输出入库测试文档，要求：

1.新建采购单，回货扫描

2.每个步骤的测试截图，以及写上测试的单号



**目录**

[一．采购单生成 3](#_Toc62832607)

[采购单生成方式1：库存管理-采购单列表-新建采购单 3](#_Toc62832608)

[采购单生成方式2：建议采购-生成采购单（需上架） 19](#_Toc62832609)

[二、确认采购： 22](#_Toc62832610)

[三、扫描：收发货管理里面依次进行回货扫描-快递单扫描-QC扫描-入库扫描操作 32](#_Toc62832611)

[回货扫描 32](#_Toc62832612)

[快递单扫描 41](#_Toc62832613)

[QC扫描 46](#_Toc62832614)

[入库扫描 50](#_Toc62832615)

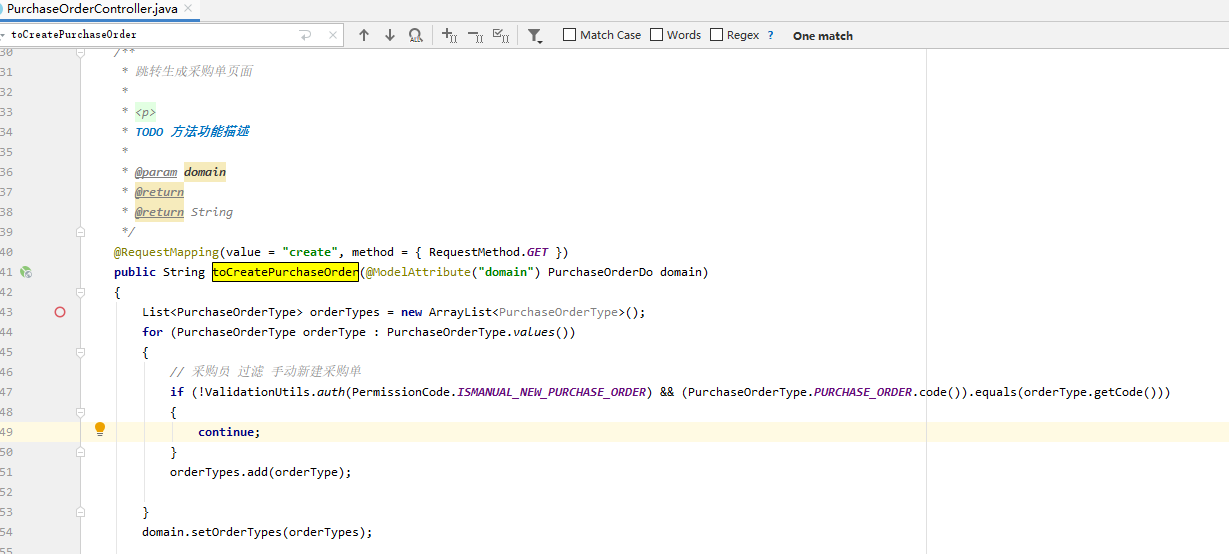
一．采购单生成

# 采购单生成方式1：库存管理-采购单列表-新建采购单



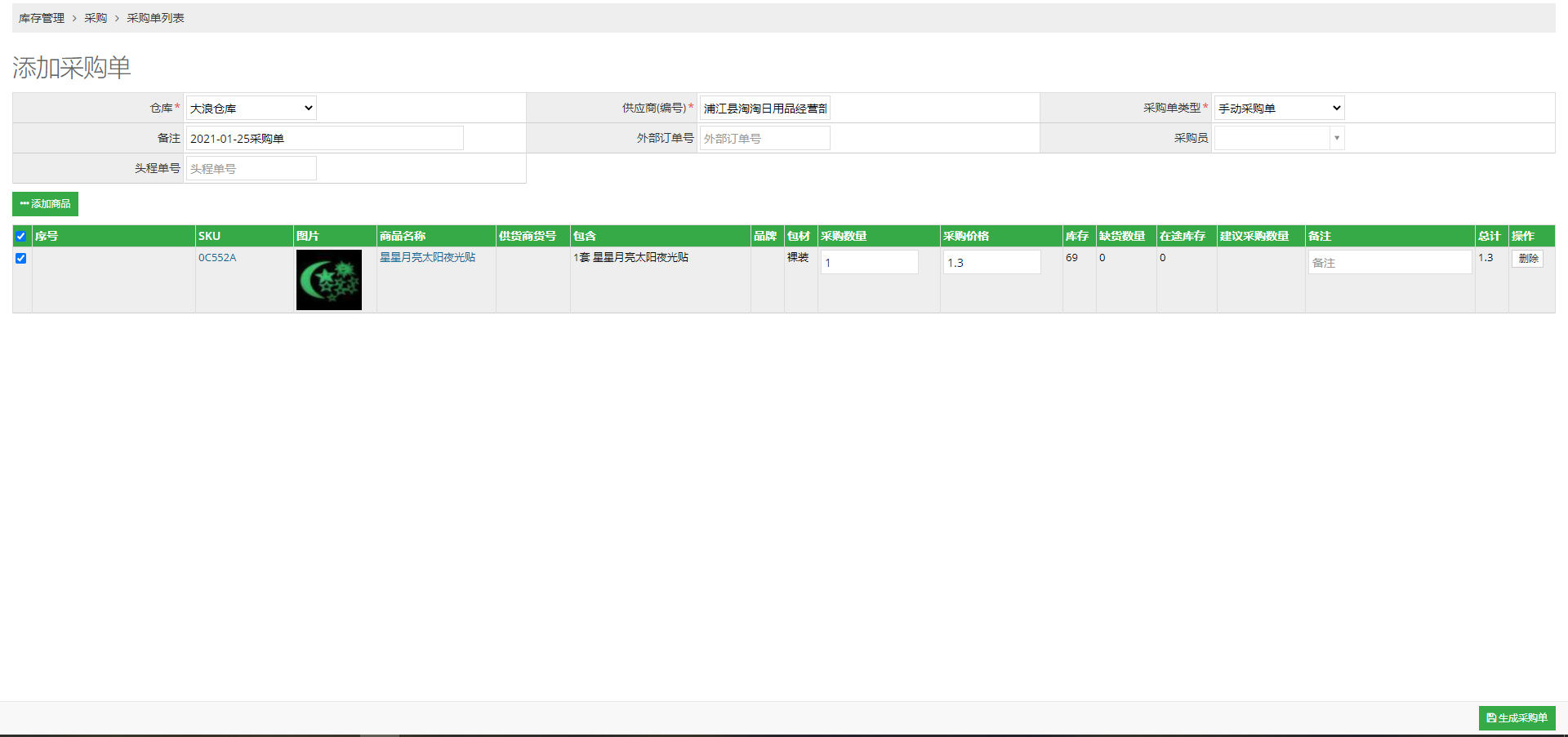


\*采购单列表-点击新建采购单请求后台代码PurchaseOrderController

