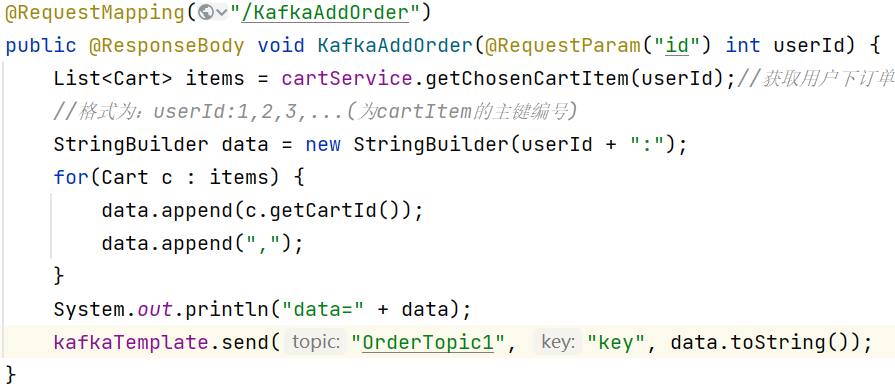
作业2

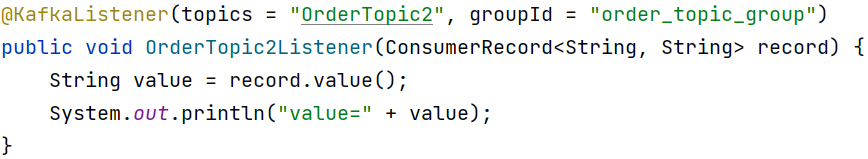
程序设计方案：

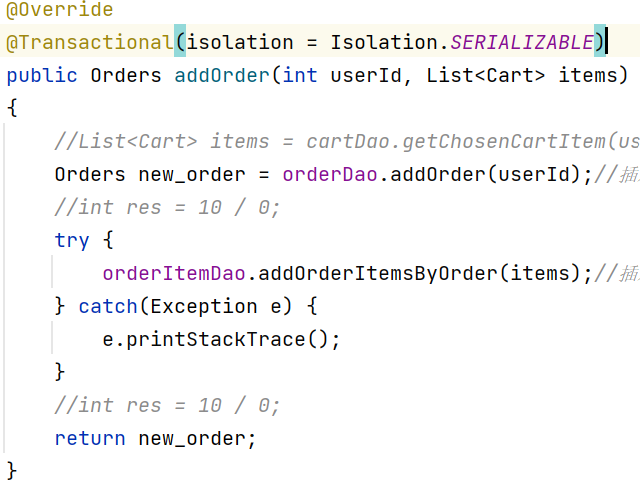
1. 逻辑结构与老师上课提供的实例类似，主要是controller层在接收前端的请求之后，将前端发送过来的数据拆分重新组装成我们自定义的格式，之后将消息异步的发送到topic，之后有listener来监听发过来的请求并处理，并将结果发送到另一个不同的topic中去。
2. 消息数据格式为：userId:cartId,cartId,cartId,.....。其中cartId为用户在购物车中下订单时选中的购物车条目的在数据库中对应的主键id。
3. Topic配置项没有做额外的配置，采用springBoot默认的配置。
4. 我使用的订单结果呈现方式为：使用的是老师的作业要求中的**第二种方式，即通过前端额外的Ajax请求来主动的获取新添加的订单的状态**。具体实现逻辑为：在发送新的订单生成请求之前，想通过fetch获取当前的最大订单主键编号，(由于在本系统中当用户删除订单时候会将数据库中对应的finish属性设置为2，所以并不会因为删除而出现错误)之后在发送异步创建订单的请求。在请求发出后，每个500ms向后端发送一个请求检查当前的订单编号是否已经变为+1，若已经变成+1，则说明已经处理成功，在前端显示成功信息，并移除购物车中对应项，若还没有变成+1，则说明还在处理，在前端显示正在处理的相关信息。
5. 代码截图如下：

后端：





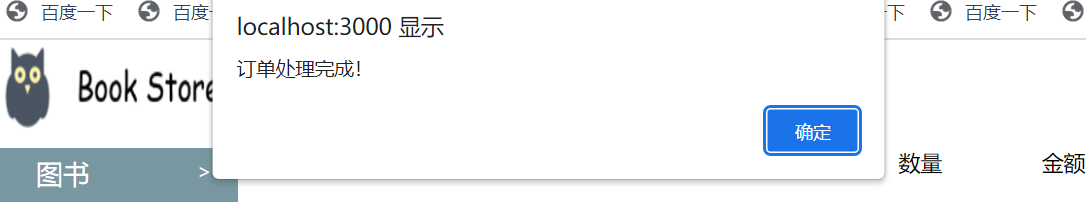




前端：







三种前端呈现方式的优缺点分析：

第一种：

优点：只在前端做监听，后端不需要额外的监听器，简化了后端的代码结构，并且可以在后端完成之后前端立刻做出对应的刷新。

缺点：使得前端的关于订单结果的处理部分边的比较复杂，同时这种在前端直接监听后端进行处理的topic中消息处理结果的方式也与我们想要实现的在后端一个topic用于处理消息，一个topic用于接受结果并将结果返回到前端的方式相矛盾。

第二种：

优点：相对于监听器而言，方法2只需要持续的发送ajax请求就可以，实现逻辑简单，前后端代码结构都比较简单。

缺点：需要频繁的ajax请求，在用户量大时可能会产生阻塞，另外，是每过一段固定时间才会发送，若在时间间隔内后端处理完成，则会导致前端处理结果显示有时间上的延迟。并且该方法很难处理下订单失败的情况。

第三种：

优点：在前端的代码逻辑中只需要获取后端返回的消息处理结果就可以直接的将结果刷新到页面上去，而不需要像前两种方式一样在发出请求之后还需要主动的去获取处理结果。使得前端实现比较简单。

缺点：在后端需要额外实现webSocket逻辑，使得后端的用于获取消息处理结果的topic中的处理逻辑变得复杂，代码结构也更加复杂。