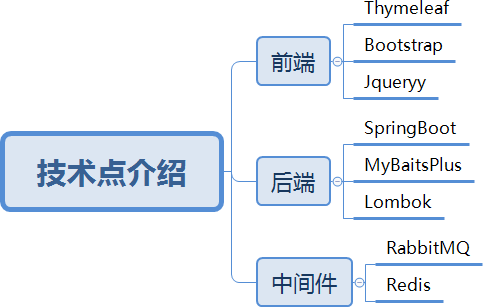
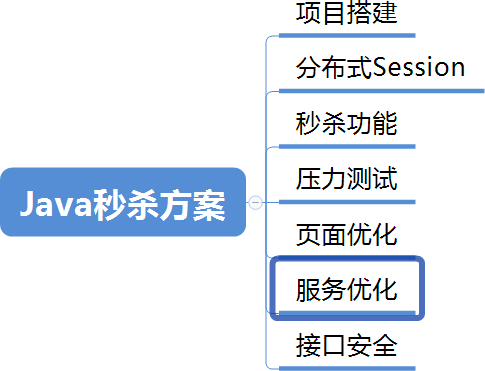
**Java秒杀方案**

# 课程介绍

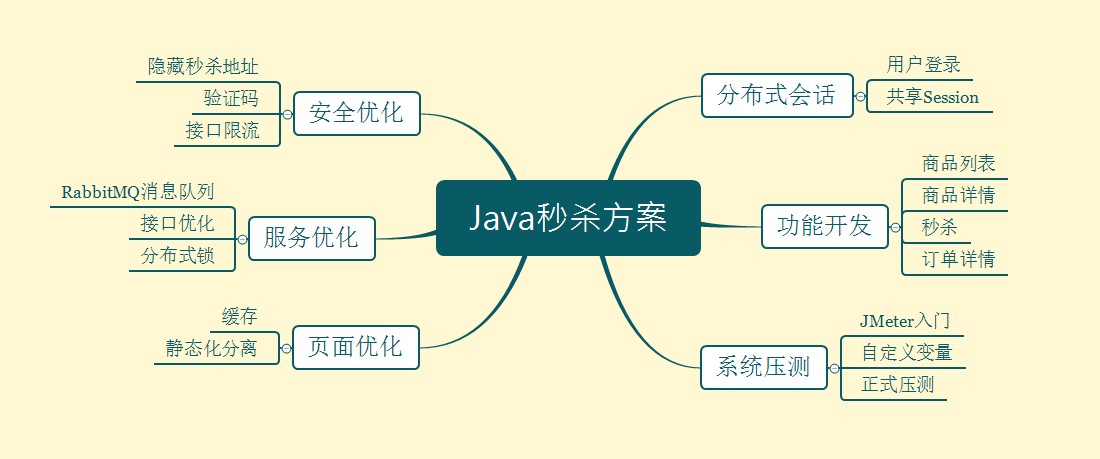
## 技术点介绍



**课程介绍**



**学习目标**



**如何设计一个秒杀系统**

##### 秒杀，对我们来说，都不是一个陌生的东西。每年的双11,618以及时下流行的直播等等。秒杀然而，这 对于我们系统而言是一个巨大的考验。

那么，如何才能更好地理解秒杀系统呢？我觉得作为一个程序员，你首先需要从高维度出发，从整体上 思考问题。在我看来，**秒杀其实主要解决两个问题，一个是并发读，一个是并发写**。并发读的核心优化 理念是尽量减少用户到服务端来“读”数据，或者让他们读更少的数据；并发写的处理原则也一样，它要 求我们在数据库层面独立出来一个库，做特殊的处理。另外，我们还要针对秒杀系统做一些保护，针对 意料之外的情况设计兜底方案，以防止最坏的情况发生。

**其实，秒杀的整体架构可以概括为“稳、准、快”几个关键字。**

所谓“稳”，就是整个系统架构要满足高可用，流量符合预期时肯定要稳定，就是超出预期时也同样不能 掉链子，你要保证秒杀活动顺利完成，即秒杀商品顺利地卖出去，这个是最基本的前提。

然后就是“准”，就是秒杀 10 台 iPhone，那就只能成交 10 台，多一台少一台都不行。一旦库存不对， 那平台就要承担损失，所以“准”就是要求保证数据的一致性。

最后再看“快”，“快”其实很好理解，它就是说系统的性能要足够高，否则你怎么支撑这么大的流量呢？ 不光是服务端要做极致的性能优化，而且在整个请求链路上都要做协同的优化，每个地方快一点，整个 系统就完美了。

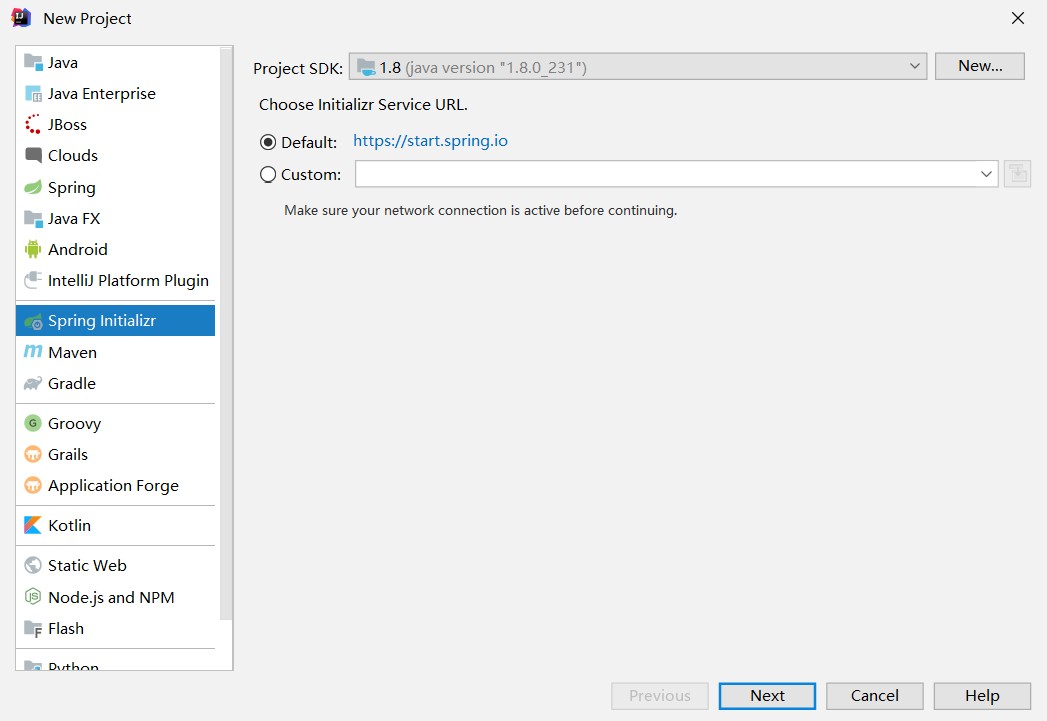
所以从技术角度上看“稳、准、快”，就对应了我们架构上的高可用、一致性和高性能的要求

**高性能。** 秒杀涉及大量的并发读和并发写，因此支持高并发访问这点非常关键。对应的方案比如动静分离方案、热点的发现与隔离、请求的削峰与分层过滤、服务端的极致优化

**一致性。** 秒杀中商品减库存的实现方式同样关键。可想而知，有限数量的商品在同一时刻被很多倍的请求同时来减库存，减库存又分为“拍下减库存”“付款减库存”以及预扣等几种，在大并发更新 的过程中都要保证数据的准确性，其难度可想而知

**高可用。** 现实中总难免出现一些我们考虑不到的情况，所以要保证系统的高可用和正确性，我们还要设计一个 PlanB 来兜底，以便在最坏情况发生时仍然能够从容应对。

**项目搭建**

**创建项目**

New Project X

Project Metadata

Grou p· 严 .xxxx 勹

Art ifact : G"江ill

Type : I Maven Proje ct (Generate a Maven based project a『 ch ive.) v La ng uage : 「Java v i

Pac kag ing: 工一 v i

Java Ve rsio n「:

8 v i

Ve rsio n: 匠产SNAPSHOT Name: 尸江ill

勹「 二 勹

De scriptio n: I se c kill

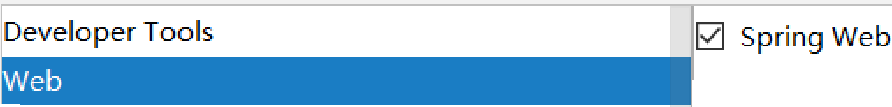
Pac kage: 匠兄.xxxx.s ec kill

r Pre vio us I I Nex t 1 1 Cance l I I He lp

Ne w Project X

Depe nde nc ies 0. Sp「ing Boo t 2.3: 勹 Selecte d Dependencies

斤e m plate Eng ines Secu rity



口 Spring Re act ive We b

SQL

No SQL

Messa g ing

1/ 0

Ops

Obse rva bil ity est ing

Spring Clo ud

Spring Clo ud Security Spring Clo ud To o ls Spring Clo ud Co nfig

Spring Clo ud Dis co ve ry Spring Clo ud Ro uting

Spring Clo ud Circu it Brea ke r Spring Clo ud Traci ng

Spring Clo ud Me ss ag ing Pivotal Clou d Foun dry

D Re s t Re posit o ries

* Spr·in g Sessio n

口 Re s t Re posit o ries HA L Browse r

口 Spring HATEOAS

* Spring We b Service s
* Je rsey

口 Vaa din

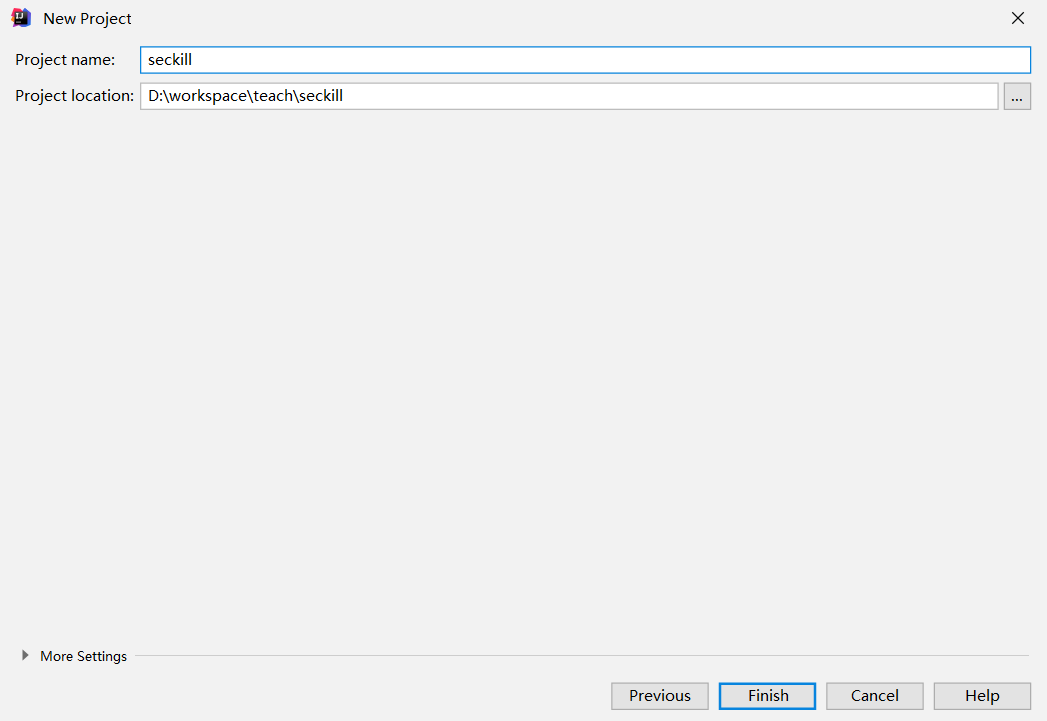
Deve lope r Tools Lo mbo k

Web

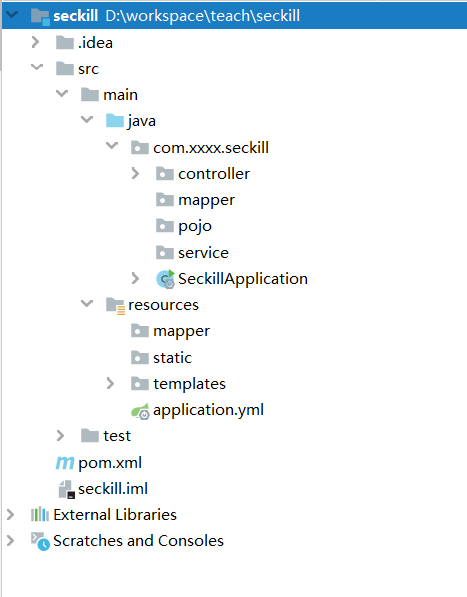
Sp ring We b

Temp late Engines Thy me le af

r Pre vio us I I Nex t 1 1 Cance l I ' He lp



**完整目录**



**添加依赖**

##### pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns=["http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi=["http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance)

xsi:schemaLocation=["http://maven.apache.org/POM/4.0.0](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>2.3.1.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.xxxx</groupId>