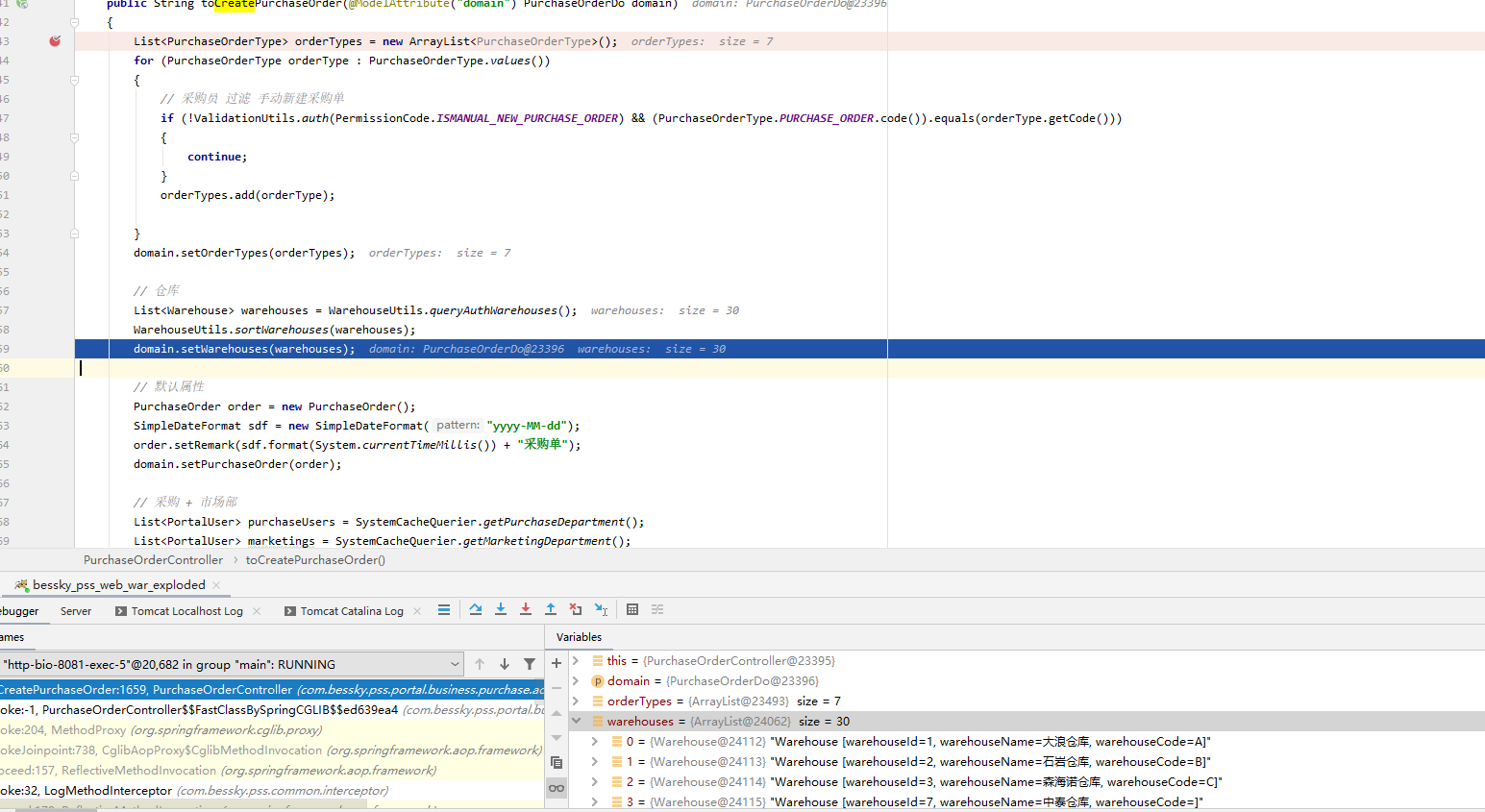


PS:若仓库下拉列表为空，需要开启warehouse微服务

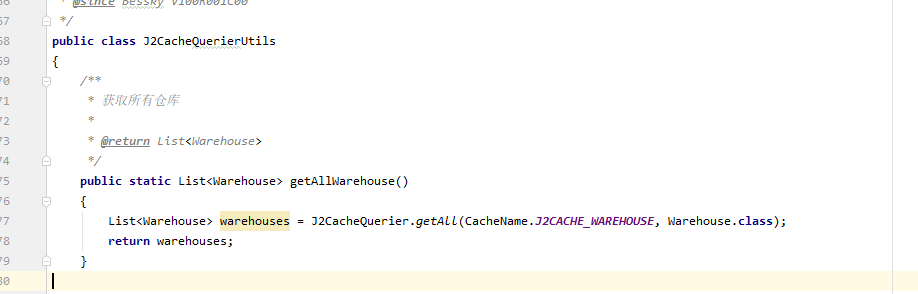
供应商应该粘贴编号而不是中文名



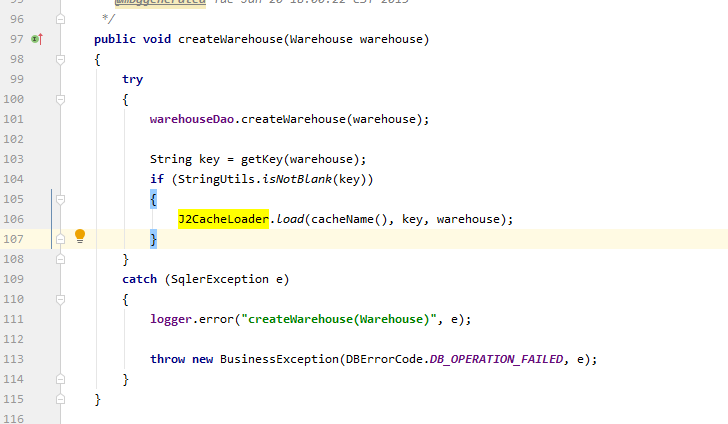
\*仓库后台代码PurchaseOrderController-toCreatePurchaseOrder：



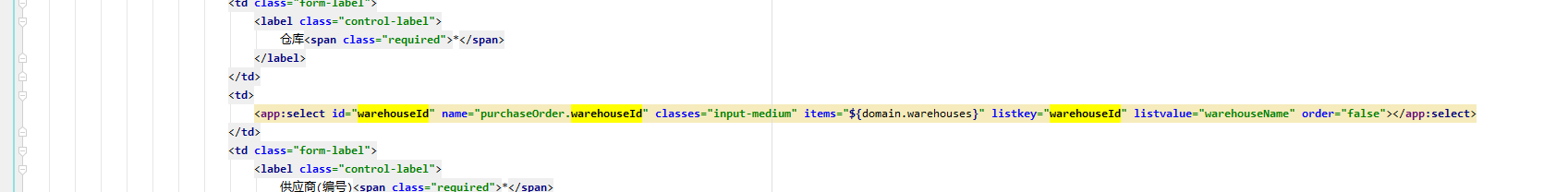
\*缓存中获取



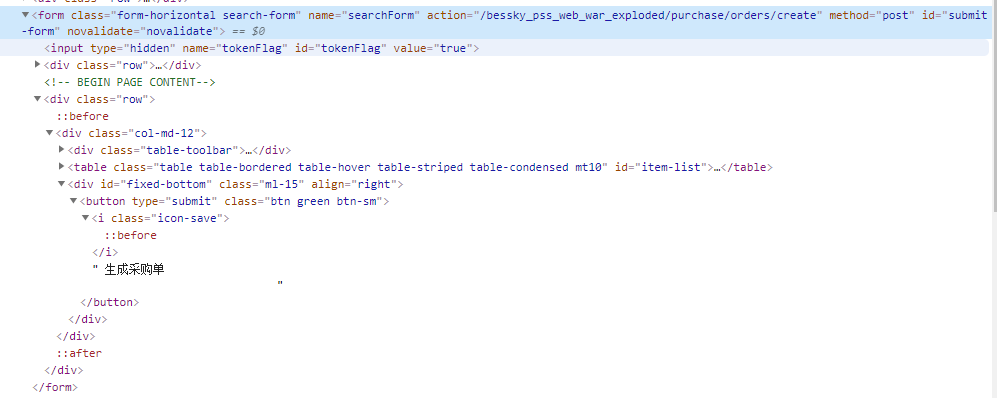
\*bessky-erp-warehouse\ WarehouseServiceImpl.java缓存数据导入（corresponds to the database table t\_warehouse）



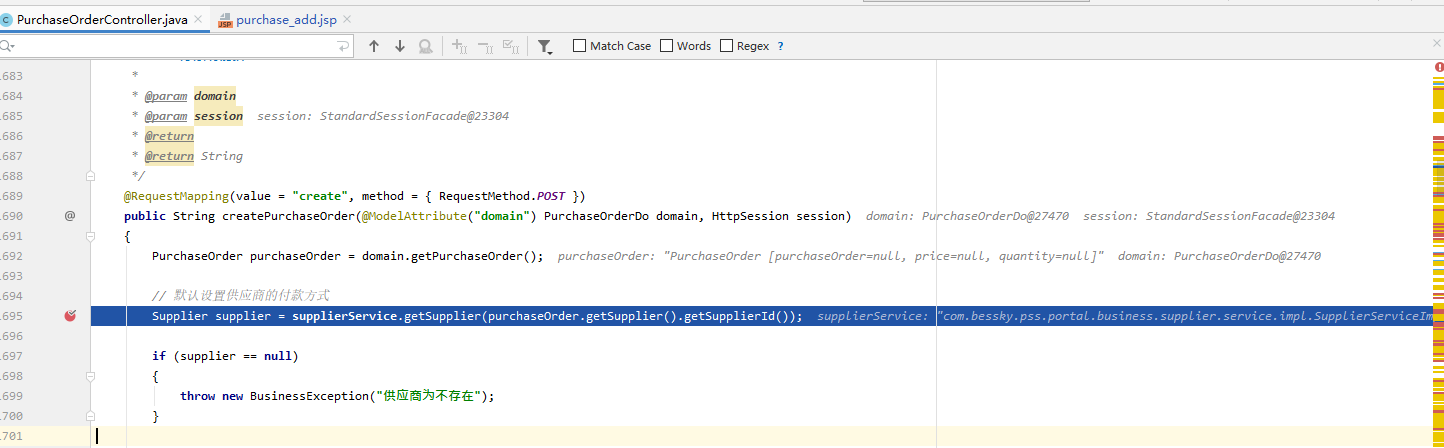
\*仓库下拉前端代码：



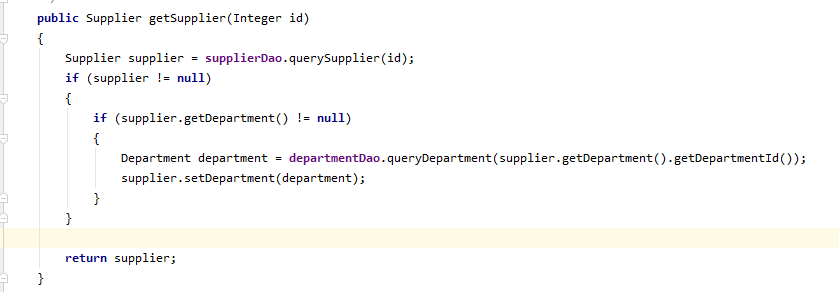
\*点击生成采购单按钮，form请求路径后台purchase/orders/create-post方法



