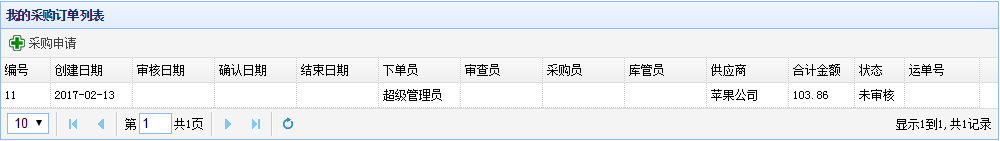
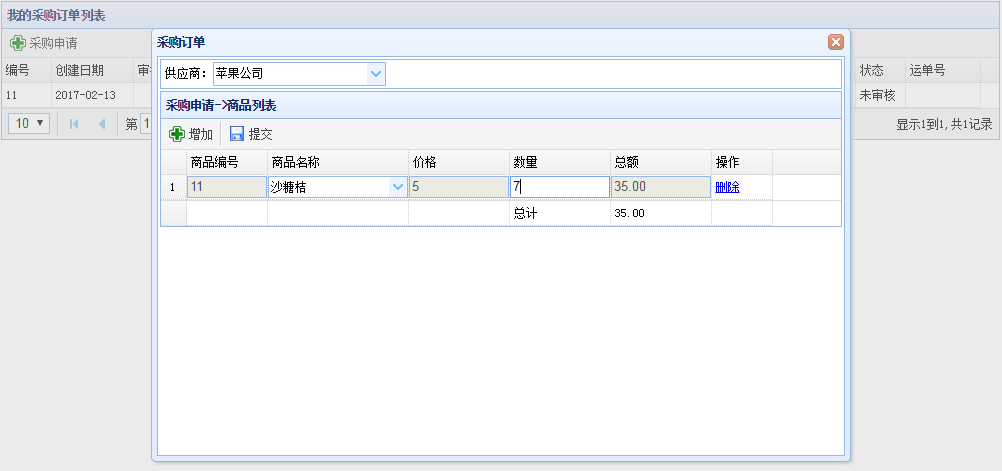
ERP项目笔记 Day-07

1. 销售与库存管理
2. 我的采购订单
   1. 需求及实现思路

只显示由我发起的订单，并将之前做的采购申请功能，整合到此页面



点击采购申请，弹出添加订单窗口

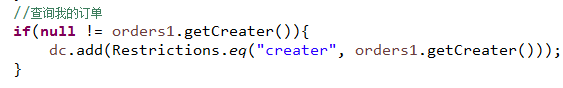


* 1. 后端实现

1. 修改 OrdersAction，增加以下代码



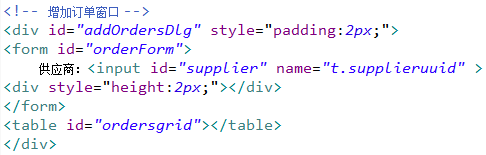
1. 修改OrdersDao中的getDetachedCriteria方法，添加：



* 1. 前端实现

把添加采购订单的页面整合到orders.html页面来。

1. 在orders.html页面中添加addOrdersDlg，再把orders\_add.html中的body部分都复制到这个DIV下，把grid改成ordersgrid

