", sum(if(indicator\_code='").append(indicator.getCode()).append("' and deviceid = ").append(device.getId()).append(",data,0)) as ");**

**selectSql.append(indicator.getCode()).append(device.getId());**

**indicatorCodes.append(",'").append(indicator.getCode()).append("'");**

**havingSql.append("||sum(if(indicator\_code='").append(indicator.getCode()).append("' and deviceid =").append(device.getId()).append(",data,0)) <> 0");**

**}**

**whereSql.append(" or ( deviceid = ").append(device.getId()).append(" and indicator\_code in (").append(indicatorCodes).append("))");**

**}**

**whereSql.append(")");**

**//添加时间查询条件**

**//增加时间参数.**

**if(model.getBeginDate()!=null){**

**whereSql.append(" and collect\_time >= '").append(model.getBeginDate()).append("'");**

**}**

**if(model.getEndDate()!=null){**

**whereSql.append(" and collect\_time <= '").append(model.getEndDate()).append("'");**

**}**

**//遍历TABLELIST组成查询语句**

**for(int i=0;i<tables.size();i++){**

**String tablename = tables.get(i).getTableName();**

**String fromSql = " from "+tablename;**

**sql.append(selectSql).append(fromSql).append(whereSql).append(groupSql).append(havingSql);**

**if(i<tables.size()-1){**

**sql.append(" union all ");**

**}**

**}**

**sql.append(orderSql);**

**rows = this.queryForList(sql.toString());**

**return rows;**

**}**

**}**

**package com.sdocean.dataQuery.dao;**

**import java.lang.reflect.Field;**

**import java.lang.reflect.InvocationTargetException;**

**import java.lang.reflect.Method;**

**import java.text.DateFormat;**

**import java.text.ParseException;**

**import java.text.SimpleDateFormat;**

**import java.util.ArrayList;**

**import java.util.Calendar;**

**import java.util.Date;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.frame.util.ReplaceUtil;**

**public class test {**

**public static void main(String[] args) {**

**List aa = new ArrayList<>();**

**aa.add(1);**

**aa.add(2);**

**aa.add(3);**

**aa.add(4);**

**aa.add(5);**

**aa.add(0, 333);**

**System.out.println(JsonUtil.toJson(aa));**

**}**

**}** **package com.sdocean.dataQuery.model;**

**import java.util.List;**

**import java.util.Map;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.device.model.DeviceModel;**

**public class DataQueryModel {**

**private int stationId;**

**private int deviceId;**

**private String deviceIds;**

**private String beginDate;**

**private String endDate;**

**private List<SelectTree> indicatorTree;**

**private String indicatorIds;**

**private List<DeviceModel> devices;**

**public List<DeviceModel> getDevices() {**

**return devices;**

**}**

**public void setDevices(List<DeviceModel> devices) {**

**this.devices = devices;**

**}**

**public String getIndicatorIds() {**

**return indicatorIds;**

**}**

**public void setIndicatorIds(String indicatorIds) {**

**this.indicatorIds = indicatorIds;**

**}**

**public List<SelectTree> getIndicatorTree() {**

**return indicatorTree;**

**}**

**public void setIndicatorTree(List<SelectTree> indicatorTree) {**

**this.indicatorTree = indicatorTree;**

**}**

**public String getDeviceIds() {**

**return deviceIds;**

**}**

**public void setDeviceIds(String deviceIds) {**

**this.deviceIds = deviceIds;**

**}**

**public int getStationId() {**

**return stationId;**

**}**

**public void setStationId(int stationId) {**

**this.stationId = stationId;**

**}**

**public int getDeviceId() {**

**return deviceId;**

**}**

**public void setDeviceId(int deviceId) {**

**this.deviceId = deviceId;**

**}**

**public String getBeginDate() {**

**return beginDate;**

**}**

**public void setBeginDate(String beginDate) {**

**this.beginDate = beginDate;**

**}**

**public String getEndDate() {**

**return endDate;**

**}**

**public void setEndDate(String endDate) {**

**this.endDate = endDate;**

**}**

**}**

**package com.sdocean.dataQuery.model;**

**import java.util.List;**

**import com.sdocean.device.model.DeviceModel;**

**public class GraphModel {**

**private String beginDate;**

**private String endDate;**

**private String indicatorIds;**

**private List<DeviceModel> devices;**

**public List<DeviceModel> getDevices() {**

**return devices;**

**}**

**public void setDevices(List<DeviceModel> devices) {**

**this.devices = devices;**

**}**

**public String getBeginDate() {**

**return beginDate;**

**}**

**public void setBeginDate(String beginDate) {**

**this.beginDate = beginDate;**

**}**

**public String getEndDate() {**

**return endDate;**

**}**

**public void setEndDate(String endDate) {**

**this.endDate = endDate;**

**}**

**public String getIndicatorIds() {**

**return indicatorIds;**

**}**

**public void setIndicatorIds(String indicatorIds) {**

**this.indicatorIds = indicatorIds;**

**}**

**}**

**package** com.sdocean.dataQuery.model;

**public** **class** MouldModel {

**private** **int** id;

**private** **int** stationId;

**private** String stationName;

**private** String mould;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **int** getStationId() {

**return** stationId;

}

**public** **void** setStationId(**int** stationId) {

**this**.stationId = stationId;

}

**public** String getStationName() {

**return** stationName;

}

**public** **void** setStationName(String stationName) {

**this**.stationName = stationName;

}

**public** String getMould() {

**return** mould;

}

