# 工作周报

# ---杨涛

# ---07-31（Start 07-31End 08-04）

**开发工作内容**

系统内开发计划的完成情况，包括计划进度，完成情况，如果未完成，写明未完成的原因或出现的问题，以及拟采用的方法措施

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **时间 任务** | **计划进度** | **完成情况** | **未完成原因** | **拟采取的方法** | **备注** |
| **07-31** | **对所做的单据进行测试，1改进了字段“会计期间”与“年度”在修改、复制等操作时的可编辑性；2修复了列表页面看不到字段内容的问题** | **完成** |  |  |  |
| **08-01** | **查询页面的bug修复：3.修复核算要素期间参考无法显示问题；4.修复筛选会计期间查询结果为0的问题** | **完成** |  |  |  |
| **08-02** | **5．分析分页功能的执行过程，并初步完成配置文件的配置** | **完成** |  |  |  |
| **08-03** | **6.完成分页条的加载和分页功能；** | **完成** |  |  |  |
| **08-04** | **7. 修复刷新后分页功能失效的问题，并完善相应的文档，为之后的分页实现提供模板参考** | **完成** |  |  |  |

**工作内容总结：**

1.复制

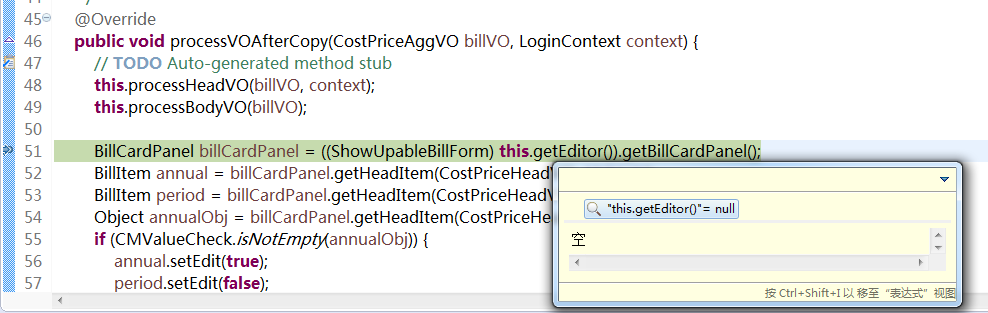
改进：点击复制时，修改“会计期间”和“年度”字段的可编辑性：当会“会计期间”有值时可编辑，而“年度”不可编辑，反正亦然。由于复制是基于现有数据的，所以正常情况下现有数据中两者有且只有一个有值，一些异常插入的数据两者可能都没值或两者都有值，也需要做处理。

问题：空指针错误

[AWT-EventQueue-0] ERROR - 未知的错误

java.lang.NullPointerException at nc.ui.mapub.costpricebase.interceptor.CopyActionProcessor.processVOAfterCopy(CopyActionProcessor.java:51) at nc.ui.mapub.costpricebase.interceptor.CopyActionProcessor.processVOAfterCopy(CopyActionProcessor.java:1)

发现无法获得编辑模板：



查看配置文件，copyAction中有editor属性，但是*copyActionProcessor*拦截器没有这个属性，无法获取到编辑模板。

<!-- 复制 -->

<bean id=*"copyAction"* class=*"nc.ui.pubapp.uif2app.actions.CopyAction"*>

<property name=*"model"* ref=*"manageAppModel"* />

<property name=*"interceptor"* ref=*"showCardInterceptor"* />

<property name=*"editor"* ref=*"billFormEditor"* />

<property name=*"copyActionProcessor"*>

<bean class=*"nc.ui.mapub.costpricebase.interceptor.CopyActionProcessor"*></bean>

</property>

</bean>

重新进行配置之后能够获取模板，

<!-- 复制 -->

<bean id=*"copyAction"* class=*"nc.ui.pubapp.uif2app.actions.CopyAction"*>

<property name=*"model"* ref=*"manageAppModel"* />

<property name=*"interceptor"* ref=*"showCardInterceptor"* />

<property name=*"editor"* ref=*"billFormEditor"* />

<property name=*"copyActionProcessor"* ref=*"copyInterceptor*>

</property>

</bean>

<!-- 复制拦截器 -->

<bean id=*"copyInterceptor"* class=*"nc.ui.mapub.costpricebase.interceptor.CopyActionProcessor"*>

