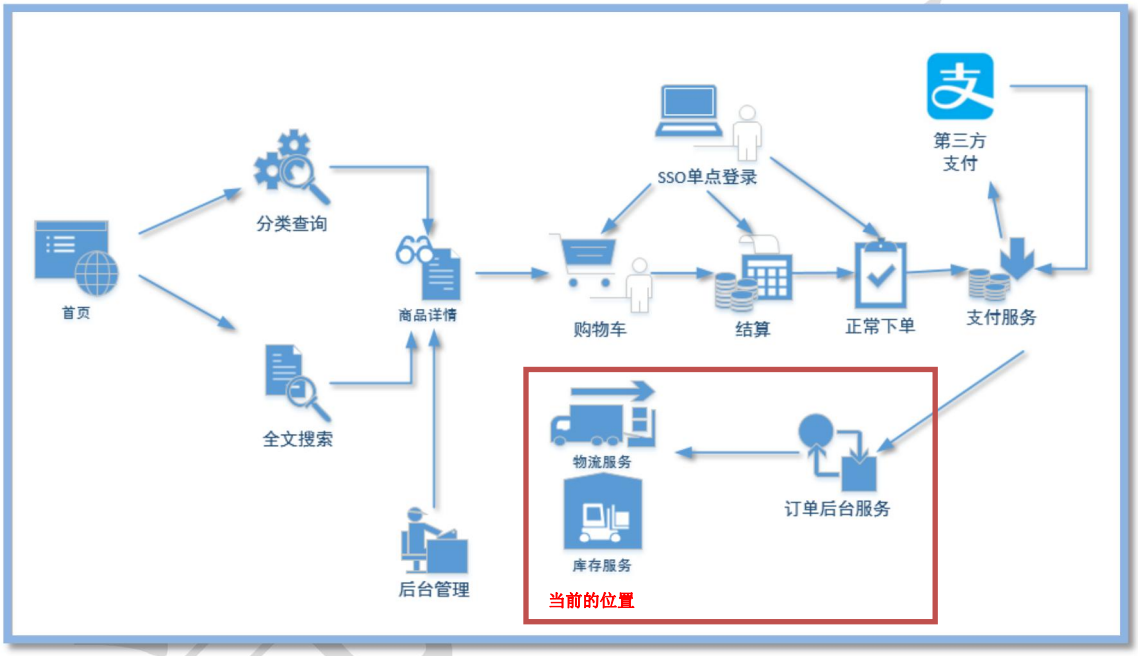
拆单

版本：V 1.0

www.atguigu.com



一、拆单简介

1 为什么要拆单

因为同一个订单中不同的商品可能会因为商家不同、仓库不同、物流渠道不同。分成多个子订单分别进行跟踪。

2 何时拆单

可以选择下单时拆分，也可以选择支付时拆单。谷粒商城的项目以京东为参考选择支付时拆单。

3 拆单对用户支付是否有影响？

用户支付时不影响拆单，如果是因为多商家进行的拆单，用户付款的金额会进行分账到不同的商家。

二 开发

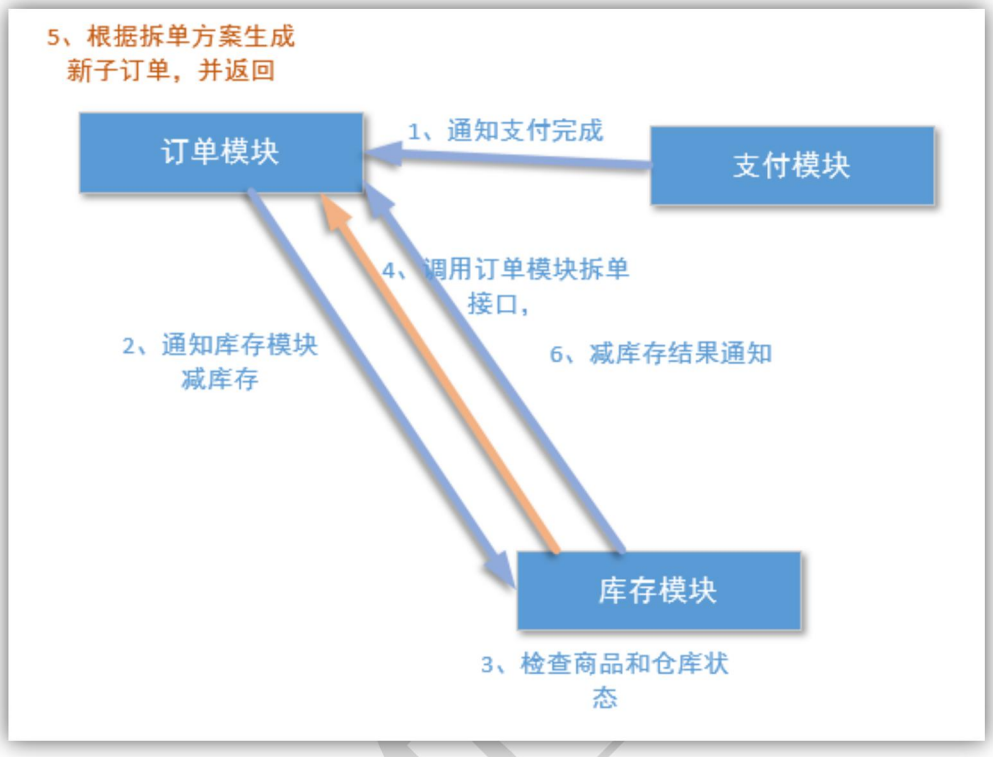
1 分析

本章案例讲解有库存系统检查发现单笔订单中的商品存在于不同仓库之中，而触发拆单动作。由库存模块调用订单模块的拆单接口。

通信方式：

由于拆单动作是不同于通知性任务，必须获得拆单后新的订单编号和结构才能继续往下进行，因此不选择利用消息队列采用异步通信。而采用 **restful** 这种同步通信的方式。

2 示意图



3 库存系统调用拆单的接口

拆单接口说明

由库存系统发起申请

调用接口	http://order.gmall.com/orderSplit	
请求参	orderId	订单系统的订单 ID

数		
	wareSkuMap	仓库编号与商品的对照关系 例如 [{"wareId":"1","skuIds":["2","10"]},{"wareId":"2","skuIds":["3"]}] 表示：sku 为 2 号，10 号的商品在 1 号仓库 sku 为 3 号，10 号的商品在 2 号仓库
请求方式	post	
返回值格式	json 集合	子订单的集合 json
	orderId	订单系统的订单 ID
	consignee	收货人
	consigneeTel	收货电话
	orderComment	订单备注
	orderBody	订单概要
	deliveryAddress	发货地址
