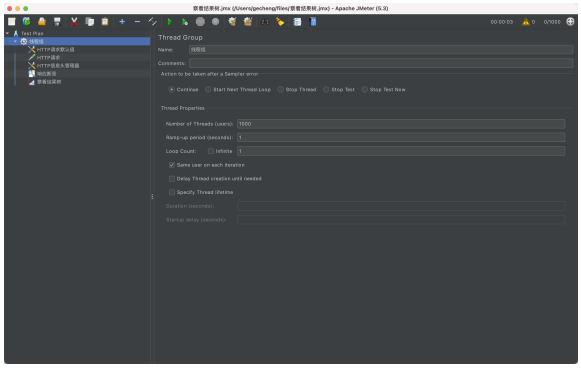
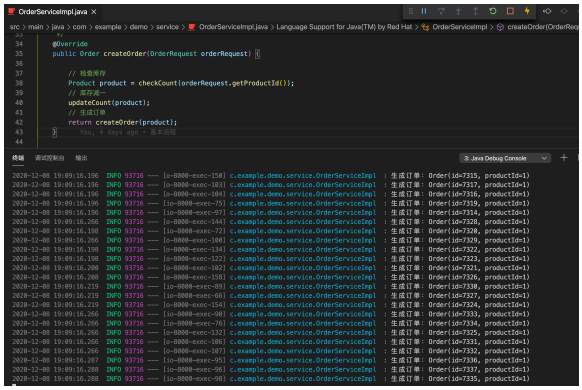
秒杀demo

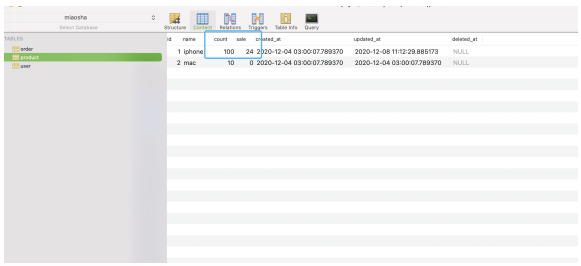
基本流程

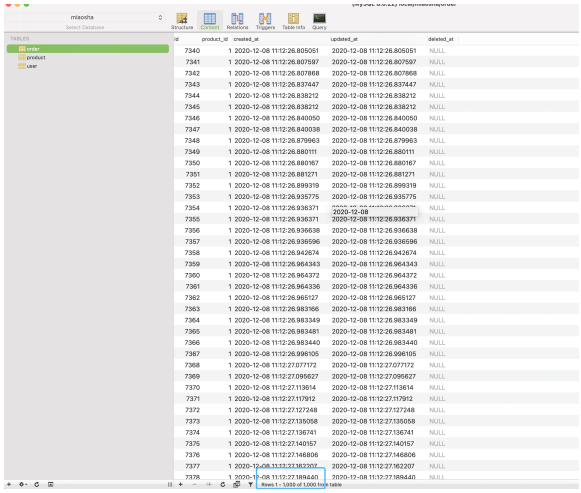
- 检查库存
- 库存减一
- 生成订单

超卖









乐观锁避免超卖

高并发下卖出率不高

OrderServiceImpl.java

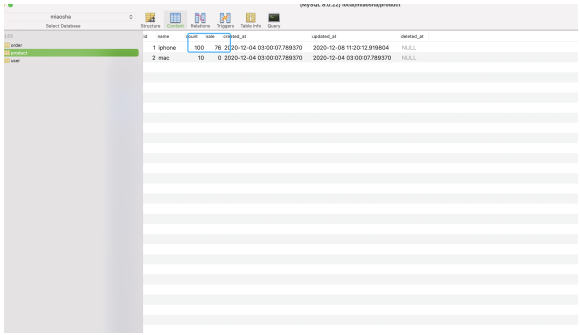
```
68              throw new RuntimeException(msg);
69          }
70          return product.get();
71      }
72
73      // 更新库存
74      private void updateCount(Product product) {
75          You, 4 days ago • 基本流程
76          int rowAffected = productService.updateByPessimisticLock(product);
77
78          if (rowAffected == 0) {
79              throw new RuntimeException("并发更新失败，乐观锁version不匹配");
80          }
81
82      }
83
84      public Order createOrder(Product product) {
```