<artifactId>seckill</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>seckill</name>

<description>seckill</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!-- thymeleaf组件 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<!-- web组件 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!--mysql 依赖-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!--mybatis-plus 依赖-->

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>3.3.1.tmp</version>

</dependency>

<!--lombok 依赖-->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<!-- test组件 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## 修改配置文件

##### application.yml

spring:

# thymeleaf配置thymeleaf:

# 关闭缓存

cache: false

# 数据源配置

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver url: jdbc:mysql://localhost:3306/seckill?

useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai username: root

password: root hikari:

# 连接池名

pool-name: DateHikariCP

# 最小空闲连接数

minimum-idle: 5

# 空闲连接存活最大时间，默认600000（10分钟）

idle-timeout: 180000

# 最大连接数，默认10 maximum-pool-size: 10

# 从连接池返回的连接的自动提交

auto-commit: true

# 连接最大存活时间，0表示永久存活，默认1800000（30分钟）

max-lifetime: 1800000

# 连接超时时间，默认30000（30秒）

connection-timeout: 30000

# 测试连接是否可用的查询语句

connection-test-query: SELECT 1

# Mybatis-plus配置mybatis-plus:

#配置Mapper映射文件

mapper-locations: classpath\*:/mapper/\*Mapper.xml

# 配置MyBatis数据返回类型别名（默认别名是类名）

type-aliases-package: com.xxxx.seckill.pojo

## Mybatis SQL 打印(方法接口所在的包，不是Mapper.xml所在的包) logging:

level:

com.xxxx.seckill.mapper: debug

**测试**

DemoController.java

package com.xxxx.seckill.controller;

import org.springframework.stereotype.Controller; import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

/\*\*

* 测试

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/demo") public class DemoController {

/\*\*

* + 测试页面跳转

\*

* + @return

\*/

@RequestMapping("/hello")

public String hello(Model model) { model.addAttribute("name", "xxxx"); return "hello";

}

}

hello.html

<!DOCTYPE html>

<html lang="en"

xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<meta charset="UTF-8">

<title>测试</title>

</head>

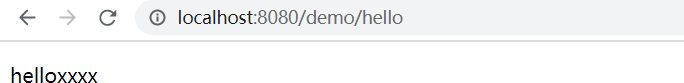
<body>

<p th:text="'hello:'+${name}"></p>

</body>

</html>

**测试结果**



**添加公共结果返回对象**

RespBeanEnum.java

package com.xxxx.seckill.vo;

import lombok.AllArgsConstructor; import lombok.Getter;

import lombok.ToString;

/\*\*

* 返回状态枚举

\*

* @author zhoubin

\* @since 1.0.0

\*/

@ToString

@Getter

@AllArgsConstructor public enum RespBeanEnum {

//通用状态码

SUCCESS(200,"success"),

ERROR(500,"服务端异常"),

//登录模块5002xx

SESSION\_ERROR(500210,"session不存在或者已经失效"),

LOGINVO\_ERROR(500211,"用户名或者密码错误"),

MOBILE\_ERROR(500212,"手机号码格式错误");

private final Integer code; private final String message;

}

RespBean.java

package com.xxxx.seckill.vo;

import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

* 通用返回结果对象

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor public class RespBean {

private long code; private String message; private Object obj;

/\*\*

\* 成功返回结果

\*/

public static RespBean success() {

return new RespBean(RespBeanEnum.SUCCESS.getCode(), RespBeanEnum.SUCCESS.getMessage(), null);

}

/\*\*

* 成功返回结果

\*

* @param obj

\*/

public static RespBean success(Object obj) {

return new RespBean(RespBeanEnum.SUCCESS.getCode(), RespBeanEnum.SUCCESS.getMessage(), obj);

}

/\*\*

* 失败返回结果

\*

* @param respBeanEnum
* @return

\*/

public static RespBean error(RespBeanEnum respBeanEnum) {

return new RespBean(respBeanEnum.getCode(), respBeanEnum.getMessage(),

null);

}

}

# 分布式会话

## 实现登录功能

### 两次**MD5**加密

##### 用户端：PASS=MD5(明文+固定Salt)

服务端：PASS=MD5(用户输入+随机Salt)

用户端MD5加密是为了防止用户密码在网络中明文传输，服务端MD5加密是为了提高密码安全性，双 重保险。

**引入pom.xml**

<!-- md5 依赖 -->

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.6</version>

</dependency>

**编写MD5工具类**

MD5Util.java

package com.xxxx.seckill.util;

import org.apache.commons.codec.digest.DigestUtils;

import org.springframework.stereotype.Component;

/\*\*

* MD5工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Component

public class MD5Util {

public static String md5(String src) { return DigestUtils.md5Hex(src);

}

private static final String salt = "1a2b3c4d";

public static String inputPassToFormPass(String inputPass) { String str = ""+salt.charAt(0)+salt.charAt(2) + inputPass

+salt.charAt(5) + salt.charAt(4); return md5(str);

}

public static String formPassToDBPass(String formPass, String salt) { String str = ""+salt.charAt(0)+salt.charAt(2) + formPass +salt.charAt(5)

+ salt.charAt(4);

return md5(str);

}

public static String inputPassToDbPass(String inputPass, String saltDB) { String formPass = inputPassToFormPass(inputPass);

String dbPass = formPassToDBPass(formPass, saltDB); return dbPass;

}

public static void main(String[] args) {

System.out.println(inputPassToFormPass("123456"));//d3b1294a61a07da9b49b6e22b2cb d7f9

System.out.println(formPassToDBPass(inputPassToFormPass("123456"), "1a2b3c4d"));

System.out.println(inputPassToDbPass("123456", "1a2b3c4d"));

}

}

**登录功能实现**

**逆向工程**

首先需要通过逆向工程基于 表生产对应的POJO、Mapper、Service、ServiceImpl、Controller

t\_user

等类，项目中使用了MybatisPlus，所以逆向工程也是用了MybatisPlus提供的AutoGenerator，代码如 下。具体可去官网查看

CodeGenerator.java

package com.xxxx.autogenerator;

import com.baomidou.mybatisplus.core.exceptions.MybatisPlusException; import com.baomidou.mybatisplus.core.toolkit.StringPool;

import com.baomidou.mybatisplus.core.toolkit.StringUtils; import com.baomidou.mybatisplus.generator.AutoGenerator; import com.baomidou.mybatisplus.generator.InjectionConfig;

import com.baomidou.mybatisplus.generator.config.DataSourceConfig; import com.baomidou.mybatisplus.generator.config.FileOutConfig; import com.baomidou.mybatisplus.generator.config.GlobalConfig; import com.baomidou.mybatisplus.generator.config.PackageConfig; import com.baomidou.mybatisplus.generator.config.StrategyConfig; import com.baomidou.mybatisplus.generator.config.TemplateConfig; import com.baomidou.mybatisplus.generator.config.po.TableInfo;

import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;

import com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;

import java.util.ArrayList;

import java.util.HashMap; import java.util.List; import java.util.Map; import java.util.Scanner;

/\*\*

* 执行 main 方法控制台输入模块表名回车自动生成对应项目目录中

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class CodeGenerator {

/\*\*

\* <p>

* + 读取控制台内容

\* </p>

\*/

public static String scanner(String tip) { Scanner scanner = new Scanner(System.in); StringBuilder help = new StringBuilder();

help.append("请输入" + tip + "：");

System.out.println(help.toString()); if (scanner.hasNext()) {

String ipt = scanner.next();

if (StringUtils.isNotEmpty(ipt)) { return ipt;

}

}

throw new MybatisPlusException("请输入正确的" + tip + "！");

}

public static void main(String[] args) {

// 代码生成器

AutoGenerator mpg = new AutoGenerator();

// 全局配置

GlobalConfig gc = new GlobalConfig();

String projectPath = System.getProperty("user.dir"); gc.setOutputDir(projectPath + "/src/main/java");

//作者

gc.setAuthor("zhoubin");

//打开输出目录gc.setOpen(false);

//xml 开 启 BaseResultMap gc.setBaseResultMap(true);

//xml 开 启 BaseColumnList gc.setBaseColumnList(true);

// 日 期 格 式 ， 采 用 Date gc.setDateType(DateType.ONLY\_DATE); mpg.setGlobalConfig(gc);

// 数据源配置

DataSourceConfig dsc = new DataSourceConfig(); dsc.setUrl("jdbc:mysql://localhost:3306/seckill?

useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia" + "/Shanghai");

dsc.setDriverName("com.mysql.cj.jdbc.Driver"); dsc.setUsername("root"); dsc.setPassword("root"); mpg.setDataSource(dsc);

// 包配置

PackageConfig pc = new PackageConfig(); pc.setParent("com.xxxx.seckill")

.setEntity("pojo")

.setMapper("mapper")

.setService("service")

.setServiceImpl("service.impl")

.setController("controller"); mpg.setPackageInfo(pc);

// 自定义配置

InjectionConfig cfg = new InjectionConfig() {

@Override

public void initMap() {

// to do nothing

Map<String,Object> map = new HashMap<>(); map.put("date1","1.0.0"); this.setMap(map);

}

};

// 如果模板引擎是 freemarker

String templatePath = "/templates/mapper.xml.ftl";

// 如果模板引擎是 velocity

// String templatePath = "/templates/mapper.xml.vm";

// 自定义输出配置

List<FileOutConfig> focList = new ArrayList<>();

// 自定义配置会被优先输出

focList.add(new FileOutConfig(templatePath) {

@Override

public String outputFile(TableInfo tableInfo) {

// 自定义输出文件名 ， 如果你 Entity 设置了前后缀、此处注意 xml 的名称会跟着发

生变化！！

return projectPath + "/src/main/resources/mapper/" +

tableInfo.getEntityName() + "Mapper"

+ StringPool.DOT\_XML;

}

});

cfg.setFileOutConfigList(focList); mpg.setCfg(cfg);

// 配置模板

TemplateConfig templateConfig = new TemplateConfig()

.setEntity("templates/entity2.java")

.setMapper("templates/mapper2.java")

.setService("templates/service2.java")

.setServiceImpl("templates/serviceImpl2.java")

.setController("templates/controller2.java");

templateConfig.setXml(null); mpg.setTemplate(templateConfig);

// 策略配置

StrategyConfig strategy = new StrategyConfig();

//数据库表映射到实体的命名策略strategy.setNaming(NamingStrategy.underline\_to\_camel);

//数据库表字段映射到实体的命名策略strategy.setColumnNaming(NamingStrategy.underline\_to\_camel);

//lombok模型strategy.setEntityLombokModel(true);

//生成 @RestController 控制器

// strategy.setRestControllerStyle(true);

strategy.setInclude(scanner("表名，多个英文逗号分割").split(",")); strategy.setControllerMappingHyphenStyle(true);

//表前缀strategy.setTablePrefix("t\_"); mpg.setStrategy(strategy);

mpg.setTemplateEngine(new FreemarkerTemplateEngine());

mpg.execute();

}

}

**ValidatorUtil**

package com.xxxx.seckill.util;

import org.springframework.util.StringUtils;

import java.util.regex.Matcher; import java.util.regex.Pattern;

/\*\*

* 校验工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class ValidatorUtil {

private static final Pattern mobile\_pattern = Pattern.compile("[1]([3-9])[0- 9]{9}$");

public static boolean isMobile(String mobile){ if (StringUtils.isEmpty(mobile)) {

return false;

}

Matcher matcher = mobile\_pattern.matcher(mobile); return matcher.matches();

}

}

**LoginController**

package com.xxxx.seckill.controller;

import com.xxxx.seckill.service.IUserService; import com.xxxx.seckill.vo.LoginVo;

import com.xxxx.seckill.vo.RespBean; import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