	paymentWay	支付方式： ‘1’ 为货到付款，‘2’为在线支付。
	details: skuId skuNum skuName	购买商品明细 例如： details:[{skuId:101,skuNum:1,skuName: ’小米手 64G’}, {skuId:201,skuNum:1,skuName:’索尼耳机’}]
	wareId	传入时的仓库编号
返回值例：	[{ "orderBody": "小米 红米 5 移动联通电信 4G 手机 双卡双待等 1 件商品", "consignee": "晨哥", "orderComment": "123123",	

```
    "wareId": "1",
    "orderId": "16",
    "deliveryAddress": "宏福科技综合楼",
    "details": [
      {
        "skuName": "小米 红米 5 动联通电信 4G 手机 双卡双待",
        "skuId": "2",
        "skuNum": 7
      }
    ],
    "paymentWay": "2"
  },
  {
    "orderBody": "小米 红米 5 Plus 全面屏手 版 4GB+64GB 黑色 等 1 件商品",
    "consignee": "晨哥",
    "orderComment": "123123",
    "wareId": "2",
    "orderId": "17",
    "deliveryAddress": "宏福科技综合楼",
    "details": [
      {
        "skuName": "小米 红米 5 Plus 全 +64GB 黑色 ",
        "skuId": "3",
        "skuNum": 3
      }
    ],
    "paymentWay": "2"
  }
]
```