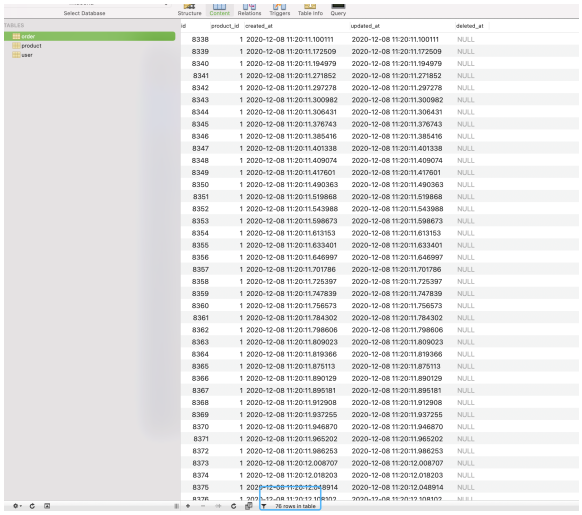
ProductMapper.xml

```
46              miaosha.product
47          <set >
48              <if test="name != null">
49                  name = #{name},
50              </if>
51              <if test="count != null">
52                  count = #{count},
53              </if>
54              <if test="sale != null">
55                  sale = sale + 1,
56              </if>
57          </set>
58          WHERE
59              id = #{id}
60              AND sale = #{sale}
61      </update>
62
63  </mapper>
```

终端 调试控制台 输出 3: Java Debug Console

```
2020-12-08 19:20:12.601  WARN 95054 --- [io-8000-exec-31] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.601  WARN 95054 --- [io-8000-exec-23] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.601  WARN 95054 --- [nio-8000-exec-5] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.603  WARN 95054 --- [io-8000-exec-76] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.612  WARN 95054 --- [o-8000-exec-154] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.623  INFO 95054 --- [io-8000-exec-69] c.example.demo.service.OrderServiceImpl  : 生成订单: Order(id=8401, productId=1)
2020-12-08 19:20:12.626  INFO 95054 --- [o-8000-exec-193] c.example.demo.service.OrderServiceImpl  : 生成订单: Order(id=8402, productId=1)
2020-12-08 19:20:12.627  WARN 95054 --- [io-8000-exec-71] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.628  WARN 95054 --- [io-8000-exec-34] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version不匹配]
2020-12-08 19:20:12.629  WARN 95054 --- [o-8000-exec-100] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 并发更新失败，乐观锁version
```

