

# 秒杀demo

基本流程

超卖

乐观锁避免超卖

悲观锁避免超卖

金牌桶限流

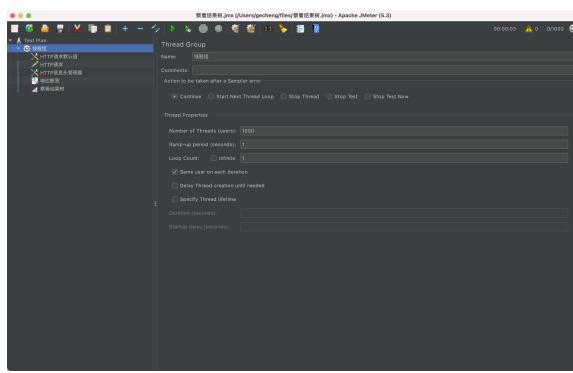
#### 责任链模式对生成订单必要条件合法性检查

消息队列 异步处理

## 基本流程

- 检查库存
  - 库存减一
  - 生成订单

超卖



maista	Salon Location		Document	File	Print	Help	Logout	
TABLES	order	id	name	count	size	created_at	updated_at	defined_at
	1 phone	100	24	2020-12-04 03:00:077899370	2020-12-08 11:29:88573	NULL		
	2 mac	10	0	2020-12-04 03:00:077899370	2020-12-04 03:00:077899370	NULL		
	user							

	masooha	Structure	Tables	Views	Triggers	Index	Query
TABLES	product_id	product_id	product_id	product_id	product_id	product_id	product_id
+  order	7340	1	2020-12-08 11:26.805051			2020-12-08 11:12.26.805051	selected_id
+  product	7341	1	2020-12-08 11:26.807597			2020-12-08 11:12.26.807597	
+  user	7342	1	2020-12-08 11:26.807896			2020-12-08 11:12.26.807896	
	7343	1	2020-12-08 11:26.808292			2020-12-08 11:12.26.808292	
	7344	1	2020-12-08 11:26.808292			2020-12-08 11:12.26.808292	
	7345	1	2020-12-08 11:26.808372			2020-12-08 11:12.26.808372	
	7346	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7347	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7348	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7349	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7350	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7351	1	2020-12-08 11:26.840005			2020-12-08 11:12.26.840005	
	7352	1	2020-12-08 11:26.889971			2020-12-08 11:12.26.889971	
	7353	1	2020-12-08 11:26.893775			2020-12-08 11:12.26.893775	
	7354	1	2020-12-08 11:26.893775			2020-12-08 11:12.26.893775	
	7355	1	2020-12-08 11:26.893775			2020-12-08 11:12.26.893775	
	7356	1	2020-12-08 11:26.893775			2020-12-08 11:12.26.893775	
	7357	1	2020-12-08 11:26.893663			2020-12-08 11:12.26.893663	
	7358	1	2020-12-08 11:26.893663			2020-12-08 11:12.26.893663	
	7359	1	2020-12-08 11:26.893663			2020-12-08 11:12.26.893663	
	7360	1	2020-12-08 11:26.893663			2020-12-08 11:12.26.893663	
	7361	1	2020-12-08 11:26.894336			2020-12-08 11:12.26.894336	
	7362	1	2020-12-08 11:26.895127			2020-12-08 11:12.26.895127	
	7363	1	2020-12-08 11:26.893836			2020-12-08 11:12.26.893836	
	7364	1	2020-12-08 11:26.893839			2020-12-08 11:12.26.893839	
	7365	1	2020-12-08 11:26.893839			2020-12-08 11:12.26.893839	
	7366	1	2020-12-08 11:26.893840			2020-12-08 11:12.26.893840	
	7367	1	2020-12-08 11:26.996095			2020-12-08 11:12.26.996095	
	7368	1	2020-12-08 11:27.207717			2020-12-08 11:12.27.207717	
	7369	1	2020-12-08 11:27.209952			2020-12-08 11:12.27.209952	
	7370	1	2020-12-08 11:27.210194			2020-12-08 11:12.27.210194	
	7371	1	2020-12-08 11:27.210194			2020-12-08 11:12.27.210194	
	7372	1	2020-12-08 11:27.21272748			2020-12-08 11:12.27.21272748	
	7373	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	
	7374	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	
	7375	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	
	7376	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	
	7377	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	
	7378	1	2020-12-08 11:27.21350562			2020-12-08 11:12.27.21350562	

