

## 项目截图：

数据库文件 t\_diners.sql

The screenshot shows the MySQL Workbench interface with a database connection to 'seckill-master'. A query window displays the results of the following SQL statement:

```
SELECT * FROM t_diners LIMIT 100;
```

The results show 100 rows of data from the t\_diners table, with columns including id, username, nickname, phone, email, password, avatar\_url, roles, is\_valid, create\_date, and update\_date. The data includes various test users like 'abc' and 'test' with their respective details.

### 4.1 相关实体类 秒杀代金券实体

The screenshot shows the IntelliJ IDEA code editor with the file 'SeckillVouchersMapper.java' open. The code defines a public interface for managing seckill vouchers. It includes methods for inserting new vouchers into the database and updating existing ones to decrease their stock. The code uses annotations from the 'org.apache.ibatis.annotations' package to specify the database operations.

```
public interface SeckillVouchersMapper {
    // 新增秒杀活动
    @Insert("insert into t_seckill_vouchers (fk_voucher_id, amount, start_time, end_time, is_valid, create_date, updated_at) values (?, ?, ?, ?, ?, ?, ?)")
    @Options(useGeneratedKeys = true, keyProperty = "id")
    int save(SeckillVouchers seckillVouchers);

    // 根据代金券 ID 查询该代金券是否参与抢购活动
    @Select("select id, fk_voucher_id, amount, start_time, end_time, is_valid " +
            "from t_seckill_vouchers where fk_voucher_id = #{voucherId}")
    SeckillVouchers selectVoucher(Integer voucherId);

    // 减库存
    @Update("update t_seckill_vouchers set amount = amount - 1 " +
            "where id = #{seckillId}")
    int stockDecrease(@Param("seckillId") int seckillId);
}
```

代金券订单实体

```

    package com.baomidou.mybatisplus.annotation;
    import com.baomidou.mybatisplus.annotation.IdType;
    import com.baomidou.mybatisplus.annotation.TableField;
    import com.baomidou.mybatisplus.annotation.TableId;
    import com.baomidou.mybatisplus.annotation.TableName;
    import io.swagger.annotations.ApiModel;
    import io.swagger.annotations.ApiModelProperty;
    import lombok.Data;
    import lombok.EqualsAndHashCode;
    import lombok.experimental.Accessors;

    /**
     * 代金券订单信息表
     */
    @TableId(value = "id", type = IdType.AUTO)
    @TableField("order_no")
    @ApiModelProperty("订单编号")
    private String orderNo;
    @TableField("voucher_id")
    @ApiModelProperty("代金券")
    private Integer fkVoucherId;
    @TableField("diner_id")
    @ApiModelProperty("下单用户")
    private Integer fkDinerId;
    @TableField("qrcode")
    @ApiModelProperty("生成二维码")
    private String qrcode;
    @TableField("payment")
    @ApiModelProperty("支付方式 0=微信支付 1=支付宝")
    private int payment;
    @TableField("status")
    @ApiModelProperty("订单状态 -1=已取消 0=未支付 1=已支付 2=已消费 3=已过期")
    private int status;
    @TableField("order_type")
    @ApiModelProperty("订单类型 0=正常订单 1=抢购订单")
    private int orderType;
    @TableField("ext")
    @ApiModelProperty("抢购订单的外键")
    private int fkSeckillId;

```

## 4.2 Mapper 接口

```

    package com.baomidou.mybatisplus.annotation;
    import com.baomidou.mybatisplus.core.mapper.BaseMapper;
    import com.baomidou.mybatisplus.core.metadata.IPage;
    import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
    import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
    import org.apache.ibatis.annotations.*;

    /**
     * 秒杀代金券 Mapper
     */
    public interface SeckillVouchersMapper {
        // 新增秒杀活动
        @Insert("insert into t_seckill_vouchers (fk_voucher_id, amount, start_time, end_time, is_valid, create_date, updated_at) values (?, ?, ?, ?, ?, now(), now())")
        @Options(useGeneratedKeys = true, keyProperty = "id")
        int save(SeckillVouchers seckillVouchers);

        // 根据代金券 ID 查询该代金券是否参与抢购活动
        @Select("select id, fk_voucher_id, amount, start_time, end_time, is_valid " +
                " from t_seckill_vouchers where fk_voucher_id = #{voucherId}")
        SeckillVouchers selectVoucher(Integer voucherId);

        // 减库存
        @Update("update t_seckill_vouchers set amount = amount - 1 " +
                " where id = #{seckillId}")
        int stockDecrease(@Param("seckillId") int seckillId);
    }

```

The screenshot shows the Eclipse IDE interface with the following details:

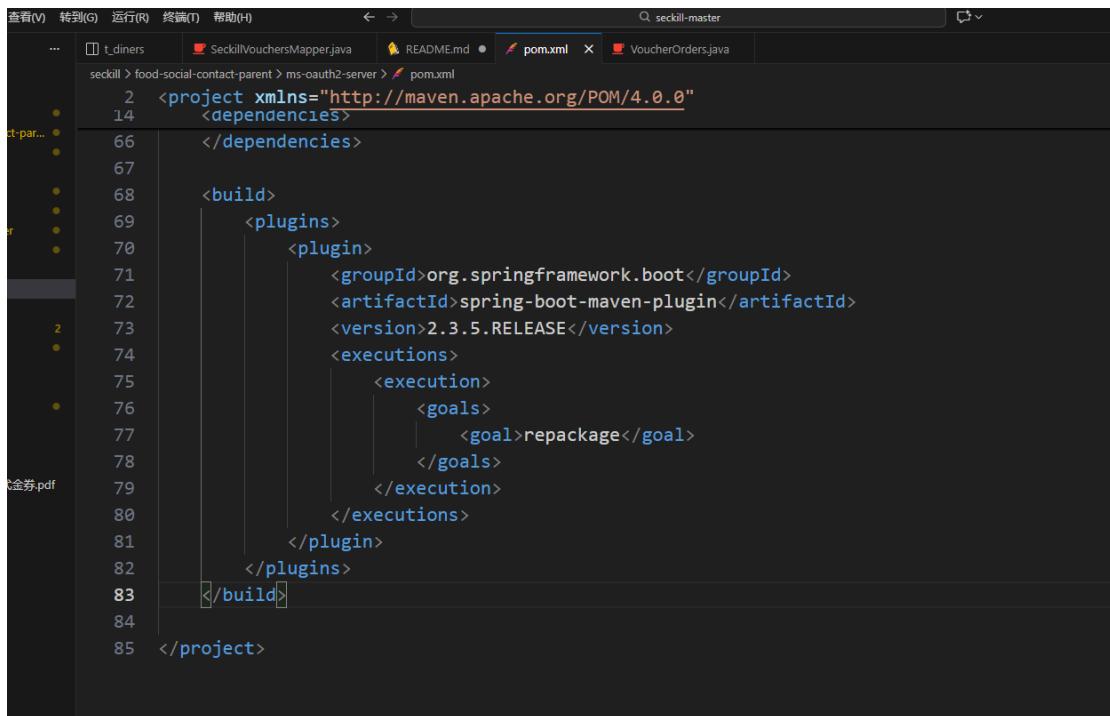
- File Path:** com/imooc/seckill/mapper/VoucherOrdersMapper.java
- Code Content:** The code defines a public interface `VoucherOrdersMapper` with methods for selecting and inserting voucher orders based on various parameters like `dinerId`, `voucherId`, and `orderType`.
- Annotations:** The code uses annotations from `org.apache.ibatis.annotations` for `Insert`, `Select`, and `Param`.
- Imports:** Imports include `com.imooc.model.VoucherOrders`, `org.apache.ibatis.annotations.Insert`, `org.apache.ibatis.annotations.Param`, and `org.apache.ibatis.annotations.Select`.
- IDE Features:** The code editor shows several Windusrefactoring annotations (e.g., `Windsurf: Refactor | Explain`) and code completion suggestions.