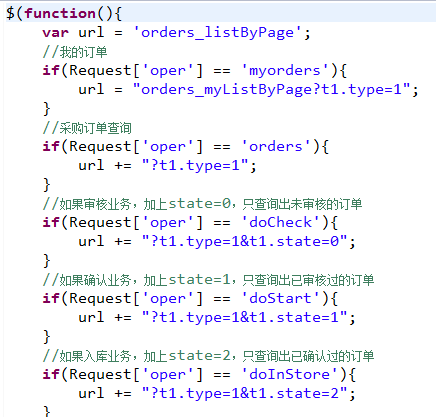


1. 在orders.html中引入orders\_add.js

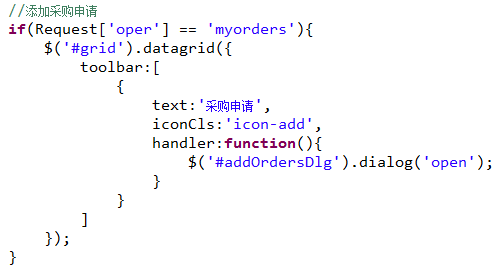


把orders\_add.js中替换#grid为#ordersgrid

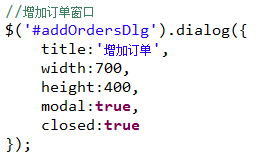
1. 在orders.js的初始化中修改：
   1. 修改请求的url



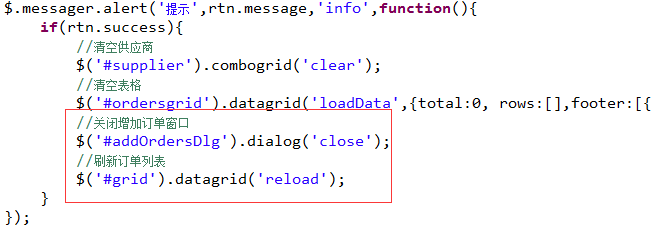
* 1. 实现动态单栏按钮：在grid初始化后加入



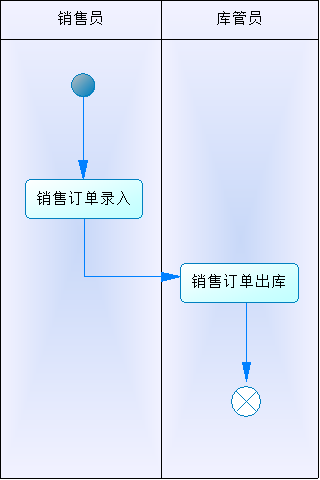
* 1. 增加订单窗口初始化



* 1. 修改orders\_add.js文件，在提交成功后，应该关闭“增加订单”窗口，同时还要刷新订单列表。添加红框代码：

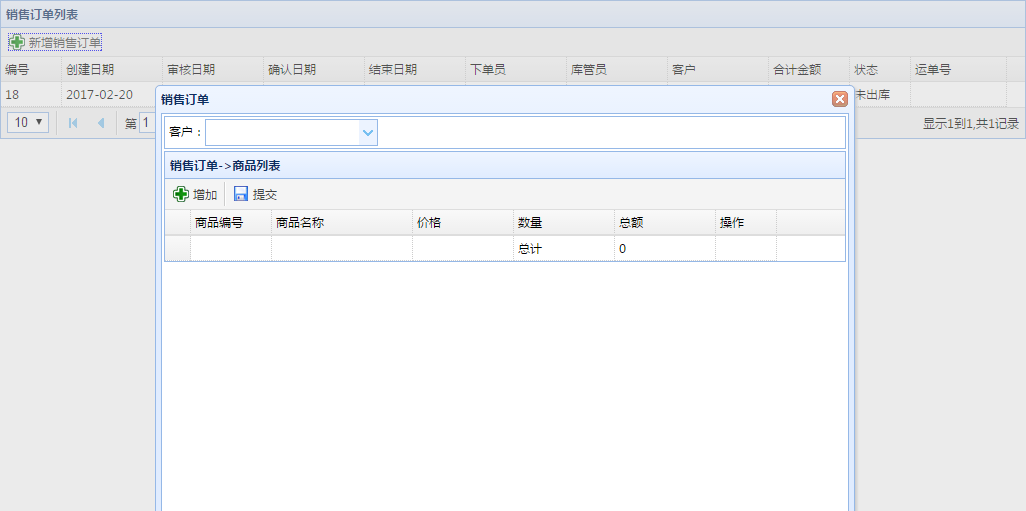


1. 我的销售订单



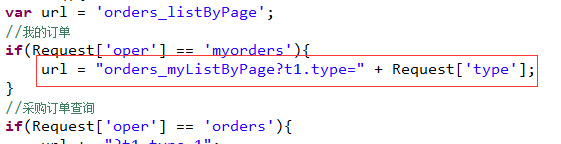
* 1. 需求及实现思路

与采购订单功能类似，只列出我的销售订单且状态是未出库的

