# **Android:玩转订单详情三层数据只需要一层RecyclerView，并不需要嵌套View**

我们先看看界面需求：



界面需求.png

那接下来就分析一下我自己的理解和做法。  
如果不想听我啰嗦的人可以直接去下载源码[源码下载链接](https://github.com/19snow93/OrderListLogic" \t "_blank)  
这样看上去，我们就看到有三个不同的item。  
1.一张大订单的含有所有信息的item：



一张大订单.png

2.一张大订单里面分为一张张小订单，每张小订单就代表一间店铺的所有商品，而且每间商铺的第一个商品需要带小订单编号和商铺名字的header：



含有商铺名和订单编号header的item.png

3.下面的就是小订单内非第一件商品的item（不含header）：

IMG_259

小订单内非第一件商品.png

看完这个界面需求，相信很多人都跟我一样，觉得用两层嵌套的RecyclerView、隐藏和显示商品的头布局不就可以完成这样的界面了？  
确实做界面的时候我是这样做的，但是当把数据插入的时候，嵌套View的问题就无缘无故出现了：



两层RecyclerView嵌套的效果.gif

可能gif图大家还看不出问题。



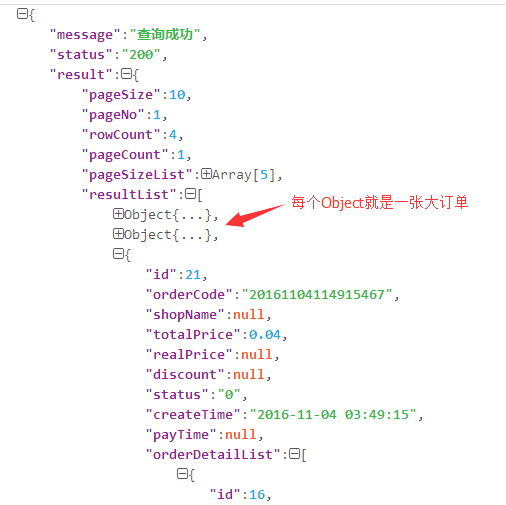
两层RecyclerView嵌套解析图一.png

明明是一张大订单里面有两张不同商铺的小订单，总共有五件商品，两张小订单号分别尾数是133和601，但当我们上拉RecyclerView然后再下拉回去的时候，我发现：