**public** **void** setMould(String mould) {

**this**.mould = mould;

}

}

**package** com.sdocean.dataQuery.model;

**public** **class** StatData {

**private** Integer value;

**private** String name;

**private** String xtime;

**private** String ydata;

**private** **int** statcount;

**private** String firstThing;

**private** String color;

**private** Double y;

**private** **int** standard\_grade;

**public** **int** getStandard\_grade() {

**return** standard\_grade;

}

**public** **void** setStandard\_grade(**int** standard\_grade) {

**this**.standard\_grade = standard\_grade;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** Double getY() {

**return** y;

}

**public** **void** setY(Double y) {

**this**.y = y;

}

**public** String getFirstThing() {

**return** firstThing;

}

**public** **void** setFirstThing(String firstThing) {

**this**.firstThing = firstThing;

}

**public** **int** getStatcount() {

**return** statcount;

}

**public** **void** setStatcount(**int** statcount) {

**this**.statcount = statcount;

}

**public** Integer getValue() {

**return** value;

}

**public** **void** setValue(Integer value) {

**this**.value = value;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getXtime() {

**return** xtime;

}

**public** **void** setXtime(String xtime) {

**this**.xtime = xtime;

}

**public** String getYdata() {

**return** ydata;

}

**public** **void** setYdata(String ydata) {

**this**.ydata = ydata;

}

}

**package com.sdocean.dataQuery.model;**

**import java.util.ArrayList;**

**import java.util.LinkedList;**

**import java.util.List;**

**public class StatisModel {**

**private String stationId;**

**private int statType;**

**private String beginDate;**

**private String endDate;**

**private List<StatData> datas; //饼形图数据**

**public List xtimes = new LinkedList(); //横坐标数据**

**public List<Object> ydatas = new ArrayList<Object>(); //纵坐标数据**

**private String indicatorIds;**

**public String getIndicatorIds() {**

**return indicatorIds;**

**}**

**public void setIndicatorIds(String indicatorIds) {**

**this.indicatorIds = indicatorIds;**

**}**

**public String getStationId() {**

**return stationId;**

**}**

**public void setStationId(String stationId) {**

**this.stationId = stationId;**

**}**

**public int getStatType() {**

**return statType;**

**}**

**public void setStatType(int statType) {**

**this.statType = statType;**

**}**

**public String getBeginDate() {**

**return beginDate;**

**}**

**public void setBeginDate(String beginDate) {**

**this.beginDate = beginDate;**

**}**

**public String getEndDate() {**

**return endDate;**

**}**

**public void setEndDate(String endDate) {**

**this.endDate = endDate;**

**}**

**public List<StatData> getDatas() {**

**return datas;**

**}**

**public void setDatas(List<StatData> datas) {**

**this.datas = datas;**

**}**

**public List getXtimes() {**

**return xtimes;**

**}**

**public void setXtimes(List xtimes) {**

**this.xtimes = xtimes;**

**}**

**public List<Object> getYdatas() {**

**return ydatas;**

**}**

**public void setYdatas(List<Object> ydatas) {**

**this.ydatas = ydatas;**

**}**

**}**

**package** com.sdocean.dataQuery.model;

**public** **class** Ydata {

**private** String value;

**private** String extra;

**public** String getValue() {

**return** value;

}

**public** **void** setValue(String value) {

**this**.value = value;

}

**public** String getExtra() {

**return** extra;

}

**public** **void** setExtra(String extra) {

**this**.extra = extra;

}

}

**package com.sdocean.dataQuery.service;**

**import java.util.List;**

**import java.util.Map;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.dataQuery.dao.DataQueryDao;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.firstpage.model.LastMetaData;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class DataQueryService {**

**@Autowired**

**private DataQueryDao dataQueryDao;**

**/\***

**\* 为实时数据添加表头**

**\*/**

**public List<UiColumn> getCols4DataQuery(DataQueryModel model){**

**return dataQueryDao.getCols4DataQuery(model);**

**}**

**/\***

**\* 为实时数据查询提供结果**

**\*/**

**public List<Map<String, Object>> getRows4DataQuery(DataQueryModel model){**

**return dataQueryDao.getRows4DataQuery(model);**

**}**

**/\***

**\* 查询出当前站点信息的最后实时数据**

**\* 为首页展示**

**\*/**

**public List<LastMetaData> getData4FirstPage(StationModel station){**

**return dataQueryDao.getData4FirstPage(station);**

**}**

**}**

**package com.sdocean.dataQuery.service;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpSession;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Echarts;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.dataQuery.dao.DataQueryDao;**

**import com.sdocean.dataQuery.dao.GraphQueryDao;**

**import com.sdocean.dataQuery.dao.SynthQueryDao;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.dataQuery.model.GraphModel;**

**import com.sdocean.device.dao.DeviceDao;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.dictionary.dao.PublicDao;**

**import com.sdocean.dictionary.model.PublicModel;**

**import com.sdocean.frame.model.ConfigInfo;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.dao.SysLoginLogDao;**

**import com.sdocean.log.model.SysLoginLogModel;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.position.dao.SysPositionDao;**