Redis 实验

OAuth2 服务用于用户认证和授权

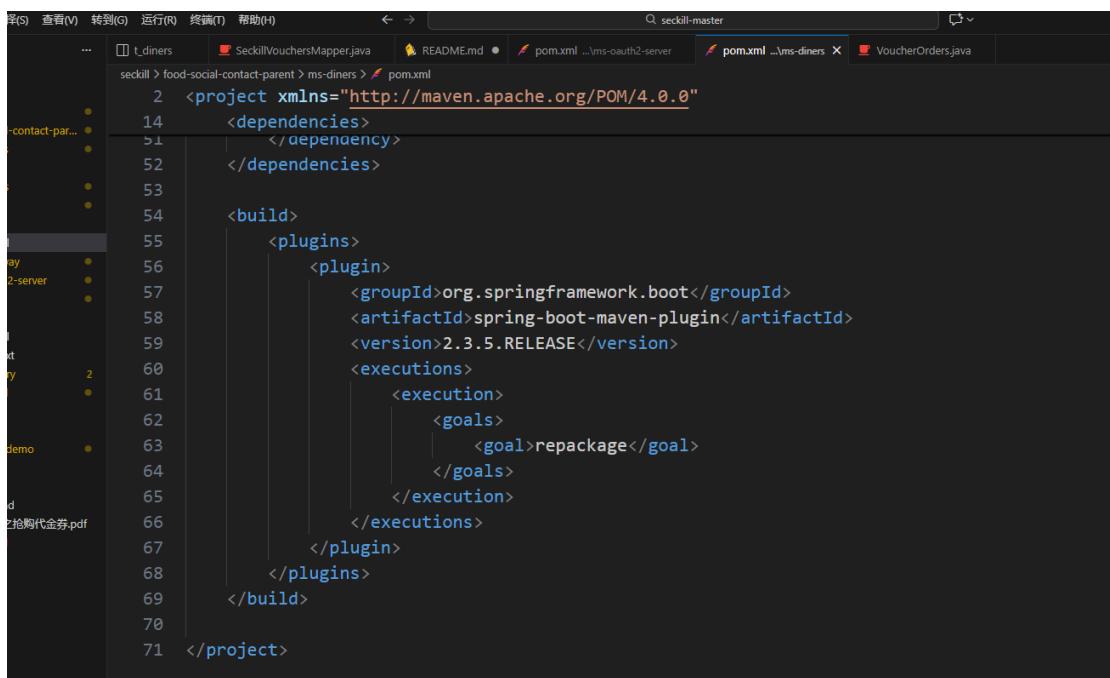
抢购接口需要 access\_token，由此服务提供

OAuth2 服务的 pom.xml 缺少 Spring Boot Maven 插件配置，导致无法打包成可执行 jar



```
查看(V) 转到(G) 运行(R) 终端(T) 帮助(H) ← → Q seckill-master
... t_diners SeckillVouchersMapper.java README.md pom.xml VoucherOrders.java
seckill > food-social-contact-parent > ms-oauth2-server > pom.xml
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
14   <dependencies>
...
66     </dependencies>
67
68   <build>
69     <plugins>
70       <plugin>
71         <groupId>org.springframework.boot</groupId>
72         <artifactId>spring-boot-maven-plugin</artifactId>
73         <version>2.3.5.RELEASE</version>
74         <executions>
75           <execution>
76             <goals>
77               <goal>repackage</goal>
78             </goals>
79           </execution>
80         </executions>
81       </plugin>
82     </plugins>
83   </build>
84
85 </project>
```

为 ms-diners 服务添加 Spring Boot Maven 插件配置、



```
查看(V) 转到(G) 运行(R) 终端(T) 帮助(H) ← → Q seckill-master
... t_diners SeckillVouchersMapper.java README.md ...\\ms-oauth2-server pom.xml ...\\ms-diners VoucherOrders.java
seckill > food-social-contact-parent > ms-diners > pom.xml
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
14   <dependencies>
...
51     </dependencies>
52
53
54   <build>
55     <plugins>
56       <plugin>
57         <groupId>org.springframework.boot</groupId>
58         <artifactId>spring-boot-maven-plugin</artifactId>
59         <version>2.3.5.RELEASE</version>
60         <executions>
61           <execution>
62             <goals>
63               <goal>repackage</goal>
64             </goals>
65           </execution>
66         </executions>
67       </plugin>
68     </plugins>
69   </build>
70
71 </project>
```

## 5.1 启动服务

Eureka 注册中心

**spring Eureka**

HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2025-12-12T14:36:31 +0800
Data center	default	Uptime	21:08
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	12

### DS Replicas

localhost

#### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MS-DINERS	n/a (1)	{1}	UP (1) - 192.168.203.1:8081
MS-OAUTH2-SERVER	n/a (1)	{1}	UP (1) - 192.168.203.1:8082
MS-SECKILL	n/a (1)	{1}	UP (1) - 192.168.203.1:8083

### General Info

Name	Value
total-avail-memory	366mb
environment	test
num-dc-cpus	20
current-memory-usage	70mb (19%)
server-upptime	21:08
registered-replicas	http://localhost:8083/eureka/

Postman 测试：

The screenshot shows a Postman workspace titled "Fan'an's Workspace". A POST request is being made to `http://localhost:8083/add`. The request body is a JSON object:

```
{
  "EnvoucherId": 2,
  "amount": 100,
  "startTime": "2025-12-12 14:00:00",
  "endTime": "2025-12-12 23:59:59"
}
```

The response status is **200 OK**, with a response time of 29 ms and a total size of 234 B. The response body is:

```
{
  "code": 1,
  "message": "Successful.",
  "path": "/api/v1/envoucher",
  "data": "438607"
}
```

多次输入会提示

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8083/add`. The request body is a JSON object:

```
1  {
2     "id": 1,
3     "amount": 100,
4     "startTime": "2025-12-12 18:00:00",
5     "endTime": "2025-12-12 23:00:00"
6 }
```

The response status is `200 OK`, with a response time of 635 ms and a size of 240 B. The response body is:

```
{ "code": 0, "message": "任务已成功执行", "path": "/add", "data": null }
```

## JMeter

The screenshot shows the Apache JMeter 5.6.3 interface. The 'Test Plan' tab is selected. The 'User-defined variables' section contains a table:

名称:	值

Below the table are several checkboxes:

- 独立运行每个线程组 (例如在一个线程运行结束后启动下一个)
- 主线程结束后运行 tearDown 线程组
- 重放测试模式

A note states: 只有当你需要记录每个请求从服务器取得的数据到文件时才需要选择此的测试模式。选择这个选项很影响性能。

At the bottom, there are buttons: 添加 (Add), 从剪贴板添加 (Paste from clipboard), 移除 (Remove), 向上 (Up), 向下 (Down).

导入数据:

id	type	data	display_order
1	TableType	Regular 大厅 0	
2	TableType	Bar 吧台 0	
3	TableType	Window 靠窗 0	
4	TableType	Outdoor 户外 0	
5	TableType	Private 包间 0	
7	RestaurantTag	24 hours[24小时] 0	
8	RestaurantTag	Afternoon tea 下午茶 0	
9	RestaurantTag	All you can eat 不限量 0	
10	RestaurantTag	Bistro 酒馆 0	
11	RestaurantTag	Breakfast 早餐 0	
12	RestaurantTag	Bund view 外滩 0	
13	RestaurantTag	Classic Shanga 0	
14	RestaurantTag	Cocktails 鸡尾酒 0	
15	RestaurantTag	Credit cards acc 0	
16	RestaurantTag	Delivery 可送外 0	
17	RestaurantTag	Pet friendly 宠物 0	
18	RestaurantTag	Kids friendly 适 0	
19	RestaurantTag	Find dining 顶级 0	
20	RestaurantTag	Free parking 免 0	
21	RestaurantTag	Lounge 酒廊 0	
22	RestaurantTag	Lunch set 午市套餐 0	
23	RestaurantTag	Group dining 团 0	
24	RestaurantTag	Healthy 健康 0	
25	RestaurantTag	Historic building 0	

## 配置 Authorization

POST http://localhost:8082/oauth/token

Authorization: Basic Auth

Username: ms-diners

Password: [REDACTED]

## 配置 body

Key	Value	Description
grant_type	password	授权类型
username	test	用户名 (数据库中的用户)
password	123456	密码
scope	api	授权范围

报错了，不用配置，把 redis 的密码删了就好了

## JMetrer 测试运行结果：

The screenshot shows two JMeter test results windows side-by-side.