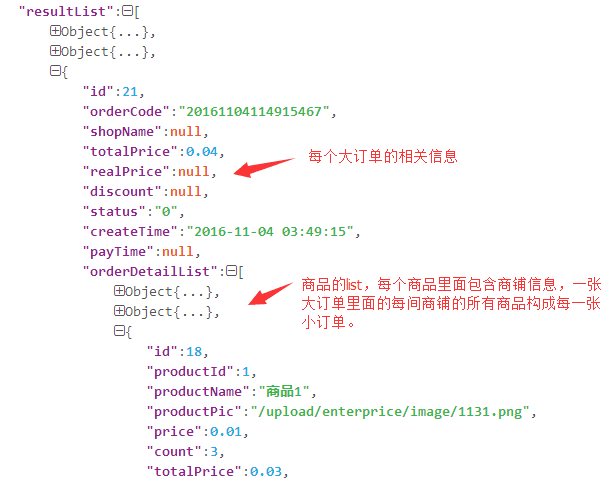


两层RecyclerView嵌套解析图二.png

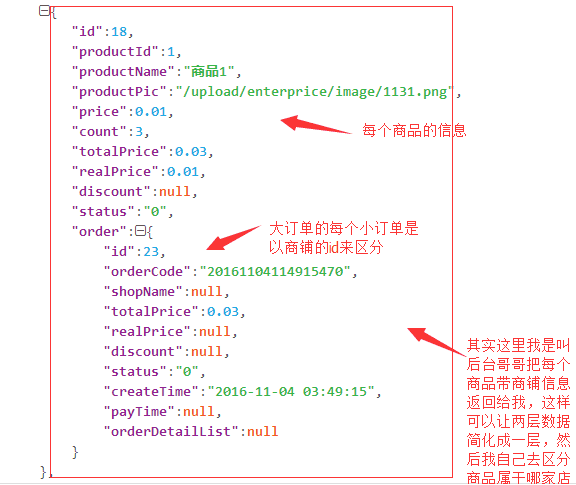
本来同属于一张大订单的尾数为133和601的两张小订单本来5件商品却无缘无故消失了剩下一件商品~  
得到这种结果令我非常诧异，无论我用什么样的方式去验证这样的嵌套RecyclerView的做法，都是会出现这样的情况。我的一位从我学习Android以来一直帮助我的师兄，他也去检查过我写的xml布局和adapter，也没发现问题，但显示的时候总会出现布局丢失的情况。  
最后我发现这是一条死胡同，一个我无法解释的bug。  
其实在用RecyclerView嵌套的时候我已经发现很多无缘无故的显示bug，所以我劝大家还是少用布局嵌套，不然就像我这样作死~  
  
既然这样走嵌套RecyclerView是一条死胡同，那只能用不嵌套的做法来做了，下面我就讲讲我的师兄，他给我的一个做法。确实在这里的做法和逻辑都是师兄写的，我只是做了实现和笔录~  
看这种做法之前，我们先看看后台返回来的数据是怎么样的，才好思考往下怎么去做：



数据结构一：大订单.png



数据结构二：大订单里面有若干个商品.png



数据结构三：每个商品信息含有该商品的商铺信息，每间商铺就是一张小订单.png

我把以上三张图的数据结构简化一下：

"result":{

"大订单List":[

大订单1{...},

大订单2 {

大订单的相关信息

"商品List":[

"商品1"{

"商品信息"

"商铺信息":{

商品所在商铺的信息, 即小订单信息

}

},

"商品2"{...},

"商品3"{...},

]

},

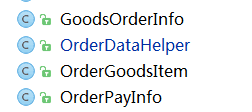
大订单3{...}

]

}

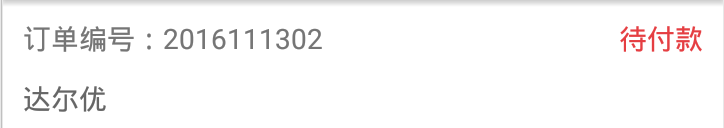
看上去，三层数据怎么把他用一层RecyclerView显示出来呢？这里，不得不说师兄想的方法非常有意思。

**他把这里的所有数据分成三种数据类型，并把RecyclerView分成三种不同的布局，对返回来的数据用不同的对象进行排序并且把排好序的数据最后放进一条List<Object>，让adapter去判断该显示哪种布局**：



三种不同的对象.png

（1）GoodsOrderInfo 表示每个小订单的头部信息（订单号、订单状态、店铺名称）



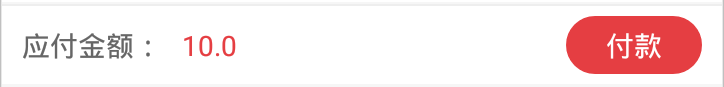
小订单的头部信息.png

（2）OrderGoodsItem 表示小订单中的商品



小订单中的商品.png

（3）OrderPayInfo 表示大订单的支付信息（金额、订单状态）



大订单的支付信息（金额、订单状态）.png

我们来看看代码：

public class OrderDataHelper {

/\*\*

\* List<Object>有三种数据类型：

\* 1、GoodsOrderInfo 表示每个小订单的头部信息（订单号、订单状态、店铺名称）

\* 2、OrderGoodsItem 表示小订单中的商品

\* 3、OrderPayInfo 表示大订单的支付信息（金额、订单状态）

\* @param resultList

\* @return

\*/

public static List<Object> getDataAfterHandle(List<OrderSummary> resultList) {

List<Object> dataList = new ArrayList<Object>();