**import com.sdocean.position.model.SysPosition;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class GraphQueryService {**

**@Autowired**

**private DeviceDao deviceDao;**

**@Autowired**

**private GraphQueryDao graphDao;**

**/\***

**\* 初始化参数列表**

**\*/**

**public void modelInfo(GraphModel model){**

**String indicatorIds = model.getIndicatorIds();**

**String[] ids = indicatorIds.split(",");**

**Map<String, String> deids = new HashMap<String, String>();**

**for(String id:ids){**

**String indicatorid = id.substring(0, id.indexOf("#"));**

**String deviceid = id.substring(id.indexOf("#")+1,id.length());**

**if(deids.containsKey(deviceid)){**

**String indi = deids.get(deviceid)+","+indicatorid;**

**deids.remove(deviceid);**

**deids.put(deviceid, indi);**

**}else{**

**deids.put(deviceid, indicatorid);**

**}**

**}**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**for(String deviceId:deids.keySet()){**

**String indicatorid = deids.get(deviceId);**

**DeviceModel device = deviceDao.getDeviceByid(deviceId, indicatorid);**

**if(device!=null&&device.getIndicators()!=null&&device.getIndicators().size()>0){**

**list.add(device);**

**}**

**}**

**model.setDevices(list);**

**}**

**/\***

**\* 水质评价 综合查询**

**\*/**

**public Echarts getEcharts4Graph(StationModel station,GraphModel model){**

**return graphDao.getEcharts4Graph(station, model);**

**}**

**}**

**package com.sdocean.dataQuery.service;**

**import java.util.List;**

**import java.util.Map;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dataQuery.dao.DataQueryDao;**

**import com.sdocean.dataQuery.dao.MouldDao;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.dataQuery.model.MouldModel;**

**import com.sdocean.firstpage.model.LastMetaData;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.station.model.StationModel;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class MouldService {**

**@Autowired**

**private MouldDao mouldDao;**

**/\***

**\* 获得当前站点的水质评价模板**

**\*/**

**public MouldModel getMouldByStationId(MouldModel model){**

**return mouldDao.getMouldByStationId(model);**

**}**

**/\***

**\* 保存当前站点的水质评价模板**

**\*/**

**public Result saveMouldSetting(MouldModel model){**

**return mouldDao.saveMouldSetting(model);**

**}**

**/\***

**\* 根据站点,获得水质评价模板**

**\*/**

**public MouldModel getMouldByStation(StationModel model){**

**return mouldDao.getMouldByStation(model);**

**}**

**}**

**package com.sdocean.dataQuery.service;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Hcharts;**

**import com.sdocean.dataQuery.dao.StatisQueryDao;**

**import com.sdocean.dataQuery.model.StatData;**

**import com.sdocean.dataQuery.model.StatisModel;**

**import com.sdocean.dataQuery.model.Ydata;**

**import com.sdocean.firstpage.model.WaterStandard;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.station.model.StationModel;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class StatisQueryService {**

**@Autowired**

**private StatisQueryDao statisQueryDao;**

**/\***

**\* 获得水质统计**

**\*/**

**public StatisModel getStatDataSearch(StatisModel model,StationModel station){**

**//获得饼形图结果以及水质统计列表数据**

**List<StatData> pielist = statisQueryDao.getData4pie(model, station, 1);**

**for(StatData stat:pielist){**

**if(stat.getStandard\_grade()==1){**

**stat.setColor("#9dc3e4");**

**}else if(stat.getStandard\_grade()==2){**

**stat.setColor("#e2efd9");**

**}else if(stat.getStandard\_grade()==3){**

**stat.setColor("#70ad45");**

**}else if(stat.getStandard\_grade()==4){**

**stat.setColor("#ffd965");**

**}else if(stat.getStandard\_grade()==5){**

**stat.setColor("#ed1e24");**

**}else if(stat.getStandard\_grade()==6){**

**stat.setColor("#000000");**

**}**

**}**

**//得到线性数据**

**List<StatData> lineList = statisQueryDao.getStatDataSearchList(model, station, 1);**

**for(StatData statData:lineList){**

**model.xtimes.add(statData.getXtime());**

**Hcharts hc = new Hcharts();**

**hc.setY(Double.parseDouble(statData.getYdata()));**

**hc.setName(statData.getName());**

**model.ydatas.add(hc);**

**}**

**model.setDatas(pielist);**

**return model;**

**}**

**/\***

**\* 为首页提供水质**

**\*/**

**public WaterStandard getWaterStandard(StationModel watchPoint){**

**return statisQueryDao.getWaterStandard(watchPoint);**

**}**

**/\***

**\* 首页显示水质等级趋势**

**\*/**

**public StatisModel getStatDataSearch4First(Long userId,StatisModel gmodel,StationModel watchPoint){**

**//得到线性数据**

**List<StatData> lineList = statisQueryDao.getStatDataSearchList(gmodel, watchPoint, 1);**

**for(StatData statData:lineList){**

**gmodel.xtimes.add(statData.getXtime());**

**Hcharts hc = new Hcharts();**

**hc.setY(Double.parseDouble(statData.getYdata()));**

**hc.setName(statData.getName());**

**gmodel.ydatas.add(hc);**

**}**

**return gmodel;**

**}**

**}**

**package com.sdocean.dataQuery.service;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpSession;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.dataQuery.dao.DataQueryDao;**