**Left Window: View Results Tree**

Name: 调查结果树  
Comments:  
Write results to file / Read from file  
Filename:  
Search: Case sensitive Regular exp Search Reset  
Text Sampler result Request Response data

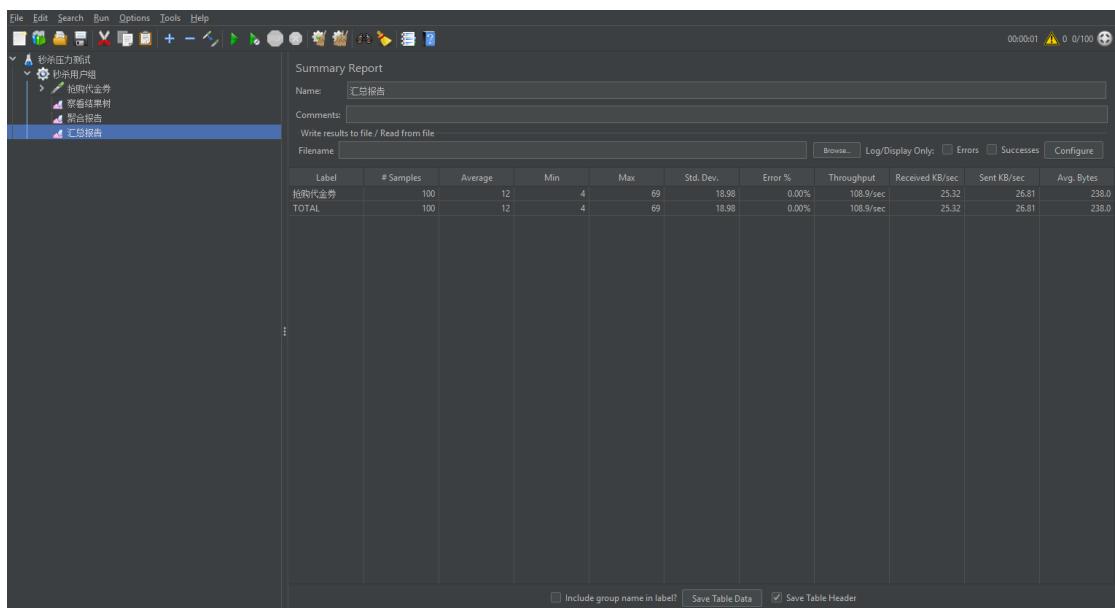
Sampler result details for "抢购代金券":  
Thread Name:秒杀用户组 1-10  
Sample Start:2023-12-12 16:26:53 CST  
Load time:55  
Connect Time:18  
Latency:44  
Size in bytes:238  
Sent bytes:232  
Headers size in bytes:162  
Body size in bytes:76  
SampleCount:1  
Error Count:0  
Data type ("text"/"bin"):text  
Response code:200  
Response message:  
HTTPSampleResult fields:  
ContentType:application/json  
DataEncoding:null

**Right Window: Aggregate Report**

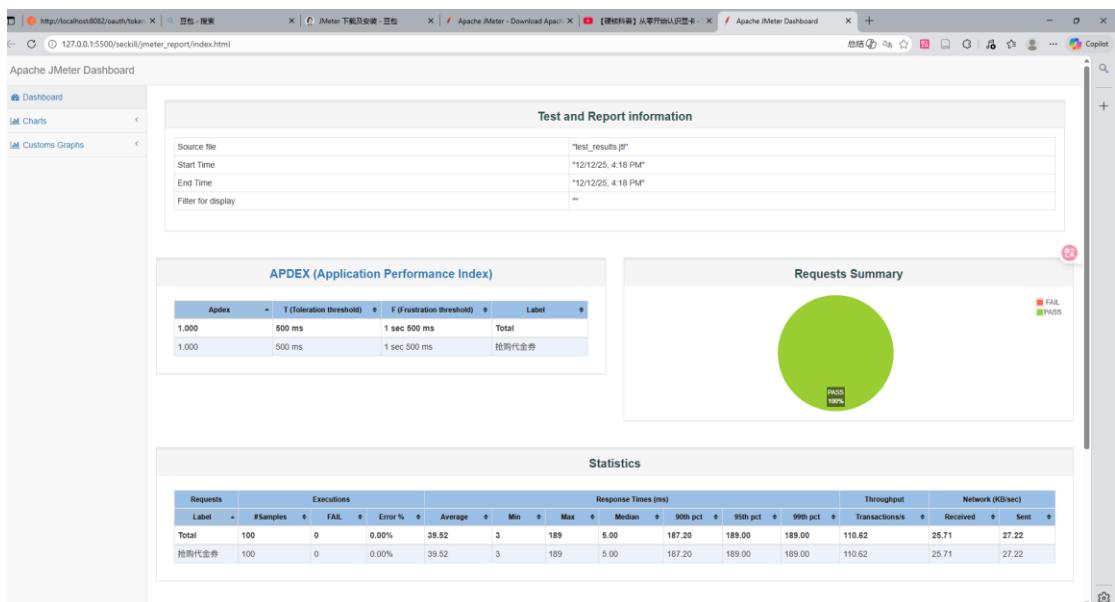
Name: 聚合报告  
Comments:  
Write results to file / Read from file  
Filename:  
Label # Samples Average Median 90% Line 95% Line 99% Line Min Maximum Error % Throughput Received KB/sec Sent KB/sec