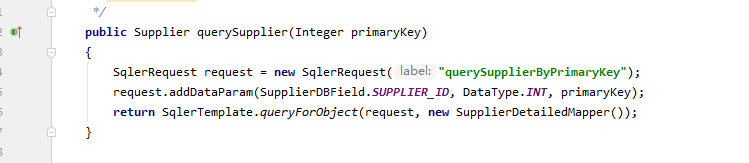
\*点击执行生成采购单按钮后台关键方法-supplierService.getSupplier（通过供应商id获取供应商同时供应商设置对应部门）



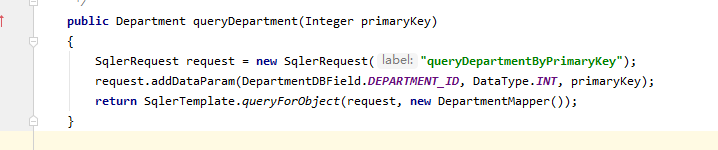
\* getSupplier的实现类 SupplierServiceImpl包含supplierDao.querySupplier查询供应商信息和departmentDao.queryDepartment查询供应商对应部门



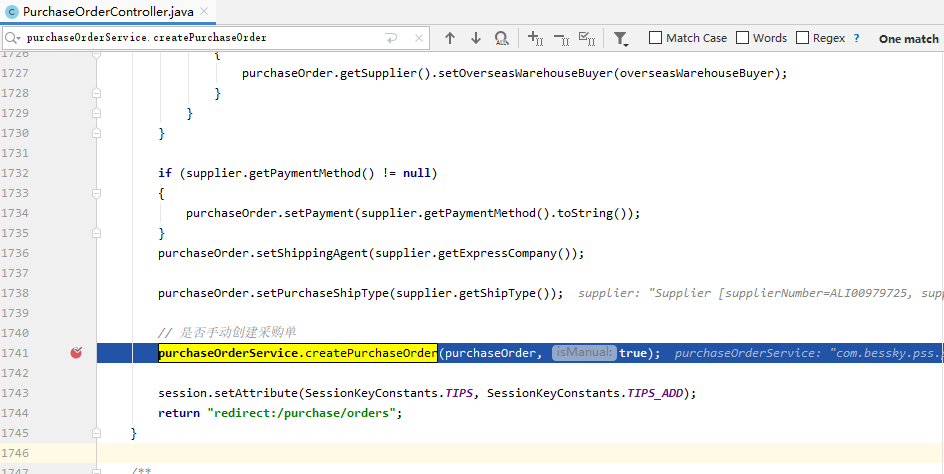
\*supplierDao.querySupplie对应sql为querySupplierByPrimaryKey from t\_supplier



\*departmentDao.queryDepartment对应sql为queryDepartmentByPrimaryKey from t\_department



\*点击执行生成采购单按钮后台关键方法- purchaseOrderService.createPurchaseOrder（purchaseOrder，isManual是否手动创建采购单）



\*实现类PurchaseOrderServiceImpl- createPurchaseOrder方法-updatePurchaseSuggestForCreate(建议采购的采购中的数量发生相应的变化，涉及数据库t\_purchase\_suggest)



\*实现类PurchaseOrderServiceImpl- createPurchaseOrder 方法-inStockDao.queryInStockWithItemCount (查询入库表，涉及数据库t\_in\_stock)

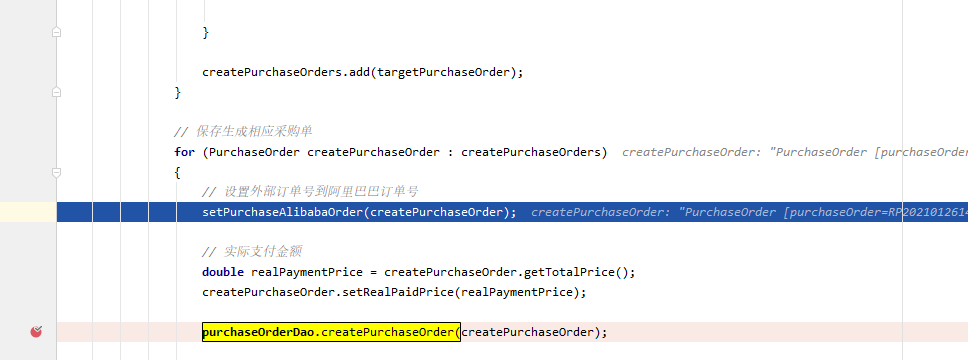


\*实现类PurchaseOrderServiceImpl- createPurchaseOrder方法-inStockItemDao.queryLastInStockPrice(查询上一次入库价格，涉及数据库t\_in\_stock\_item)

-salesService.querySaless对应salesDao.querySalesCount（查询sku销量，涉及数据库t\_sales）

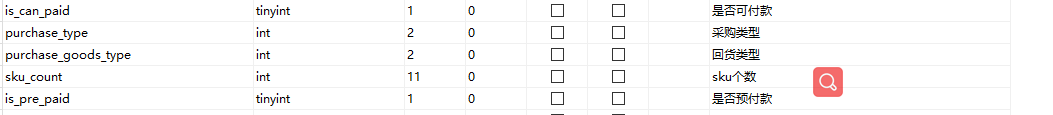


\*实现类PurchaseOrderServiceImpl- createPurchaseOrder方法- purchaseOrderDao.createPurchaseOrder(保存采购单到数据库，涉及数据库t\_purchase\_order)



数据库设计部分截图：





redirect:/purchase/orders自动跳转到采购单列表：



# 采购单生成方式2：建议采购-生成采购单（需上架）

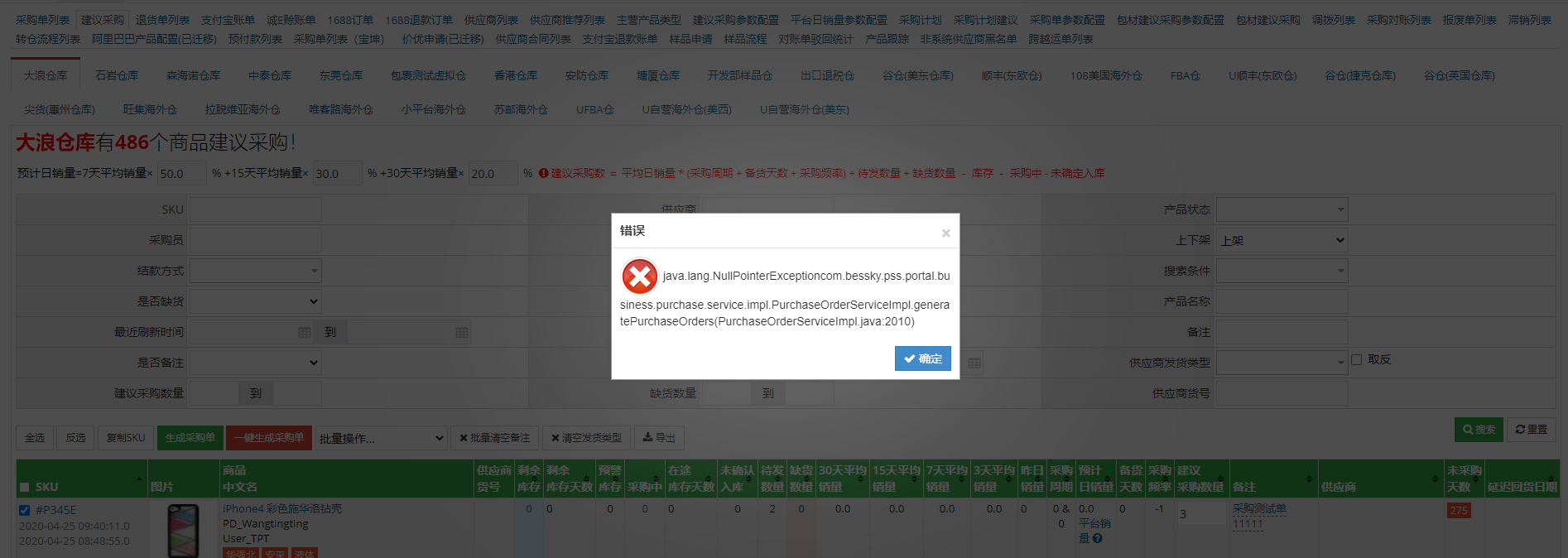
建议采购数 ＝ 平均日销量 \* (采购周期 + 备货天数 + 采购频率) + 待发数量 + 缺货数量 － 库存 － 采购中 - 未确定入库