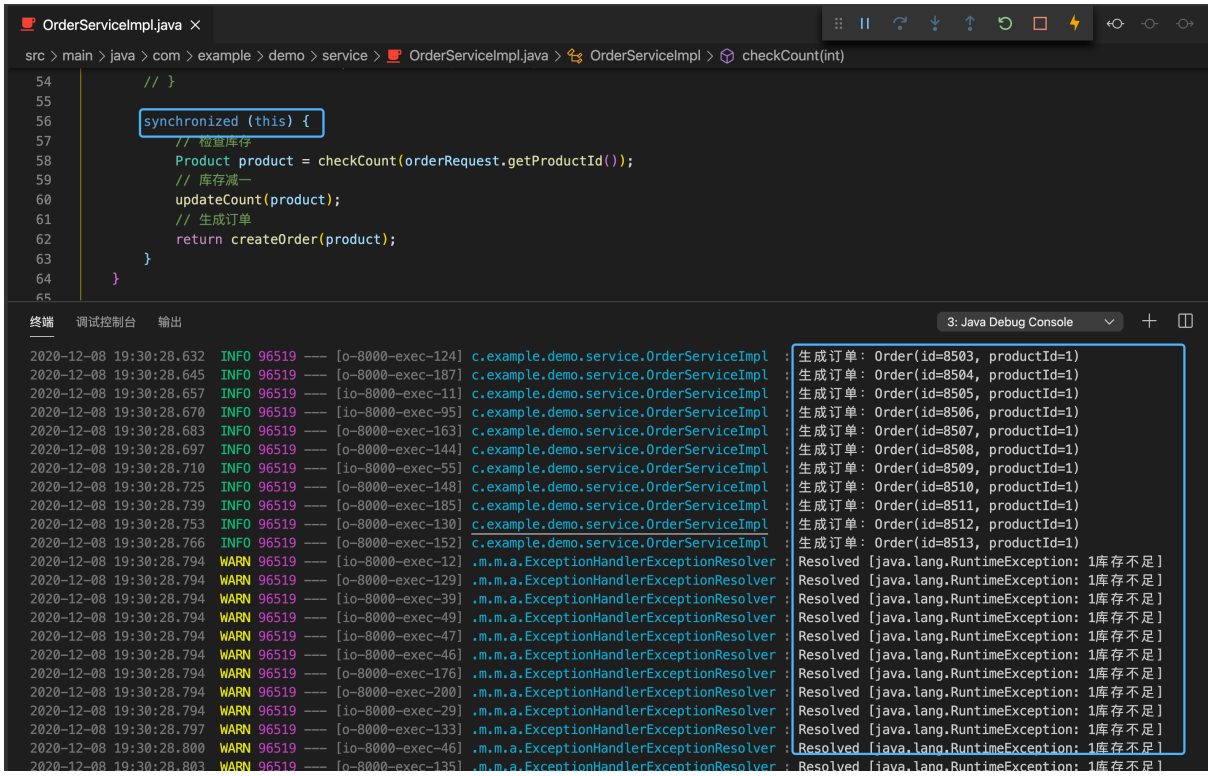


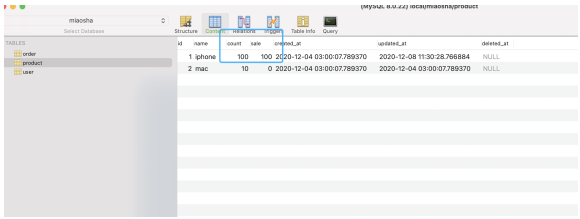


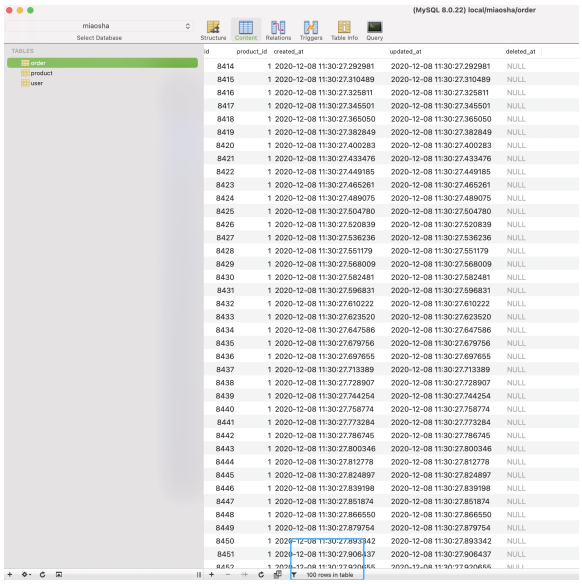
悲观锁避免超卖

- 事务 `@Transactional(rollbackFor = Exception.class, propagation = Propagation.REQUIRED)`
- synchronized