## 乐观锁避免超卖



高并发下卖出率不高

```
OrderServiceImpl.java
ice > OrderServiceImpl.java > Language Support for Java(TM) by Red Hat > OrderServiceImpl
o/          // logger.info(msg);
68      throw new RuntimeException(msg);
69  }
70  return product.get();
}
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 public Order createOrder(Product product) {
84
85
ProductMapper.xml
src > main > resources > mapper > ProductMapper.xml > mapper > update
45
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>

```

order

	id	product_id	created_at	deleted_at
1	1	100	2020-12-04 03:00:07.789370	2020-12-08 11:20:12.919804
2	2	mac	0	2020-12-04 03:00:07.789370

product

	id	name	created_at	deleted_at
1	1	shore	2020-12-08 11:20:11.723509	2020-12-08 11:20:11.723509
2	2	mac	2020-12-04 03:00:07.789370	2020-12-04 03:00:07.789370

## 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，如果请求数量巨大，悲观锁会导致后面的请求进行了长时间的阻塞等待

OrderServiceImpl.java

```

54     // ...
55
56     synchronized (this) {
57         // 检查库存
58         Product product = checkCount(orderRequest.getProductId());
59         // 库存减一
60         updateCount(product);
61         // 生成订单
62         return createOrder(product);
63     }
64 }
65

```

终端 调试控制台 输出

```

2020-12-08 19:30:28.632 INFO 96519 --- [o-8000-exec-124] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8503, productId=1)
2020-12-08 19:30:28.645 INFO 96519 --- [o-8000-exec-187] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8504, productId=1)
2020-12-08 19:30:28.657 INFO 96519 --- [io-8000-exec-11] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8505, productId=1)
2020-12-08 19:30:28.670 INFO 96519 --- [io-8000-exec-95] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8506, productId=1)
2020-12-08 19:30:28.683 INFO 96519 --- [o-8000-exec-163] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8507, productId=1)
2020-12-08 19:30:28.697 INFO 96519 --- [o-8000-exec-144] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8508, productId=1)
2020-12-08 19:30:28.710 INFO 96519 --- [io-8000-exec-55] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8509, productId=1)
2020-12-08 19:30:28.723 INFO 96519 --- [o-8000-exec-148] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8510, productId=1)
2020-12-08 19:30:28.739 INFO 96519 --- [o-8000-exec-185] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8511, productId=1)
2020-12-08 19:30:28.753 INFO 96519 --- [o-8000-exec-130] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8512, productId=1)
2020-12-08 19:30:28.766 INFO 96519 --- [o-8000-exec-152] c.example.demo.service.OrderServiceImpl : 生成订单: Order(id=8513, productId=1)
2020-12-08 19:30:28.7794 WARN 96519 --- [io-8000-exec-12] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [o-8000-exec-129] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [io-8000-exec-39] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [o-8000-exec-49] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [io-8000-exec-47] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [o-8000-exec-46] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [io-8000-exec-176] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [o-8000-exec-200] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.794 WARN 96519 --- [io-8000-exec-29] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.797 WARN 96519 --- [o-8000-exec-133] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.800 WARN 96519 --- [io-8000-exec-46] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]
2020-12-08 19:30:28.803 WARN 96519 --- [o-8000-exec-135] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 1库存不足]

```

miaosha	order
id	product_id
name	created_at
count	updated_at
sell	deleted_at
deleted_at	

miaosha	order
id	product_id
name	created_at
count	updated_at
sell	deleted_at
deleted_at	

# 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.nextInt(1000, 1000))) {
        // log.warn("限流了, 直接返回失败");
        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);
}

public class OrderServiceImpl implements OrderService {
    @Autowired
    private ProductService productService;

    @Autowired
    private OrderMapper orderMapper;

    /**
     * 每秒放行10个请求
     */
    RateLimiter rateLimiter = RateLimiter.create(10);

