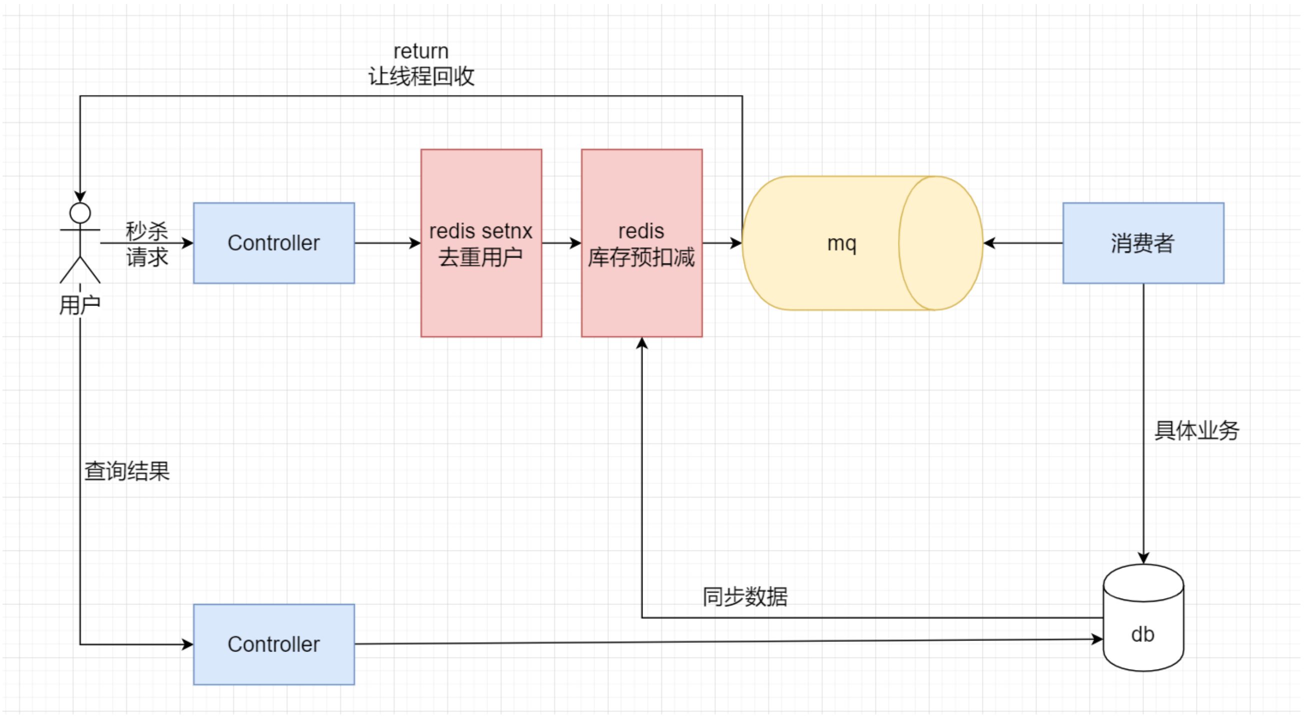
【秒杀】

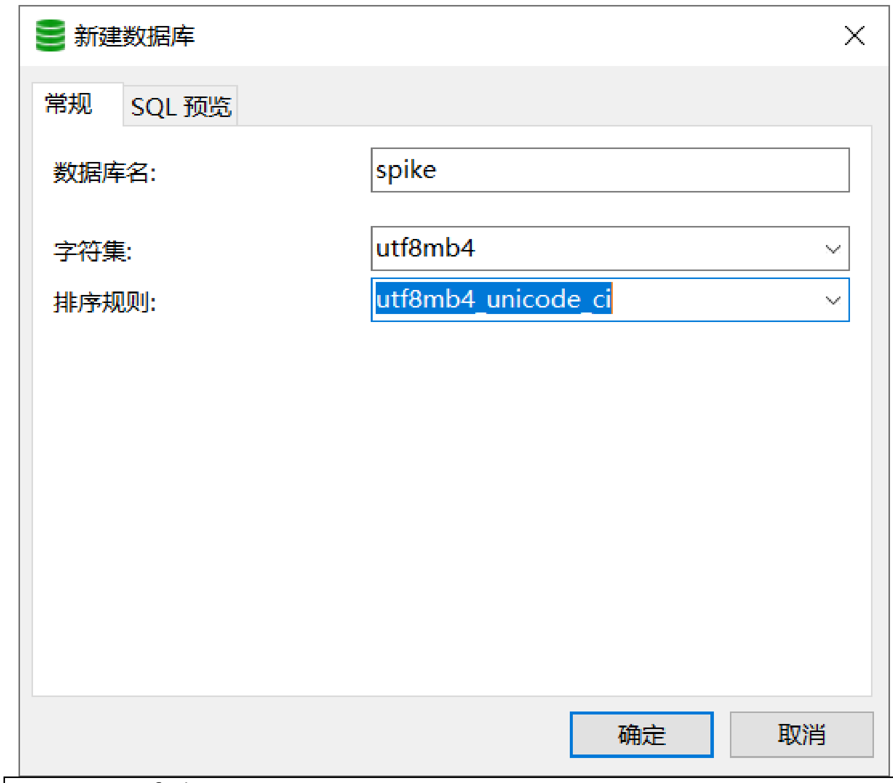
1.1. 技术选择型

- Springboot +接收请求并操作 redis 和 mysql
- Redis　用于缓存+分布式锁
- Rocketmq　用于解耦　削峰，异步
- Mysql　用于存放真实的商品信息
- Mybatis　用于操作数据库的 orm 框架

1.2. 架构图



1.3. 准备工作-数据库



数据库名: spike

字符集: utf8mb4

排序规则: utf8mb4_unicode_ci

```
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for goods
-- ----------------------------
DROP TABLE IF EXISTS `goods`;
CREATE TABLE `goods`  (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `goods_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL DEFAULT NULL,
  `price` decimal(10, 2) NULL DEFAULT NULL,
  `stocks` int(255) NULL DEFAULT NULL,
  `status` int(255) NULL DEFAULT NULL,
  `pic` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL DEFAULT NULL,
  `create_time` datetime(0) NULL DEFAULT NULL,
  `update_time` datetime(0) NULL DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of goods
-- ----------------------------
INSERT INTO `goods` VALUES (1, '小米12s', 4999.00, 1000, 2, 'xxxxxx', '2023-02-23 11:35:56', '2023-02-23 16:53:34');
INSERT INTO `goods` VALUES (2, '华为mate50', 6999.00, 10, 2, 'xxxx', '2023-02-23 11:35:56', '2023-02-23 11:35:56');
INSERT INTO `goods` VALUES (3, '锤子pro2', 1999.00, 100, 1, NULL, '2023-02-23 11:35:56', '2023-02-23 11:35:56');

-- ----------------------------
-- Table structure for order_records
-- ----------------------------
DROP TABLE IF EXISTS `order_records`;
CREATE TABLE `order_records`  (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NULL DEFAULT NULL,
  `order_sn` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL DEFAULT NULL,
  `goods_id` int(11) NULL DEFAULT NULL,
  `create_time` datetime(0) NULL DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci ROW_FORMAT = Dynamic;

SET FOREIGN_KEY_CHECKS = 1;
```

1.4. 创建项目选择依赖 spike-web（接受用户秒杀请求）

1.4.1. Pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.13</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.powernode</groupId>
    <artifactId>spike-web</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spike-web</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- rocketmq 的依赖 -->
        <dependency>
            <groupId>org.apache.rocketmq</groupId>
            <artifactId>rocketmq-spring-boot-starter</artifactId>
            <version>2.2.1</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba.fastjson2</groupId>
            <artifactId>fastjson2</artifactId>
            <version>2.0.14</version>
        </dependency>

