# 核心技术整理

## 增强BaseDao

抽取分页条件查询，集成复杂条件封装

|  |
| --- |
| /\*\*  \* 根据传过来的Query类，封装DetachedCriteria条件  \* **@param** t  \* **@return**  \*/  **protected** DetachedCriteria getDetachedCriteria(T t) {  **return** DetachedCriteria.*forClass*(beanClass);  }    @Override  **public** List<T> findAll(T t) {  DetachedCriteria dc = **this**.getDetachedCriteria(t);  **return** (List<T>) **this**.getHibernateTemplate().findByCriteria(dc);  }  @Override  **public** List<T> findAll(T t, **int** offset, **int** limit) {  DetachedCriteria dc = **this**.getDetachedCriteria(t);  **return** (List<T>) **this**.getHibernateTemplate().findByCriteria(dc, offset, limit);  }    /\*\*  \* 遍历实体对象的属性，把值为空白字符串的属性设置为null，非空白字符串trim一下  \*/  **protected** **void** Blank2Null(Object entity) {  **try** {  PropertyUtilsBean propertyUtilsBean = **new** PropertyUtilsBean();  PropertyDescriptor[] descriptors = propertyUtilsBean.getPropertyDescriptors(entity);    **for** (**int** i = 0; i < descriptors.length; i++) {    String name = descriptors[i].getName(); //属性名称    **if** (!StringUtils.*equals*(name, "class")) { ///判断如果不是class    Object value = propertyUtilsBean.getNestedProperty(entity, name); //获取该属性的值    **if** (value **instanceof** String) {//如果这个值是String  String valueStr = (String)value;  **if** (StringUtils.*isBlank*(valueStr)) {//如果属性值为空串  propertyUtilsBean.setNestedProperty(entity, name, **null**);//设置属性值为null  } **else** {  propertyUtilsBean.setNestedProperty(entity, name, valueStr.trim());//设置属性值为null  }  }    }  }  } **catch** (Exception e) {  e.printStackTrace();  }  }    /\*\*  \* 获得Example把实体对象封装为条件，同时把实体对象的值为空白字符串的属性设置为null，文本类型使用ilike模糊匹配  \*/  **protected** Example getExample(Object entiry) {  **if** (entiry==**null**) {  **return** **null**;  }  Blank2Null(entiry);  **return** Example.*create*(entiry).enableLike(MatchMode.***ANYWHERE***).ignoreCase().excludeZeroes();  } |

## 统一三角色采购单查询

### Service层

|  |
| --- |
| @Override  **public** **int** getTotalRecord(YycgdQueryCustom yycgdQueryCustom) {  **return** yycgdDao.getTotalRecord(getYycgQuery(yycgdQueryCustom));  }  @Override  **public** List<Yycgd> findAll(YycgdQueryCustom yycgdQueryCustom, **int** offset, **int** limit) {  List<Yycgd> list = yycgdDao.findAll(getYycgQuery(yycgdQueryCustom), offset, limit);    **return** list;  }  // 查询条件封装  **private** YycgdQuery getYycgQuery(YycgdQueryCustom yycgdQueryCustom) {  YycgdQuery yycgdQuery = **new** YycgdQuery();  **if** (yycgdQueryCustom != **null**) {  // 拷贝属性，如果传入yycgdQueryCustom对象不为null  BeanUtils.*copyProperties*(yycgdQueryCustom, yycgdQuery);  }  // 获得当前用户，根据角色设定查询条件  ActiveUser activeUser = (ActiveUser) SecurityUtils.*getSubject*().getPrincipal();  String groupid = activeUser.getGroupid();  SysDictInfo sysDictInfo = sysDictInfoDao.findById(groupid);  String dictcode = sysDictInfo.getDictcode();  // 根据dictcode判断单位  **if** ("1".equals(dictcode) || "2".equals(dictcode)) {  // 查询卫生局或卫生院  DwWsy dwWsy = dwWsyDao.findById(activeUser.getSysid());  // 获得地区  SysArea sysArea = dwWsy.getSysArea();  // 存入YycgdQuery  yycgdQuery.setSysArea(sysArea);  } **else** **if** ("3".equals(dictcode)) { // 卫生室  DwWss dwWss = **new** DwWss();  dwWss.setId(activeUser.getSysid());  yycgdQuery.setDwWss(dwWss);  } **else** **if** ("4".equals(dictcode)) { // 供应商  DwGys dwGys = **new** DwGys();  dwGys.setId(activeUser.getSysid());  yycgdQuery.setDwGys(dwGys);  }  **return** yycgdQuery;  } |

### Query对象

|  |
| --- |
| **public** **class** YycgdQuery **extends** Yycgd {    **private** SysArea sysArea; //卫生室或卫生局地区 |