    /**
     * 令牌桶限流
     */
}

```

调试控制台 输出

```

2020-06:01:744 WARN 958 --- [o-8000-exec-164] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:745 WARN 958 --- [o-8000-exec-173] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:747 WARN 958 --- [o-8000-exec-122] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:748 WARN 958 --- [io-8000-exec-17] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:748 WARN 958 --- [nio-8000-exec-5] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:748 WARN 958 --- [o-8000-exec-116] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:748 WARN 958 --- [o-8000-exec-25] .m.m.a.ExceptionHandlerExceptionResolver : Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]
2020-06:01:814 INFO 958 --- [io-8000-exec-22] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8625, productId=1}
2020-06:01:911 INFO 958 --- [io-8000-exec-23] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8626, productId=1}
2020-06:02:012 INFO 958 --- [io-8000-exec-24] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8627, productId=1}
2020-06:02:112 INFO 958 --- [io-8000-exec-25] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8628, productId=1}
2020-06:02:226 INFO 958 --- [io-8000-exec-26] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8629, productId=1}
2020-06:02:314 INFO 958 --- [io-8000-exec-27] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8630, productId=1}
2020-06:02:409 INFO 958 --- [io-8000-exec-28] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8631, productId=1}
2020-06:02:510 INFO 958 --- [io-8000-exec-29] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8632, productId=1}
2020-06:02:611 INFO 958 --- [io-8000-exec-30] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8633, productId=1}
2020-06:02:711 INFO 958 --- [io-8000-exec-31] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8634, productId=1}
2020-06:02:808 INFO 958 --- [io-8000-exec-34] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8635, productId=1}
2020-06:02:908 INFO 958 --- [io-8000-exec-32] c.example.demo.service.OrderServiceImpl : 生成订单: Order{id=8636, productId=1}

```

MySQL 8.0.22 local/miaosha/product						
	Tables	Structure	Content	Relationships	Triggers	Table Info
miaosha	Select Database					
order	1 phone 100 100 2020-12-04 03:00:07789370 2020-12-08 12:06:10.605297 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
product	2 mei 10 0 2020-12-04 03:00:07789370 2020-12-04 03:00:07789370 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
user						

MySQL 8.0.22 local/miaosha/order						
	Tables	Structure	Content	Relationships	Triggers	Table Info
miaosha	Select Database					
order	1 8626 1 2020-12-08 12:06:01.935152 2020-12-08 12:06:01.935152 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
product	8627 1 2020-12-08 12:06:02.035703 2020-12-08 12:06:02.035703 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
user	8628 1 2020-12-08 12:06:02.138425 2020-12-08 12:06:02.138425 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8629 1 2020-12-08 12:06:02.235798 2020-12-08 12:06:02.235798 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8630 1 2020-12-08 12:06:02.335061 2020-12-08 12:06:02.335061 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8631 1 2020-12-08 12:06:02.433955 2020-12-08 12:06:02.433955 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8632 1 2020-12-08 12:06:02.534286 2020-12-08 12:06:02.534286 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8633 1 2020-12-08 12:06:02.636352 2020-12-08 12:06:02.636352 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8634 1 2020-12-08 12:06:02.735983 2020-12-08 12:06:02.735983 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8635 1 2020-12-08 12:06:02.834348 2020-12-08 12:06:02.834348 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8636 1 2020-12-08 12:06:02.933258 2020-12-08 12:06:02.933258 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8637 1 2020-12-08 12:06:03.032974 2020-12-08 12:06:03.032974 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8638 1 2020-12-08 12:06:03.131707 2020-12-08 12:06:03.131707 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8639 1 2020-12-08 12:06:03.234124 2020-12-08 12:06:03.234124 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8640 1 2020-12-08 12:06:03.334495 2020-12-08 12:06:03.334495 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8641 1 2020-12-08 12:06:03.434497 2020-12-08 12:06:03.434497 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8642 1 2020-12-08 12:06:03.533416 2020-12-08 12:06:03.533416 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8643 1 2020-12-08 12:06:03.634721 2020-12-08 12:06:03.634721 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8644 1 2020-12-08 12:06:03.739887 2020-12-08 12:06:03.739887 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8645 1 2020-12-08 12:06:03.839574 2020-12-08 12:06:03.839574 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8646 1 2020-12-08 12:06:03.935426 2020-12-08 12:06:03.935426 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8647 1 2020-12-08 12:06:03.030985 2020-12-08 12:06:03.030985 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8648 1 2020-12-08 12:06:03.137203 2020-12-08 12:06:03.137203 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8649 1 2020-12-08 12:06:03.236007 2020-12-08 12:06:03.236007 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8650 1 2020-12-08 12:06:03.338154 2020-12-08 12:06:03.338154 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8651 1 2020-12-08 12:06:03.440295 2020-12-08 12:06:03.440295 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8652 1 2020-12-08 12:06:03.540594 2020-12-08 12:06:03.540594 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8653 1 2020-12-08 12:06:03.638794 2020-12-08 12:06:03.638794 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8654 1 2020-12-08 12:06:03.738892 2020-12-08 12:06:03.738892 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8655 1 2020-12-08 12:06:03.838308 2020-12-08 12:06:03.838308 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8656 1 2020-12-08 12:06:03.938204 2020-12-08 12:06:03.938204 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8657 1 2020-12-08 12:06:03.037860 2020-12-08 12:06:03.037860 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8658 1 2020-12-08 12:06:03.138423 2020-12-08 12:06:03.138423 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8659 1 2020-12-08 12:06:03.237931 2020-12-08 12:06:03.237931 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8660 1 2020-12-08 12:06:03.338229 2020-12-08 12:06:03.338229 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8661 1 2020-12-08 12:06:03.437903 2020-12-08 12:06:03.437903 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8662 1 2020-12-08 12:06:03.538130 2020-12-08 12:06:03.538130 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8663 1 2020-12-08 12:06:03.638120 2020-12-08 12:06:03.638120 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				
	8664 1 2020-12-08 12:06:03.737079 2020-12-08 12:06:03.737079 NULL	Resolved [java.lang.RuntimeException: 限流了, 直接返回失败]				

## 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

The screenshot shows a Java IDE interface with several tabs open. The main tab contains the code for `OrderServiceImpl.java`, which includes logic for handling order requests, performing checks (product, time, frequency), and creating orders. Below the code is a terminal window showing log entries from December 12, 2020, at 20:20:42. It logs various check results and a message about a user not having enough stock.