* 登录

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/login")

@Slf4j

public class LoginController {

@Autowired

private IUserService userService;

/\*\*

* + 跳转登录页

\*

* + @return

\*/

@RequestMapping("/toLogin") public String toLogin() {

return "login";

}

/\*\*

* + 登录
  + @return

\*/

@RequestMapping("/doLogin")

@ResponseBody

public RespBean doLogin(LoginVo loginVo) { log.info(loginVo.toString());

return userService.login(loginVo);

}

}

#### **IUserService**

package com.xxxx.seckill.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.xxxx.seckill.pojo.User; import com.xxxx.seckill.vo.LoginVo;

import com.xxxx.seckill.vo.RespBean;

/\*\*

\* <p>

* 服务类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface IUserService extends IService<User> {

/\*\*

* 登录
* @param loginVo
* @return

\*/

RespBean login(LoginVo loginVo);

}

**UserServiceImpl**

package com.xxxx.seckill.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.xxxx.seckill.mapper.UserMapper;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IUserService; import com.xxxx.seckill.util.MD5Util;

import com.xxxx.seckill.util.ValidatorUtil; import com.xxxx.seckill.vo.LoginVo;

import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import org.springframework.util.StringUtils;

/\*\*

\* <p>

* 服务实现类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements IUserService {

@Autowired

private UserMapper userMapper;

/\*\*

* 登录
* @param loginVo
* @return

\*/

@Override

public RespBean login(LoginVo loginVo) { String mobile = loginVo.getMobile(); String password = loginVo.getPassword();

if (StringUtils.isEmpty(mobile)||StringUtils.isEmpty(password)){ return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

}

if (!ValidatorUtil.isMobile(mobile)){

return RespBean.error(RespBeanEnum.MOBILE\_ERROR);

}

//根据手机号获取用户

User user = userMapper.selectById(mobile); if (null==user){

return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

}

//校验密码if

(!MD5Util.formPassToDBPass(password,user.getSalt()).equals(user.getPassword())){ return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

}

return RespBean.success();

}

}

**login.html**

<!DOCTYPE html>

<html lang="en"

xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<meta charset="UTF-8">

<title>登录</title>

<!-- jquery -->

<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" th:href="@{/bootstrap/css/bootstrap.min.css}"/>

<script type="text/javascript" th:src="@{/bootstrap/js/bootstrap.min.js}">

</script>

<!-- jquery-validator -->

<script type="text/javascript" th:src="@{/jquery- validation/jquery.validate.min.js}"></script>

<script type="text/javascript" th:src="@{/jquery- validation/localization/messages\_zh.min.js}"></script>

<!-- layer -->

<script type="text/javascript" th:src="@{/layer/layer.js}"></script>

<!-- md5.js -->

<script type="text/javascript" th:src="@{/js/md5.min.js}"></script>

<!-- common.js -->

<script type="text/javascript" th:src="@{/js/common.js}"></script>

</head>

<body>

<form name="loginForm" id="loginForm" method="post" style="width:50%; margin:0 auto">

<h2 style="text-align:center; margin-bottom: 20px">用户登录</h2>

<div class="form-group">

<div class="row">

<label class="form-label col-md-4">请输入手机号码</label>

<div class="col-md-5">

<input id="mobile" name="mobile" class="form-control" type="text" placeholder="手机号码" required="true"

minlength="11" maxlength="11"/>

</div>

<div class="col-md-1">

</div>

</div>

</div>

<div class="form-group">

<div class="row">

<label class="form-label col-md-4">请输入密码</label>

<div class="col-md-5">

<input id="password" name="password" class="form-control" type="password" placeholder="密码"

required="true" minlength="6" maxlength="16"/>

</div>

</div>

</div>

<div class="row">

<div class="col-md-5">

<button class="btn btn-primary btn-block" type="reset"

onclick="reset()">重置</button>

</div>

<div class="col-md-5">

<button class="btn btn-primary btn-block" type="submit" onclick="login()">登录</button>

</div>

</div>

</form>

</body>

<script>

function login() {

$("#loginForm").validate({ submitHandler: function (form) {

doLogin();

}

});

}

function doLogin() { g\_showLoading();

var inputPass = $("#password").val(); var salt = g\_passsword\_salt;

var str = "" + salt.charAt(0) + salt.charAt(2) + inputPass + salt.charAt(5) + salt.charAt(4);

var password = md5(str);

$.ajax({

url: "/login/doLogin", type: "POST",

data: {

mobile: $("#mobile").val(), password: password

},

success: function (data) { layer.closeAll();

if (data.code == 200) { layer.msg("成功");

} else {

layer.msg(data.message);

}

},

error: function () { layer.closeAll();

}

});

}

</script>

</html>

### 测试

##### 手机号码格式不正确



手机号码或者密码不正确



正确登录



**参数校验**

每个类都写大量的健壮性判断过于麻烦，我们可以使用 validation 简化我们的代码

### 添加依赖

##### pom.xml

<!-- validation组件 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

**自定义手机号码验证规则**

IsMobileValidator.java

package com.xxxx.seckill.validator;

import com.xxxx.seckill.util.ValidatorUtil; import org.springframework.util.StringUtils;

import javax.validation.ConstraintValidator; import javax.validation.ConstraintValidatorContext;

/\*\*

* 手机号码验证规则

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class IsMobileValidator implements ConstraintValidator<IsMobile,String> {

private boolean required = false;

@Override

public void initialize(IsMobile constraintAnnotation) { required = constraintAnnotation.required();

}

@Override

public boolean isValid(String value, ConstraintValidatorContext context) { if (required){

return ValidatorUtil.isMobile(value);

}else {

if (StringUtils.isEmpty(value)){ return true;

}else {

return ValidatorUtil.isMobile(value);

}

}

}

}

**自定义注解**

IsMobile.java

package com.xxxx.seckill.validator;

import javax.validation.Constraint; import javax.validation.Payload; import java.lang.annotation.Documented; import java.lang.annotation.Retention; import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.ANNOTATION\_TYPE; import static java.lang.annotation.ElementType.CONSTRUCTOR; import static java.lang.annotation.ElementType.FIELD;

import static java.lang.annotation.ElementType.METHOD; import static java.lang.annotation.ElementType.PARAMETER; import static java.lang.annotation.ElementType.TYPE\_USE; import static java.lang.annotation.RetentionPolicy.RUNTIME;

/\*\*

* 验证手机号

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Target({METHOD, FIELD, ANNOTATION\_TYPE, CONSTRUCTOR, PARAMETER, TYPE\_USE})

@Retention(RUNTIME)

@Documented

@Constraint(validatedBy = {IsMobileValidator.class}) public @interface IsMobile {

boolean required() default true;

String message() default "手机号码格式错误"; Class<?>[] groups() default {};

Class<? extends Payload>[] payload() default {};

}

**修改LoginVo**

LoginVo.java

package com.xxxx.seckill.vo;

import com.xxxx.seckill.validator.IsMobile; import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import org.hibernate.validator.constraints.Length;

import javax.validation.constraints.NotNull;

/\*\*

* 登录入参

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor public class LoginVo {

@NotNull

@IsMobile

private String mobile;

@NotNull

@Length(min = 32) private String password;

}

**其他修改**

**LoginController**

入参添加

@Valid

/\*\*

* 登录
* @return

\*/

@RequestMapping("/doLogin")

@ResponseBody

public RespBean doLogin(@Valid LoginVo loginVo) { log.info(loginVo.toString());

return userService.login(loginVo);

}

**UserServiceImpl**

注释掉之前的健壮性判断即可

/\*\*

* 登录
* @param loginVo
* @return

\*/

@Override

public RespBean login(LoginVo loginVo) {

String mobile = loginVo.getMobile(); String password = loginVo.getPassword();

// if (StringUtils.isEmpty(mobile)||StringUtils.isEmpty(password)){

//

// }

return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

// if (!ValidatorUtil.isMobile(mobile)){

//

// }

return RespBean.error(RespBeanEnum.MOBILE\_ERROR);

//根据手机号获取用户

User user = userMapper.selectById(mobile); if (null==user){

return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

}

//校验密码if

(!MD5Util.formPassToDBPass(password,user.getSalt()).equals(user.getPassword())){ return RespBean.error(RespBeanEnum.LOGINVO\_ERROR);

}

return RespBean.success();

}

**测试**



**异常处理**

我们知道，系统中异常包括：编译时异常和运行时异常 ，前者通过捕获异常从而获

RuntimeException

取异常信息，后者主要通过规范代码开发、测试通过手段减少运行时异常的发生。在开发中，不管是 dao层、service层还是controller层，都有可能抛出异常，在Springmvc中，能将所有类型的异常处理从各处理过程解耦出来，既保证了相关处理过程的功能较单一，也实现了异常信息的统一处理和维护。SpringBoot全局异常处理方式主要两种：

使用使用

@ControllerAdvice

ErrorController类

区别：

和

##### 来实现

@ExceptionHandler

注解。

##### 方式只能处理控制器抛出的异常。此时请求已经进入控制器中。

@ControllerAdvice

1. 方式可以处理所有的异常，包括未进入控制器的错误，比如404,401等错误

ErrorController类

1. 如果应用中两者共同存在，则 方式处理控制器抛出的异常，

@ControllerAdvice

4.

信息，自由度更大。

方式处理未进入控制器的异常。

方式可以定义多个拦截方法，拦截不同的异常类，并且可以获取抛出的异常

ErrorController类

@ControllerAdvice

**GlobalException**

package com.xxxx.seckill.exception;

import com.xxxx.seckill.vo.RespBeanEnum; import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

* 全局异常

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

public class GlobalException extends RuntimeException { private RespBeanEnum respBeanEnum;

}

**GlobalExceptionHandler**

package com.xxxx.seckill.exception;

import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.validation.BindException;

import org.springframework.web.bind.annotation.ExceptionHandler; import org.springframework.web.bind.annotation.RestControllerAdvice;

/\*\*

\* 全局异常处理

\*

\* @author zhoubin

\* @since 1.0.0

\*/

@RestControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(Exception.class)

public RespBean ExceptionHandler(Exception e) { if (e instanceof GlobalException) {

GlobalException ex = (GlobalException) e; return RespBean.error(ex.getRespBeanEnum());

} else if (e instanceof BindException) { BindException ex = (BindException) e;

RespBean respBean = RespBean.error(RespBeanEnum.BIND\_ERROR);

respBean.setMessage(" 参 数 校 验 异 常 ：" + ex.getBindingResult().getAllErrors().get(0).getDefaultMessage());

return respBean;

}

return RespBean.error(RespBeanEnum.ERROR);

}

}

**修改之前代码**

直接返回RespBean改为直接抛 异常

GlobalException

/\*\*

* 登录
* @param loginVo
* @return

\*/

@Override

public RespBean login(LoginVo loginVo) { String mobile = loginVo.getMobile(); String password = loginVo.getPassword();

//根据手机号获取用户

User user = userMapper.selectById(mobile); if (null==user){

throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

//校验密码if

(!MD5Util.formPassToDBPass(password,user.getSalt()).equals(user.getPassword())){ throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

return RespBean.success();

}

**测试**





**分布式Session**

**完善登录功能**

使用cookie+session记录用户信息

**准备工具类**

CookieUtil.java

package com.xxxx.seckill.util;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.UnsupportedEncodingException; import java.net.URLDecoder;

import java.net.URLEncoder;

/\*\*

* Cookie工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

public final class CookieUtil {

/\*\*

* + 得到Cookie的值, 不编码

\*

* + @param request
  + @param cookieName
  + @return

\*/

public static String getCookieValue(HttpServletRequest request, String cookieName) {

return getCookieValue(request, cookieName, false);

}

/\*\*

* + 得到Cookie的值,

\*

* + @param request
  + @param cookieName
  + @return

\*/

public static String getCookieValue(HttpServletRequest request, String cookieName, boolean isDecoder) {

Cookie[] cookieList = request.getCookies();

if (cookieList == null || cookieName == null) { return null;

}

String retValue = null; try {

for (int i = 0; i < cookieList.length; i++) {

if (cookieList[i].getName().equals(cookieName)) { if (isDecoder) {

retValue = URLDecoder.decode(cookieList[i].getValue(),

"UTF-8");

} else {

retValue = cookieList[i].getValue();

}

break;

}

}

} catch (UnsupportedEncodingException e) { e.printStackTrace();