<property name=*"model"* ref=*"manageAppModel"* />

<property name=*"editor"* ref=*"billFormEditor"* />

</bean>

对年度和会计期间的克编辑性设置如下：

BillCardPanel billCardPanel = ((ShowUpableBillForm) **this**.getEditor()).getBillCardPanel();

BillItem annual = billCardPanel.getHeadItem(CostPriceHeadVO.*ANNUAL*);

BillItem period = billCardPanel.getHeadItem(CostPriceHeadVO.*VPERIOD*);

Object annualObj = annual.getValueObject();

Object periodObj = period.getValueObject();

// 年度不为空，会计为空

**if** (CMValueCheck.*isNotEmpty*(annualObj) && CMValueCheck.*isEmpty*(periodObj)) {

annual.setEdit(**true**);

period.setEdit(**false**);

}

// 年度为空，会计不为空

**else** **if** (CMValueCheck.*isEmpty*(annualObj) && CMValueCheck.*isNotEmpty*(periodObj)) {

annual.setEdit(**false**);

period.setEdit(**true**);

}

// 年度为空，会计也为空or 年度不为空，会计也不为空

**else** {

annual.setEdit(**true**);

period.setEdit(**true**);

}

2.查询时，选择“年度”后，查询出错。

Caused by: nc.vo.pub.BusinessException: sqlexception caught while run query.sql: select innercode from bd\_accperiodscheme where bd\_accperiod.pk\_accperiod in ('10011410000000000009') ORA-00904: "BD\_ACCPERIOD"."PK\_ACCPERIOD": 标识符无效

at nc.bs.dbcache.impl.DBCacheBS.runSQLQuery(DBCacheBS.java:274)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)

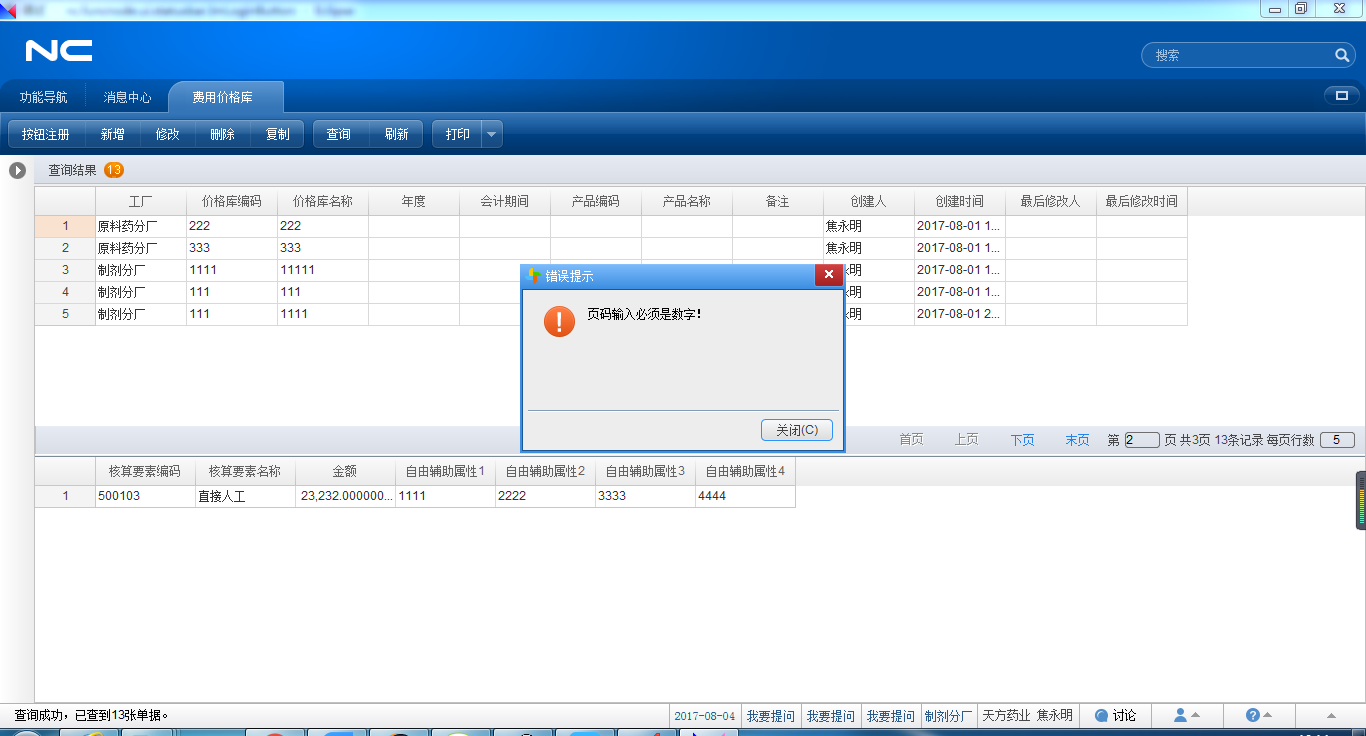
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)