### Dao 层

|  |
| --- |
| @Override  **protected** DetachedCriteria getDetachedCriteria(Yycgd t) {  DetachedCriteria dc = **super**.getDetachedCriteria(t);  YycgdQuery query = (YycgdQuery) t;    //判断是否有状态参数  **boolean** hasZt = query.getSysDictInfoByzt()!=**null** && StringUtils.*isNotBlank*(query.getSysDictInfoByzt().getId());    **if** (query.getDwWss()!=**null**) { //卫生室  dc.add(Restrictions.*eq*("dwWss", query.getDwWss()));  } **else** **if** (query.getDwGys()!=**null**) { //供应商    dc.createCriteria("dwGys").add(Restrictions.*idEq*(query.getDwGys().getId()));  // 供应商只能查看状态为已审核01003和已受理01005的采购单  **if** (!hasZt) {  dc.add(Restrictions.*or*(  Restrictions.*eq*("sysDictInfoByzt.id", "01003"),  Restrictions.*eq*("sysDictInfoByzt.id", "01005")  ));  }  } **else** { //卫生院或卫生局  SysArea sysArea = query.getSysArea();  //加入子查询dwWss再加入子查询sysArea，地区id模糊匹配卫生局或卫生院的地区码wssDq  dc.createCriteria("dwWss").createCriteria("sysArea").add(Restrictions.*like*("id", sysArea.getId(), MatchMode.***START***));  //卫生院或卫生局 只能够查询 提交未审核 01002和审核未通过01004两种状态  **if** (!hasZt) {  dc.add(Restrictions.*or*(  Restrictions.*eq*("sysDictInfoByzt.id", "01002"),  Restrictions.*eq*("sysDictInfoByzt.id", "01003")  ));  }  }    //采购单状态00101  **if** (hasZt) {  dc.add(Restrictions.*eq*("sysDictInfoByzt.id", query.getSysDictInfoByzt().getId()));  }    //封装其它查询条件，采购单名称  dc.add(getExample(t));    //根据采购单编号id精确 查询  Integer id = t.getId();  **if** (id!=**null** && id!=0) {  dc.add(Restrictions.*idEq*(id));  }  **return** dc;  } |

## 通过药品信息子条件查询采购单明细

|  |
| --- |
| @Override  **protected** DetachedCriteria getDetachedCriteria(YycgdMx t) {  YycgdMxQuery yycgdMxQuery = (YycgdMxQuery) t;  DetachedCriteria detachedCriteria = DetachedCriteria.*forClass*(YycgdMx.**class**);  **if** (yycgdMxQuery != **null**) {    // 采购单对应的采购单详情  **if** (yycgdMxQuery.getYycgd() != **null**) {  detachedCriteria.add(Restrictions.*eq*("yycgd", yycgdMxQuery.getYycgd()));  }  // 和药品信息有关的条件  **if** (yycgdMxQuery.getYpxx() != **null**) {  //创建药品相关条件  DetachedCriteria dcYpxx = detachedCriteria.createCriteria("ypxx");    // 交易状态  SysDictInfo jyzt = yycgdMxQuery.getYpxx().getSysDictInfoByJyzt();  **if** (jyzt != **null** && StringUtils.*isNotBlank*(jyzt.getId())) {  dcYpxx.add(Restrictions.*eq*("sysDictInfoByJyzt", jyzt));  }    // 药品类别  SysDictInfo lb = yycgdMxQuery.getYpxx().getSysDictInfoByLb();  **if** (lb != **null** && StringUtils.*isNotBlank*(lb.getId())) {  dcYpxx.add(Restrictions.*eq*("sysDictInfoByLb", lb));  }    // 价格范围  Float zbjglower = yycgdMxQuery.getZbjglower();  **if** (zbjglower != **null** && zbjglower != 0) {  dcYpxx.add(Restrictions.*gt*("zbjg", yycgdMxQuery.getZbjglower()));  }  Float zbjgupper = yycgdMxQuery.getZbjgupper();  **if** ( zbjgupper!= **null** && zbjgupper != 0) {  dcYpxx.add(Restrictions.*lt*("zbjg", yycgdMxQuery.getZbjgupper()));  }    //转换系数精确匹配  String zhxs = yycgdMxQuery.getYpxx().getZhxs();  **if**(StringUtils.*isNotBlank*(zhxs)) {  dcYpxx.add(Restrictions.*eqOrIsNull*("zhxs", zhxs));  }    //普通文本字段统一处理为模糊匹配  dcYpxx.add(getExample(t.getYpxx()));  }    // 采购状态  SysDictInfo cgzt = yycgdMxQuery.getSysDictInfoCgzt();  **if** (cgzt != **null** && StringUtils.*isNotBlank*(cgzt.getId())) {  detachedCriteria.add(Restrictions.*eq*("sysDictInfoCgzt.id", cgzt.getId()));  }  }  **return** detachedCriteria;  } |