}

return retValue;

}

/\*\*

* + 得到Cookie的值,

\*

* + @param request
  + @param cookieName
  + @return

\*/

public static String getCookieValue(HttpServletRequest request, String cookieName, String encodeString) {

Cookie[] cookieList = request.getCookies();

if (cookieList == null || cookieName == null) { return null;

}

String retValue = null; try {

for (int i = 0; i < cookieList.length; i++) {

if (cookieList[i].getName().equals(cookieName)) {

retValue = URLDecoder.decode(cookieList[i].getValue(),

encodeString);

}

}

break;

} catch (UnsupportedEncodingException e) { e.printStackTrace();

}

return retValue;

}

/\*\*

* + 设置Cookie的值 不设置生效时间默认浏览器关闭即失效,也不编码

\*/

public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName,

String cookieValue) { setCookie(request, response, cookieName, cookieValue, -1);

}

/\*\*

* + 设置Cookie的值 在指定时间内生效,但不编码

\*/

public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName,

false);

}

String cookieValue, int cookieMaxage) { setCookie(request, response, cookieName, cookieValue, cookieMaxage,

/\*\*

* + 设置Cookie的值 不设置生效时间,但编码

\*/

public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName,

String cookieValue, boolean isEncode) { setCookie(request, response, cookieName, cookieValue, -1, isEncode);

}

/\*\*

* + 设置Cookie的值 在指定时间内生效, 编码参数

\*/

public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName,

String cookieValue, int cookieMaxage, boolean

isEncode) {

doSetCookie(request, response, cookieName, cookieValue, cookieMaxage, isEncode);

}

/\*\*

* + 设置Cookie的值 在指定时间内生效, 编码参数(指定编码)

\*/

public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName,

String cookieValue, int cookieMaxage, String

encodeString) {

doSetCookie(request, response, cookieName, cookieValue, cookieMaxage, encodeString);

}

/\*\*

* + 删除Cookie带cookie域名

\*/

public static void deleteCookie(HttpServletRequest request, HttpServletResponse response,

String cookieName) { doSetCookie(request, response, cookieName, "", -1, false);

}

/\*\*

* + 设置Cookie的值，并使其在指定时间内生效

\*

* + @param cookieMaxage cookie生效的最大秒数

\*/

private static final void doSetCookie(HttpServletRequest request, HttpServletResponse response,

String cookieName, String cookieValue,

int cookieMaxage, boolean isEncode) { try {

if (cookieValue == null) { cookieValue = "";

} else if (isEncode) {

cookieValue = URLEncoder.encode(cookieValue, "utf-8");

}

Cookie cookie = new Cookie(cookieName, cookieValue); if (cookieMaxage > 0)

cookie.setMaxAge(cookieMaxage);

if (null != request) {// 设置域名的cookie

String domainName = getDomainName(request); System.out.println(domainName);

if (!"localhost".equals(domainName)) { cookie.setDomain(domainName);

}

}

cookie.setPath("/"); response.addCookie(cookie);

} catch (Exception e) { e.printStackTrace();

}

}

/\*\*

* + 设置Cookie的值，并使其在指定时间内生效

\*

* + @param cookieMaxage cookie生效的最大秒数

\*/

private static final void doSetCookie(HttpServletRequest request, HttpServletResponse response,

String cookieName, String cookieValue,

int cookieMaxage, String encodeString) { try {

if (cookieValue == null) { cookieValue = "";

} else {

cookieValue = URLEncoder.encode(cookieValue, encodeString);

}

Cookie cookie = new Cookie(cookieName, cookieValue); if (cookieMaxage > 0) {

cookie.setMaxAge(cookieMaxage);

}

if (null != request) {// 设置域名的cookie

String domainName = getDomainName(request); System.out.println(domainName);

if (!"localhost".equals(domainName)) { cookie.setDomain(domainName);

}

}

cookie.setPath("/"); response.addCookie(cookie);

} catch (Exception e) { e.printStackTrace();

}

}

/\*\*

* + 得到cookie的域名

\*/

private static final String getDomainName(HttpServletRequest request) { String domainName = null;

// 通过request对象获取访问的url地址

String serverName = request.getRequestURL().toString(); if (serverName == null || serverName.equals("")) {

domainName = "";

} else {

// 将url地下转换为小写

serverName = serverName.toLowerCase();

// 如果url地址是以[http://开头](http://开头/) 将[http://截取](http://截取/)

if (serverName.startsWith("http://")) { serverName = serverName.substring(7);

}

int end = serverName.length();

// 判断url地址是否包含"/"

if (serverName.contains("/")) {

//得到第一个"/"出现的位置

end = serverName.indexOf("/");

}

// 截取

serverName = serverName.substring(0, end);

// 根据"."进行分割

final String[] domains = serverName.split("\\."); int len = domains.length;

if (len > 3) {

// [www.xxx.com.cn](http://www.xxx.com.cn/)

domainName = domains[len - 3] + "." + domains[len - 2] + "." + domains[len - 1];

} else if (len <= 3 && len > 1) {

// xxx.com or xxx.cn

domainName = domains[len - 2] + "." + domains[len - 1];

} else {

domainName = serverName;

}

}

if (domainName != null && domainName.indexOf(":") > 0) { String[] ary = domainName.split("\\:");

domainName = ary[0];

}

return domainName;

}

}

**UUIDUtil.java**

package com.xxxx.seckill.util;

import java.util.UUID;

/\*\*

* UUID工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class UUIDUtil {

public static String uuid() {

return UUID.randomUUID().toString().replace("-", "");

}

}

#### **IUserService**

package com.xxxx.seckill.service;

import com.baomidou.mybatisplus.extension.service.IService; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.vo.LoginVo; import com.xxxx.seckill.vo.RespBean;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;

/\*\*

\* <p>

* 服务类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface IUserService extends IService<User> {

/\*\*

* + 登录
  + @param loginVo
  + @return

\*/

RespBean login(HttpServletRequest request, HttpServletResponse response, LoginVo loginVo);

}

#### **UserServiceImpl**

/\*\*

* 登录
* @param loginVo
* @return

\*/

@Override

public RespBean login(HttpServletRequest request, HttpServletResponse response,LoginVo loginVo) {

String mobile = loginVo.getMobile(); String password = loginVo.getPassword();

//根据手机号获取用户

User user = userMapper.selectById(mobile); if (null==user){

throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

//校验密码if

(!MD5Util.formPassToDBPass(password,user.getSalt()).equals(user.getPassword())){ throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

//生成cookie

String ticket = UUIDUtil.uuid(); request.getSession().setAttribute(ticket,user);

CookieUtil.setCookie(request,response,"userTicket",ticket); return RespBean.success(ticket);

}

#### **LoginController**

/\*\*

* 登录

\*

* @return

\*/

@RequestMapping("/doLogin")

@ResponseBody

public RespBean doLogin(HttpServletRequest request, HttpServletResponse response, @Valid LoginVo loginVo) {

log.info(loginVo.toString());

return userService.login(request, response, loginVo);

}

#### **GoodsController**

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import org.springframework.stereotype.Controller; import org.springframework.ui.Model;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpSession;

/\*\*

* 商品

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/goods") public class GoodsController {

/\*\*

* + 跳转登录页

\*

* + @return

\*/

@RequestMapping("/toList")

public String toLogin(HttpSession session, Model model,

@CookieValue("userTicket") String ticket) { if (StringUtils.isEmpty(ticket)) {

return "login";

}

User user = (User) session.getAttribute(ticket); if (null == user) {

return "login";

}

model.addAttribute("user", user); return "goodsList";

}

}

#### **login.html**

$.ajax({

url: "/login/doLogin", type: "POST",

data: {

mobile: $("#mobile").val(), password: password

},

success: function (data) { layer.closeAll();

if (data.code == 200) { layer.msg("成功");

window.location.href = "/goods/toList";

} else {

layer.msg(data.message);

}

},

error: function () { layer.closeAll();

}

});

#### **goodsList.html**

<!DOCTYPE html>

<html lang="en"

xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<meta charset="UTF-8">

<title>商品列表</title>

</head>

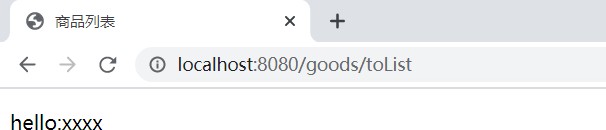
<body>

<p th:text="'hello:'+${user.username}"></p>

</body>

</html>

#### 测试



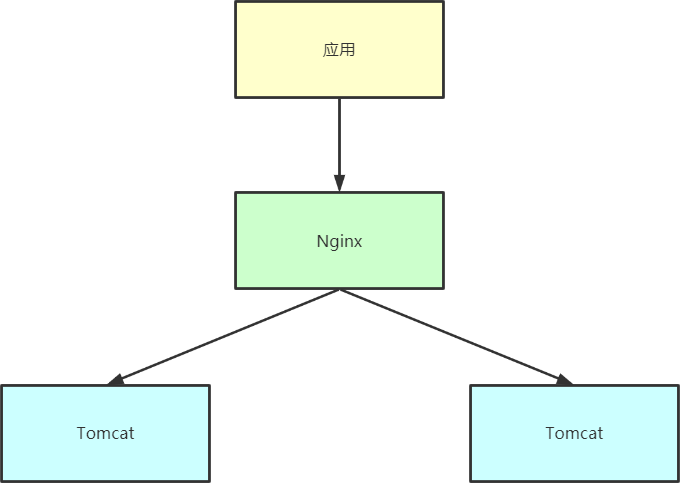
**分布式Session问题**

##### 之前的代码在我们之后一台应用系统，所有操作都在一台Tomcat上，没有什么问题。当我们部署多台 系统，配合Nginx的时候会出现用户登录的问题

原因

由于 Nginx 使用默认负载均衡策略（**轮询**），请求将会按照时间顺序逐一分发到后端应用上。

也就是说刚开始我们在 Tomcat1 登录之后，用户信息放在 Tomcat1 的 **Session** 里。过了一会，请求又被 Nginx 分发到了 Tomcat2 上，这时 Tomcat2 上 **Session** 里还没有用户信息，于是又要登录。



解决方案：

Session复制

优点缺点

前端存储

优点缺点

无需修改代码，只需要修改Tomcat配置

Session同步传输占用内网带宽多台Tomcat同步性能指数级下降

Session占用内存，无法有效水平扩展

不占用服务端内存存在安全风险

数据大小受cookie限制占用外网带宽

Session粘滞

优点

无需修改代码

服务端可以水平扩展

缺点

增加新机器，会重新Hash，导致重新登录应用重启，需要重新登录

后端集中存储

优点

安全

容易水平扩展

缺点

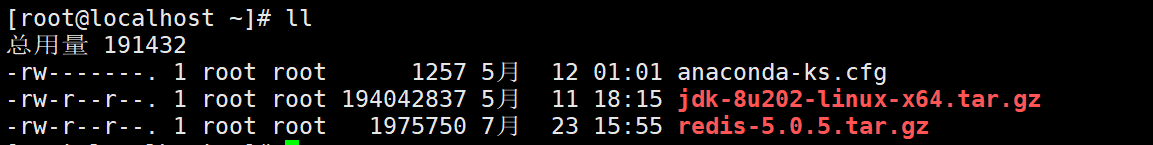
增加复杂度需要修改代码

**Redis安装**

**下载地址**

<http://redis.io/>

将下载好的安装包上传至服务器

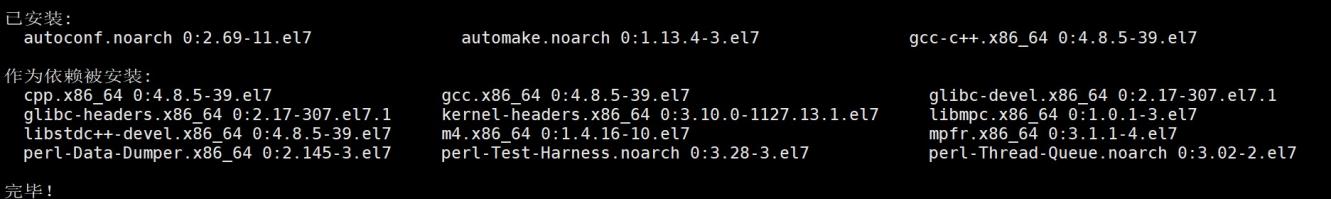


解压

tar zxvf redis-5.0.3.tar.gz

##### 安装依赖

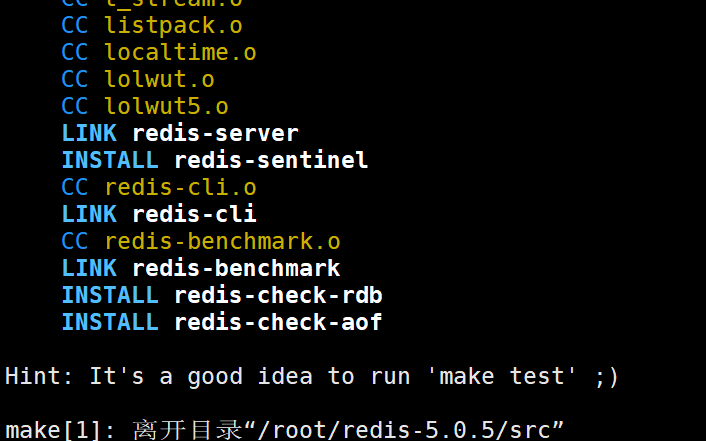
yum -y install gcc-c++ autoconf automake