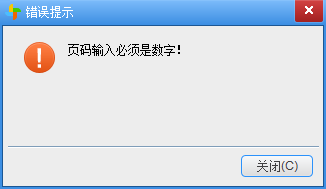
at java.lang.reflect.Method.invoke(Unknown Source) at nc.bs.framework.aop.rt.MethodProceedingJoinpoint$3.invoke(MethodProceedingJoinpoint.java:74) at nc.bs.framework.aop.rt.MethodProceedingJoinpoint.proceed(MethodProceedingJoinpoint.java:184) at nc.bs.framework.server.SecurityAspect.aroundMethod1(SecurityAspect.java:55)

问题分析：调试发现进行查询之前，平台会对查询的条件进行验证，由于年度为参照，需要验证所选择的年度是否在数据库中存在，问题在于在表accperidscheme中无法找到bd.accperiod.pk\_accperiod字段，分析原因为年度的参照设置不正确，但是重新设置参照后问题依旧。

1. 分页问题错误及解决

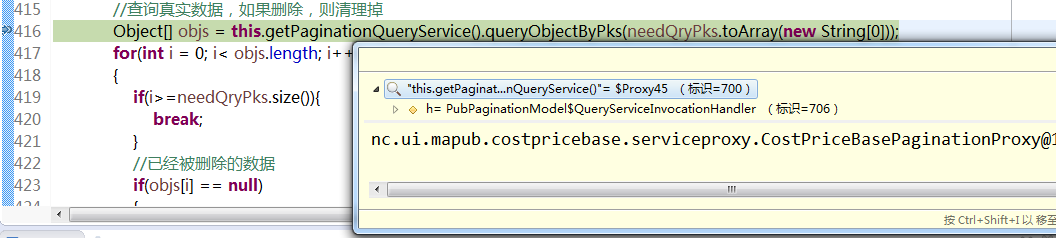
问题分析：查询后分页功能正常，但是刷新之后，无法点击下一页，设置页码和每页行数时出错

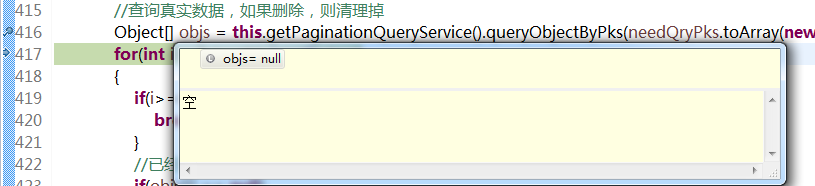




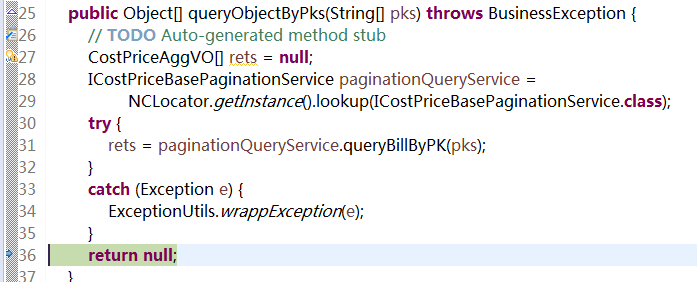
on in thread "AWT-EventQueue-0" java.lang.NullPointerExceptionat nc.ui.uif2.components.pagination.PaginationModel.prepareData(PaginationModel.java:417)at nc.ui.uif2.components.pagination.PaginationModel.structChange(PaginationModel.java:368)at nc.ui.uif2.components.pagination.PaginationModel.setPageIndex(PaginationModel.java:128)at nc.ui.uif2.components.pagination.PaginationModel.next(PaginationModel.java:302)at nc.ui.uif2.components.pagination.PaginationBar.actionPerformed(PaginationBar.java:174)at javax.swing.AbstractButton.fireActionPerformed(Unknown Source) ...