//遍历每一张大订单

for (OrderSummary orderSummary : resultList) {

//大订单支付的金额和订单状态

OrderPayInfo orderPayInfo = new OrderPayInfo();

orderPayInfo.setTotalAmount(orderSummary.getTotalPrice());

orderPayInfo.setStatus(orderSummary.getStatus());

orderPayInfo.setId(orderSummary.getId());

//小订单商品

List<OrderGoodsItem> orderDetailList = orderSummary.getOrderDetailList();

Map<String, List<OrderGoodsItem>> orderGoodsMap = new HashMap<String, List<OrderGoodsItem>>();

Map<String, GoodsOrderInfo> orderInfoMap = new HashMap<String, GoodsOrderInfo>();

//遍历每个大订单里面的商品

for (OrderGoodsItem orderGoodsItem : orderDetailList) {

//获取商品里面的商铺信息的订单号

String orderCode = orderGoodsItem.getOrder().getOrderCode();

orderGoodsItem.setOrderid(orderSummary.getId());

//拿到相对应订单号的所有商品

List<OrderGoodsItem> goodsList = orderGoodsMap.get(orderCode);

//如果goodsList为空，则新建；

//而且把这个订单号的orderGoodsMap持有订单的对象goodsList

if (goodsList == null) {

goodsList = new ArrayList<>();

orderGoodsMap.put(orderCode, goodsList);

}

//goodsList添加商品的对象，因为orderGoodsMap

//已经持有这个订单号的goodsList对象，所以不用重新put

goodsList.add(orderGoodsItem);

//把小订单的店铺信息赋给GoodsOrderInfo对象，并加入到orderInfoMap

GoodsOrderInfo orderInfo = orderInfoMap.get(orderCode);

if(orderInfo == null) {

orderInfo = new GoodsOrderInfo();

orderInfo.setOrderCode(orderCode);

orderInfo.setShopName(orderGoodsItem.getOrder().getShopName());

orderInfo.setStatus(orderGoodsItem.getOrder().getStatus());

orderInfoMap.put(orderCode, orderInfo);

}

}

//把所有数据按照头部、内容和尾部三个部分排序好

Set<String> keySet = orderGoodsMap.keySet();

for(String orderCode : keySet) {

dataList.add(orderInfoMap.get(orderCode));

dataList.addAll(orderGoodsMap.get(orderCode));

}

dataList.add(orderPayInfo);

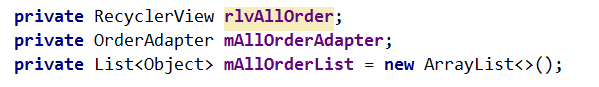
}

return dataList;

}

}

代码很简单，只是一个简单的工具类，就把后台返回来的三层数据分为了一层，一条已经排好序的List<Object>。  
然后我们就去看看简单的adapter数据显示：