## 动态菜单及权限维护

菜单排序问题解决，老师提供的思路，本组与第二组各自不同时间独立完成。其它同学已经讲过，此处省略一万字。

权限表增加和删除

权限表修改

## 主题设置功能

common\_css.jsp中引入样式，设置id

|  |
| --- |
| <LINK id=*"themeCssId"* rel=*"stylesheet"* type=*"text/css"* href=*"*${baseurl}*/js/easyui/themes/default/easyui.css"* /> |

common\_js.jsp中引入脚本

|  |
| --- |
| <script type=*"text/javascript"* src=*"*${baseurl}*/js/easyui/easytheme.js"*></script> |

first.jsp中在适当位置编写一个select设置class为changeTheme

|  |
| --- |
| <SPAN style="padding-left: *10px*; font-size: *16px*;"><IMG  align=*"absmiddle"* src=*"images/blocks.gif"* width=*"20"* height=*"20"*>  医药集中采购系统</SPAN> <SPAN style="padding-left: *15px*;" id=*"News"*>  <!-- 主题切换 -->主题设置  <select id=*"changeThemeIdselect"* class=*"changeTheme"*></select>  </SPAN> |

|  |
| --- |
| /\*\*  \* 主题修改  \* 王吉@黑马程序员就业班37期  \*  \* 引入样式themeCssId  \* 修改样式方法changeTheme  \* 修改样式themesId  \*/  **var** themeName = "theme"; //在cookie中保存主题名的name名，即主题变量名  **var** themeClass = "changeTheme"; //自动绑定修改主题事件到此class上  **var** themeCss = "themeCssId";  **var** themesUrl = $("#themeCssId")[0].href;  $(**function**(){  //给class=变量themeClass的select元素绑定修改主题事件  $("."+themeClass).changeTheme();  themesUrl = $("#themeCssId")[0].href;  mytheme = getCookie(themeName);  changeTheme(mytheme);  });  //$(对象).changeTheme();  $.fn.changeTheme = **function**(){  //this 代表jquery选择器被选中的对象，是jquery对象  $this = **this**;  **var** themesUrl = $("#themeCssId")[0].href;  //初始化主题列表  **var** themes = {"default":"默认主题",  "bootstrap":"布特斯穿破",  "material":"棉花糖",  "ui-pepper-grinder":"胡椒粉",  "gray":"星空灰",  "black":"梦幻黑",  "ui-dark-hive":"黑暗蜂巢",  "ui-sunny":"夕阳晚霞",  "ui-cupertino":"蓝瘦香菇",  "metro-blue":"美俏天空蓝",  "metro-gray":"美俏都市灰 ",  "metro-green":"美俏典雅绿 ",  "metro-orange":"美俏动感橙 ",  "metro-red":"美俏樱桃红",  "metro":"美俏象牙白"};  **for** (**var** key **in** themes) {  $("<option></option>").val(key).text(themes[key]).appendTo($this);  }    //读取Cookie看有没有保存主题变量，如果有设置select的默认值  mytheme = getCookie(themeName);  **if** (mytheme!="") {  //console.log($this[0].id);  //按class修改，变量themeClass的值是class的值  //var $curOption = $("."+themeClass+" option[value='"+mytheme+"']");  //$curOption.prop("selected",true);  //按id修改，this[0].id是id  //$curOption = $("#"+this[0].id+" option[value='"+mytheme+"']");  //$curOption.prop("selected",true);  //按select的子元素修改  $this.children("[value='"+mytheme+"']").prop("selected",**true**);;  }    $this.bind("change",**function**(event) {  changeTheme(**this**.value);  saveTheme(**this**.value);  //location.reload();  //刷新所有框架  $("iframe").each(**function**(idx, ele){  ele.contentWindow.location.reload();  });  $("frame").each(**function**(idx, ele){  ele.contentWindow.location.reload();  });  });  };  //调用changeTheme(主题名);  **function** changeTheme(theme){  **if** (**typeof** theme=="string" && theme!="") {  **var** href = themesUrl.replace("default",theme);  $("#themeCssId")[0].href = href;  }  }  //保存主题  **function** saveTheme(theme) {  // save Theme  **if** (**typeof** theme=="string" && theme!="" ){  setCookie(themeName,theme,100);  }  }  **function** setCookie(c\_name,value,expiredays)  {  **var** exdate=**new** Date()  exdate.setDate(exdate.getDate()+expiredays)  document.cookie=c\_name+ "=" +escape(value)+  ((expiredays==**null**) ? "" : ";expires="+exdate.toGMTString())  }  **function** getCookie(c\_name)  {  **if** (document.cookie.length>0)  {  c\_start=document.cookie.indexOf(c\_name + "=")  **if** (c\_start!=-1)  {  c\_start=c\_start + c\_name.length+1  c\_end=document.cookie.indexOf(";",c\_start)  **if** (c\_end==-1) c\_end=document.cookie.length  **return** unescape(document.cookie.substring(c\_start,c\_end))  }  }  **return** ""  } |