问题分析及解决：

调试发现刷新时候通过代理服务与后台交互，而查询则是通过查询服务接口，所以导致查询时正常而刷新时出错。

接着调试得到查询结果为空，但实际上是有数据的，说明查询数据过程中出错。进入queryObjectPks方法中查看。

发现了问题所在如下



最后返回查询结果时写错成返回空了，所以得不到查询结果导致分页页面出现的错误。

从以上的调试过程中发现，进行查询和刷新时，所使用的方法不同，刷新通过代理进行后台的交互，且由于刷新时已经进行过查询，即已经获得了查询数据的主键数组，刷新只需要从主键数组中选择当前页面所需要显示的行数，而不需要返回所有主键数组的结果。

4.刷新页面配置及实现过程

分页配置：

（1）在列表视图中添加分页控件，新增内容为红色字体部分：

<bean id=*"listView"* class=*"nc.ui.mapub.costpricebase.view.CostPriceListView"*

init-method=*"initUI"*>

<property name=*"model"* ref=*"manageAppModel"*/>

<property name=*"templateContainer"* ref=*"templateContainer"*/>

<property name=*"paginationBar"* ref=*"paginationBar"*></property>

</bean>

**注：**若上述listView使用的是平台封装的类nc.ui.pubapp.uif2app.view.ShowUpableBillListView，则不需要对分页条进行初始化，因为平台的初始化方法中已经设置了。但如果这里才自己的类来实现，由于重写了initUI方法，需要加入分页条，如红色字体代码部分：

**public** **class** CostPriceListView **extends** ShowUpableBillListView {

**private** **static** **final** **long** ***serialVersionUID*** = -8906686830841878902L;

**private** PaginationBar paginationBar;

@Override

**public** PaginationBar getPaginationBar() {

**return** **this**.paginationBar;

}

@Override

**public** **void** setPaginationBar(PaginationBar paginationBar) {

**this**.paginationBar = paginationBar;

}

@Override

**public** **void** initUI() {

**super**.initUI();

**new** CostPriceBaseScaleUtil().setListScale(**this**.getModel().getContext().getPk\_group(), **this**.getBillListPanel());

**this**.getBillListPanel().getHeadBillModel().setSortColumn(**new** String[] {

CostPriceHeadVO.*PK\_ORG*, CostPriceHeadVO.*CCOSTPRICEID*

});

**this**.billListPanel.addMouseListener(**this**);

**if** (**null** != **this**.paginationBar) {

**this**.billListPanel.addHeadNavigatePanel(**this**.paginationBar);

}

}

（2）增加分页控件和分页模型，如下，均采用平台的类

<!-- 分页控件 -->

<bean id=*"paginationBar"* class=*"nc.ui.uif2.components.pagination.PaginationBar"* init-method=*"registeCallbak"*>

<property name=*"paginationModel"* ref=*"paginationmodel"*></property>

<property name=*"context"* ref=*"context"* />

</bean>

<!-- 分页模型 -->

<bean id=*"paginationmodel"* class=*"nc.ui.pubapp.uif2app.model.pagination.PubPaginationModel"* init-method=*"init"*>

<property name=*"paginationQueryService"* ref=*"paginationProxy"*></property>

</bean>

（3）修改之前的查询服务代理，对查询结果进行分页处理，重新配置代理为：

~~<!-- 单据查询代理 -->~~

~~<bean id=~~*~~"queryProxy"~~* ~~class=~~*~~"nc.ui.mapub.costpricebase.serviceproxy.CostPriceBaseQueryProxy"~~* ~~/>~~

<!-- 分页代理 -->

<bean id=*"paginationProxy"* class=*"nc.ui.mapub.costpricebase.serviceproxy.CostPriceBasePaginationProxy"* />

(4)修改数据模型管理器，并添加分页服务接口，需要将之前的查询代理删除，并修改数据模型管理器的类为平台封装的分页模型数据管理类：*nc.ui.pubapp.uif2app.model.pagination.PaginationModelDataManager。*分页服务接口定义了查询时访问的方法。

<!-- 数据模型管理器，主要负责各种方式的模型初始化 -->

<bean id=*"modelDataManager"* class=*"nc.ui.pubapp.uif2app.model.pagination.PaginationModelDataManager"*>