> 悲观锁在大量请求的请求下，有着更好的卖出成功率。但是需要注意的是，如果请求量巨大，悲观锁会导致后面的请求进行了长时间的阻塞等待

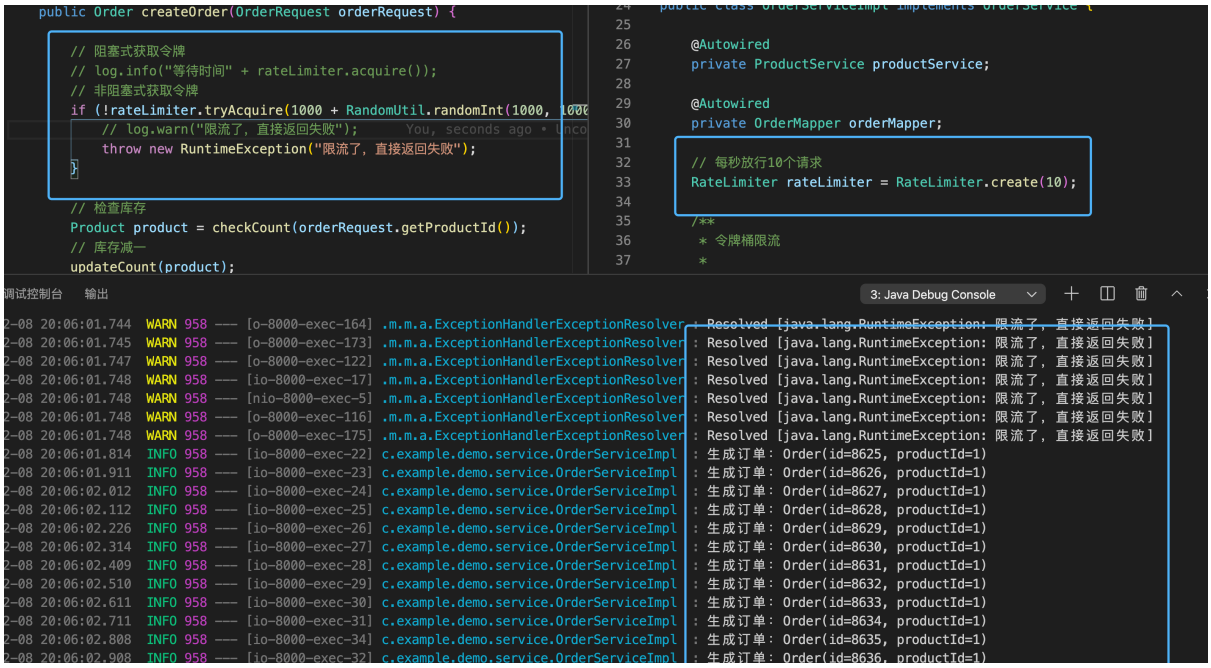






令牌桶限流

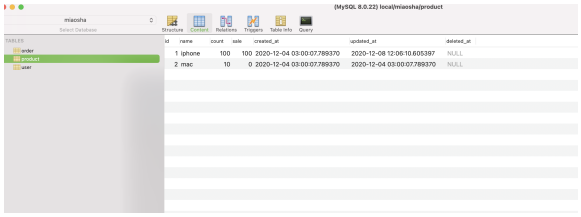
控制流量 提高乐观锁卖出成功率

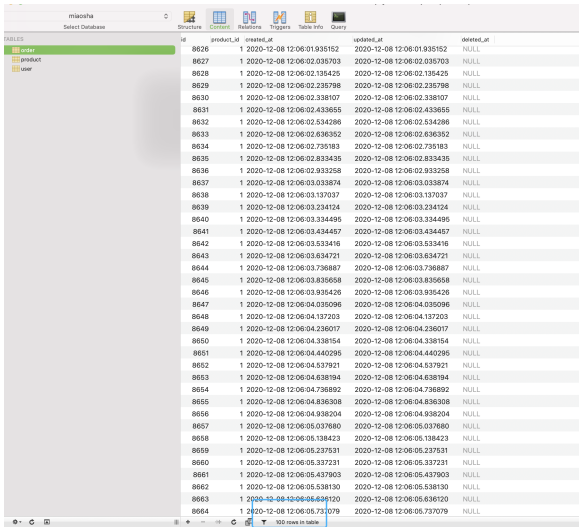


```
public Order createOrder(OrderRequest orderRequest) {

    // 阻塞式获取令牌
    // log.info("等待时间" + rateLimiter.acquire());
    // 非阻塞式获取令牌
    if (!rateLimiter.tryAcquire(1000 + RandomUtil.randomInt(1000, 1000
        // log.warn("限流了，直接返回失败");        You, seconds ago • Unco
        throw new RuntimeException("限流了，直接返回失败");
    }

    // 检查库存
    Product product = checkCount(orderRequest.getProductId());
    // 库存减一
    updateCount(product);
```

```
25
26      @Autowired
27      private ProductService productService;
28
29      @Autowired
30      private OrderMapper orderMapper;
31
32      // 每秒放行10个请求
33      RateLimiter rateLimiter = RateLimiter.create(10);
34
35      /**
36       * 令牌桶限流
37       *
```

```
调试控制台    输出                                                    3: Java Debug Console

2-08 20:06:01.744  WARN 958 --- [o-8000-exec-164] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.745  WARN 958 --- [o-8000-exec-173] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.747  WARN 958 --- [o-8000-exec-122] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.748  WARN 958 --- [io-8000-exec-17] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.748  WARN 958 --- [nio-8000-exec-5] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.748  WARN 958 --- [o-8000-exec-116] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.748  WARN 958 --- [o-8000-exec-175] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了，直接返回失败]
2-08 20:06:01.814  INFO 958 --- [io-8000-exec-22] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8625, productId=1)
2-08 20:06:01.911  INFO 958 --- [io-8000-exec-23] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8626, productId=1)
2-08 20:06:02.012  INFO 958 --- [io-8000-exec-24] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8627, productId=1)
2-08 20:06:02.112  INFO 958 --- [io-8000-exec-25] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8628, productId=1)
2-08 20:06:02.226  INFO 958 --- [io-8000-exec-26] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8629, productId=1)
2-08 20:06:02.314  INFO 958 --- [io-8000-exec-27] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8630, productId=1)
2-08 20:06:02.409  INFO 958 --- [io-8000-exec-28] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8631, productId=1)
2-08 20:06:02.510  INFO 958 --- [io-8000-exec-29] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8632, productId=1)
2-08 20:06:02.611  INFO 958 --- [io-8000-exec-30] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8633, productId=1)
2-08 20:06:02.711  INFO 958 --- [io-8000-exec-31] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8634, productId=1)
2-08 20:06:02.808  INFO 958 --- [io-8000-exec-34] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8635, productId=1)
2-08 20:06:02.908  INFO 958 --- [io-8000-exec-32] c.example.demo.service.OrderServiceImpl  : 生成订单：Order(id=8636, productId=1)
```



