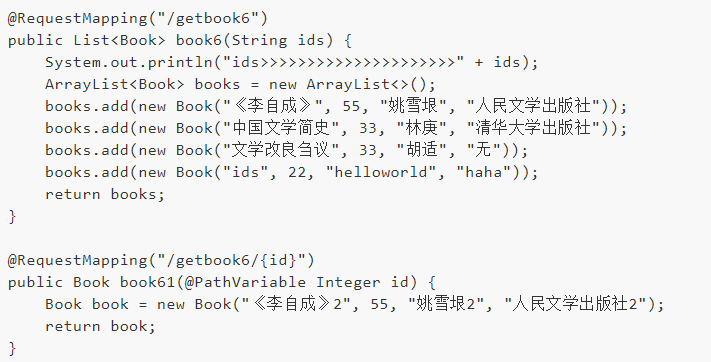
1. 在服务提供者中提供两个接口供服务消费者调用

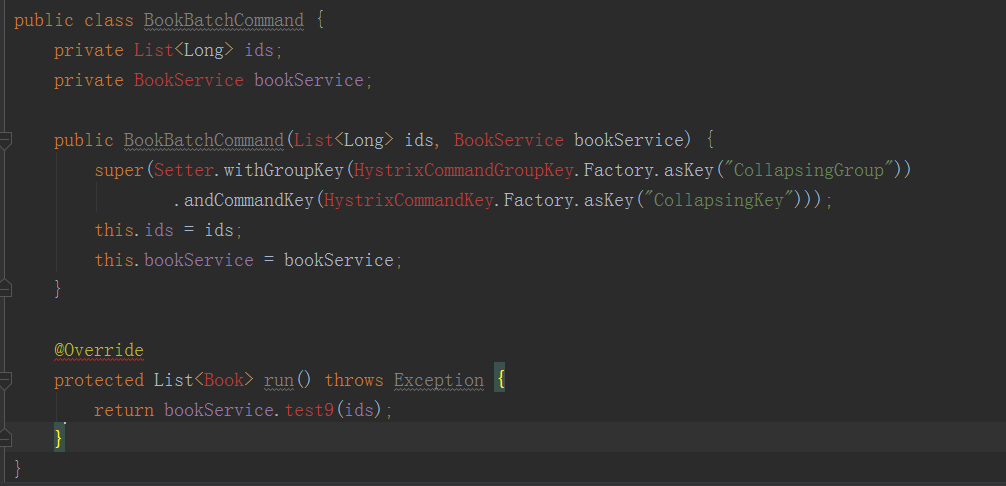


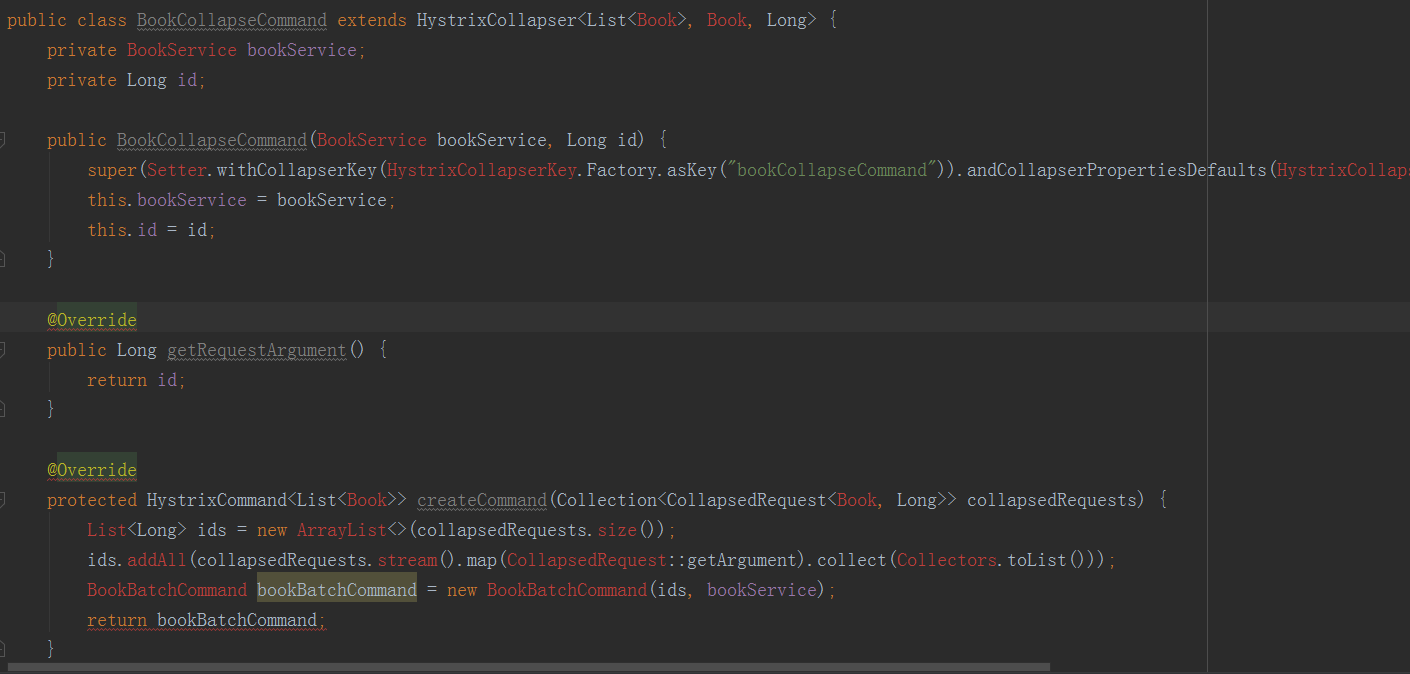
第一个接口是一个批处理接口，第二个接口是一个处理单个请求的接口

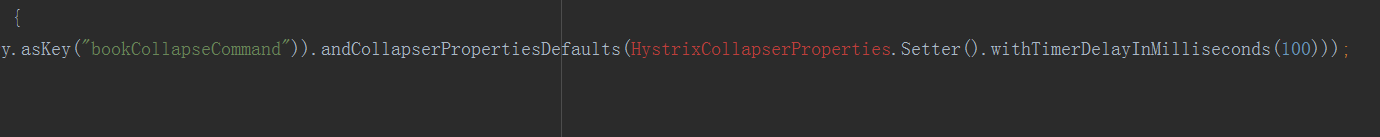
1. 服务提供者处理好之后，服务消费者 在BookService中添加两个方法用来调用服务提供者提供的接口

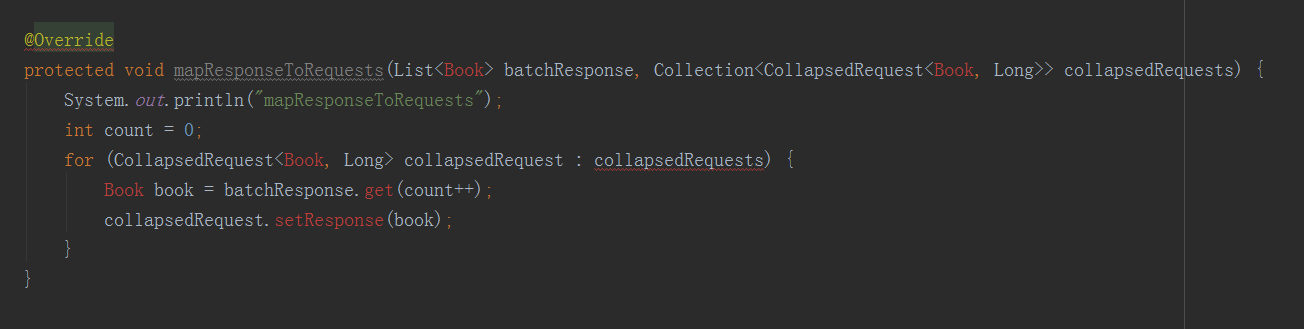


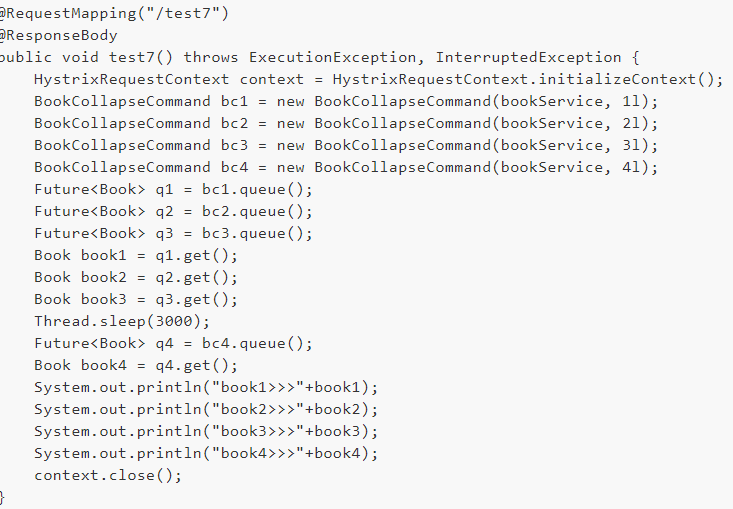
test8用来调用提供单个id的接口，test9用来调用批处理的接口

1. 创建一个BookBatchCommand
2. 创建BookCollapseCommand继承自HystrixCollapser来实现请求合并







1. 首先在构造方法中，我们设置了请求时间窗为100ms，即请求时间间隔在100ms之内的请求会被合并为一个请求。  
   2.createCommand方法主要用来合并请求，在这里获取到各个单个请求的id，将这些单个的id放到一个集合中，然后再创建出一个BookBatchCommand对象，用该对象去发起一个批量请求。  