**import com.sdocean.dataQuery.dao.SynthQueryDao;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.dao.DeviceDao;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.dictionary.dao.PublicDao;**

**import com.sdocean.dictionary.model.PublicModel;**

**import com.sdocean.frame.model.ConfigInfo;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.dao.SysLoginLogDao;**

**import com.sdocean.log.model.SysLoginLogModel;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.position.dao.SysPositionDao;**

**import com.sdocean.position.model.SysPosition;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class SynthQueryService {**

**@Autowired**

**private SynthQueryDao synthQueryDao;**

**@Autowired**

**private DeviceDao deviceDao;**

**/\***

**\* 为综合查询添加表头**

**\*/**

**public List<UiColumn> getCols4SynQuery(DataQueryModel model){**

**return synthQueryDao.getCols4SynQuery(model);**

**}**

**/\***

**\* 为综合查询查询结果**

**\*/**

**public List<Map<String, Object>> getRows4SynQuery(DataQueryModel model){**

**return synthQueryDao.getRows4SynQuery(model);**

**}**

**}**

**package com.sdocean.device.action;**

**import java.util.ArrayList;**

**import java.util.List;**

**import javax.annotation.Resource;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.device.service.DeviceService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Controller**

**public class DeviceAction {**

**private static Logger log = Logger.getLogger(DeviceAction.class);**

**@Autowired**

**private DeviceService deviceService;**

**@Resource**

**OperationLogService logService;**

**@RequestMapping("info\_device.do")**

**public ModelAndView info\_public(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/device/deviceInfo");**

**return mav;**

**}**

**/\***

**\* 为设备管理查询结果**

**\*/**

**@RequestMapping(value="showDevices.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showDevices(@ModelAttribute("model") DeviceModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//为查询结果增加表头**

**List<UiColumn> cols = deviceService.getCols4List();**

**result.setCols(cols);**

**List<DeviceModel> rows = deviceService.getDevices(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 保存修改的公共代码**

**\*/**

**@RequestMapping(value="saveDeviceChange.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveDeviceChange(@ModelAttribute("model") DeviceModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = deviceService.saveDeviceChange(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增的公共代码**

**\*/**

**@RequestMapping(value="saveNewDevice.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveNewDevice(@ModelAttribute("model") DeviceModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = deviceService.saveNewDevice(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增的公共代码**

**\*/**

**@RequestMapping(value="getDeviceListByStation.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getDeviceListByStation(@ModelAttribute("model") StationModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**list = deviceService.getDeviceListByStation(model);**

**return JsonUtil.toJson(list);**

**}**

**/\***

**\* 获得当前站点下的所有的设备的列表**

**\*/**

**@RequestMapping(value="getDevices4Station.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getDevices4Station(@ModelAttribute("model") StationModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**list = deviceService.getDevices4Station(model);**

**return JsonUtil.toJson(list);**

**}**

**/\***

**\* 获得当前站点下有展示权限的设备的列表**

**\*/**

**@RequestMapping(value="getDevices4Station4Show.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getDevices4Station4Show(@ModelAttribute("model") StationModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**list = deviceService.getDevices4Station4Show(model);**

**return JsonUtil.toJson(list);**

**}**

**/\***

**\* 查询所有的设备列表**

**\*/**

**@RequestMapping(value="getDevices.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getDevices(HttpServletRequest request,**

**HttpServletResponse response){**

**DeviceModel model = new DeviceModel();**

**List<DeviceModel> rows = deviceService.getDevices(model);**

**return JsonUtil.toJson(rows);**

**}**

**}**

**package com.sdocean.device.dao;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.springframework.stereotype.Component;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.common.model.ZTreeModel;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.dictionary.model.PublicModel;**

**import com.sdocean.frame.dao.OracleEngine;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.indicator.model.IndicatorModel;**

**import com.sdocean.log.model.SysLoginLogModel;**

**import com.sdocean.menu.model.CurrMenu;**

**import com.sdocean.menu.model.SysMenu;**

**import com.sdocean.role.model.RoleModel;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.users.model.SysUser;**

**@Component**

**public class DeviceDao extends OracleEngine{**

**/\***

**\* 查询符合条件的设备列表**

**\*/**

**public List<DeviceModel> getDevices(DeviceModel model){**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.id,a.name,a.code,a.deviceModel,a.brief,a.detail,a.ordercode,");**

**sql.append(" group\_concat(m.title SEPARATOR '|') as indicatorNames,");**

**sql.append(" group\_concat(m.indicatorid) as indicatorIds,pointnum,mainnum");**

**sql.append(" from device\_catalog a left join (select b.catalogId,b.indicatorid,c.title");**

**sql.append(" from device\_catalog\_indicator b,dm\_indicator c");**

**sql.append(" where b.indicatorid = c.id and c.isactive = 1) m");**

**sql.append(" on a.id = m.catalogid");**

**sql.append(" where 1=1");**

**//添加查询条件**

**if(model!=null&&model.getCode()!=null&&model.getCode().length()>0){**

**sql.append(" and ( a.name like '%").append(model.getCode()).append("%' or");**

**sql.append(" a.code like '%").append(model.getCode()).append("%') ");**

**}**

**sql.append(" group by a.id");**

**//添加排序**

**sql.append(" order by ordercode ");**

**list = this.queryObjectList(sql.toString(), DeviceModel.class);**

**return list;**

**}**

**/\***

**\* 根据设备ID,查询设备信息**

**\*/**

**public DeviceModel getDeviceByDeviceId(int deviceId){**

**DeviceModel device = new DeviceModel();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.id,a.name,a.code,a.deviceModel,a.brief,a.detail,a.ordercode,");**