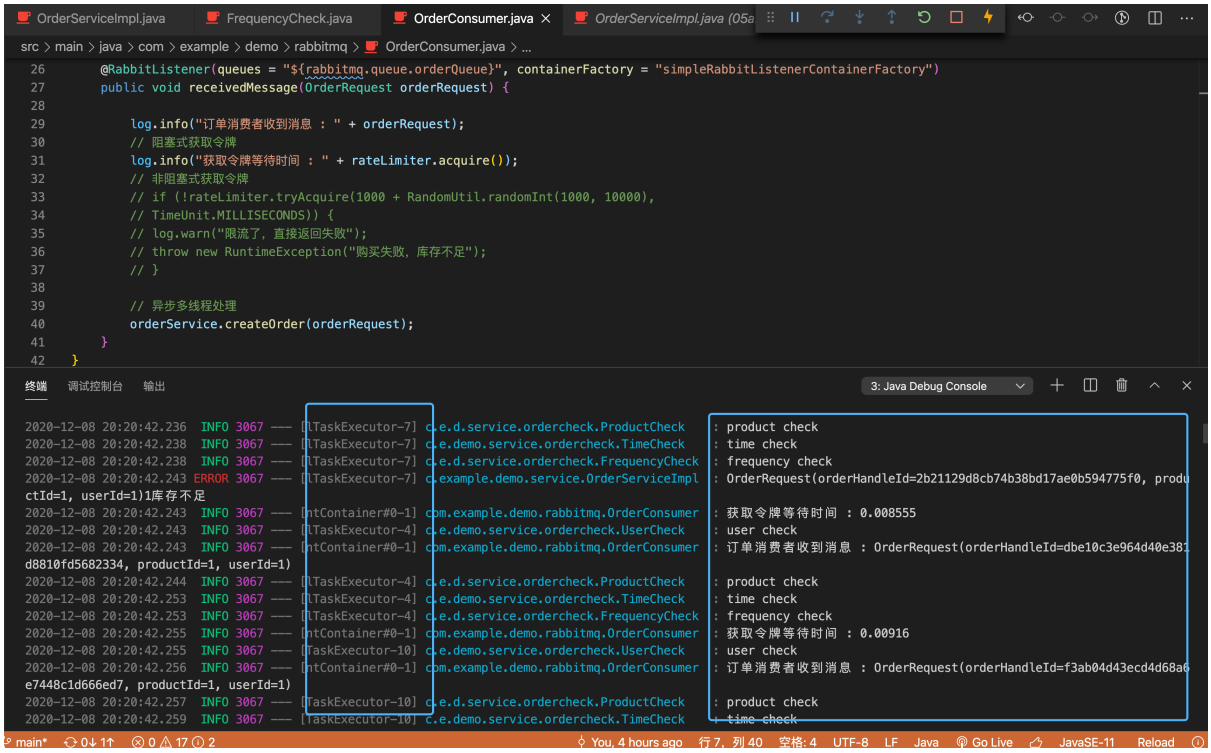


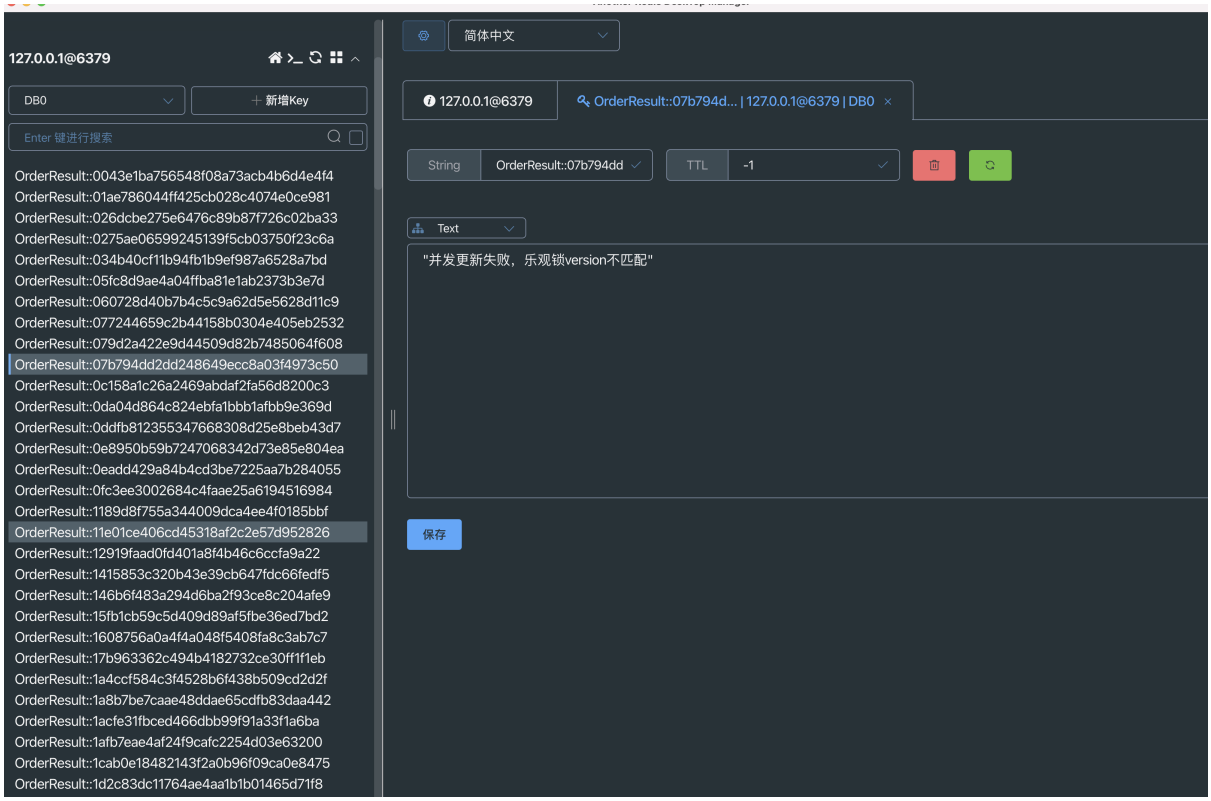
责任链模式对生成订单必要条件合法性检查

- 商品ID
- 用户ID
- 访问频率
- 是否已经购买
- 是否到了购买时间

```
@Slf4j
@Component
@Order(4)
public class FrequencyCheck implements OrderCheck {

  @Autowired
  private RedisUtil redisUtil;

  @Override
  public void check(OrderRequest orderRequest) {

    log.info("frequency check");

    // redis 频率限制key
    String limitKey = "miaosha_" + orderRequest.getUserId() + "_" + orderRequest.getProductId();

    Object value = redisUtil.get(limitKey);
    if (value != null) {
      if ((int) value > 1000) {
        throw new RuntimeException("购买失败，超过频率限制");
      }
      redisUtil.incr(limitKey, 1);
    } else {
      redisUtil.set(limitKey, 1, 60);
    }

  }

}
//--------
@Autowired
private List<OrderCheck> orderCheckChain;
//-----------

for (int i = 0; i < orderCheckChain.size(); i++) {
      orderCheckChain.get(i).check(orderRequest);
}
```

消息队列 异步处理


OrderServiceImpl.java
FrequencyCheck.java
OrderConsumer.java
src > main > java > com > example > demo > rabbitmq > OrderConsumer.java > ...
@RabbitListener(queues = "${rabbitmq.queue.orderQueue}", containerFactory = "simpleRabbitListenerContainerFactory")