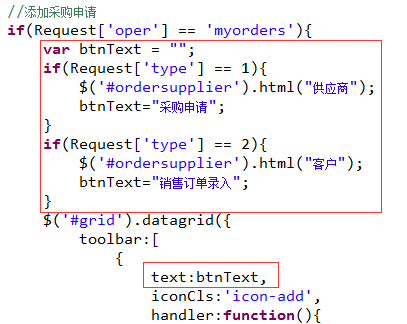


* 1. 代码实现

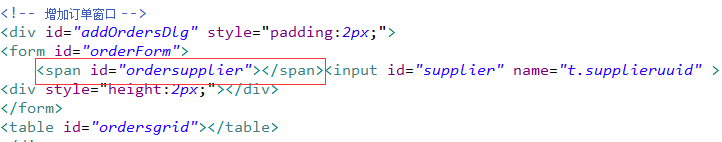
1. 修改orders.js中初始化代码，查询我的订单URL中把t1.type进行拼接，通过静态传参获取type类型，1=采购，2=销售：



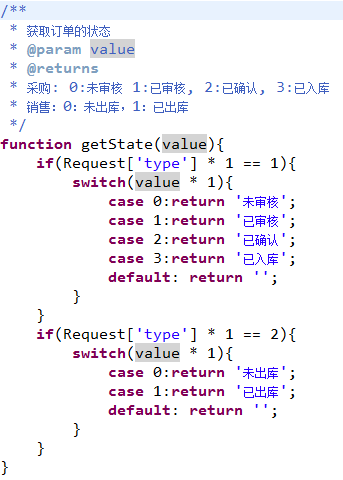
1. 修改采购申请按钮代码块：根据不同的type来显示供应商或客户，动态显示按钮文本名



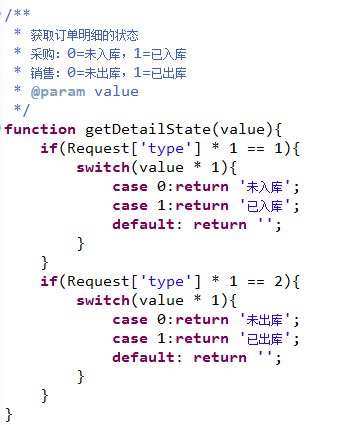
1. 修改orders.html，把供应商替换成如下，加上span



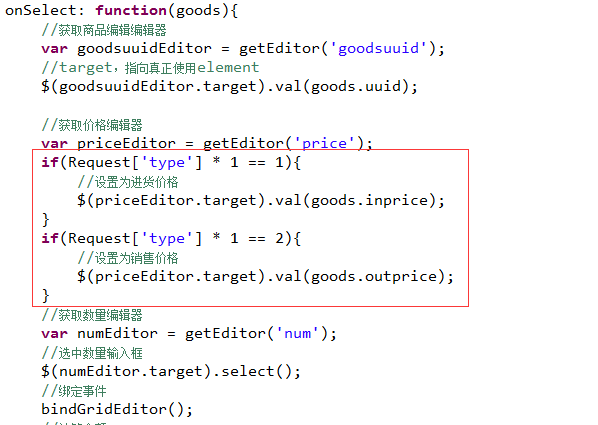
1. 修改订单列表的状态字段显示，如果type=2为销售订单，则它的状态只有0：未出库，1：已出库



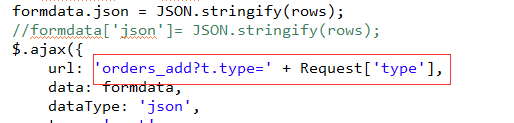
1. 修改订单明细里的状态：



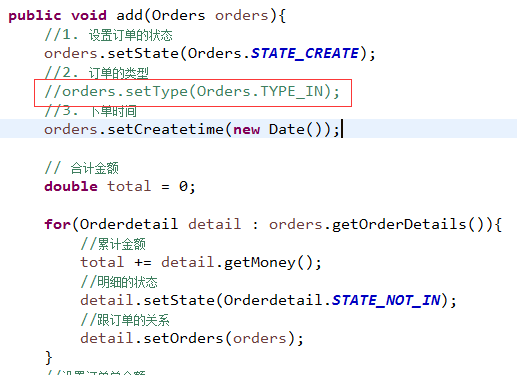
1. 修改 orders\_add.js，在选择商品时根据页面参数 type 来读取不同的价格，当type=1的时候，取的是进货价；当type=2的时候要取销售价。