        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

1.4.2. 修改配置文件

```
server:
  port: 7001
spring:
  application:
    name: spike-web
  redis:
    host: 127.0.0.1
    port: 6379
    database: 0
    lettuce:
      pool:
        enabled: true
        max-active: 100
        max-idle: 20
        min-idle: 5
rocketmq:
  name-server: 192.168.188.129:9876      # rocketMq 的 nameServer 地址
  producer:
    group: powernode-group         # 生产者组别
    send-message-timeout: 3000  # 消息发送的超时时间
    retry-times-when-send-async-failed: 2  # 异步消息发送失败重试次数
    max-message-size: 4194304        # 消息的最大长度
```

1.4.3. 创建 SpikeController

```java
package com.powernode.controller;

import com.alibaba.fastjson.JSON;
import org.apache.rocketmq.client.producer.SendCallback;
import org.apache.rocketmq.client.producer.SendResult;
import org.apache.rocketmq.spring.core.RocketMQTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;
import java.util.concurrent.atomic.AtomicInteger;

@RestController
public class SeckillController {


    @Autowired
    private StringRedisTemplate redisTemplate;

    @Autowired
    private RocketMQTemplate rocketMQTemplate;

    /**
     * 压测时自动是生成用户 id
     */
    AtomicInteger ai = new AtomicInteger(0);

    /**
     * 1.一个用户针对一种商品只能抢购一次
     * 2.做库存的预扣减  拦截掉大量无效请求
     * 3.放入 mq 异步化处理订单
     *
     * @return
     */
    @GetMapping("doSeckill")
    public String doSeckill(Integer goodsId /*, Integer userId*/) {
        int userId = ai.incrementAndGet();
        // unique key 唯一标记 去重
        String uk = userId + "-" + goodsId;
        // set nx  set if not exist
         Boolean flag = redisTemplate.opsForValue().setIfAbsent("seckillUk:" + uk, "");
        if (!flag) {
            return "您以及参与过该商品的抢购，请参与其他商品抢购!";
        }
        // 假设库存已经同步了  key:goods_stock:1  val:10
         Long count = redisTemplate.opsForValue().decrement("goods_stock:" + goodsId);
        // getkey  java  setkey    先查再写 再更新 有并发安全问题
        if (count < 0) {
            return "该商品已经被抢完，请下次早点来哦O(∩_∩)O";
        }
        // 放入 mq
         HashMap<String, Integer> map = new HashMap<>(4);
        map.put("goodsId", goodsId);
        map.put("userId", userId);
        rocketMQTemplate.asyncSend("seckillTopic3", JSON.toJSONString(map), new SendCallback() {
            @Override
            public void onSuccess(SendResult sendResult) {
                System.out.println("发送成功" + sendResult.getSendStatus());
            }

            @Override
            public void onException(Throwable throwable) {
                System.err.println("发送失败" + throwable);
            }
        });
        return "拼命抢购中,请稍后去订单中心查看";
    }

}
```

1.5. 创建项目选择依赖 spike-service（处理秒杀）

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.13</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.powernode</groupId>
    <artifactId>spike-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spike-service</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>2.3.0</version>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid-spring-boot-starter</artifactId>
            <version>1.2.6</version>
        </dependency>
        <!-- rocketmq 的依赖 -->
        <dependency>
            <groupId>org.apache.rocketmq</groupId>
            <artifactId>rocketmq-spring-boot-starter</artifactId>
            <version>2.2.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
        </dependency>
        <dependency>
            <groupId>com.alibaba.fastjson2</groupId>
            <artifactId>fastjson2</artifactId>
            <version>2.0.14</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