   3.mapResponseToRequests方法主要用来为每个请求设置请求结果。该方法的第一个参数batchResponse表示批处理请求的结果，第二个参数collapsedRequests则代表了每一个被合并的请求，然后我们通过遍历batchResponse来为collapsedRequests设置请求结果。
2. 测试：

1.首先要初始化HystrixRequestContext  
2.创建BookCollapseCommand类的实例来发起请求，先发送3个请求，然后睡眠3秒钟，再发起1个请求，这样，前3个请求就会被合并为一个请求，第四个请求因为间隔的时间比较久，所以不会被合并，而是单独创建一个线程去处理。

**通过注解实现请求合并**

在BookService中添加两个方法



在test10方法上添加@HystrixCollapser注解实现请求合并，用batchMethod属性指明请求合并后的处理方法，collapserProperties属性指定其他属性。

在BookService中写好之后，直接调用就可以了

## 总结

多个请求被合并为一个请求进行一次性处理，可以有效节省网络带宽和线程池资源，但是，有优点必然也有缺点，设置请求合并之后，本来一个请求可能5ms就搞定了，但是现在必须再等10ms看看还有没有其他的请求一起的，这样一个请求的耗时就从5ms增加到15ms了，不过，如果要发起的命令本身就是一个高延迟的命令，那么这个时候就可以使用请求合并了，因为这个时候时间窗的时间消耗就显得微不足道了，另外高并发也是请求合并的一个非常重要的场景