##### 预编译

#切换到解压目录 cd redis-5.0.5/

#预编译make



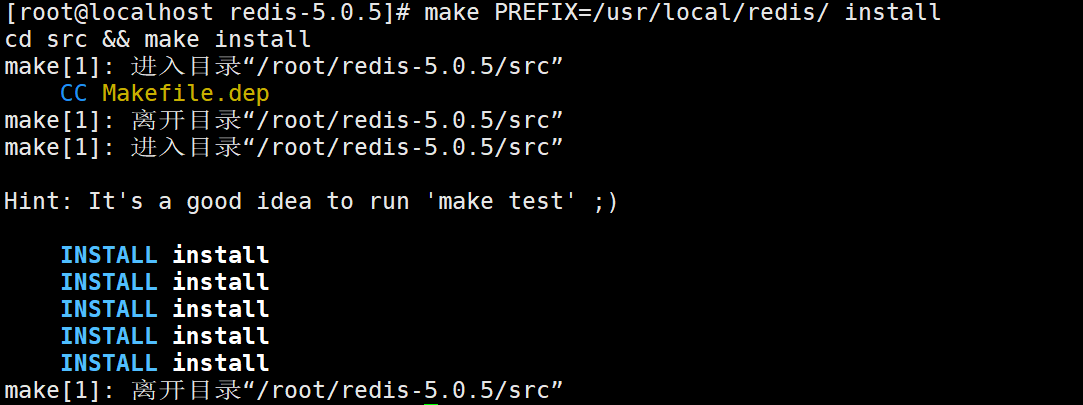
##### 安装

#创建安装目录

mkdir -p /usr/local/redis

#安装

make PREFIX=/usr/local/redis/ install



##### 修改配置文件

#复制redis.conf至安装路径下

cp redis.conf /usr/local/redis/bin/

#修改配置文件

vim /usr/local/redis/bin/redis.conf

##### 修改内容如下

#方便学习，注释掉该行。可以使所有ip访问redis

#bind 127.0.0.1

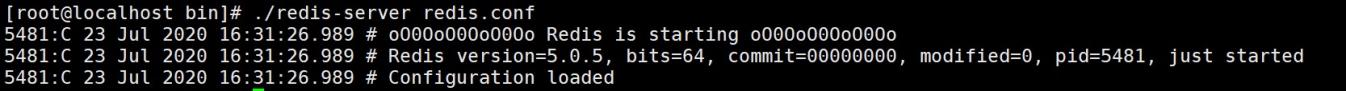
#关闭保护模式protected-mode no

#后台启动daemonize yes

#添加访问认证requirepass root

##### 启动redis

./redis-server redis.conf



### **Redis**实现分布式**Session**

#### 方法一：使用**SpringSession**实现

##### 添加依赖

pom.xml

<!-- spring data redis 依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

<!-- commons-pool2 对象池依赖 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-pool2</artifactId>

</dependency>

<!-- spring-session 依赖 -->

<dependency>

<groupId>org.springframework.session</groupId>

<artifactId>spring-session-data-redis</artifactId>

</dependency>

##### 添加配置

application.yml

spring: redis:

#超时时间

timeout: 10000ms

#服务器地址

host: 192.168.10.100

#服务器端口port: 6379

#数据库database: 0

# 密 码 password: root lettuce:

pool:

#最大连接数，默认8 max-active: 1024

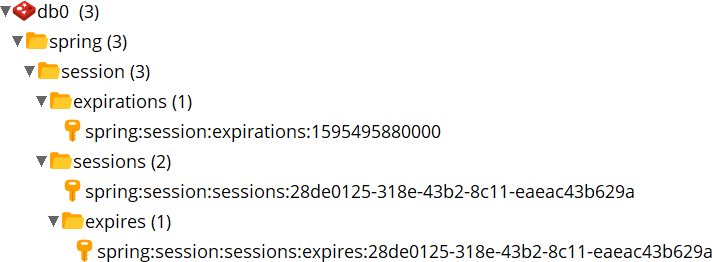
#最大连接阻塞等待时间，默认-1 max-wait: 10000ms

#最大空闲连接max-idle: 200

#最小空闲连接min-idle: 5

##### 测试

其余代码暂时不动，重新登录测试。会发现session已经存储在Redis上



**方法二：将用户信息存入Redis**

依赖

<!-- spring data redis 依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

<!-- commons-pool2 对象池依赖 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-pool2</artifactId>

</dependency>

##### 添加配置

application.yml

spring: redis:

#超时时间

timeout: 10000ms

#服务器地址

host: 192.168.10.100

#服务器端口port: 6379

#数据库database: 0

# 密 码 password: root lettuce:

pool:

#最大连接数，默认8 max-active: 1024

#最大连接阻塞等待时间，默认-1 max-wait: 10000ms

#最大空闲连接max-idle: 200

#最小空闲连接min-idle: 5

##### RedisConfig.java

package com.xxxx.seckill.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.redis.connection.RedisConnectionFactory; import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer; import org.springframework.data.redis.serializer.StringRedisSerializer;

/\*\*

* Redis配置类

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RedisConfig {

@Bean

public RedisTemplate<String,Object> redisTemplate(RedisConnectionFactory connectionFactory){

RedisTemplate<String,Object> redisTemplate = new RedisTemplate<>();

//key序列器

redisTemplate.setKeySerializer(new StringRedisSerializer());

//value序列器redisTemplate.setValueSerializer(new

GenericJackson2JsonRedisSerializer());

//Hash类型 key序列器

redisTemplate.setHashKeySerializer(new StringRedisSerializer());

//Hash类型 value序列器redisTemplate.setHashValueSerializer(new

GenericJackson2JsonRedisSerializer()); redisTemplate.setConnectionFactory(connectionFactory); return redisTemplate;

}

}

##### 工具类

JsonUtil.java

package com.xxxx.seckill.util;

import com.fasterxml.jackson.core.JsonParseException; import com.fasterxml.jackson.core.JsonProcessingException; import com.fasterxml.jackson.databind.JavaType;

import com.fasterxml.jackson.databind.JsonMappingException; import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.IOException; import java.util.List;

/\*\*

* Json工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class JsonUtil {

private static ObjectMapper objectMapper = new ObjectMapper();

/\*\*

* + 将对象转换成json字符串

\*

* + @param obj
  + @return

\*/

public static String object2JsonStr(Object obj) { try {

return objectMapper.writeValueAsString(obj);

} catch (JsonProcessingException e) {

//打印异常信息e.printStackTrace();

}

return null;

}

/\*\*

* + 将字符串转换为对象

\*

* + @param <T> 泛型

\*/

public static <T> T jsonStr2Object(String jsonStr, Class<T> clazz) { try {

return objectMapper.readValue(jsonStr.getBytes("UTF-8"), clazz);

} catch (JsonParseException e) { e.printStackTrace();

} catch (JsonMappingException e) { e.printStackTrace();

} catch (IOException e) { e.printStackTrace();

}

return null;

}

/\*\*

* + 将json数据转换成pojo对象list
  + <p>Title: jsonToList</p>
  + <p>Description: </p>

\*

* + @param jsonStr
  + @param beanType
  + @return

\*/

public static <T> List<T> jsonToList(String jsonStr, Class<T> beanType) { JavaType javaType =

objectMapper.getTypeFactory().constructParametricType(List.class, beanType); try {

List<T> list = objectMapper.readValue(jsonStr, javaType); return list;

} catch (Exception e) { e.printStackTrace();

}

return null;

}

}

##### 修改之前代码

IUserService.java

/\*\*

* 根据cookie获取用户
* @param userTicket
* @param request
* @param response
* @return

\*/

User getByUserTicket(String userTicket,HttpServletRequest request,HttpServletResponse response);

##### UserServiceImpl.java

package com.xxxx.seckill.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.xxxx.seckill.exception.GlobalException;

import com.xxxx.seckill.mapper.UserMapper; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IUserService; import com.xxxx.seckill.util.CookieUtil; import com.xxxx.seckill.util.JsonUtil;

import com.xxxx.seckill.util.MD5Util; import com.xxxx.seckill.util.UUIDUtil; import com.xxxx.seckill.vo.LoginVo; import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.stereotype.Service;

import org.springframework.util.StringUtils;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;

/\*\*

\* <p>

* 服务实现类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements IUserService {

@Autowired

private UserMapper userMapper;

@Autowired

private RedisTemplate redisTemplate;

/\*\*

* + 登录

\*

* + @param loginVo
  + @return

\*/

@Override

public RespBean login(HttpServletRequest request, HttpServletResponse response, LoginVo loginVo) {

String mobile = loginVo.getMobile(); String password = loginVo.getPassword();

//根据手机号获取用户

User user = userMapper.selectById(mobile); if (null == user) {

throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

//校验密码

if (!MD5Util.formPassToDBPass(password, user.getSalt()).equals(user.getPassword())) {

throw new GlobalException(RespBeanEnum.LOGINVO\_ERROR);

}

//生成cookie

String ticket = UUIDUtil.uuid(); redisTemplate.opsForValue().set("user:" + ticket,

JsonUtil.object2JsonStr(user));

CookieUtil.setCookie(request, response, "userTicket", ticket); return RespBean.success(ticket);

}

/\*\*

* + 根据cookie获取用户
  + @param userTicket
  + @param request
  + @param response
  + @return

\*/

@Override

public User getByUserTicket(String userTicket, HttpServletRequest request, HttpServletResponse response) {

if (StringUtils.isEmpty(userTicket)) { return null;

}

String userJson = (String) redisTemplate.opsForValue().get("user:" + userTicket);

User user = JsonUtil.jsonStr2Object(userJson, User.class); if (null != user) {

CookieUtil.setCookie(request,response,"userTicket",userTicket);

}

return user;

}

}

##### GoodsController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import org.springframework.stereotype.Controller; import org.springframework.ui.Model;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpSession;

/\*\*

* 商品

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/goods") public class GoodsController {

@Autowired

private IUserService userService;

/\*\*

* + 跳转登录页

\*

* + @return

\*/

@RequestMapping("/toList")

public String toLogin(HttpServletRequest request,HttpServletResponse response, Model model, @CookieValue("userTicket") String ticket) {

if (StringUtils.isEmpty(ticket)) { return "login";

}

User user = userService.getByUserTicket(ticket,request,response); if (null == user) {

return "login";

}

model.addAttribute("user", user); return "goodsList";

}

}

##### 测试



**优化登录功能**

UserArgumentResolver.java

package com.xxxx.seckill.config;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IUserService; import com.xxxx.seckill.util.CookieUtil;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.core.MethodParameter;

import org.springframework.stereotype.Component; import org.springframework.util.StringUtils;

import org.springframework.web.bind.support.WebDataBinderFactory; import org.springframework.web.context.request.NativeWebRequest;

import org.springframework.web.method.support.HandlerMethodArgumentResolver; import org.springframework.web.method.support.ModelAndViewContainer;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Component

public class UserArgumentResolver implements HandlerMethodArgumentResolver {

@Autowired

private IUserService userService;

@Override

public boolean supportsParameter(MethodParameter parameter) { Class<?> clazz = parameter.getParameterType();

return clazz == User.class;

}

@Override

public Object resolveArgument(MethodParameter parameter, ModelAndViewContainer mavContainer,

NativeWebRequest webRequest, WebDataBinderFactory binderFactory) throws Exception {