1. 修改 orders\_add.js，提交时追加参数



1. 修改OrdersBiz的add方法，将下列代码中红框部分注释。因为这个值是由前端传递过来的



浏览器打开url测试：

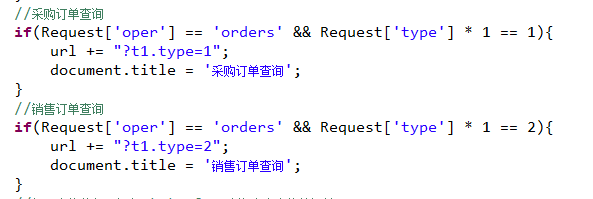
http://localhost:8080/erp/orders.html?oper=myorders&type=2

1. 销售订单查询
   1. 需求及实现思路

向 orders.html页面传递参数type 为2, 显示销售订单; 参数type为1, 显示采购订单

* 1. 代码实现

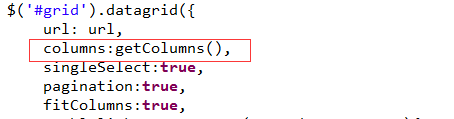
1. 修改 orders.js 中的初始化方法中，动态显示文档标题



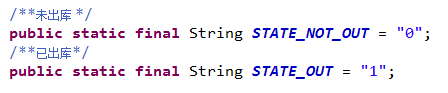
1. 抽出columns定义，根据不同的订单类型，订单列表显示不同的列，新增getColumns方法



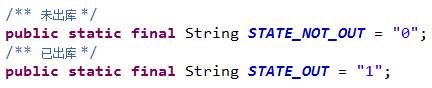
1. 修改订单列表columns属性配置，调用getColumns方法获取相应的列



1. 销售订单出库（重点）
   1. 需求及实现思路
2. 
3. 此功能与“采购订单入库”功能极为类似
4. 需要注意的问题是出库时需要校验库存是否大于出库量
5. 销售订单的流程比采购流程简单，没有审核、确认环节
   1. 代码实现
      1. 后端实现
6. 在Orderdetail类中，加入两个常量定义



1. 在Orders类中，加入两个常量定义：



1. 在OrderdetailBiz中增加相应接口和实现方法doOutStore

|  |
| --- |
| /\*\*  \* 销售出库操作  \* **@param** empuuid  \* **@param** uuid  \* **@param** storeuuid  \*/  **public** **void** doOutStore(Long empuuid, Long uuid, Long storeuuid){  //获取订单明细  Orderdetail orderDetail = orderdetailDao.get(uuid);  **if**(!"0".equals(orderDetail.getState())){  **throw** **new** ErpException("亲！该明细已经出库了，不能重复出库哦!");  }  //更新订单明细  orderDetail.setEnder(empuuid);  orderDetail.setEndtime(**new** Date());  orderDetail.setState("1");  orderDetail.setStoreuuid(storeuuid);  //查询库存  Storedetail storeDetail = **new** Storedetail();  storeDetail.setGoodsuuid(orderDetail.getGoodsuuid());  storeDetail.setStoreuuid(storeuuid);  List<Storedetail> storedetailList = storedetailDao.getList(storeDetail, **null**, **null**);  //商品仓库库存数量  **long** num = -1l;  //如果存在库存，检查库存是否足够  **if**(**null** != storedetailList && storedetailList.size() > 0){  storeDetail = storedetailList.get(0);  num = storeDetail.getNum().longValue() - orderDetail.getNum().longValue();  }  **if**(num > 0){  //库存充足，则更新库存数量  storeDetail.setNum(num);  }**else**{  //库存不足，则提示用户  **throw** **new** ErpException("库存不足!");  }  //添加库存变更操作记录  Storeoper log = **new** Storeoper();  log.setEmpuuid(empuuid);  log.setGoodsuuid(orderDetail.getGoodsuuid());  log.setNum(orderDetail.getNum());  log.setOpertime(orderDetail.getEndtime());  log.setStoreuuid(storeuuid);  log.setType("2");  storeoperDao.add(log);  //检查是否订单下的所有明细都已经出库  Orderdetail queryParam = **new** Orderdetail();  Orders orders = orderDetail.getOrders();  queryParam.setOrders(orders);  queryParam.setState("0");  Long cnt = orderdetailDao.getCount(queryParam, **null**, **null**);  **if**(cnt == 0){  //所有明细都已经出库，则更新订单状态为已出库  orders.setState("1");  orders.setEnder(empuuid);  orders.setEndtime(orderDetail.getEndtime());  }  } |