预计日销量=7天平均销量×50%+15天平均销量×30%+30天平均销量×20%

PS:缺货数量若大于建议采货数量，需要去缺货未发去查看核对



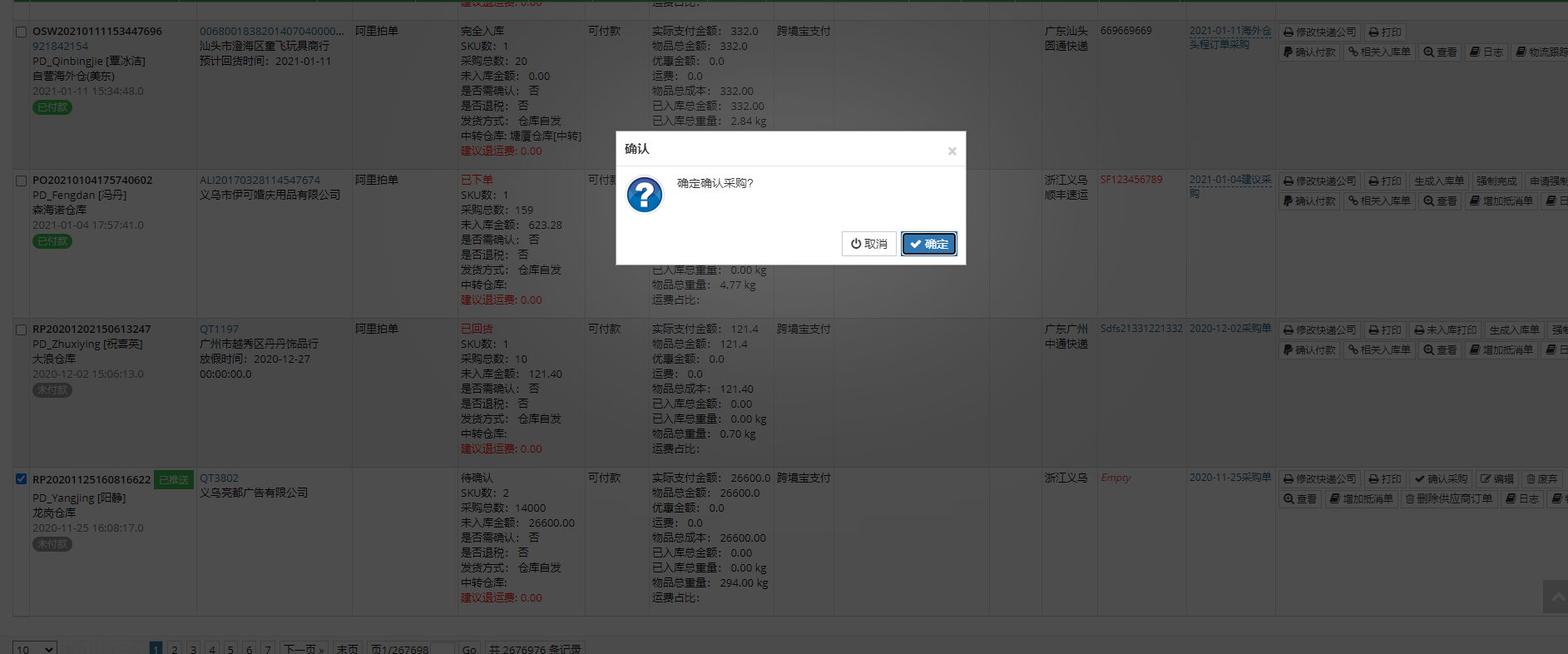
PS：供应商为null，报空指针



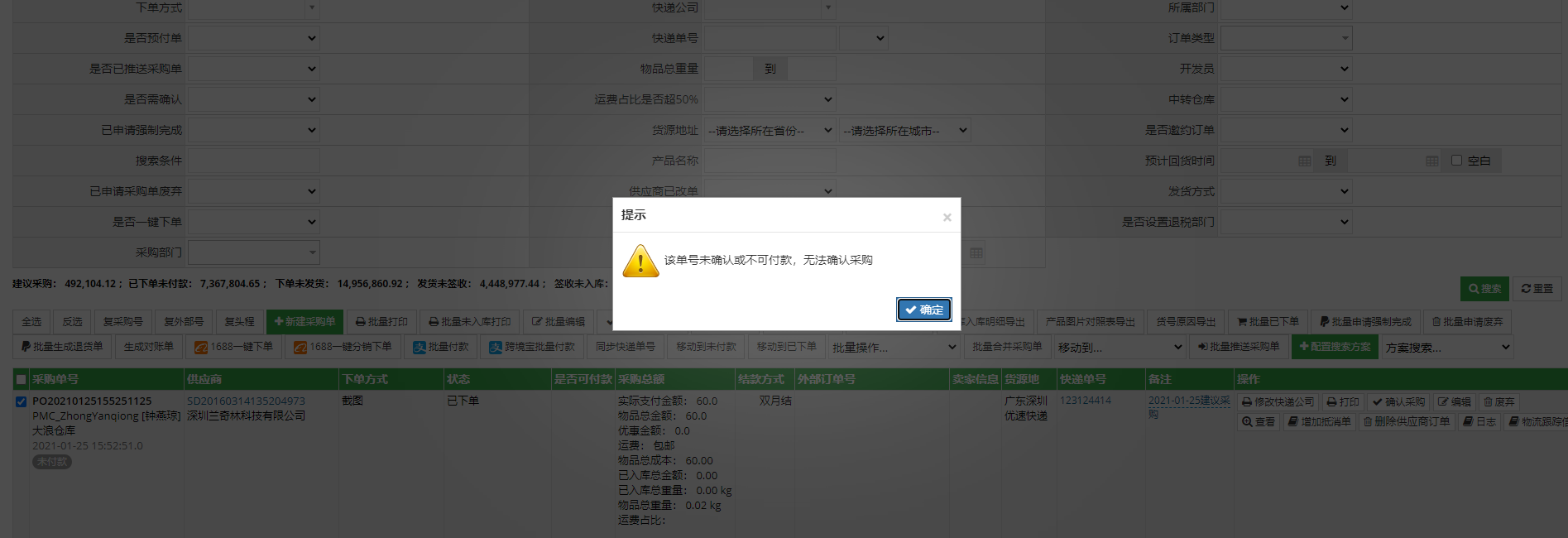
\*空指针代码PurchaseOrderServiceImpl



二、确认采购：



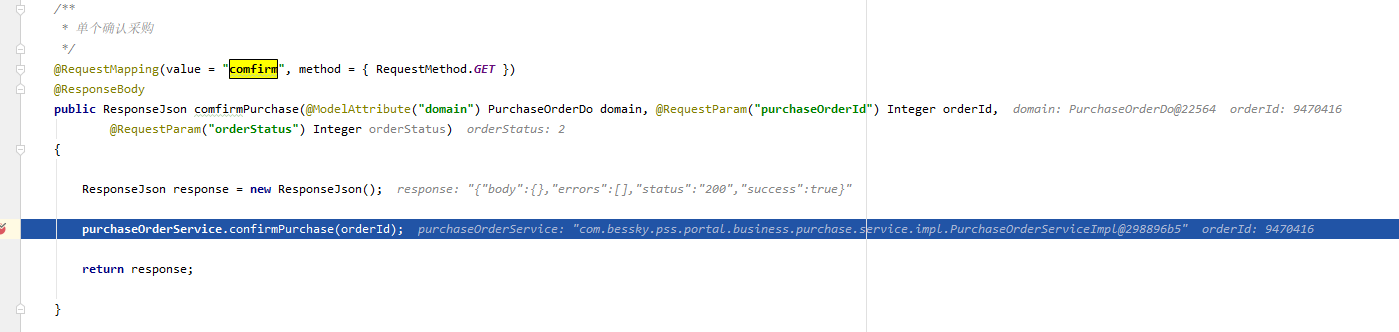
PS:状态为”待确认”且是否可付款类型（筛选“是”，搜索）才能采购



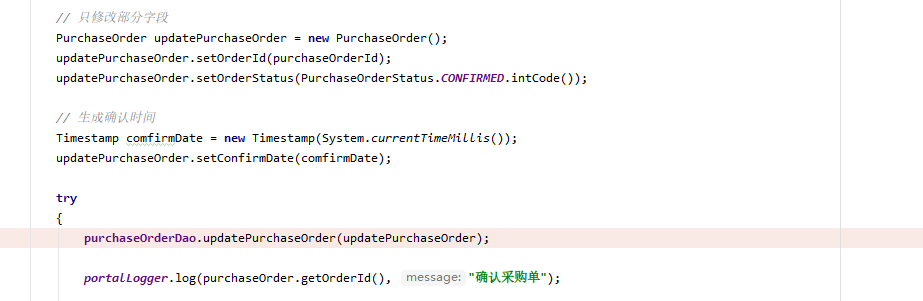
\*确认采购前端js



\*确认采购后台

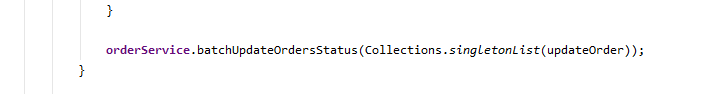