1.5.1. 修改 yml 文件

```
server:
  port: 7002
spring:
  application:
    name: spike-service
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://127.0.0.1:3306/spike?useUnicode=true&characterEncoding=UTF-8&serverTimezone
=UTC
    username: root
    password: 123456
    type: com.alibaba.druid.pool.DruidDataSource
  redis:
    host: 127.0.0.1
    port: 6379
    database: 0
    lettuce:
      pool:
        enabled: true
        max-active: 100
        max-idle: 20
        min-idle: 5
mybatis:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
  mapper-locations: classpath*:mapper/*.xml
rocketmq:
  name-server: 192.168.188.129:9876
```

1.5.2. 逆向生成实体类等

1.5.3. 修改启动类

```
@SpringBootApplication
@MapperScan(basePackages = {"com.powernode.mapper"})
public class SpikeServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpikeServiceApplication.class, args);
    }
}
```

1.5.4. 修改 GoodsMapper

```
List<Goods> selectSeckillGoods();
```

1.5.5. 修改 GoodsMapper.xml

```
<!--  查询数据库中需要参于秒杀的商品数据 status = 2 -->
<select id="selectSeckillGoods" resultMap="BaseResultMap">
  select `id`,`stocks` from goods where `status` = 2
  </select>
```

1.5.6. 同步 mysql 数据到 redis

1.1.1.1 方法 1

```
package com.powernode.config;


import com.powernode.domain.Goods;
import com.powernode.mapper.GoodsMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import org.springframework.util.CollectionUtils;


import javax.annotation.PostConstruct;
import java.util.List;


/**
 * 将 mysql 的参与抢购的商品的数据
 * 同步到 redis 里面去
 * 在上游服务需要使用 redis 来做库存的预扣减
 */
@Component
public class DataSyncConfig {
    @Autowired
    private GoodsMapper goodsMapper;

    @Autowired
    private StringRedisTemplate redisTemplate;

    // 业务场景是搞一个定时任务 每天 10 点开启
   // 为了 测试方便 项目已启动就执行一次

   /**
     * spring bean 的生命周期
    * 在当前对象 实例化完以后
    * 属性注入以后
    * 执行 PostConstruct 注解的方法
    */
    @PostConstruct
    @Scheduled(cron = "0 10 0 0 0 ?")
    public void initData() {
        List<Goods> goodsList = goodsMapper.selectSeckillGoods();
        if (CollectionUtils.isEmpty(goodsList)) {
            return;
        }
        goodsList.forEach(goods -> redisTemplate.opsForValue().set("goods_stock:" + goods.getId(), goods.getStocks().toString()));
    }
}
```

1.1.1.2 方法 2

```
package com.powernode.data;

import com.powernode.domain.Goods;
import com.powernode.mapper.GoodsMapper;
import org.springframework.boot.CommandLineRunner;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.core.ValueOperations;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
```

```java
import java.util.List;

/**
 * 描述: 暂无
 * 作者: 动力节点
 * 时间: 11:12
 */
@Component
public class MySqlToRedis2 implements CommandLineRunner {


    @Resource
    private GoodsMapper goodsMapper;

    @Resource
    private StringRedisTemplate stringRedisTemplate;

    @Override
    public void run(String... args) throws Exception {
        initData();
    }

    private void initData() {
        //1,查询数据库中需要参于秒杀的商品数据
        List<Goods> goodsList = goodsMapper.querySpikeGoods();
        ValueOperations<String, String> operations = stringRedisTemplate.opsForValue();
//          //2,把数据同步到Redis
         for (Goods goods : goodsList) {
            operations.set("goods:" + goods.getGoodsId(), goods.getTotalStocks().toString());
        }
    }
}
```