public void receivedMessage(OrderRequest orderRequest) {
log.info("订单消费者收到消息 : " + orderRequest);
// 阻塞式获取令牌
log.info("获取令牌等待时间 : " + rateLimiter.acquire());
// 非阻塞式获取令牌
// if (!rateLimiter.tryAcquire(1000 + RandomUtil.randomInt(1000, 10000),
// TimeUnit.MILLISECONDS)) {
// log.warn("限流了，直接返回失败");
// throw new RuntimeException("购买失败，库存不足");
// }
// 异步多线程处理
orderService.createOrder(orderRequest);
}
}
终端
调试控制台
输出
3: Java Debug Console
2020-12-08 20:20:42.236 INFO 3067 --- [lTaskExecutor-7] c.e.d.service.ordercheck.ProductCheck : product check
2020-12-08 20:20:42.238 INFO 3067 --- [lTaskExecutor-7] c.e.demo.service.ordercheck.TimeCheck : time check
2020-12-08 20:20:42.238 INFO 3067 --- [lTaskExecutor-7] c.e.d.service.ordercheck.FrequencyCheck : frequency check
2020-12-08 20:20:42.243 ERROR 3067 --- [lTaskExecutor-7] c.example.demo.service.OrderServiceImpl : OrderRequest(orderHandleId=2b21129d8cb74b38bd17ae0b594775f0, produ
ctId=1, userId=1)1库存不足
2020-12-08 20:20:42.243 INFO 3067 --- [ntContainer#0-1] com.example.demo.rabbitmq.OrderConsumer : 获取令牌等待时间 : 0.008555
2020-12-08 20:20:42.243 INFO 3067 --- [lTaskExecutor-4] c.e.demo.service.ordercheck.UserCheck : user check