**sql.append(" group\_concat(m.title SEPARATOR '|') as indicatorNames,");**

**sql.append(" group\_concat(m.indicatorid) as indicatorIds,pointnum,mainnum");**

**sql.append(" from device\_catalog a left join (select b.catalogId,b.indicatorid,c.title");**

**sql.append(" from device\_catalog\_indicator b,dm\_indicator c");**

**sql.append(" where b.indicatorid = c.id and c.isactive = 1) m");**

**sql.append(" on a.id = m.catalogid");**

**sql.append(" where a.id =").append(deviceId);**

**sql.append(" limit 1");**

**device = this.queryObject(sql.toString(), DeviceModel.class);**

**return device;**

**}**

**/\***

**\* 保存修改的设备信息**

**\*/**

**public Result saveDeviceChange(DeviceModel model){**

**Result result = new Result();**

**result.setDotype(result.UPDATE);**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("修改成功");**

**//验证code 是否唯一**

**StringBuffer checkSql = new StringBuffer("");**

**checkSql.append(" select count(1) from device\_catalog where code ='").append(model.getCode()).append("' and id <>").append(model.getId());**

**int cou = 0;**

**try {**

**cou = this.queryForInt(checkSql.toString(), null);**

**} catch (Exception e) {**

**e.printStackTrace();**

**result.setResult(result.FAILED);**

**result.setMessage("验证唯一性时失败");**

**return result;**

**}**

**if(cou>0){**

**result.setResult(result.FAILED);**

**result.setMessage("CODE重复,违反唯一性原则");**

**return result;**

**}**

**//开始保存数据**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" update device\_catalog set name=?,code=?,devicemodel=?,brief=?,detail=?,ordercode=?,pointNum=?,mainnum=? where id =?");**

**Object[] params = new Object[]{**

**model.getName(),model.getCode(),model.getDeviceModel(),**

**model.getBrief(),model.getDetail(),model.getOrderCode(),model.getPointNum(),model.getMainnum(),model.getId()**

**};**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), params);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("保存失败");**

**return result;**

**}**

**this.saveDeviceIndicator(model);**

**return result;**

**}**

**/\***

**\* 保存新增的设备**

**\*/**

**public Result saveNewDevice(DeviceModel model){**

**Result result = new Result();**

**result.setDotype(result.UPDATE);**

**result.setModel(JsonUtil.toJson(model));**

**result.setResult(result.SUCCESS);**

**result.setMessage("新增成功");**

**//验证code 是否唯一**

**StringBuffer checkSql = new StringBuffer("");**

**checkSql.append(" select count(1) from device\_catalog where code ='").append(model.getCode()).append("' ");**

**int cou = 0;**

**try {**

**cou = this.queryForInt(checkSql.toString(), null);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("验证唯一性时失败");**

**return result;**

**}**

**if(cou>0){**

**result.setResult(result.FAILED);**

**result.setMessage("CODE重复,违反唯一性原则");**

**return result;**

**}**

**//开始保存数据**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" insert into device\_catalog(name,code,devicemodel,brief,detail,ordercode,pointnum,mainnum) values(?,?,?,?,?,?,?,?)");**

**Object[] params = new Object[]{**

**model.getName(),model.getCode(),model.getDeviceModel(),**

**model.getBrief(),model.getDetail(),model.getOrderCode(),model.getPointNum(),model.getMainnum()**

**};**

**int res = 0;**

**try {**

**res = this.update(sql.toString(), params);**

**} catch (Exception e) {**

**result.setResult(result.FAILED);**

**result.setMessage("新增失败");**

**return result;**

**}**

**//获得保存的ID值\**

**String last = "SELECT LAST\_INSERT\_ID()";**

**int lastId = this.queryForInt(last, null);**

**model.setId(lastId);**

**this.saveDeviceIndicator(model);**

**return result;**

**}**

**public void saveDeviceIndicator(DeviceModel model){**

**//删除设备的参数**

**StringBuffer deleSql = new StringBuffer("");**

**deleSql.append(" delete from device\_catalog\_indicator where catalogid = ").append(model.getId());**

**this.update(deleSql.toString(), null);**

**String ids = model.getIndicatorIds();**

**String[] idStrings = ids.split(",");**

**StringBuffer val = new StringBuffer("(0,0)");**

**for(String id:idStrings){**

**val.append(",(").append(model.getId()).append(",").append(id).append(")");**

**}**

**StringBuffer insertSql = new StringBuffer("");**

**insertSql.append(" insert into device\_catalog\_indicator(catalogid,indicatorid) values").append(val);**

**insertSql.append(" ON DUPLICATE KEY UPDATE indicatorid = indicatorid ");**

**this.update(insertSql.toString(), null);**

**}**

**/\***

**\* 获得当前站点所关联的设备列表**

**\*/**

**public List<SelectTree> getDeviceListByStation(StationModel model){**

**List<SelectTree> list = new ArrayList<SelectTree>();**

**//添加设备列表的首层**

**SelectTree first = new SelectTree();**

**first.setId("0");**

**first.setName("设备");**

**first.setIsExpanded(true);**

**first.setIsActive(true);**

**//获得所有有效的设备的列表**

**List<SelectTree> children = new ArrayList<SelectTree>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.id,a.name, case when b.aiot\_watch\_point\_id is null then 'false' else 'true' end as selected");**