1.5.7. 创建秒杀监听

```java
package com.powernode.listener;


import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.powernode.service.GoodsService;
import org.apache.rocketmq.common.message.MessageExt;
import org.apache.rocketmq.spring.annotation.RocketMQMessageListener;
import org.apache.rocketmq.spring.core.RocketMQListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.stereotype.Component;

import java.util.concurrent.TimeUnit;

/**
 * 时间 14:27
 * 默认负载均衡模式
 * 默认多线程消费
 */
@Component
@RocketMQMessageListener(topic = "seckillTopic3", consumerGroup = "seckill-consumer-group")
public class SeckillMsgListener implements RocketMQListener<MessageExt> {

    @Autowired
    private GoodsService goodsService;

    @Autowired
    private StringRedisTemplate redisTemplate;

    // 20s
    int time = 20000;

    @Override
    public void onMessage(MessageExt message) {
        String s = new String(message.getBody());
        JSONObject jsonObject = JSON.parseObject(s);
        Integer goodsId = jsonObject.getInteger("goodsId");
        Integer userId = jsonObject.getInteger("userId");
        // 做真实的抢购业务  减库存 写订单表    todo 答案2  但是不符合分布式
//        synchronized (SeckillMsgListener.class) {
//            goodsService.realDoSeckill(goodsId, userId);
//        }
```

```
        // 自旋锁  一般 mysql 每秒1500/s 写  看数量 合理的设置自旋时间  todo 答案3
        int current = 0;
        while (current <= time) {
            // 一般在做分布式锁的情况下  会给锁一个过期时间 防止出现死锁的问题
           Boolean flag = redisTemplate.opsForValue().setIfAbsent("goods_lock:" + goodsId, "", 10, TimeUnit.SECONDS);
            if (flag) {
                try {
                    goodsService.realDoSeckill(goodsId, userId);
                    return;
                } finally {
                    redisTemplate.delete("goods_lock:" + goodsId);
                }
            } else {
                current += 200;
                try {
                    Thread.sleep(200);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

1.5.8. 修改 GoodsService

```
void realDoSeckill(Integer goodsId, Integer userId);
```

1.5.9. 修改 GoodsServiceImpl

```
    @Resource
    private GoodsMapper goodsMapper;

    @Autowired
    private OrderRecordsMapper orderRecordsMapper;

    /**
     *
     * @param goodsId
     * @param userId
     */
    @Override
    @Transactional(rollbackFor = RuntimeException.class)
    public void realDoSeckill(Integer goodsId, Integer userId) {
        // 扣减库存  插入订单表
        Goods goods = goodsMapper.selectByPrimaryKey(goodsId);
        int finalStock = goods.getStocks() - 1;
        if (finalStock < 0) {
            // 只是记录日志 让代码停下来   这里的异常用户无法感知
            throw new RuntimeException("库存不足: " + goodsId);
        }
        goods.setStocks(finalStock);
        goods.setUpdateTime(new Date());
        // insert 要么成功 要么报错  update 会出现i<=0 的情况
        // update goods set stocks =  1 where id = 1  没有行锁
        int i = goodsMapper.updateByPrimaryKey(goods);
        if (i > 0) {
            // 写订单表
            OrderRecords orderRecords = new OrderRecords();
            orderRecords.setGoodsId(goodsId);
            orderRecords.setUserId(userId);
            orderRecords.setCreateTime(new Date());
            // 时间戳生成订单号
            orderRecords.setOrderSn(String.valueOf(System.currentTimeMillis()));
            orderRecordsMapper.insert(orderRecords);
        }
    }

    /**
     * mysql 行锁  innodb  行锁
    * 分布式锁
    * todo 答案1
     *
     * @param goodsId
     * @param userId
     */
//    @Override
//    @Transactional(rollbackFor = RuntimeException.class)
//    public void realDoSeckill(Integer goodsId, Integer userId) {
//        // update goods set stocks = stocks - 1 ,update_time = now() where id = #{value}
//        int i = goodsMapper.updateStocks(goodsId);
//        if (i > 0) {
//            // 写订单表
//            OrderRecords orderRecords = new OrderRecords();
//            orderRecords.setGoodsId(goodsId);
//            orderRecords.setUserId(userId);
//            orderRecords.setCreateTime(new Date());
//            // 时间戳生成订单号
//            orderRecords.setOrderSn(String.valueOf(System.currentTimeMillis()));
//            orderRecordsMapper.insert(orderRecords);
//        }
//    }
```