2020-12-08 20:20:42.243 INFO 3067 --- [ntContainer#0-1] com.example.demo.rabbitmq.OrderConsumer : 订单消费者收到消息 : OrderRequest(orderHandleId=dbe10c3e964d40e381
d8810fd5682334, productId=1, userId=1)
2020-12-08 20:20:42.244 INFO 3067 --- [lTaskExecutor-4] c.e.d.service.ordercheck.ProductCheck : product check
2020-12-08 20:20:42.253 INFO 3067 --- [lTaskExecutor-4] c.e.demo.service.ordercheck.TimeCheck : time check
2020-12-08 20:20:42.253 INFO 3067 --- [lTaskExecutor-4] c.e.d.service.ordercheck.FrequencyCheck : frequency check
2020-12-08 20:20:42.255 INFO 3067 --- [ntContainer#0-1] com.example.demo.rabbitmq.OrderConsumer : 获取令牌等待时间 : 0.00916
2020-12-08 20:20:42.255 INFO 3067 --- [TaskExecutor-10] c.e.demo.service.ordercheck.UserCheck : user check
2020-12-08 20:20:42.256 INFO 3067 --- [ntContainer#0-1] com.example.demo.rabbitmq.OrderConsumer : 订单消费者收到消息 : OrderRequest(orderHandleId=f3ab04d43ecd4d68a6
e7448c1d666ed7, productId=1, userId=1)
2020-12-08 20:20:42.257 INFO 3067 --- [TaskExecutor-10] c.e.d.service.ordercheck.ProductCheck : product check
2020-12-08 20:20:42.259 INFO 3067 --- [TaskExecutor-10] c.e.demo.service.ordercheck.TimeCheck : time check
main*
0↓ 1↑
You, 4 hours ago
行 7，列 40
空格: 4
UTF-8
LF
Java
Go Live
JavaSE-11
Reload