HttpServletRequest request = webRequest.getNativeRequest(HttpServletRequest.class);

HttpServletResponse response = webRequest.getNativeResponse(HttpServletResponse.class);

String ticket = CookieUtil.getCookieValue(request, "userTicket"); if (StringUtils.isEmpty(ticket)) {

return null;

}

return userService.getByUserTicket(ticket, request, response);

}

}

##### WebConfig.java

package com.xxxx.seckill.config;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.annotation.Configuration;

import org.springframework.web.method.support.HandlerMethodArgumentResolver; import org.springframework.web.servlet.config.annotation.EnableWebMvc; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import java.util.List;

/\*\*

* MVC配置类

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

@EnableWebMvc

public class WebConfig implements WebMvcConfigurer {

@Autowired

private UserArgumentResolver userArgumentResolver;

@Override

public void addArgumentResolvers(List<HandlerMethodArgumentResolver> resolvers) {

resolvers.add(userArgumentResolver);

}

}

##### GoodsController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import org.springframework.stereotype.Controller; import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

/\*\*

* 商品

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/goods") public class GoodsController {

/\*\*

* + 跳转登录页

\*

* + @return

\*/

@RequestMapping("/toList")

public String toLogin(Model model,User user) { model.addAttribute("user", user);

return "goodsList";

}

}

# 秒杀功能

## 商品列表页

##### 用逆向工程生成所需的所有类

**GoodsVo**

同时查询商品表和秒杀商品表的返回对象

package com.xxxx.seckill.vo;

import com.xxxx.seckill.pojo.Goods; import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

import java.math.BigDecimal; import java.util.Date;

/\*\*

* 商品返回对象

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

public class GoodsVo extends Goods { private BigDecimal seckillPrice; private Integer stockCount; private Date startDate;

private Date endDate;

}

**GoodsMapper**

##### GoodsMapper.java

package com.xxxx.seckill.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.xxxx.seckill.pojo.Goods;

import com.xxxx.seckill.vo.GoodsVo;

import java.util.List;

/\*\*

\* <p>

* Mapper 接口

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface GoodsMapper extends BaseMapper<Goods> {

/\*\*

* + 获取商品列表
  + @return

\*/

List<GoodsVo> findGoodsVo();

}

##### GoodsMapper.xml

<!-- 获取商品列表 -->

<select id="findGoodsVo" resultType="com.xxxx.seckill.vo.GoodsVo">

SELECT

g.id, g.goods\_name, g.goods\_title, g.goods\_img, g.goods\_detail, g.goods\_price, g.goods\_stock, sg.seckill\_price, sg.stock\_count, sg.start\_date, sg.end\_date

FROM

t\_goods AS g

LEFT JOIN t\_seckill\_goods AS sg ON g.id = sg.goods\_id

</select>

### **GoodsService**

##### IGoodService.java

package com.xxxx.seckill.service;

import com.baomidou.mybatisplus.extension.service.IService; import com.xxxx.seckill.pojo.Goods;

import com.xxxx.seckill.vo.GoodsVo;

import java.util.List;

/\*\*

\* <p>

* 服务类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface IGoodsService extends IService<Goods> {

/\*\*

* + 获取商品列表
  + @return

\*/

List<GoodsVo> findGoodsVo();

}

##### GoodsServiceImpl.java

package com.xxxx.seckill.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.xxxx.seckill.mapper.GoodsMapper;

import com.xxxx.seckill.pojo.Goods;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.vo.GoodsVo;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* <p>

* 服务实现类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

public class GoodsServiceImpl extends ServiceImpl<GoodsMapper, Goods> implements IGoodsService {

@Autowired

private GoodsMapper goodsMapper;

/\*\*

* + 获取商品列表
  + @return

\*/

@Override

public List<GoodsVo> findGoodsVo() { return goodsMapper.findGoodsVo();

}

}

### **GoodsController**

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

/\*\*

* 商品

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/goods") public class GoodsController {

@Autowired

private IGoodsService goodsService;

/\*\*

* + 跳转商品列表页

\*

* + @return

\*/

@RequestMapping("/toList")

public String toLogin(Model model, User user) { model.addAttribute("user", user); model.addAttribute("goodsList", goodsService.findGoodsVo()); return "goodsList";

}

}

### **MvcConfig**

##### 如果出现图片无法访问的情况需要修改此配置类。否则无需修改此配置类

@Override

public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/\*\*").addResourceLocations("classpath:/static/");

}

### **goodsList.html**

<!DOCTYPE html>

<html lang="en"

xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<meta charset="UTF-8">

<title>商品列表</title>

<!-- jquery -->

<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" th:href="@{/bootstrap/css/bootstrap.min.css}"/>

<script type="text/javascript" th:src="@{/bootstrap/js/bootstrap.min.js}">

</script>

<!-- layer -->

<script type="text/javascript" th:src="@{/layer/layer.js}"></script>

<!-- common.js -->

<script type="text/javascript" th:src="@{/js/common.js}"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀商品列表</div>

<table class="table" id="goodslist">

<tr>

</td>

<td>商品名称</td>

<td>商品图片</td>

<td>商品原价</td>

<td>秒杀价</td>

<td>库存数量</td>

<td>详情</td>

</tr>

<tr th:each="goods,goodsStat : ${goodsList}">

<td th:text="${goods.goodsName}"></td>

<td><img th:src="@{${goods.goodsImg}}" width="100" height="100"/>

<td th:text="${goods.goodsPrice}"></td>

<td th:text="${goods.seckillPrice}"></td>

<td th:text="${goods.stockCount}"></td>

<td><a th:href="'/goods/toDetail/'+${goods.id}">详情</a></td>

</tr>

</table>

</div>

</body>

</html>

### 测试



**商品详情页**

**GoodsMapper**

##### GoodsMapper.java

/\*\*

* 根据商品id获取商品详情
* @param goodsId
* @return

\*/

GoodsVo findGoodsVoByGoodsId(Long goodsId);

##### GoodsMapper.xml

<!-- 根据商品id获取商品详情 -->

<select id="findGoodsVoByGoodsId" resultType="com.xxxx.seckill.vo.GoodsVo"> SELECT

g.id, g.goods\_name, g.goods\_title, g.goods\_img, g.goods\_detail, g.goods\_price, g.goods\_stock, sg.seckill\_price, sg.stock\_count, sg.start\_date, sg.end\_date

FROM

t\_goods AS g

LEFT JOIN t\_seckill\_goods AS sg ON g.id = sg.goods\_id WHERE g.id = #{goodsId}

</select>

### **GoodsService**

##### IGoodsService.java

/\*\*

* 根据商品id获取商品详情
* @param goodsId
* @return

\*/

GoodsVo findGoodsVoByGoodsId(Long goodsId);

##### GoodsServiceImpl.java

/\*\*

* 根据商品id获取商品详情
* @param goodsId
* @return

\*/

@Override

public GoodsVo findGoodsVoByGoodsId(Long goodsId) { return goodsMapper.findGoodsVoByGoodsId(goodsId);

}

### **GoodsController**

/\*\*

* 跳转商品详情页

\*

* @param model
* @param user
* @param goodsId
* @return

\*/

@RequestMapping("/toDetail/{goodsId}")

public String toDetail(Model model, User user, @PathVariable Long goodsId) { model.addAttribute("user", user);

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId); model.addAttribute("goods", goods);

Date startDate = goods.getStartDate(); Date endDate = goods.getEndDate(); Date nowDate = new Date();

//秒杀状态

int secKillStatus = 0;

//剩余开始时间

int remainSeconds = 0;

//秒杀还未开始

if (nowDate.before(startDate)) {

remainSeconds = (int) ((startDate.getTime()-nowDate.getTime())/1000);

// 秒杀已结束

} else if (nowDate.after(endDate)) { secKillStatus = 2;

remainSeconds = -1;

// 秒杀中

} else {

secKillStatus = 1;

remainSeconds = 0;

}

model.addAttribute("secKillStatus",secKillStatus); model.addAttribute("remainSeconds",remainSeconds); return "goodsDetail";

}

### **goodsDetail.html**

<!DOCTYPE html>

<html lang="en"

xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<meta charset="UTF-8">

<title>商品详情</title>

<!-- jquery -->

<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" th:href="@{/bootstrap/css/bootstrap.min.css}"/>

<script type="text/javascript" th:src="@{/bootstrap/js/bootstrap.min.js}">

</script>

<!-- layer -->

<script type="text/javascript" th:src="@{/layer/layer.js}"></script>

<!-- common.js -->

<script type="text/javascript" th:src="@{/js/common.js}"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀商品详情</div>

<div class="panel-body">

<span th:if="${user eq null}"> 您还没有登录，请登陆后再操作<br/></span>

<span>没有收货地址的提示。。。</span>

</div>

<table class="table" id="goods">

<tr>

<td>商品名称</td>

<td colspan="3" th:text="${goods.goodsName}"></td>

</tr>

<tr>

<td>商品图片</td>

<td colspan="3"><img th:src="@{${goods.goodsImg}}" width="200" height="200"/></td>

</tr>

<tr>

<td>秒杀开始时间</td>

<td th:text="${#dates.format(goods.startDate, 'yyyy-MM-dd HH:mm:ss')}"></td>

<td id="seckillTip">

<input type="hidden" id="remainSeconds" th:value="${remainSeconds}"/>

<span th:if="${secKillStatus eq 0}">秒杀倒计时：<span

id="countDown"

th:text="${remainSeconds}"></span>秒</span>

<span th:if="${secKillStatus eq 1}">秒杀进行中</span>

<span th:if="${secKillStatus eq 2}">秒杀已结束</span>

</td>

<td>

<form id="seckillForm" method="post" action="/seckill/doSeckill">

<button class="btn btn-primary btn-block" type="submit"

id="buyButton">立即秒杀</button>

<input type="hidden" name="goodsId" th:value="${goods.id}"/>

</form>

</td>

</tr>

<tr>

<td>商品原价</td>

<td colspan="3" th:text="${goods.goodsPrice}"></td>

</tr>

<tr>

<td>秒杀价</td>

<td colspan="3" th:text="${goods.seckillPrice}"></td>

</tr>

<tr>

<td>库存数量</td>

<td colspan="3" th:text="${goods.stockCount}"></td>

</tr>

</table>

</div>

</body>

<script>

$(function(){

countDown();

});

function countDown(){

var remainSeconds = $("#remainSeconds").val(); var timeout;

//秒杀还没开始，倒计时

if(remainSeconds > 0){

$("#buyButton").attr("disabled", true); timeout = setTimeout(function(){

$("#countDown").text(remainSeconds - 1);

$("#remainSeconds").val(remainSeconds - 1); countDown();

},1000);

//秒杀进行中

}else if(remainSeconds == 0){

$("#buyButton").attr("disabled", false); if(timeout){

clearTimeout(timeout);

}

$("#seckillTip").html("秒杀进行中");

//秒杀已经结束

}else{

$("#buyButton").attr("disabled", true);

$("#seckillTip").html("秒杀已经结束");