**sql.append(" from device\_catalog a");**

**sql.append(" left join map\_awp\_device\_catalog b on a.id = b.device\_catalog\_id");**

**sql.append(" and b.aiot\_watch\_point\_id = ").append(model.getId());**

**children = this.queryObjectList(sql.toString(), SelectTree.class);**

**//将设备列表添加的首层的child中**

**first.setChildren(children);**

**//将首层添加到结果集中**

**list.add(first);**

**return list;**

**}**

**/\***

**\* 获得当前站点下的设备的列表**

**\*/**

**public List<DeviceModel> getDevices4Station(StationModel model){**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select distinct a.id,a.name,a.code,a.deviceModel,a.brief,a.detail,a.ordercode");**

**sql.append(" from device\_catalog a,map\_awp\_device\_catalog b");**

**sql.append(" where a.id = b.device\_catalog\_id");**

**//添加查询条件**

**if(model!=null&&model.getId()>0){**

**sql.append(" and b.aiot\_watch\_point\_id =").append(model.getId());**

**}**

**sql.append(" order by a.ordercode");**

**list = this.queryObjectList(sql.toString(), DeviceModel.class);**

**return list;**

**}**

**/\***

**\* 获得当前站点下有展示权限的设备的列表**

**\*/**

**public List<DeviceModel> getDevices4Station4Show(StationModel model){**

**List<DeviceModel> list = new ArrayList<DeviceModel>();**

**StringBuffer sql = new StringBuffer("");**

**sql.append(" select a.id,a.name,a.code,a.deviceModel,a.brief,a.detail,a.ordercode");**

**sql.append(" from view\_show\_station\_device a");**

**sql.append(" where a.stationid =").append(model.getId());**

**sql.append(" order by a.ordercode");**

**list = this.queryObjectList(sql.toString(), DeviceModel.class);**

**return list;**

**}**

**/\***

**\*根据deviceid以及indicators**

**\*/**

**public DeviceModel getDeviceByid(String deviceId,String indicatorids){**

**DeviceModel model = new DeviceModel();**

**//查询出设备**

**StringBuffer devicesql = new StringBuffer("");**

**devicesql.append(" select a.id,a.name,a.code,a.deviceModel,a.brief,a.detail,a.ordercode");**

**devicesql.append(" from device\_catalog a where a.id = ").append(deviceId);**

**model = this.queryObject(devicesql.toString(), DeviceModel.class);**

**//查询出参数列表**

**StringBuffer indicatorSql = new StringBuffer("");**

**indicatorSql.append(" select a.id,a.code,a.title,a.groupId,a.unitId,a.description,a.isactive");**

**indicatorSql.append(" from dm\_indicator a where a.id in (").append(indicatorids).append(")");**

**List<IndicatorModel> indicators = new ArrayList<IndicatorModel>();**

**indicators = this.queryObjectList(indicatorSql.toString(), IndicatorModel.class);**

**if(indicators!=null&&indicators.size()>0){**

**model.setIndicators(indicators);**

**}**

**return model;**

**}**

**/\***

**\* 读取站点内设备读取数据的时间间隔**

**\*/**

**public DeviceModel getInterval4device(int stationid,DeviceModel device){**

**StringBuffer sql = new StringBuffer("");**

**sql.append("select max(betweentime) from aiot\_station\_device\_comm where stationid in (0, ").append(stationid).append(") and deviceid in (0").append(device.getId()).append(")");**