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
抢购代金券	100	12	6	44	69	69	4	69	0.00%	108.9/sec	25.32	26.81
TOTAL	100	12	6	44	69	69	4	69	0.00%	108.9/sec	25.32	26.81

Include group name in label? Save Table Data Save Table Header



图形化页面：



6 Redis 防止超卖 (Lua 脚本原子性扣减库存)

```

1 if (redis.call('hexists', KEYS[1], KEYS[2]) == 1) then
2     local stock = tonumber(redis.call('hget', KEYS[1], KEYS[2]));
3     if (stock > 0) then
4         redis.call('hincrby', KEYS[1], KEYS[2], -1);
5         return stock;
6     end;
7     return 0;
8 end;

```

```

35 public class SeckillService {
62     public ResultInfo doSeckill(Integer voucherId, String accessToken, String path) {
137         // 采用 Redis
138         // 扣库存
139         // count = redisTemplate.opsForHash().increment(key, "amount", -1);
140         // AssertUtil.isTrue(count < 0, "该券已经卖完了");
141
142         // 采用 Redis + Lua 解决问题
143         // 扣库存
144         List<String> keys = new ArrayList<>();
145         keys.add(key);
146         keys.add("amount");
147         Long amount = (Long) redisTemplate.execute(defaultRedisScript, keys);
148         AssertUtil.isTrue(amount == null || amount < 1, ...message: "该券已经卖完了");
149     }
150 } catch (Exception e) {
151     // 手动回滚事务
152     TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();
153     // 自定义 Redis 解锁
154     // redisLock.unlock(lockName, lockKey);
155
156     // Redisson 解锁
157     lock.unlock();
158     if (e instanceof ParameterException) {

```

## Redisson 分布式锁实现

```

... README.md stock.lua SeckillService.java 9+ index.html JMeter测试报告.md pom.xml ...\\ms-oauth2-server pom.xml ...\\ms-diners t_dictionary

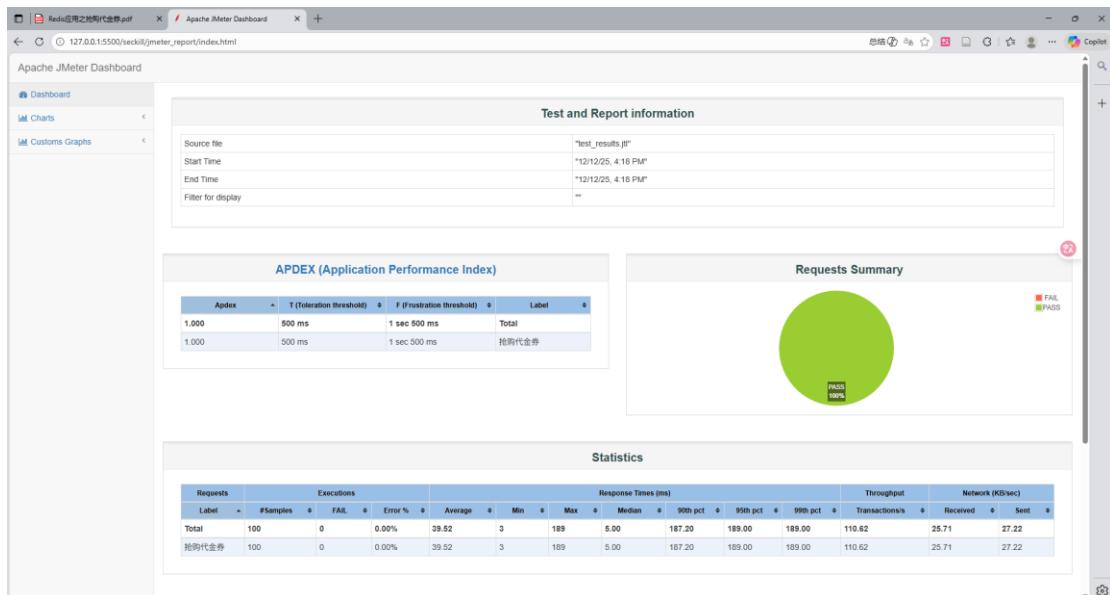
seckill > food-social-contact-parent > ms-seckill > src > main > java > com > imooc > seckill > service > SeckillService.java > {} com.imooc.seckill.service

 35 public class SeckillService {
 62     public ResultInfo doSeckill(Integer voucherId, String accessToken, String path) {
105         // 使用 Redis 锁一个账号只能购买一次
106         String lockName = RedisKeyConstant.lock_key.getKey()
107             + dinerInfo.getId() + ":" + voucherId;
108         long expireTime = seckillVouchers.getEndTime().getTime() - now.getTime();
109
110         // 自定义 Redis 分布式锁
111         //String lockKey = redisLock.tryLock(lockName, expireTime);
112
113         // Redisson 分布式锁
114         RLock lock = redissonClient.getLock(lockName);
115
116         try {
117             // 不为空意味着拿到锁了，执行下单
118             // 自定义 Redis 分布式锁处理
119             // if (StrUtil.isNotBlank(lockKey)) {
120
121                 // Redisson 分布式锁处理
122                 boolean isLocked = lock.tryLock(expireTime, TimeUnit.MILLISECONDS);
123                 if (isLocked) {
124                     // 下单
125                     VoucherOrders voucherOrders = new VoucherOrders();
126                     voucherOrders.setFkDinerId(dinerInfo.getId());
127                     // Redis 中不需要维护外键信息
128                     // voucherOrders.setFkSeckillId(seckillVouchers.getId());
129                     voucherOrders.setFkVoucherId(seckillVouchers.getFkVoucherId());
130                     String orderNo = IdUtil.getSnowflake(1, 1).nextIdStr();
131                     voucherOrders.setOrderNo(orderNo);
132                     voucherOrders.setOrderType(orderType: 1);