}

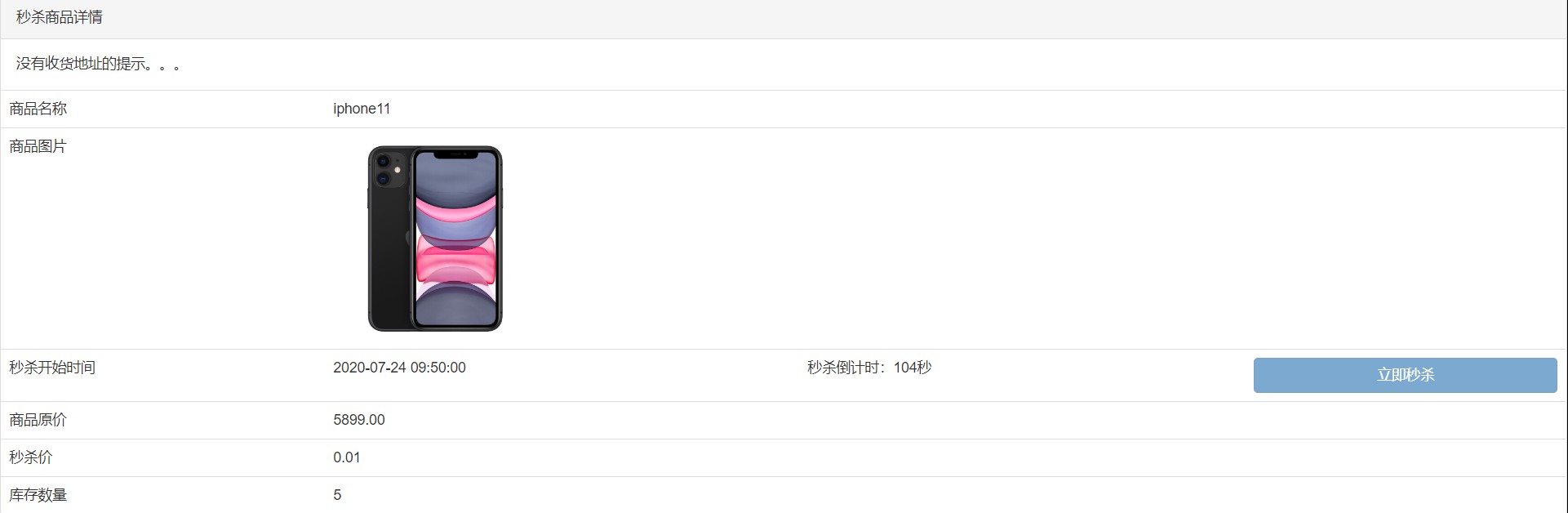
}

</script>

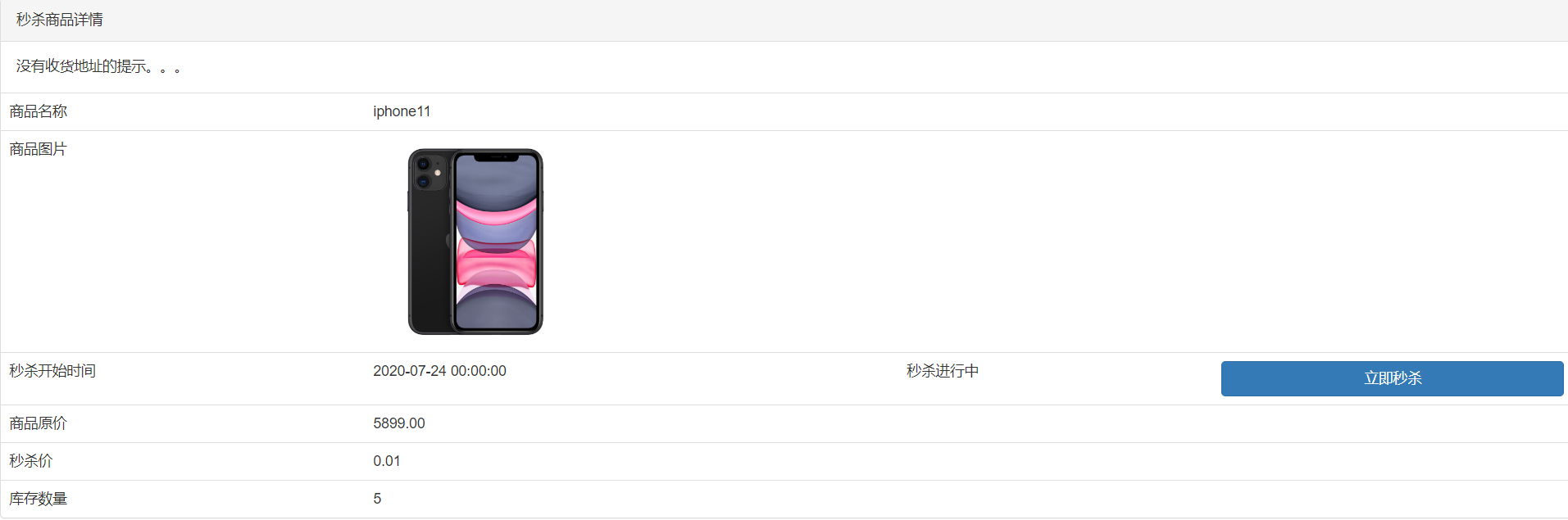
</html>

### 测试

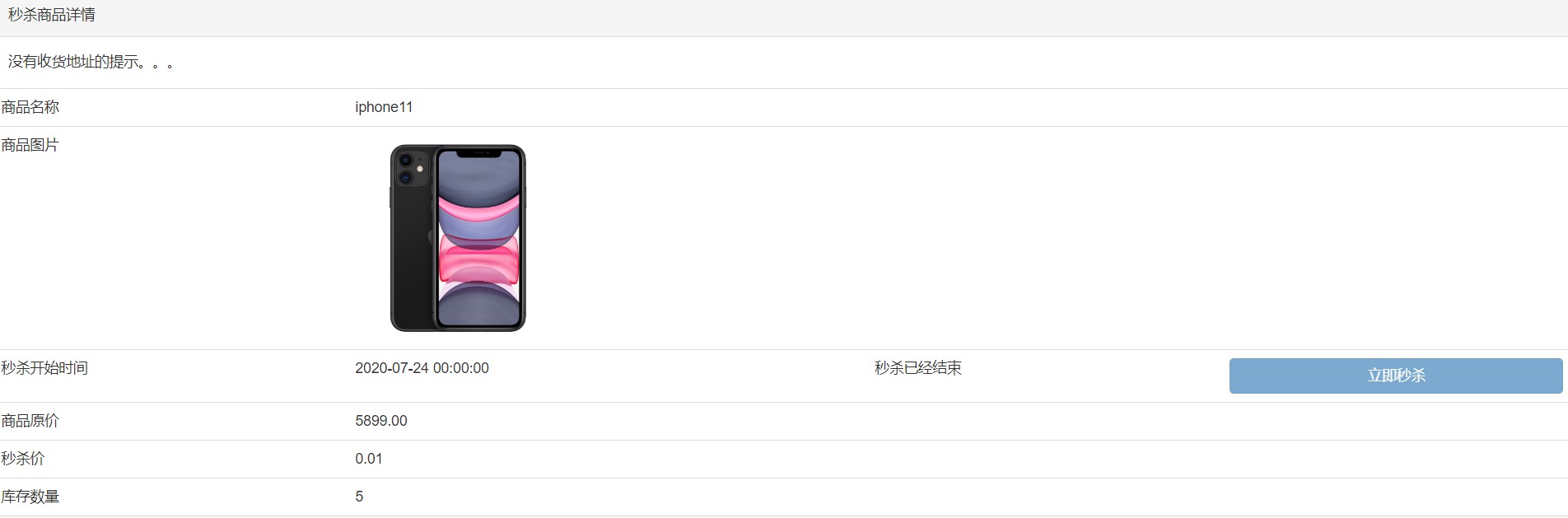
##### 秒杀未开始



秒杀进行中



秒杀已结束



**秒杀功能实现**

**OrderService**

IOrderService.java

package com.xxxx.seckill.service;

import com.baomidou.mybatisplus.extension.service.IService; import com.xxxx.seckill.pojo.Order;

import com.xxxx.seckill.pojo.User; import com.xxxx.seckill.vo.GoodsVo;

/\*\*

\* <p>

* 服务类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface IOrderService extends IService<Order> {

/\*\*

* + 秒杀
  + @param user
  + @param goods
  + @return

\*/

Order seckill(User user, GoodsVo goods);

}

##### OrderServiceImpl.java

package com.xxxx.seckill.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.xxxx.seckill.mapper.OrderMapper;

import com.xxxx.seckill.pojo.Order;

import com.xxxx.seckill.pojo.SeckillGoods; import com.xxxx.seckill.pojo.SeckillOrder; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillGoodsService; import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.vo.GoodsVo;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.Date;

/\*\*

\* <p>

* 服务实现类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

public class OrderServiceImpl extends ServiceImpl<OrderMapper, Order> implements IOrderService {

@Autowired

private ISeckillGoodsService seckillGoodsService;

@Autowired

private IGoodsService goodsService;

@Autowired

private OrderMapper orderMapper;

@Autowired

private ISeckillOrderService seckillOrderService;

/\*\*

* + 秒杀
  + @param user
  + @param goods
  + @return

\*/

@Override

@Transactional

public Order seckill(User user, GoodsVo goods) {

//秒杀商品表减库存

SeckillGoods seckillGoods = seckillGoodsService.getOne(new QueryWrapper<SeckillGoods>().eq("goods\_id",

goods.getId())); seckillGoods.setStockCount(seckillGoods.getStockCount()-1); seckillGoodsService.updateById(seckillGoods);

//生成订单

Order order = new Order(); order.setUserId(user.getId()); order.setGoodsId(goods.getId()); order.setDeliveryAddrId(0L); order.setGoodsName(goods.getGoodsName()); order.setGoodsCount(1); order.setGoodsPrice(seckillGoods.getSeckillPrice()); order.setOrderChannel(1);

order.setStatus(0); order.setCreateDate(new Date()); orderMapper.insert(order);

//生成秒杀订单

SeckillOrder seckillOrder = new SeckillOrder(); seckillOrder.setOrderId(order.getId()); seckillOrder.setUserId(user.getId()); seckillOrder.setGoodsId(goods.getId()); seckillOrderService.save(seckillOrder);

return order;

}

}

### **SeckillController**

package com.xxxx.seckill.controller;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.xxxx.seckill.pojo.Order;

import com.xxxx.seckill.pojo.SeckillOrder; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.vo.GoodsVo;

import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller; import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill") public class SeckillController {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@RequestMapping("/doSeckill")

public String doSeckill(Model model, User user, Long goodsId) { if (user == null) {

return "login";

}

model.addAttribute("user", user);

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) {

model.addAttribute("errmsg", RespBeanEnum.EMPTY\_STOCK.getMessage()); return "seckillFail";

}

//判断是否重复抢购

SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id", user.getId()).eq(

"goods\_id", goodsId));

if (seckillOrder != null) {

model.addAttribute("errmsg", RespBeanEnum.REPEATE\_ERROR.getMessage()); return "seckillFail";

}

Order order = orderService.seckill(user, goods); model.addAttribute("order",order); model.addAttribute("goods",goods);

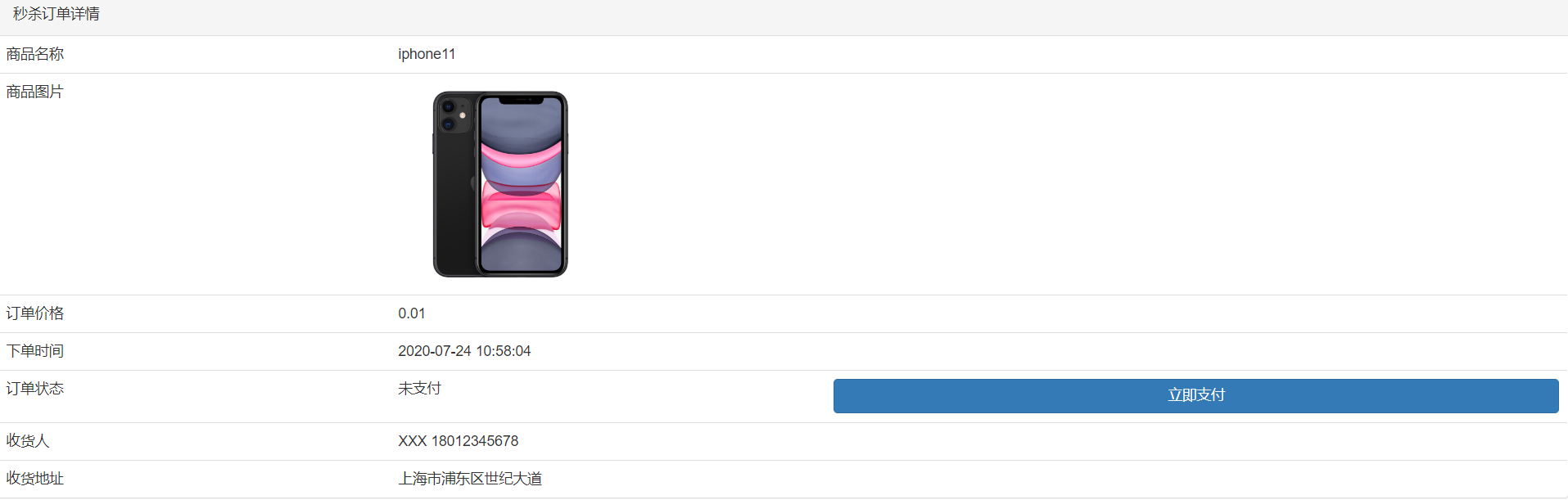
return "orderDetail";