\*实现类PurchaseOrderServiceImpl- confirmPurchase方法-purchaseOrderDao.updatePurchaseOrder(updatePurchaseOrder)(更新部分数据，涉及数据库更新t\_purchase\_order)



\*若为FBA执行

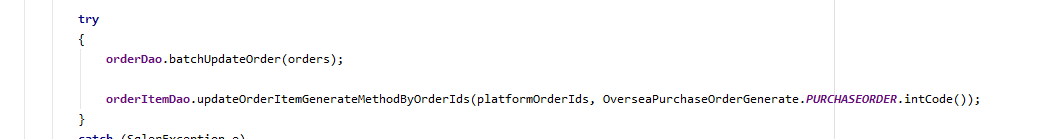
orderService.batchUpdateOrdersStatus(Collections.*singletonList*(updateOrder))



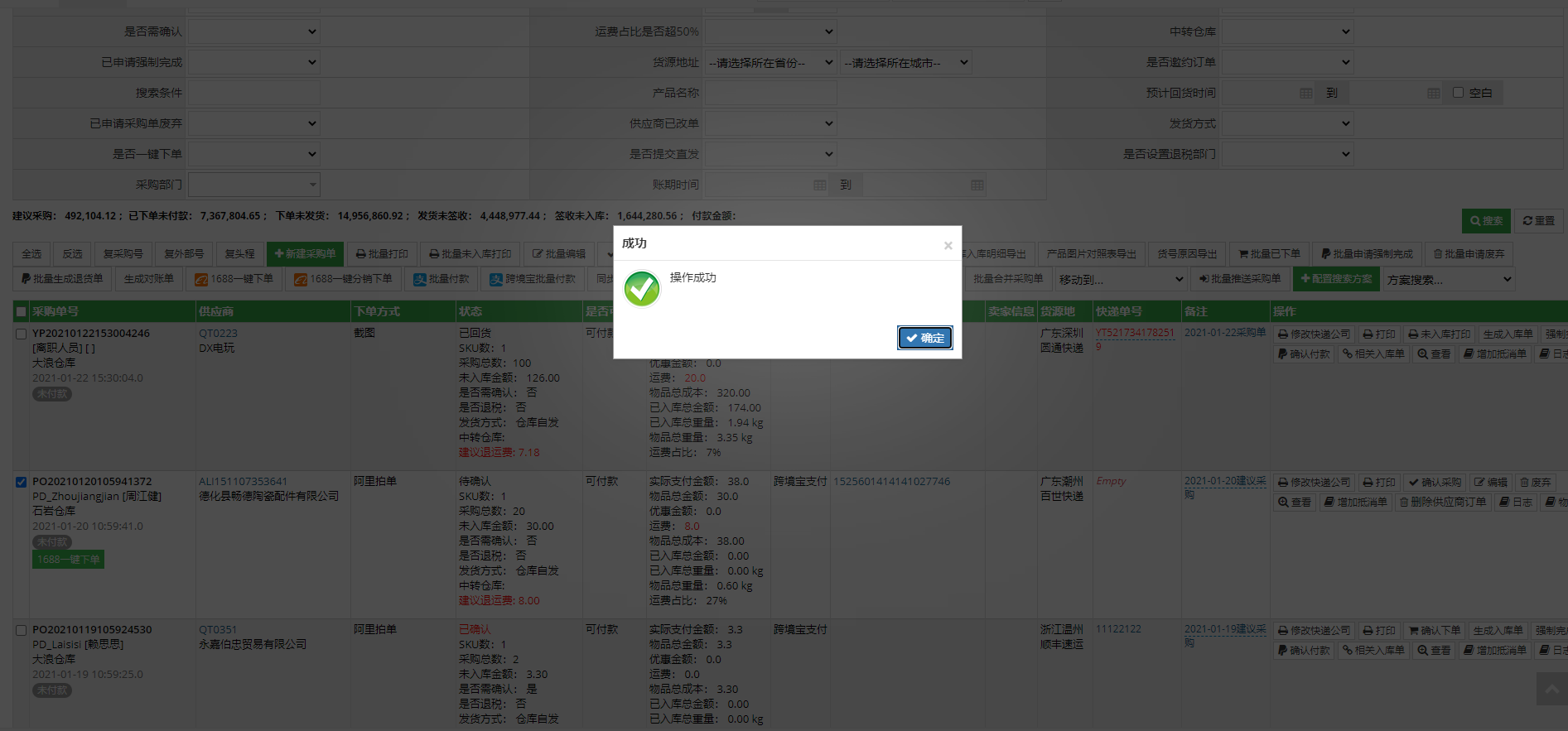
\*实现类OrderServiceImpl包含orderDao.batchUpdateOrder(UPDATE t\_purchase\_order

&orderItemDao.updateOrderItemGenerateMethodByOrderIds(UPDATE t\_order\_item

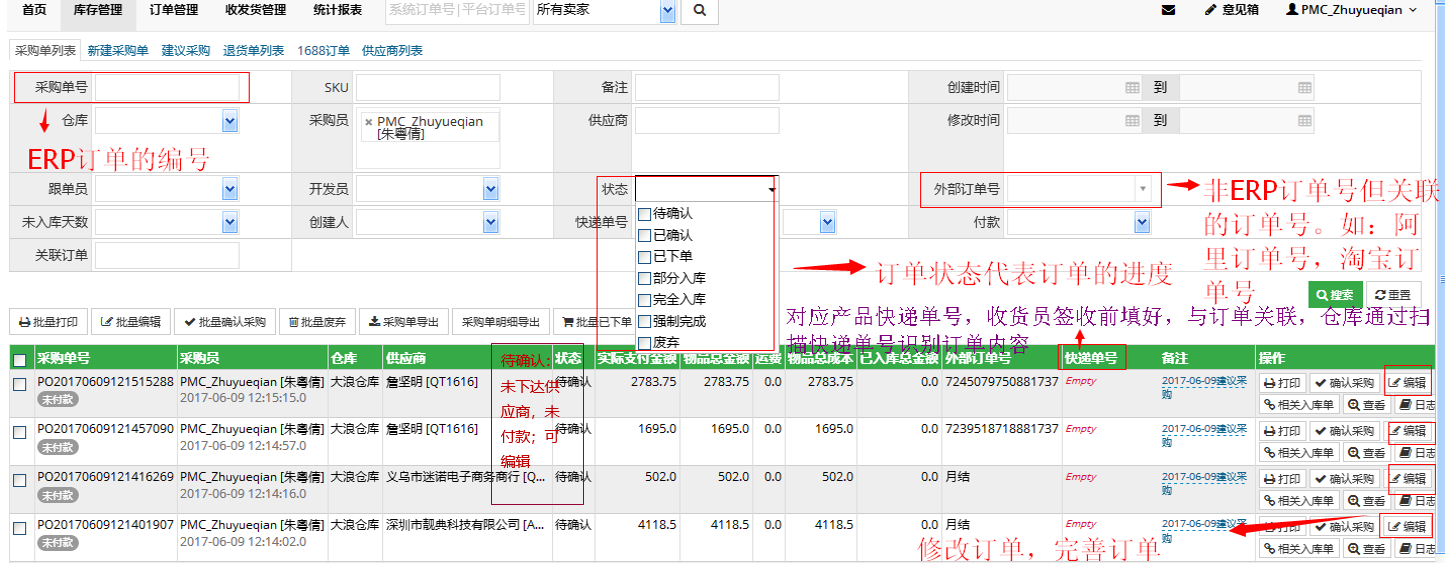
)