1. Action中增加doOutStore方法：

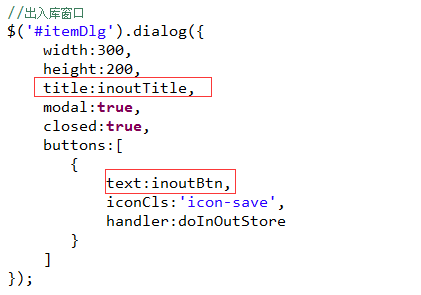


* + 1. 前端实现

1. 定义oper=doOutStore为出库操作



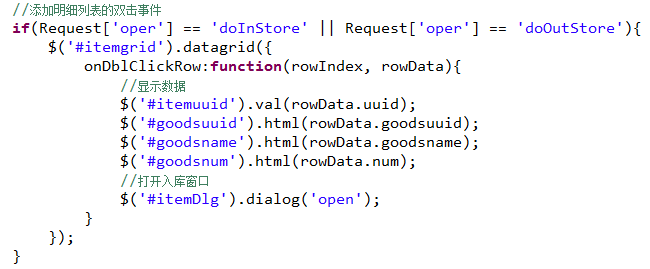
1. 修改出入库窗口：



1. 在orders.js中修改原来的doInStore方法，改成doInOutStore：



1. 在订单明细列表里增加双击行事件



1. 库存查询
   1. 需求分析



* 1. 代码实现

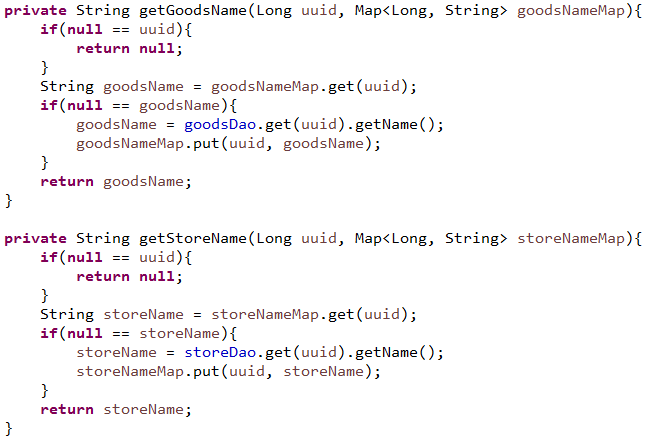
1. 修改StoredetailBiz

引入：

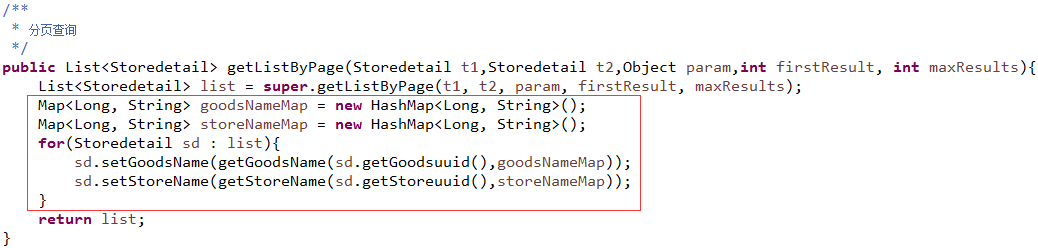


配置spring文件：略

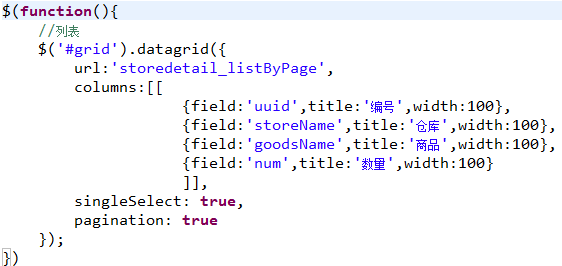
增加获取方法：



重写getListByPage方法：



1. 增加storedetail.js



1. 修改storedetail.html:



1. 库存变动记录查询
   1. 需求分析

需求：库存变动记录表查询，查询条件为商品（自动补全下拉列表）、仓库（下拉列表）、日期范围搜索、操作员（自动补全下拉列表）、类型（出库、入库）

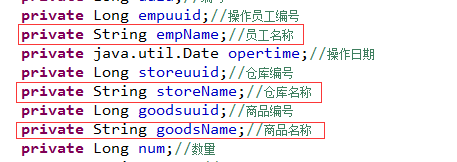


* 1. 代码实现

1. 修改StoreoperDao的getDetachedCriteria方法



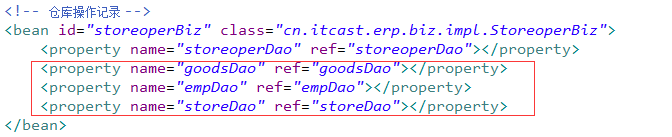
1. 在Storeoper实现类中添加3个属性，并生成getter与setter