变量的声明.png

下面就是拿到数据之后的使用：

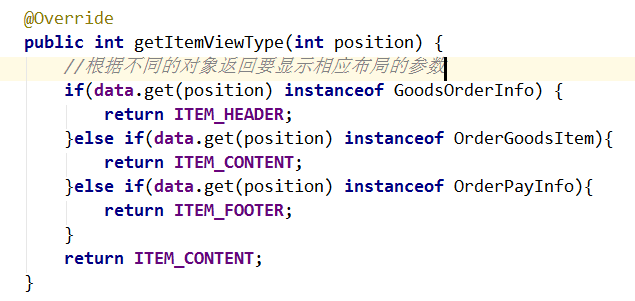
IMG_271

使用.png

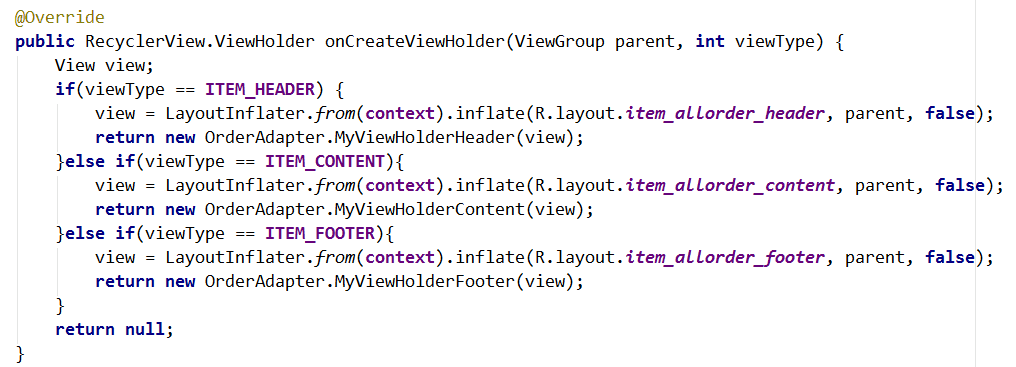
看看adapter怎么写：

IMG_272

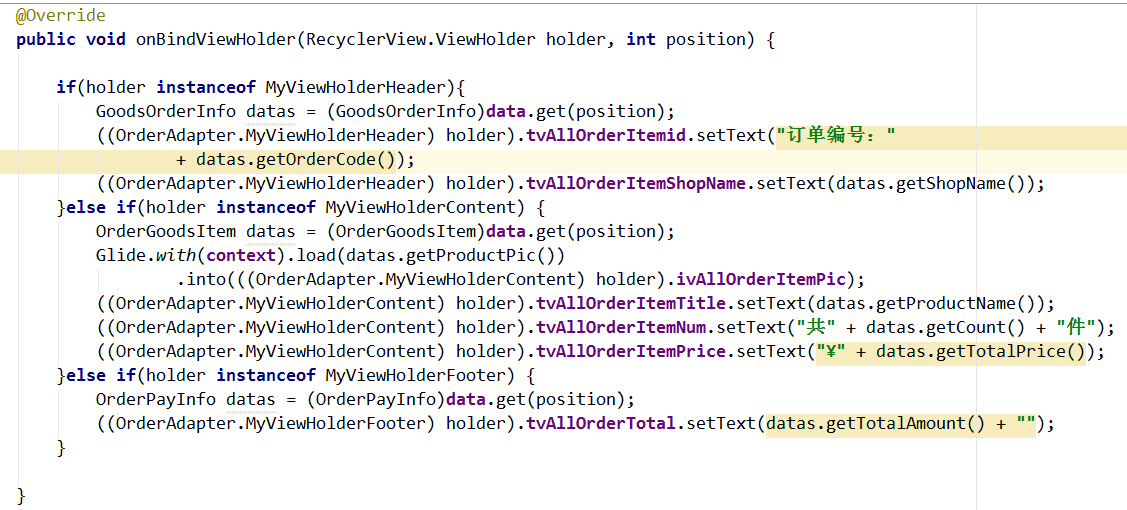
类型.png



根据不同的对象返回要显示相应布局的参数.png



根据不同的参数显示相应的ViewHolder.png



根据不同的布局做相关的处理.png



三种布局最后拼接成了理想的界面.png

是不是很简单？**其实师兄的这种做法同时也利用了RecyclerView可以显示返回不同的布局，然后通过三种不同的布局拼接来完成一层RecyclerView控制三层数据的相关做法**，而且这样做并不会用到该死的嵌套RecyclerView，不会有那该死的不知道拿来的显示bug，我个人觉得真的很赞！你们觉得如何？

文／19snow93（简书作者）  
原文链接：http://www.jianshu.com/p/b25340903671  
著作权归作者所有，转载请联系作者获得授权，并标注“简书作者”。