127.0.0.1@6379
DB0
新增Key
Enter 键进行搜索
OrderResult::0043e1ba756548f08a73acb4b6d4e4f4
OrderResult::01ae786044ff425cb028c4074e0ce981
OrderResult::026dcbe275e6476c89b87f726c02ba33
OrderResult::0275ae06599245139f5cb03750f23c6a
OrderResult::034b40cf11b94fb1b9ef987a6528a7bd
OrderResult::05fc8d9ae4a04ffba81e1ab2373b3e7d
OrderResult::060728d40b7b4c5c9a62d5e5628d11c9
OrderResult::077244659c2b44158b0304e405eb2532
OrderResult::079d2a422e9d44509d82b7485064f608
OrderResult::07b794dd2dd248649ecc8a03f4973c50
OrderResult::0c158a1c26a2469abdaf2fa56d8200c3
OrderResult::0da04d864c824ebfa1bbb1afbb9e369d
OrderResult::0ddfb812355347668308d25e8beb43d7
OrderResult::0e8950b59b7247068342d73e85e804ea
OrderResult::0eadd429a84b4cd3be7225aa7b284055
OrderResult::0fc3ee3002684c4faae25a6194516984
OrderResult::1189d8f755a344009dca4ee4f0185bbf
OrderResult::11e01ce406cd45318af2c2e57d952826
OrderResult::12919faad0fd401a8f4b46c6ccfa9a22
OrderResult::1415853c320b43e39cb647fdc66fedf5
OrderResult::146b6f483a294d6ba2f93ce8c204afe9
OrderResult::15fb1cb59c5d409d89af5fbe36ed7bd2
OrderResult::1608756a0a4f4a048f5408fa8c3ab7c7
OrderResult::17b963362c494b4182732ce30ff1f1eb
OrderResult::1a4ccf584c3f4528b6f438b509cd2d2f
OrderResult::1a8b7be7caae48ddae65cdfb83daa442
OrderResult::1acfe31fbced466dbb99f91a33f1a6ba
OrderResult::1afb7eae4af24f9cafc2254d03e63200
OrderResult::1cab0e18482143f2a0b96f09ca0e8475
OrderResult::1d2c83dc11764ae4aa1b1b01465d71f8
简体中文
127.0.0.1@6379
OrderResult::07b794d... | 127.0.0.1@6379 | DB0
String
OrderResult::07b794dd
TTL
-1
Text
"并发更新失败，乐观锁version不匹配"
保存