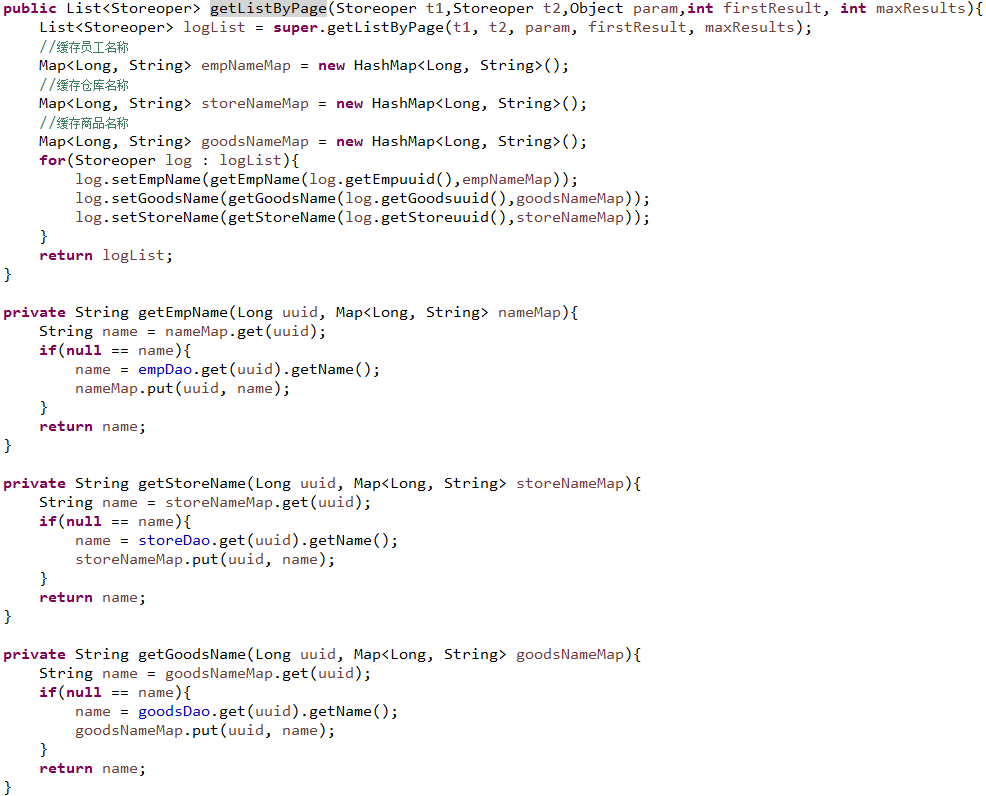
1. 在StoreoperBiz中注入：



1. 配置spring文件

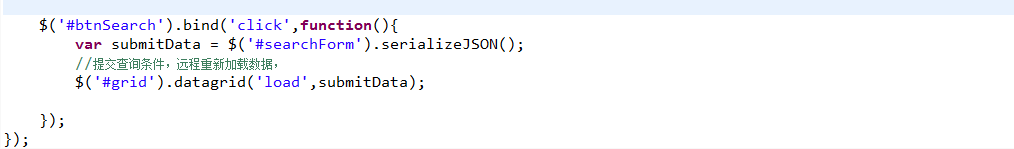


1. 在StoreoperBiz中，重写listByPage方法：



1. 添加storeoper.js文件

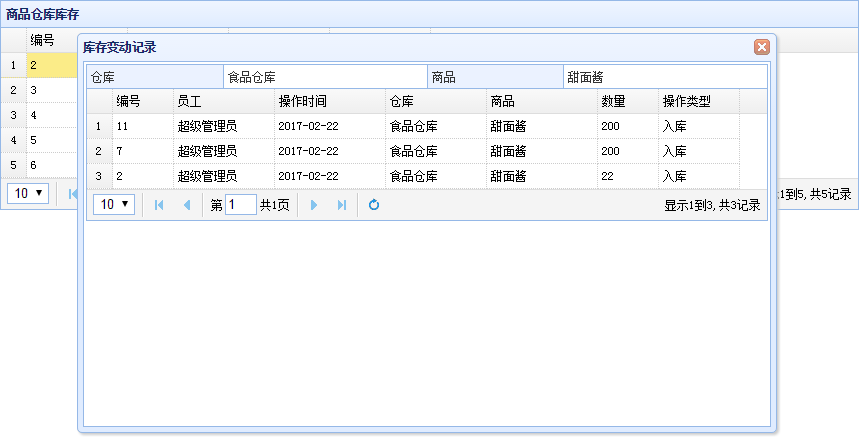




1. 修改storeoper.html