**Integer inter = this.queryForInt(sql.toString(), null);**

**if(inter==null){**

**inter = 0;**

**}**

**device.setInterval(inter);**

**return device;**

**}**

**}**

**package com.sdocean.device.model;**

**import java.util.List;**

**import com.sdocean.indicator.model.IndicatorModel;**

**public class DeviceModel {**

**private int id;**

**private String name;**

**private String code;**

**private String deviceModel;**

**private String brief;**

**private String detail;**

**private String orderCode;**

**private String indicatorIds;**

**private String indicatorNames;**

**private List<IndicatorModel> indicators;**

**private Integer interval;**

**private Integer pointNum; //小数点保留位数**

**private int mainnum; //例行维护周期**

**public int getMainnum() {**

**return mainnum;**

**}**

**public void setMainnum(int mainnum) {**

**this.mainnum = mainnum;**

**}**

**public Integer getPointNum() {**

**return pointNum;**

**}**

**public void setPointNum(Integer pointNum) {**

**this.pointNum = pointNum;**

**}**

**public Integer getInterval() {**

**return interval;**

**}**

**public void setInterval(Integer interval) {**

**this.interval = interval;**

**}**

**public List<IndicatorModel> getIndicators() {**

**return indicators;**

**}**

**public void setIndicators(List<IndicatorModel> indicators) {**

**this.indicators = indicators;**

**}**

**public String getIndicatorIds() {**

**return indicatorIds;**

**}**

**public void setIndicatorIds(String indicatorIds) {**

**this.indicatorIds = indicatorIds;**

**}**

**public String getIndicatorNames() {**

**return indicatorNames;**

**}**

**public void setIndicatorNames(String indicatorNames) {**

**this.indicatorNames = indicatorNames;**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getCode() {**

**return code;**

**}**

**public void setCode(String code) {**

**this.code = code;**

**}**

**public String getDeviceModel() {**

**return deviceModel;**

**}**

**public void setDeviceModel(String deviceModel) {**

**this.deviceModel = deviceModel;**

**}**

**public String getBrief() {**

**return brief;**

**}**

**public void setBrief(String brief) {**

**this.brief = brief;**

**}**

**public String getDetail() {**

**return detail;**

**}**

**public void setDetail(String detail) {**

**this.detail = detail;**

**}**

**public String getOrderCode() {**

**return orderCode;**

**}**

**public void setOrderCode(String orderCode) {**

**this.orderCode = orderCode;**

**}**

**}**

**package com.sdocean.device.service;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Propagation;**

**import org.springframework.transaction.annotation.Transactional;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.common.model.SelectTree;**

**import com.sdocean.dataQuery.model.DataQueryModel;**

**import com.sdocean.device.dao.DeviceDao;**

**import com.sdocean.device.model.DeviceModel;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**@Service**

**@Transactional(rollbackFor=Exception.class, propagation=Propagation.REQUIRED)**

**public class DeviceService {**

**@Autowired**

**private DeviceDao deviceDao;**

**/\***

**\* 为公共代码管理提供表头**

**\*/**

**public List<UiColumn> getCols4List(){**

**List<UiColumn> cols = new ArrayList<UiColumn>();**

**UiColumn col1 = new UiColumn("id", "id", true, "\*");**

**UiColumn col2 = new UiColumn("设备名称", "name", true, "\*");**

**UiColumn col3 = new UiColumn("设备编码", "code", true, "\*");**

**UiColumn col4 = new UiColumn("设备模型", "deviceModel", true, "\*");**

**UiColumn col5 = new UiColumn("介绍", "brief", true, "\*");**

**UiColumn col6 = new UiColumn("明细", "detail", true, "\*");**

**UiColumn col19 = new UiColumn("orderCode", "orderCode", true, "\*");**

**UiColumn col10 = new UiColumn("保留位数", "pointNum", true, "\*");**

**UiColumn col12 = new UiColumn("维护周期(天)", "mainnum", true, "\*");**

**UiColumn col11 = new UiColumn("监测参数", "indicatorNames", true, "\*");**

**cols.add(col1);**

**cols.add(col2);**

**cols.add(col3);**

**cols.add(col6);**

**cols.add(col4);**

**cols.add(col5);**

**cols.add(col19);**

**cols.add(col10);**

**cols.add(col12);**

**cols.add(col11);**

**return cols;**

**}**

**/\***

**\* 查询符合条件的设备列表**

**\*/**

**public List<DeviceModel> getDevices(DeviceModel model){**

**return deviceDao.getDevices(model);**

**}**

**/\***

**\* 保存修改的设备信息**

**\*/**

**public Result saveDeviceChange(DeviceModel model){**

**return deviceDao.saveDeviceChange(model);**

**}**

**/\***

**\* 保存新增的设备**

**\*/**

**public Result saveNewDevice(DeviceModel model){**

**return deviceDao.saveNewDevice(model);**

**}**

**/\***

**\* 获得当前站点所关联的设备列表**

**\*/**

**public List<SelectTree> getDeviceListByStation(StationModel model){**

**return deviceDao.getDeviceListByStation(model);**

**}**

**/\***

**\* 获得当前站点下的设备的列表**

**\*/**

**public List<DeviceModel> getDevices4Station(StationModel model){**

**return deviceDao.getDevices4Station(model);**

**}**

**/\***

**\* 获得当前站点下有展示权限的设备的列表**

**\*/**

**public List<DeviceModel> getDevices4Station4Show(StationModel model){**

**return deviceDao.getDevices4Station4Show(model);**

**}**

**}**

**package com.sdocean.dictionary.action;**

**import java.util.ArrayList;**

**import java.util.List;**

**import javax.annotation.Resource;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.apache.log4j.Logger;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dictionary.model.PublicModel;**

**import com.sdocean.dictionary.model.UnitModel;**

**import com.sdocean.dictionary.service.PublicService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.position.model.SysPosition;**

**import com.sdocean.position.service.SysPositionService;**

**import com.sdocean.users.model.SysUser;**

**@Controller**

**public class PublicAction {**

**private static Logger log = Logger.getLogger(PublicAction.class);**

**@Autowired**

**private PublicService publicService;**

**@Resource**

**OperationLogService logService;**

**@RequestMapping("info\_public.do")**

**public ModelAndView info\_public(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dictionary/publicInfo");**

**return mav;**

**}**

**/\***

**\* 通过parentCode查询到公共代码列表**

**\*/**

**@RequestMapping(value="getPublicList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getPublicList(@ModelAttribute("model") PublicModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<PublicModel> publics = publicService.getPublicsByParent(model.getParentCode());**