```

## 总结：

说明：当前忽略了启动 redis 的步骤，需要自行启动 VM 和 CRT  
总测试结果和聚合报告



```

1 [ {
2   "Total" : {
3     "transaction" : "Total",
4     "sampleCount" : 100,
5     "errorCount" : 0,
6     "errorPct" : 0.0,
7     "meanResTime" : 39.52000000000002,
8     "medianResTime" : 5.0,
9     "minResTime" : 3.0,
10    "maxResTime" : 189.0,
11    "pct1ResTime" : 187.20000000000005,
12    "pct2ResTime" : 189.0,
13    "pct3ResTime" : 189.0,
14    "throughput" : 110.61946902654867,
15    "receivedKBytesPerSec" : 25.710384402654867,
16    "sentKBytesPerSec" : 27.222759955752213
17  },
18  "抢购代金券" : {
19    "transaction" : "抢购代金券",
20    "sampleCount" : 100,
21    "errorCount" : 0,
22    "errorPct" : 0.0,
23    "meanResTime" : 39.52000000000002,
24    "medianResTime" : 5.0,
25    "minResTime" : 3.0,
26    "maxResTime" : 189.0,
27    "pct1ResTime" : 187.20000000000005,
28    "pct2ResTime" : 189.0,
29    "pct3ResTime" : 189.0,
30    "throughput" : 110.61946902654867,
31    "receivedKBytesPerSec" : 25.710384402654867,
32    "sentKBytesPerSec" : 27.222759955752213
33  }
34 }

```

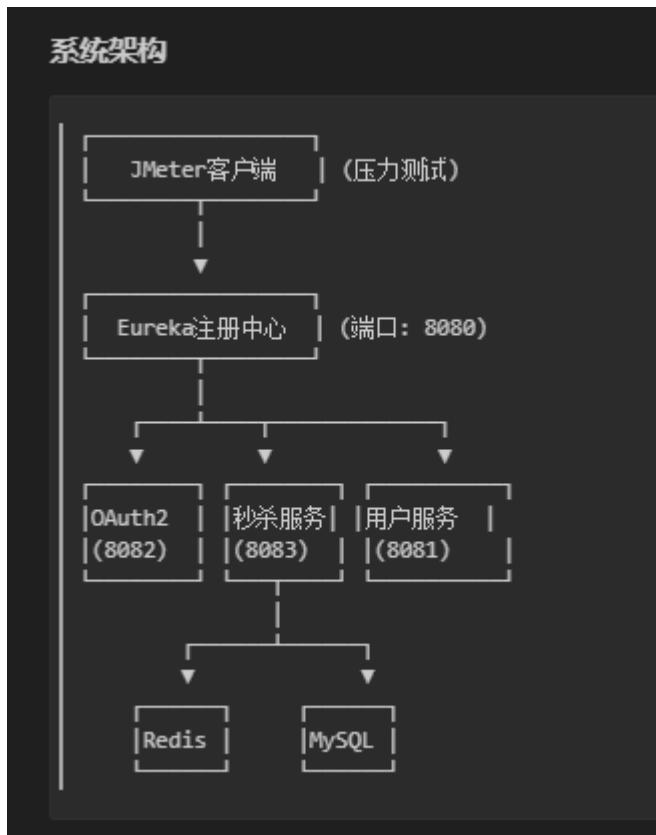
## 附录：

一些数据库表汇总：

The screenshot shows the MySQL Workbench interface with the database 'seckill' selected. The left sidebar shows the database structure with tables like t\_dictionary, t\_diners, t\_feed, etc. The main area displays the data for the 't\_dictionary' table.