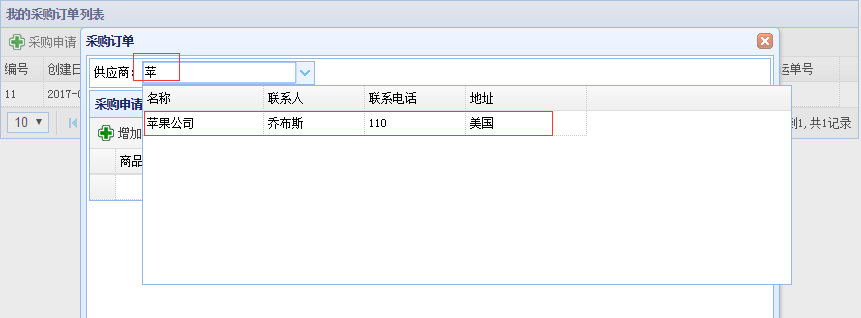
1. 库存查询添加查看变动记录功能（学员实现）



1. 自动补全
   1. 需求分析

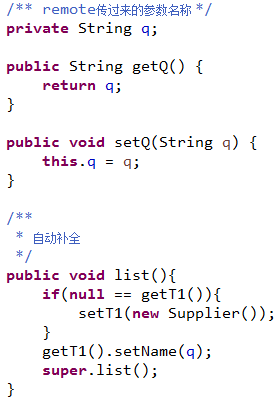
**在采购申请（销售订单录入）界面中，当用户在供应商（客户）输入部分内容，**

**下拉面板中的数据会自动模糊搜索给出结果**



* 1. 代码实现

1. 修改SupplierAction，加入



1. 修改 order\_add.js 添加 mode 属性为 remote