操作成功：采购单号为**PO20210120105941372**，成功后自动刷新页面



确认下单（empty输入快递编号）

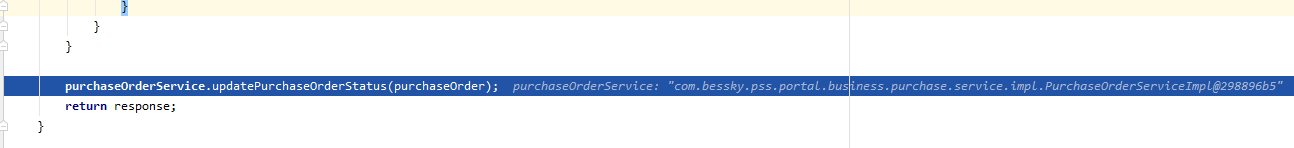




\*确认下单前端js

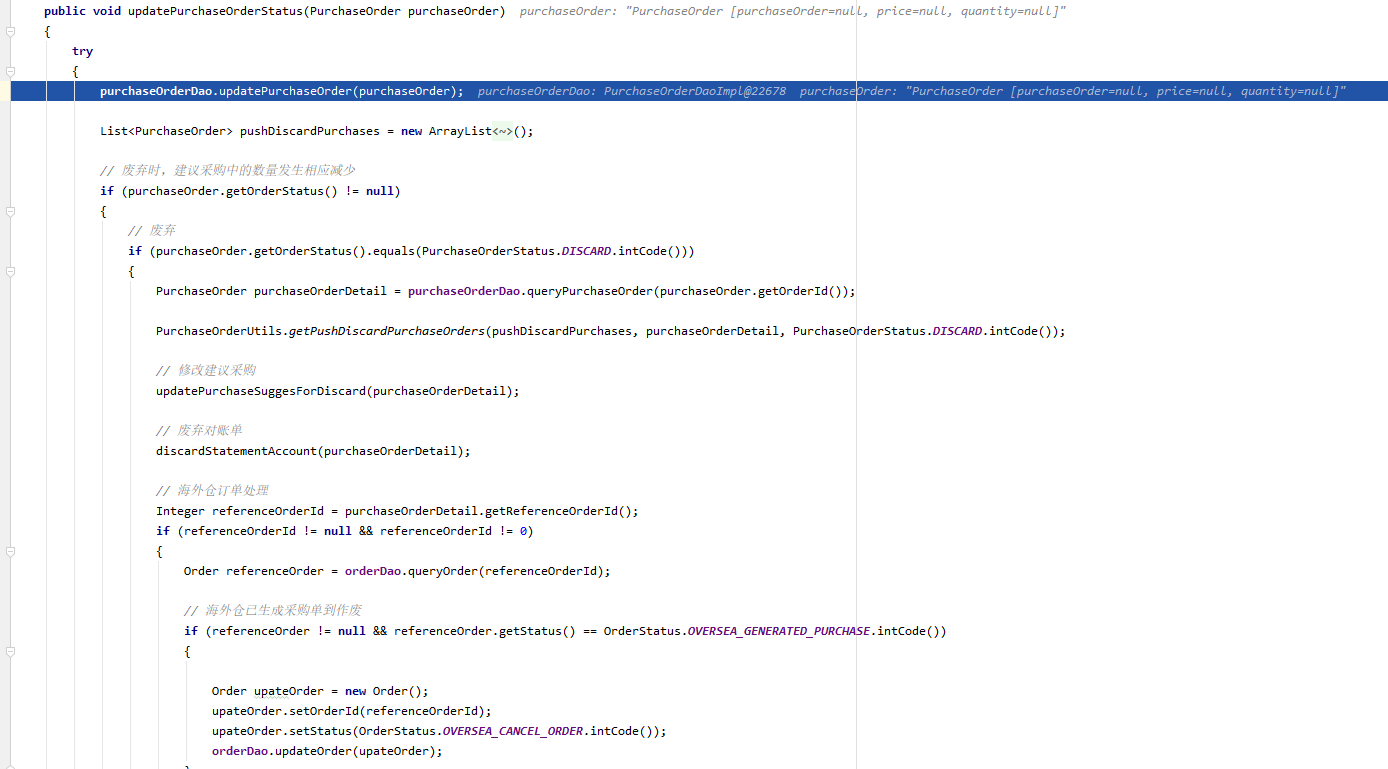


\*后台实现类PurchaseOrderServiceImpl- changePurchaseStatus



\* changePurchaseStatus方法-purchaseOrderDao.updatePurchaseOrder(purchaseOrder)(同上更新数据库t\_purchase\_orde)

采购单状态不为空且为废弃时- purchaseOrderDao.queryPurchaseOrder（查询t\_purchase\_order）......



三、扫描：收发货管理里面依次进行回货扫描-快递单扫描-QC扫描-入库扫描操作

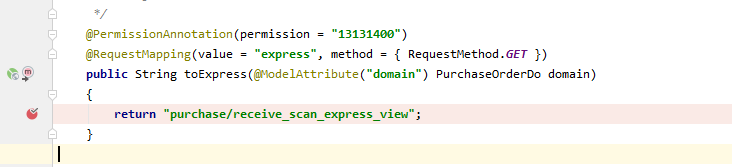
# 回货扫描



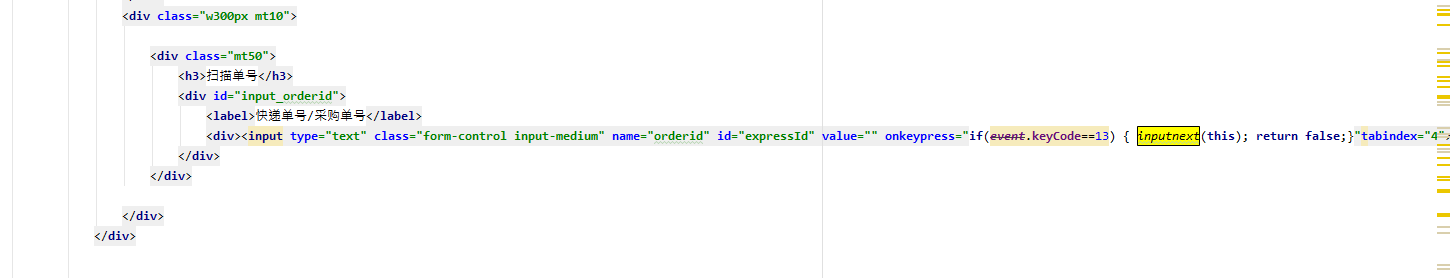
回货扫描：测试回货单**RP20201007115306257**,成功会播放/theme/default/sound/ok.wav