三 代码实现

OrderController

```
@RequestMapping(value = "orderSplit",method = RequestMethod.POST)
@ResponseBody
public String orderSplit(HttpServletRequest request){
    String orderId = request.getParameter("orderId");
    String wareSkuMapJson = request.getParameter("wareSkuMap");
     List<OrderInfo> subOrderInfoList = orderService.splitOrder(orderId,wareSkuMapJson);
```

```
        List wareSkuMapList=new ArrayList();
        for (OrderInfo orderInfo : subOrderInfoList) {
            Map map = orderService.initWareOrder(orderInfo);
            wareSkuMapList.add(map);
        }
        String newWareSkuMapJson = JSON.toJSONString(wareSkuMapList);
        return newWareSkuMapJson;
    }
```

OrderServiceImpl

```
public    List<OrderInfo>    splitOrder(String orderId,String wareSkuMapJson){
        List<OrderInfo> subOrderList=new ArrayList<>();
        //转 List    循环 list 获得  仓库与 sku 的对应关系
        List<Map> wareSkuMapList = JSON.parseArray(wareSkuMapJson, Map.class);
        OrderInfo orderInfo = getOrderInfo(orderId);
        for (Map map : wareSkuMapList) {
            String wareId = (String)map.get("wareId");
            List<String> skuidList = (List)map.get("skuIds");

            //每个仓库拆分成一个子订单
            //构造主表
            OrderInfo subOrderInfo=new OrderInfo();
            try {
                BeanUtils.copyProperties(subOrderInfo,orderInfo);
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                e.printStackTrace();
            }
            subOrderInfo.setParentOrderId(orderInfo.getId());
            subOrderInfo.setId(null);
            subOrderInfo.setWareId(wareId);
            List<OrderDetail> subOrderDetailList=new ArrayList<>();
            for (String skuId : skuidList) {
                for (OrderDetail orderDetail :orderInfo.getOrderDetailList()) {
                    if(skuId.equals(orderDetail.getSkuId())){
                        orderDetail.setOrderId(null);
                        orderDetail.setId(null);
                        subOrderDetailList.add(orderDetail);
                    }
                }
            }
            subOrderInfo.setOrderDetailList(subOrderDetailList);
```

```
            subOrderInfo.sumTotalAmount();

            saveOrder(subOrderInfo);

            updateOrderStatus(orderInfo.getId(),ProcessStatus.SPLIT);

            subOrderList.add(subOrderInfo);
        }

    return subOrderList;
}
```

在不影响原程序下改造 initWareOrder 方法

```
public String initWareOrder(String orderId) {
     OrderInfo orderInfo = getOrderInfo(orderId);
     Map map = initWareOrder(orderInfo);
     return JSON.toJSONString(map);
}


public Map  initWareOrder(OrderInfo orderInfo){

    Map map=new HashMap();
    map.put("orderId",orderInfo.getId());
    map.put("consignee", orderInfo.getConsignee());
    map.put("consigneeTel",orderInfo.getConsigneeTel());
    map.put("orderComment",orderInfo.getOrderComment());
    map.put("orderBody",orderInfo.getTradeBody());
    map.put("deliveryAddress",orderInfo.getDeliveryAddress());
    map.put("paymentWay","2");
    map.put("wareId",orderInfo.getWareId());

    List detailList=new ArrayList();
    List<OrderDetail> orderDetailList = orderInfo.getOrderDetailList();
    for (OrderDetail orderDetail : orderDetailList) {
        Map  detailMap =new HashMap();
        detailMap.put("skuId",orderDetail.getSkuId());
        detailMap.put("skuName",orderDetail.getSkuName());
```

```
        detailMap.put("skuNum",orderDetail.getSkuNum());

        detailList.add(detailMap);
    }

    map.put("details",detailList);

    return  map;

}
```