	<b>id</b>	<b>type</b>	<b>data</b>	<b>display_order</b>
1	1	TableType	Regular 大厅	0
2	2	TableType	Bar 吧台	0
3	3	TableType	Window 靠窗	0
4	4	TableType	Outdoor 户外	0
5	5	TableType	Private 包间	0
7	7	RestaurantTag	24 hours 24小时	0
8	8	RestaurantTag	Afternoon tea 下午茶	0
9	9	RestaurantTag	All you can eat i	0
10	10	RestaurantTag	Bistros 酒馆	0
11	11	RestaurantTag	Breakfast 早餐	0
12	12	RestaurantTag	Bund view 外滩	0
13	13	RestaurantTag	Classic Shangha	0
14	14	RestaurantTag	Cocktails 鸡尾酒	0
15	15	RestaurantTag	Credit cards acc	0
16	16	RestaurantTag	Delivery 可送外	0
17	17	RestaurantTag	Pet friendly 宠物	0
18	18	RestaurantTag	Kids friendly   适	0
19	19	RestaurantTag	Fine dining 顶级	0
20	20	RestaurantTag	Free parking 免	0
21	21	RestaurantTag	Lounge 酒廊	0
22	22	RestaurantTag	Lunch set 午市套餐	0
23	23	RestaurantTag	Group dining 团	0
24	24	RestaurantTag	Healthy 健康	0
25	25	RestaurantTag	Historic building	0
26	26	RestaurantTag	Hotel restaurant	0
27	27	RestaurantTag	Ice cream 冰激凌	0
28	28	RestaurantTag	Late night dining 夜宵	0
29	29	RestaurantTag	Non-smoking 禁烟	0
30	30	RestaurantTag	Notable wine 美酒	0
32	32	RestaurantTag	Outdoor seating 户外	0

Readme 未在打包文件中展示，固本项目相关介绍和指令在此一并补充：



## 1.数据库准备

```
# 创建数据库
mysql -h localhost -u root -p123456 -e "CREATE DATABASE IF NOT EXISTS db_imooc
DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;"
```

## 2. Redis 服务准备

```
# 检查 Redis 是否运行
redis-cli -h 192.168.153.135 -p 6379 ping
# 预期输出: PONG
# 如果 Redis 未启动, 启动 Redis 服务
redis-server /path/to/redis.conf
```

## 3. 项目编译

设置好环境变量 (Java11, 报错方面修改了很多, 原项目对 11 和 17 都不兼容, 需修改, 比如 Spring Boot Maven 插件配置、还有 Redis 密码删掉等等, 在此不一一列举)

编译秒杀服务:

```
cd D:\你的路径\seckill-master\seckill\food-social-contact-parent\ms-seckill
mvn clean package -DskipTests
```

编译 Eureka 注册中心 (ms-registry)

```
cd D:\你的路径\seckill-master\seckill\food-social-contact-parent\ms-registry
mvn clean package -DskipTests
```

编译 OAuth2

```
cd D:\你的路径\seckill-master\seckill\food-social-contact-parent\ms-oauth2-server  
mvn clean package -DskipTests
```

编译用户服务

```
cd D:\你的路径\seckill-master\seckill\food-social-contact-parent\ms-diners  
mvn clean package -DskipTests
```

## 相关验证测试

测试 1: 添加秒杀活动

工具: Postman 或 curl

接口: POST http://localhost:8083/add

请求头:

Content-Type: application/json

请求体:

```
{  
    "fkVoucherId": 1,  
    "amount": 100,  
    "startTime": "2025-12-11 17:00:00",  
    "endTime": "2025-12-11 23:59:59"  
}
```

使用 curl 测试 (Windows CMD) :

```
curl -X POST http://localhost:8083/add -H "Content-Type: application/json" -d  
>{"fkVoucherId": 1, "amount": 100, "startTime": "2025-12-11 17:00:00",  
"endTime": "2025-12-11 23:59:59"}
```

预期响应:

```
{  
    "code": 1,  
    "message": "Successful.",  
    "path": "/add",  
    "data": "添加成功"  
}
```

---

测试 2: 验证 Redis 数据

运行位置: Linux 虚拟机 (192.168.153.135)

# 查看秒杀券数据

```
redis-cli -h 192.168.153.135 -p 6379 HGETALL "seckill_vouchers:1"
```

预期输出:

- 1) "fkVoucherId"
- 2) "1"
- 3) "amount"
- 4) "100"
- 5) "startTime"
- 6) "1765443600000"

```
7) "endTime"
8) "1765468740000"
9) "id"
10) ""
11) "createDate"
12) "1765447356239"
13) "updateDate"
14) "1765447356239"
15) "isValid"
16) "1"
```

说明:

- 数据成功存储在 Redis 的 Hash 结构中
- key 格式: seckill\_vouchers:{voucherId}
- 时间戳格式为毫秒级 Unix 时间戳

---

测试 3: 抢购代金券 (需要 OAuth2 服务)

接口: POST http://localhost:8083/{voucherId}?access\_token={token}

步骤:

1. 先获取 access\_token (需要 OAuth2 服务)
2. 调用抢购接口

使用 Postman 测试:

POST http://localhost:8083/1?access\_token=your\_access\_token\_here

预期响应:

```
{  
  "code": 1,  
  "message": "Successful.",  
  "path": "/1",  
  "data": "抢购成功"  
}
```