\*页面加载后台请求purchase/receive\_scan\_express\_view.jsp



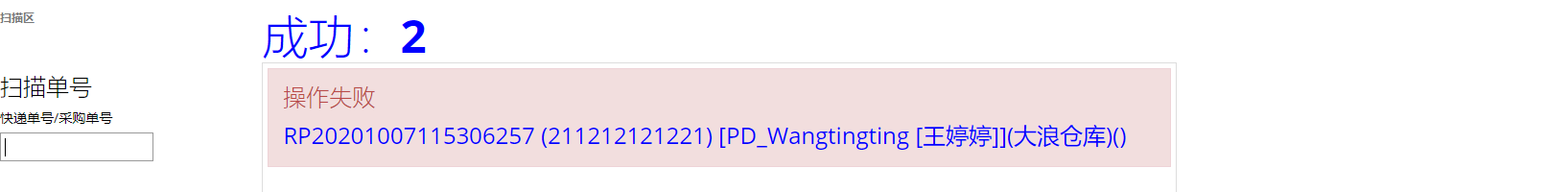
\*前端对应jsp



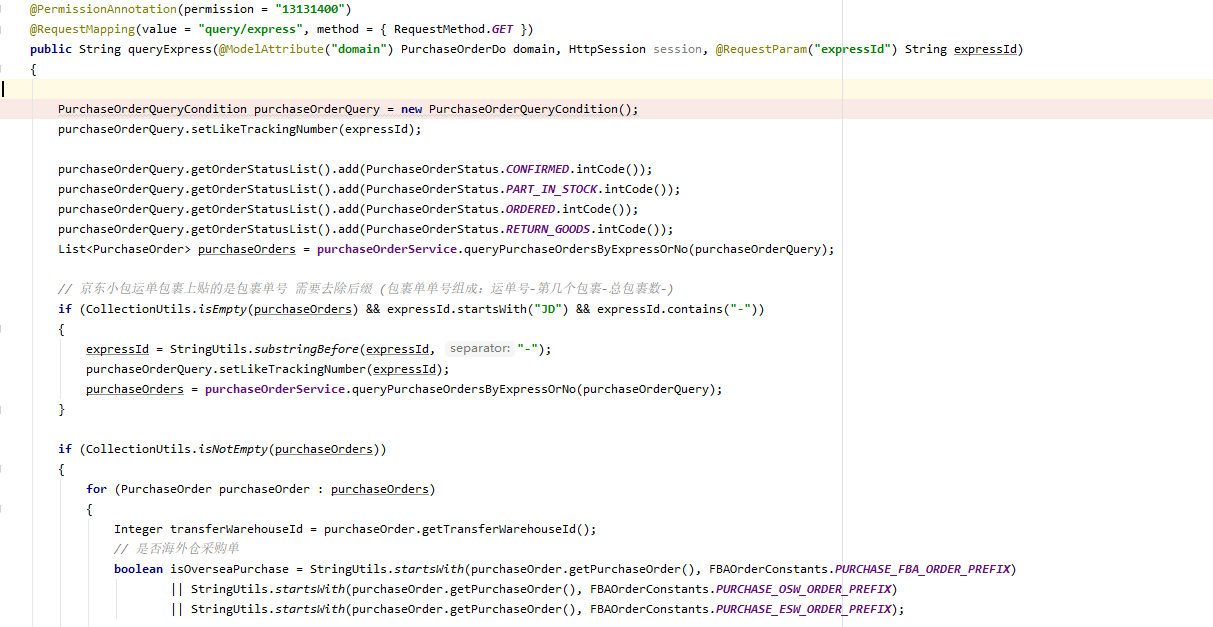
\*执行inputnext方法：



ps：重复单号会失败



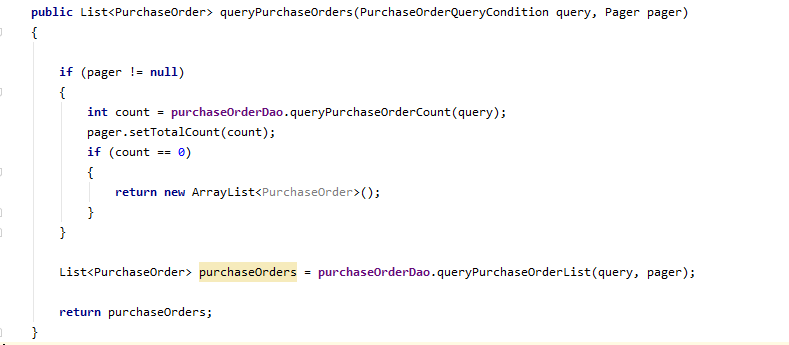
\*后端入库代码



\*实现类PurchaseReceiveScanController- queryExpress

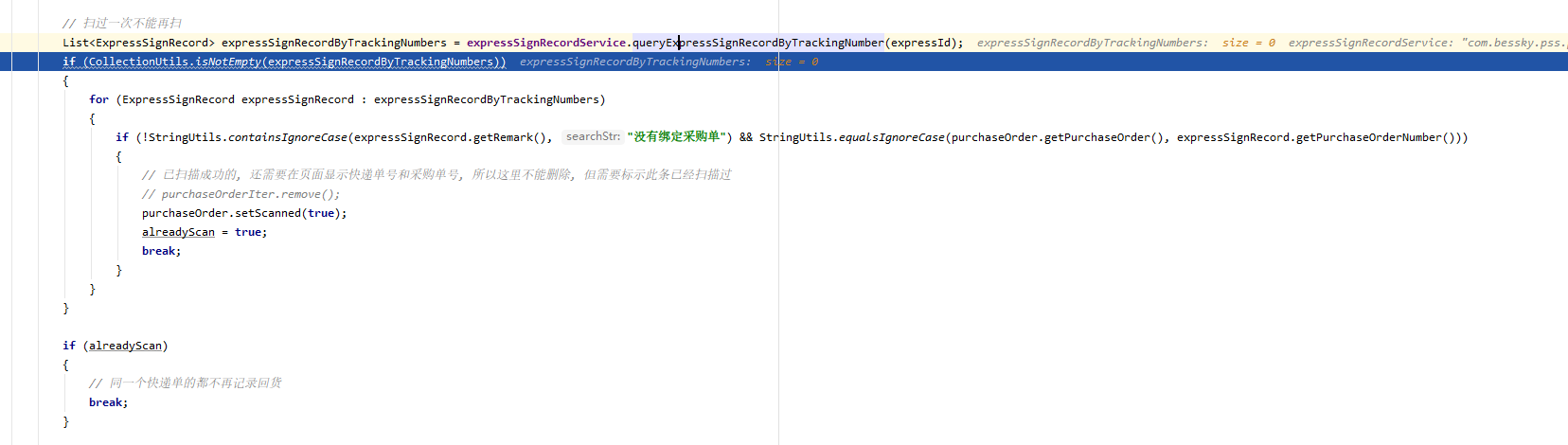
方法- purchaseOrderService.queryPurchaseOrdersByExpressOrNo（*根据快递单号查询*）

- purchaseOrderDao.queryPurchaseOrderList（*t\_purchase\_order*）

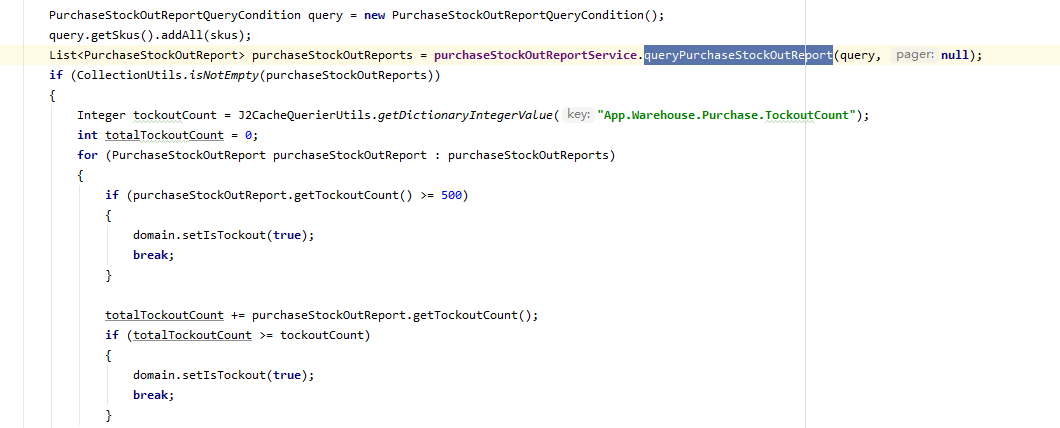


\*实现类PurchaseReceiveScanController- queryExpress方法-expressSignRecordService.queryExpressSignRecordByTrackingNumber(expressId)（查询已经扫描快递单list，涉及数据库t\_express\_sign\_record

）如果已经扫描会显示单号，但不会记录回货信息

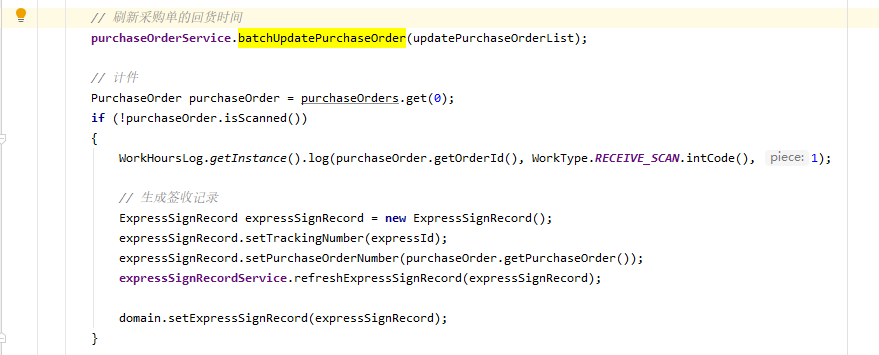


\*实现类PurchaseReceiveScanController- queryExpress方法- purchaseStockOutReportService.queryPurchaseStockOutReport(query, null)（查询缺货订单大于500设为缺货，涉及数据库t\_stock，t\_order）



\*实现类PurchaseReceiveScanController- queryExpress方法- purchaseOrderService. batchUpdatePurchaseOrder

(purchaseOrders)（批量修改采购单, UPDATE t\_purchase\_order）和expressSignRecordService.refreshExpressSignRecord(expressSignRecord)*( 刷新签收记录,涉及数据库*t\_express\_sign\_record)



# 快递单扫描

步骤：

“扫描单号”输入快递单号

采购单备注，货号原因，操作等下拉框可以选择对应类型

填入入库数量（大于0）

点击“生成入库单”按钮执行buildInStock（）方法



\*对应check\_scan\_express.jsp

进行逻辑判断：

采购单需要绑定快递单号，扫描枪需要检测到正确的快递单号（可注释掉相关判断）

不能为“强制完成状态”，“停产产品”..