<property name=*"model"* ref=*"manageAppModel"*></property>

~~<!-- <property name="service" ref="queryProxy"></property> -->~~

<property name=*"paginationModel"* ref=*"paginationmodel"*></property>

<property name=*"pageQueryService"* ref=*"pageQueryService"* />

</bean>

<!-- 分页服务接口 -->

<bean id=*"pageQueryService"* class=*"nc.ui.pubapp.uif2app.model.pagination.UIPageQueryService"*>

<property name=*"allPagePkQueryServiceMethod"* value=*"nc.itf.mapub.costpricebase.ICostPriceBasePaginationService.queryPKs"*></property>

<property name=*"dataOfPksQueryServiceMethod"* value=*"nc.itf.mapub.costpricebase.ICostPriceBasePaginationService.queryBillByPK"*></property>

</bean>

分页代理类*paginationProxy*，代理类实现了IPaginationQueryService接口。在实现的接口方法queryObjectByPks中，通过NCLocator.getInstance().lookup(ICostPriceBasePaginationService.class)向后台请求数据，因此需要定义接口ICostPriceBasePaginationService及实现CostPriceBasePaginationServiceImpl。

public class CostPriceBasePaginationProxy implements IPaginationQueryService {

/\*根据主键数组查询，返回实际的VO

\* 参数：主键数组pks

\* @see nc.ui.uif2.components.pagination.IPaginationQueryService#queryObjectByPks(java.lang.String[])

\*/

@Override

public Object[] queryObjectByPks(String[] pks) throws BusinessException {

// TODO Auto-generated method stub

CostPriceAggVO[] rets = null;

ICostPriceBasePaginationService paginationQueryService =

NCLocator.getInstance().lookup(ICostPriceBasePaginationService.class);

try {

rets = paginationQueryService.queryBillByPK(pks);

}

catch (Exception e) {

ExceptionUtils.wrappException(e);

}

return rets;

}

}

页查询接口，ICostPriceBasePaginationService

public interface ICostPriceBasePaginationService {

//根据查询条件进行查询，返回主键数组

public String[] queryPKs(IQueryScheme queryScheme) throws BusinessException;

//根据主键数组进行查询，返回VO

public CostPriceAggVO[] queryBillByPK(String[] pks) throws BusinessException;

}

分页查询接口实现，CostPriceBasePaginationServiceImpl。实现类实现了接口定义的两个方法，方法queryPKs根据查询条件从数据库中获取符合条件的所有主键，返回主键数组keys，可根据业务需求对keys进行过滤，筛选出需要的数据。方法queryBillByPK根据主键数组查询实际的VO。

public String[] queryPKs(IQueryScheme queryScheme) throws BusinessException {

StringBuffer sql = new StringBuffer();

QuerySchemeProcessor processor = new QuerySchemeProcessor(queryScheme);

// 默认组织权限

processor.appendFuncPermissionOrgSql();

// 获得主表别名

String mainAlias = processor.getMainTableAlias();

sql.append("select ");

sql.append(mainAlias);

sql.append(".");

sql.append(CostPriceHeadVO.CCOSTPRICEID);

sql.append(processor.getFinalFromWhere()); // 获得加工之后最终的sql

sql.append(" order by ").append(CostPriceHeadVO.CREATETIME);

DataAccessUtils dao = new DataAccessUtils();

IRowSet rowSet = dao.query(sql.toString());

String[] keys = rowSet.toOneDimensionStringArray(); // 转化为一维字符串数组

return keys; //可根据业务需求对keys进行过滤

}

/\*根据主键数组查询

\*/

@Override

public CostPriceAggVO[] queryBillByPK(String[] pks) throws BusinessException {

// TODO Auto-generated method stub

CostPriceAggVO[] bills = null;

BillQuery<CostPriceAggVO> billQuery = new BillQuery<CostPriceAggVO>(CostPriceAggVO.class);

bills = billQuery.query(pks);

return PaginationUtils.filterNotExistBills(bills, pks);

}

注：ICostPriceBasePaginationService和CostPriceBasePaginationServiceImpl实现之后需要在upm文件中进行配置：

<!--费用价格库分页查询 begin-->

<component priority="0" singleton="true" remote="true" tx="CMT" supportAlias="true"> <interface>nc.itf.mapub.costpricebase.ICostPriceBasePaginationService</interface> <implementation>nc.impl.mapub.costpricebase.CostPricePaginationBaseServiceImpl</implementation>

</component>

<!--费用价格库分页查询 end-->