## 7.1 Redis 缓存秒杀券信息 测试

步骤 1: 添加秒杀活动

在 Windows 命令提示符 (CMD) 中执行:

```
curl -X POST http://localhost:8083/add -H "Content-Type: application/json" -d  
{"fkVoucherId": 1, "amount": 100, "startTime": "2025-12-12 00:00:00",  
"endTime": "2025-12-12 23:59:59"}
```

预期响应:

```
{  
  "code": 1,  
  "message": "Successful.",  
  "path": "/add",  
  "data": "添加成功"  
}
```

步骤 2: 在 Redis 中查看缓存数据

在 Linux 虚拟机上执行:

```
# 查看秒杀券的完整信息  
redis-cli -h 192.168.153.135 -p 6379 HGETALL "seckill_vouchers:1"
```

预期输出:

```
1) "fkVoucherId"  
2) "1"  
3) "amount"  
4) "100"  
5) "startTime"  
6) "1734019200000"  
7) "endTime"  
8) "1734105599000"  
9) "isValid"  
10) "1"  
11) "createDate"  
12) "1734019356239"  
13) "updateDate"  
14) "1734019356239"
```

## 7.2 Redis 防止超卖 (Lua 脚本) 测试

步骤 1: 准备测试环境

确保已添加秒杀活动 (库存 100)，并获取 access\_token。

步骤 2: 使用 JMeter 进行并发测试

配置 JMeter:

- 线程数: 100 (模拟 100 个用户同时抢购)
- Ramp-Up 时间: 1 秒
- 循环次数: 1 次

执行测试后，查看 Redis 中的库存:

```
# 查看剩余库存  
redis-cli -h 192.168.153.135 -p 6379 HGET "seckill_vouchers:1" "amount"
```

## 7.3 Redis 限制一人一单 (分布式锁) 测试

步骤 1: 获取 access\_token

```
curl -X POST "http://localhost:8082/oauth/token" ^  
-H "Content-Type: application/x-www-form-urlencoded" ^  
-u "appId:123456" ^  
-d "grant_type=password&username=test&password=123456&scope=api"
```

记录返回的 accessToken。

步骤 2: 第一次抢购 (应该成功)

```
curl -X POST "http://localhost:8083/1?access_token=your_access_token_here"
```

预期响应:

```
{  
  "code": 1,  
  "message": "Successful.",
```

```
        "path": "/1",
        "data": "抢购成功"
    }
```

步骤 3: 第二次抢购 (应该失败)

使用相同的 access\_token 再次请求:

```
curl -X POST "http://localhost:8083/1?access_token=your_access_token_here"
```

预期响应:

```
{
    "code": 0,
    "message": "该用户已抢到该代金券，无需再抢",
    "path": "/1",
    "data": null
}
```

步骤 4: 查看 Redis 中的分布式锁

在 Linux 虚拟机上执行:

```
# 查看分布式锁的 key (抢购过程中可能存在)
```

```
redis-cli -h 192.168.153.135 -p 6379 KEYS "lockby:*
```

## JMeter 压力测试

步骤 1: 获取 OAuth2 访问令牌

运行位置: Windows 命令提示符 (CMD)

```
# 获取 access_token
```

```
curl -X POST "http://localhost:8082/oauth/token" ^
-H "Content-Type: application/x-www-form-urlencoded" ^
-u "appId:123456" ^
-d "grant_type=password&username=test&password=123456&scope=api"
```

步骤 2: 添加秒杀活动

```
# 添加秒杀活动 (库存 100)
```

```
curl -X POST http://localhost:8083/add ^
-H "Content-Type: application/json" ^
-d "{\"fkVoucherId\": 1, \"amount\": 100, \"startTime\": \"2025-12-12 00:00:00\",
\"endTime\": \"2025-12-12 23:59:59\"}"
```

非 GUI 执行压力测试

```
# 执行 JMeter 测试 (非 GUI 模式)
```

```
E:\apache-jmeter-5.6.3\bin\jmeter.bat -n -t seckill_test.jmx -l test_results.jtl -e -o
jmeter_report
```

也可使用图形化页面, 找到相应的 bat 文件打开, 再打开我们的工作目录 (*seckill\_test.jmx*) 即可

默认测试配置:

并发用户: 100

Ramp-Up 时间: 1s

循环次数: 1 次

总请求数: 100 个

注意事项:

1. 每次测试前清除结果 (GUI 模式)
  - 点击 运行 → 清除全部
  - 避免数据混淆
2. 重置秒杀活动 (如需重复测试)
3. ~~# 删除旧的秒杀活动~~
4. redis-cli -h 192.168.153.135 -p 6379 DEL "seckill\_vouchers:1"
- 5.
6. ~~# 重新添加秒杀活动~~
7. curl -X POST http://localhost:8083/add -H "Content-Type: application/json" -d "{\"fkVoucherId\": 1, \"amount\": 100, \"startTime\": \"2025-12-12 00:00:00\", \"endTime\": \"2025-12-12 23:59:59\"}"
8. 测试规模建议
  - GUI 模式: 适合 < 500 个用户
  - 命令行模式: 适合 > 500 个用户的大规模测试
9. 观察服务器状态
  - 测试时注意观察后台服务日志
  - 如果出现大量错误, 检查服务是否正常运行
10. access\_token 过期
  - 如果测试时出现 401 错误, 重新获取 access\_token
  - 默认过期时间约 30 天