\*请求后台PurchaseCheckScanController-purchases/scans/instock

代码解析：

强制完成状态抛出异常

设置发货时间

是否可以走快速发货流程

采购单备注别名产品：**purchaseOrderService**.getPurchaseOrder（获得采购单信息，涉及数据库t\_purchase\_order）

**purchaseOrderService**.updatePurchaseOrder（更新采购单信息，涉及数据库t\_purchase\_order）

入库数量大于配置数量：**purchaseOrderService**.queryOrderStockOutQuantity（根据SKU查找订单缺货数量，返回map，涉及数据库SELECT t\_order\_item， t\_order ）

**stockService**.queryWithholdStock（实时计算剩余库存，涉及数据库t\_product）

生成多个入库单：**inStockService**.createMultiSkuInStock-**inStockDao**.createInStock（INSERT INTO t\_in\_stock）和**inStockItemDao**.batchCreateInStockItem（批量生产）

入库数量超过1千不计算产能



成功界面：



打印二维码-贴到货品上面

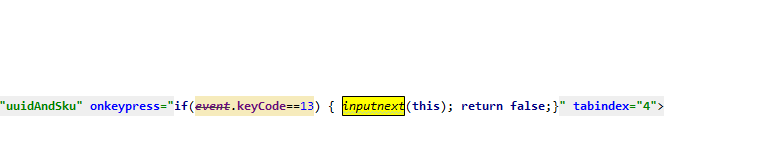
# QC扫描

复制二维码旁边生成的UUID.SKU- 210129101844053001.XYM70225901PK38回车



\*前端check\_scan\_product\_view.jsp

回车执行Inputnext（）方法，会调用inputInit(),请求后台"products/scans/query/sku"



\*后台查询方法

代码分析：

查询待检QC产品：**productScanService**.checkWaitingQCProduct；

获取标签**skuTraceService**.getSkuTraceByUuidAndSku(uuidAndSku)，涉及数据库t\_sku\_trace；

海外仓的不在这里扫描；

查询采购单信息和子条目信息: **purchaseOrderService**.getPurchaseOrderItemInfo- **purchaseOrderDao**.queryPurchaseOrderList

,涉及数据库t\_purchase\_order+ **purchaseItemDao**.queryPurchaseItemList，涉及数据库t\_purchase\_item；

调拨单判断：七天销量大于50；

InStockItem的进库产品需大于0；

已入库数量: **inStockDao**.queryInStockedQuantityMap(查询并且计算总已入库数量，涉及数据库t\_in\_stock )

, SqlerTemplate.queryForList(request)不能为null,为null时 return result前可增加代码result.put("XYM70225901PK38",1);  
返回给前端domain商品数量，标签；

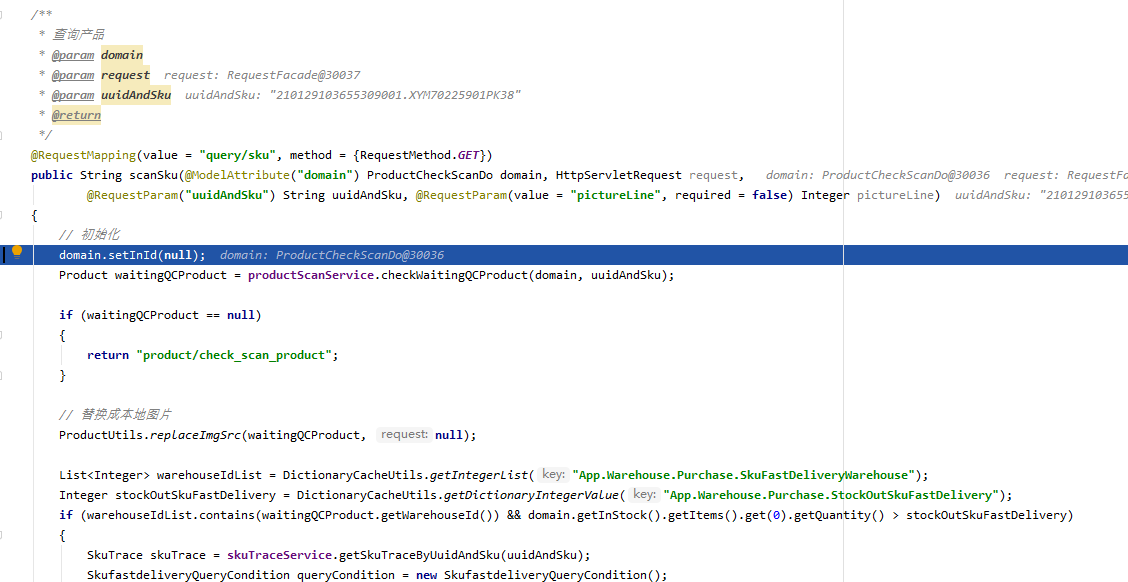
条件查询单个产品详情(包括描述，供应商):**productService**.getProductDetails, **“queryProductDetail”**,涉及数据库t\_sku\_trace

返回前端产品详情；

设置入库信息；

查询同一采购单入库单等待QC (PASS)状态的记录：**inStockService**.searchInStockQCPass(condition)，**"searchInStockQCPass"**，涉及数据库t\_in\_stock，若跑不出来可注释；

重量范围：**weightCheckFacade**.queryWeightCheckByWeight【需配置和开启相关微服务】，若没开启成功可注释



成功页面：



最后点击绿色勾勾即可！

# 入库扫描

页面输入UUID+SKU



\*前端check\_scan\_instock\_view.jsp请求后台路径**"instocks/scans/query/trace"**



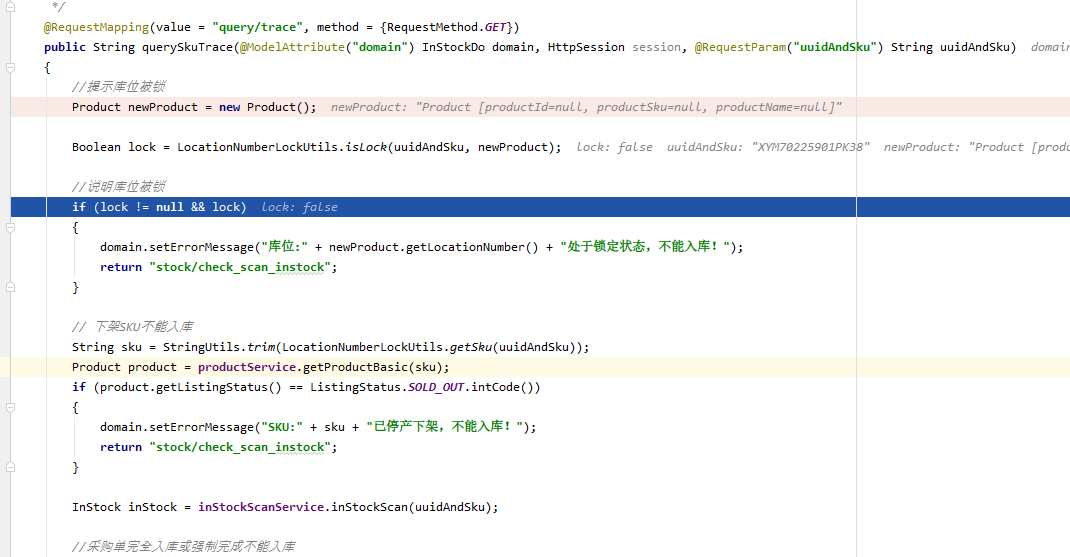
\*后台InStockScanController

LocationNumberLockUtils判断库位是否被锁；

下架SKU不能入库productService.getProductBasic（根据SKU查询产品基本信息(不包括描述)*，***"queryProductBasicByCondition"**

，涉及数据库t\_product）；

采购单完全入库或强制完成不能入库：inStockScanService.inStockScan(uuidAndSku)，入库扫描：获取标签skuTraceService.getSkuTraceByUuidAndSku(uuidAndSku)，若获取失败，数据库t\_sku\_trace里面放入sku和UUID（注意不要粘贴空格）



成功页面：



扫描成功后，状态会由等待上架变成已入库。