**return JsonUtil.toJson(publics);**

**}**

**/\***

**\* 为公共代码管理查询结果**

**\*/**

**@RequestMapping(value="showPublics.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showPublics(@ModelAttribute("model") PublicModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//为查询结果增加表头**

**List<UiColumn> cols = publicService.getCols4List();**

**result.setCols(cols);**

**List<PublicModel> rows = publicService.getPublics(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 保存修改的公共代码**

**\*/**

**@RequestMapping(value="savePublicChange.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String savePublicChange(@ModelAttribute("model") PublicModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = publicService.savePublicChange(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增的公共代码**

**\*/**

**@RequestMapping(value="saveNewPublic.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveNewPublic(@ModelAttribute("model") PublicModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = publicService.saveNewPublic(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 删除选中的编码**

**\*/**

**@RequestMapping(value="delePublics.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String delePublics(@ModelAttribute("model") PublicModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = publicService.delePublic(model.getIds());**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**}**

**package com.sdocean.dictionary.action;**

**import java.util.List;**

**import javax.annotation.Resource;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dictionary.model.UnitModel;**

**import com.sdocean.dictionary.service.UnitService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**@Controller**

**public class UnitAction {**

**@Resource**

**UnitService unitService;**

**@Resource**

**OperationLogService logService;**

**@RequestMapping("info\_unit.do")**

**public ModelAndView info\_unit(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dictionary/unitInfo");**

**return mav;**

**}**

**@RequestMapping(value="showUnitList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showUnitList(@ModelAttribute("model") UnitModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//为查询结果增加表头**

**List<UiColumn> cols = unitService.getCols4GroupList();**

**result.setCols(cols);**

**//为查询结果增加列表**

**List<UnitModel> rows = unitService.getUnitList(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 获得单位列表**

**\*/**

**@RequestMapping(value="getUnitList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getUnitList(@ModelAttribute("model") UnitModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**List<UnitModel> rows = unitService.getUnitList(model);**

**return JsonUtil.toJson(rows);**

**}**

**/\***

**\* 保存修改信息**

**\*/**

**@RequestMapping(value="saveUnitChange.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveUnitChange(@ModelAttribute("model") UnitModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = unitService.saveUnitChange(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增信息**

**\*/**

**@RequestMapping(value="saveNewUnit.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveNewUnit(@ModelAttribute("model") UnitModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = unitService.saveNewUnit(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**}** **package com.sdocean.dictionary.action;**

**import java.util.ArrayList;**

**import java.util.List;**

**import javax.annotation.Resource;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.web.bind.annotation.ModelAttribute;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RequestMethod;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**import org.springframework.web.servlet.ModelAndView;**

**import com.sdocean.common.model.Result;**

**import com.sdocean.dictionary.dao.UnitGroupDao;**

**import com.sdocean.dictionary.model.UnitGroupModel;**

**import com.sdocean.dictionary.service.UnitGroupService;**

**import com.sdocean.frame.util.JsonUtil;**

**import com.sdocean.log.service.OperationLogService;**

**import com.sdocean.page.model.NgColumn;**

**import com.sdocean.page.model.PageResult;**

**import com.sdocean.page.model.UiColumn;**

**import com.sdocean.station.model.StationModel;**

**import com.sdocean.station.service.StationService;**

**@Controller**

**public class UnitGroupAction {**

**@Resource**

**UnitGroupService unitGroupService;**

**@Resource**

**OperationLogService logService;**

**@RequestMapping("info\_unit\_group.do")**

**public ModelAndView info\_unit\_group(HttpServletRequest request,**

**HttpServletResponse response)throws Exception{**

**ModelAndView mav = new ModelAndView("/dictionary/unitGroup");**

**return mav;**

**}**

**@RequestMapping(value="showGroupList.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String showGroupList(@ModelAttribute("model") UnitGroupModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**PageResult result = new PageResult();**

**//为查询结果增加表头**

**List<UiColumn> cols = unitGroupService.getCols4GroupList();**

**result.setCols(cols);**

**//为查询结果增加列表**

**List<UnitGroupModel> rows = unitGroupService.getUnitGroups(model);**

**result.setRows(rows);**

**return JsonUtil.toJson(result);**

**}**

**/\***

**\* 保存修改信息**

**\*/**

**@RequestMapping(value="saveGroupChange.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveGroupChange(@ModelAttribute("model") UnitGroupModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = unitGroupService.saveGroupChange(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 保存新增信息**

**\*/**

**@RequestMapping(value="saveNewGroup.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String saveNewGroup(@ModelAttribute("model") UnitGroupModel model,HttpServletRequest request,**

**HttpServletResponse response){**

**Result result = new Result();**

**result = unitGroupService.saveNewGroup(model);**

**logService.saveOperationLog(result,request);**

**return result.getMessage();**

**}**

**/\***

**\* 获得有效的单位组的集合**

**\*/**

**@RequestMapping(value="getUnitGroups.do", method = RequestMethod.POST,produces = "application/json;charset=UTF-8")**

**@ResponseBody**

**public String getUnitGroups(HttpServletRequest request,**

**HttpServletResponse response){**

**List<UnitGroupModel> list = new ArrayList<UnitGroupModel>();**

**list = unitGroupService.getUnitGroups();**

**return JsonUtil.toJson(list);**

**}**

**}**