```

26     @RabbitListener(queues = "${rabbitmq.queue.orderQueue}", containerFactory = "simpleRabbitListenerContainerFactory")
27     public void receivedMessage(OrderRequest orderRequest) {
28
29         log.info("订单消费者收到消息 : " + orderRequest);
30         // 阻塞式获取令牌
31         log.info("获取令牌等待时间 : " + rateLimiter.acquire());
32         // 非阻塞式获取令牌
33         if (!rateLimiter.tryAcquire(1000 + RandomUtil.nextInt(1000, 10000),
34             TimeUnit.MILLISECONDS)) {
35             log.warn("限流了，直接返回失败");
36             throw new RuntimeException("购买失败，库存不足");
37         }
38
39         // 异步多线程处理
40         orderService.createOrder(orderRequest);
41     }
42 }

```

The screenshot shows a MySQL database client interface. A table named `OrderResult` is displayed, containing numerous rows of data. The columns visible are `String` and `TTL`. The TTL column is set to -1. The data in the String column consists of unique identifiers for each order result entry.

	String	TTL
1	OrderResult::0043e1ba756548f08a73acb4b6d4e4f4	-1
2	OrderResult::01ae786044f425cb028c4074e0ce981	
3	OrderResult::026dcbe275e6476c89b87f726c02ba33	
4	OrderResult::0275ae06599245139f5cb03750f23c6a	
5	OrderResult::034b40cf1b94fb1b9ef987a6528a7bd	
6	OrderResult::05fc8d9ae4a04ffba81efab2373b3e7d	
7	OrderResult::060728d40b7b4c5c9a62d5e5628d11c9	
8	OrderResult::077244659c2b44158b0304e405eb2532	
9	OrderResult::079d2a22e9d44509d82b7485064f608	
10	OrderResult::07b794d2dd248649ecc8a03f4973c50	
11	OrderResult::0c158a1c26a2469abda2f2f56d8200c3	
12	OrderResult::0da04d64cb24ebfa1bbbfabbb9e369d	
13	OrderResult::0ddfb812355347668308d25e8beb43d7	
14	OrderResult::0e8950b59b7247068342d73e85e804ea	
15	OrderResult::0eaddd429a84b4cd3be7225aa7b284055	
16	OrderResult::0fc3ee3002684c4faae25a6194516984	
17	OrderResult::1189d8f755a344009dca4e4af0185bbf	
18	OrderResult::11e01ce406cd45318af2c2e57d952826	
19	OrderResult::12919faad0fd401a8f4b466ccfa9a22	
20	OrderResult::1415853c320b43e39cb647fdc66fedf5	
21	OrderResult::146b61483a294d6ba2f93ce8c204afe9	
22	OrderResult::15fb1cb59c5d409d89af5fbe36ed7bd2	
23	OrderResult::1608756a0a4f4a048f5408fa8c3ab7c7	
24	OrderResult::17b963362c494b4182732ce30fff11eb	
25	OrderResult::1a4ccf584c3f4528b6f438b509cd2d2f	
26	OrderResult::1a8b7be7cae48ddae65cdfb83daa442	
27	OrderResult::1acf31fbced466dbb99f91a33f1a6ba	
28	OrderResult::1afb7eae4af24f9cafcc2254d03e63200	
29	OrderResult::1cab0e18482143f2a0b96f09ca0e8475	
30	OrderResult::1d2c83dc11764ae4aa1b1b01465d71f8	