}

}

### 测试

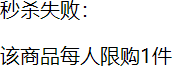
秒杀成功进入订单详情**注意查看库存是否正确扣减，订单是否正确生成**



##### 库存不足



重复抢购



**订单详情页**

本课程重点针对秒杀，所以订单详情只做简单页面展示，随后的支付等功能也不在本课程体现

**OrderDetail.html**

<!DOCTYPE HTML>

<html xmlns:th=["http://www.thymeleaf.org"](http://www.thymeleaf.org/)>

<head>

<title>订单详情</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<!-- jquery -->

<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" th:href="@{/bootstrap/css/bootstrap.min.css}" />

<script type="text/javascript" th:src="@{/bootstrap/js/bootstrap.min.js}">

</script>

<!-- layer -->

<script type="text/javascript" th:src="@{/layer/layer.js}"></script>

<!-- common.js -->

<script type="text/javascript" th:src="@{/js/common.js}"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀订单详情</div>

<table class="table" id="order">

<tr>

<td>商品名称</td>

<td th:text="${goods.goodsName}" colspan="3"></td>

</tr>

<tr>

<td>商品图片</td>

<td colspan="2"><img th:src="@{${goods.goodsImg}}" width="200" height="200" /></td>

</tr>

<tr>

<td>订单价格</td>

<td colspan="2" th:text="${order.goodsPrice}"></td>

</tr>

<tr>

<td>下单时间</td>

<td th:text="${#dates.format(order.createDate, 'yyyy-MM-dd HH:mm:ss')}" colspan="2"></td>

</tr>

<tr>

<td>订单状态</td>

<td >

<span th:if="${order.status eq 0}">未支付</span>

<span th:if="${order.status eq 1}">待发货</span>

<span th:if="${order.status eq 2}">已发货</span>

<span th:if="${order.status eq 3}">已收货</span>

<span th:if="${order.status eq 4}">已退款</span>

<span th:if="${order.status eq 5}">已完成</span>

</td>

<td>

<button class="btn btn-primary btn-block" type="submit"

id="payButton">立即支付</button>

</td>

</tr>

<tr>

<td>收货人</td>

<td colspan="2">XXX 18012345678</td>

</tr>

<tr>

<td>收货地址</td>

<td colspan="2">上海市浦东区世纪大道</td>

</tr>

</table>

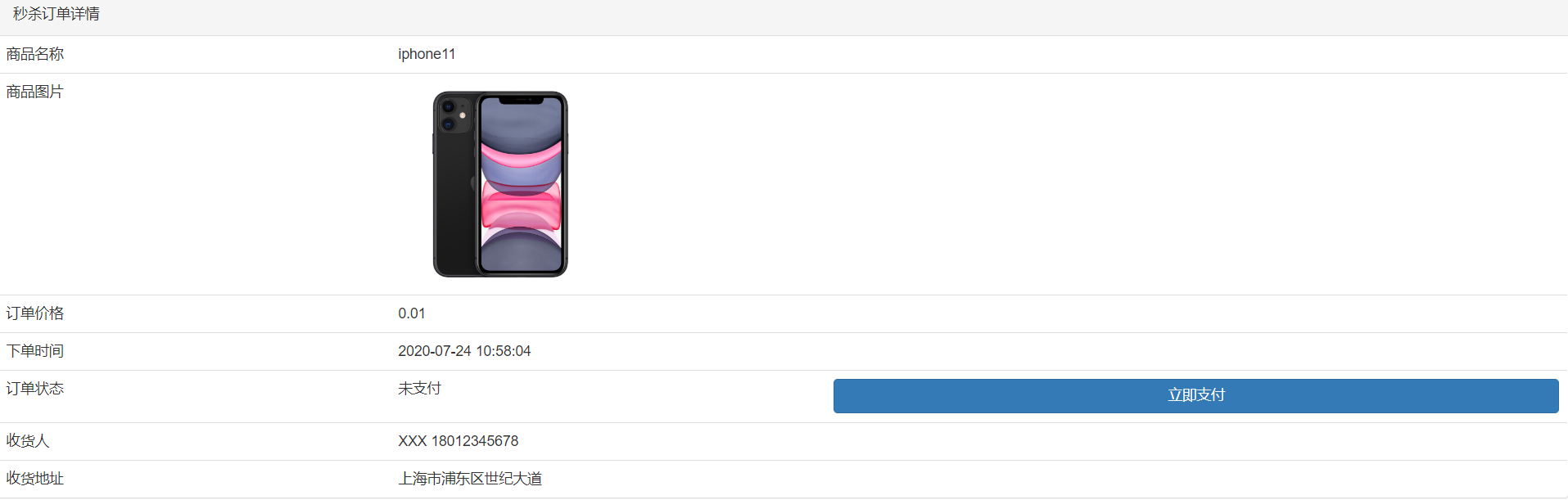
</div>

</body>

</html>

**测试**

##### 至此，简单的秒杀功能逻辑就完成了，下面进入优化阶段



**系统压测**

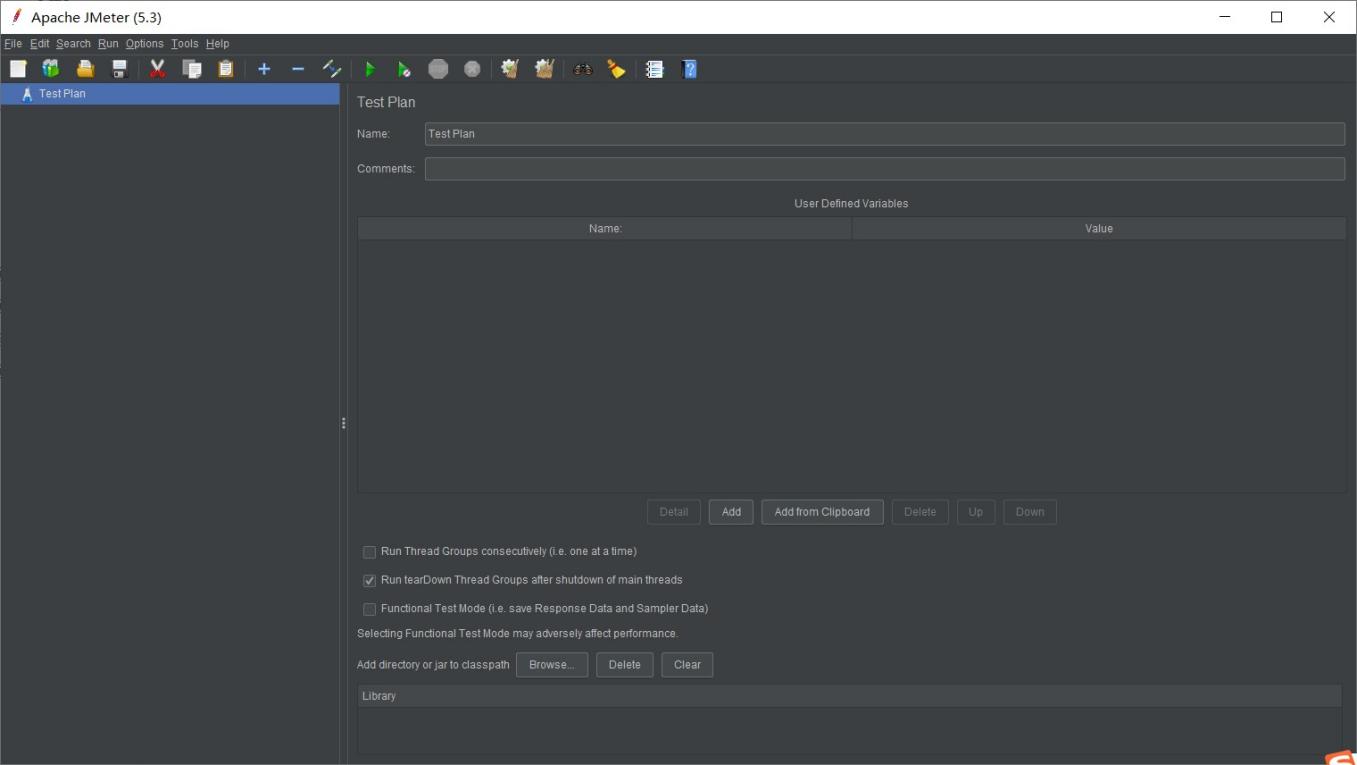
**JMeter 入门**

**安装**

官网：<https://jmeter.apache.org/>

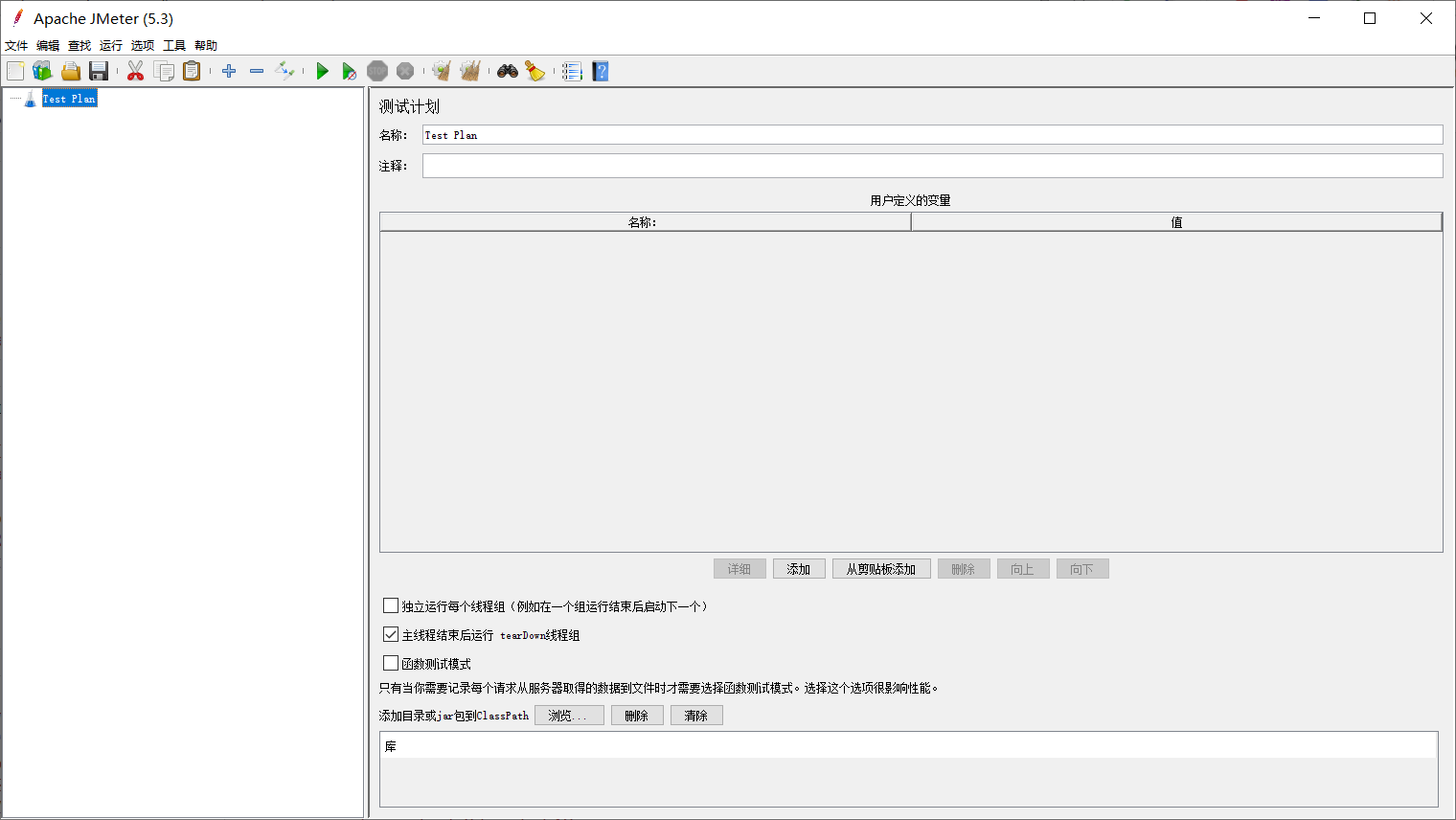
下载地址：<https://jmeter.apache.org/download_jmeter.cgi>

下载解压后直接在 bin 目录里双击 jmeter.bat 即可启动（**Lunix系统通过** jmeter.sh **启动**）



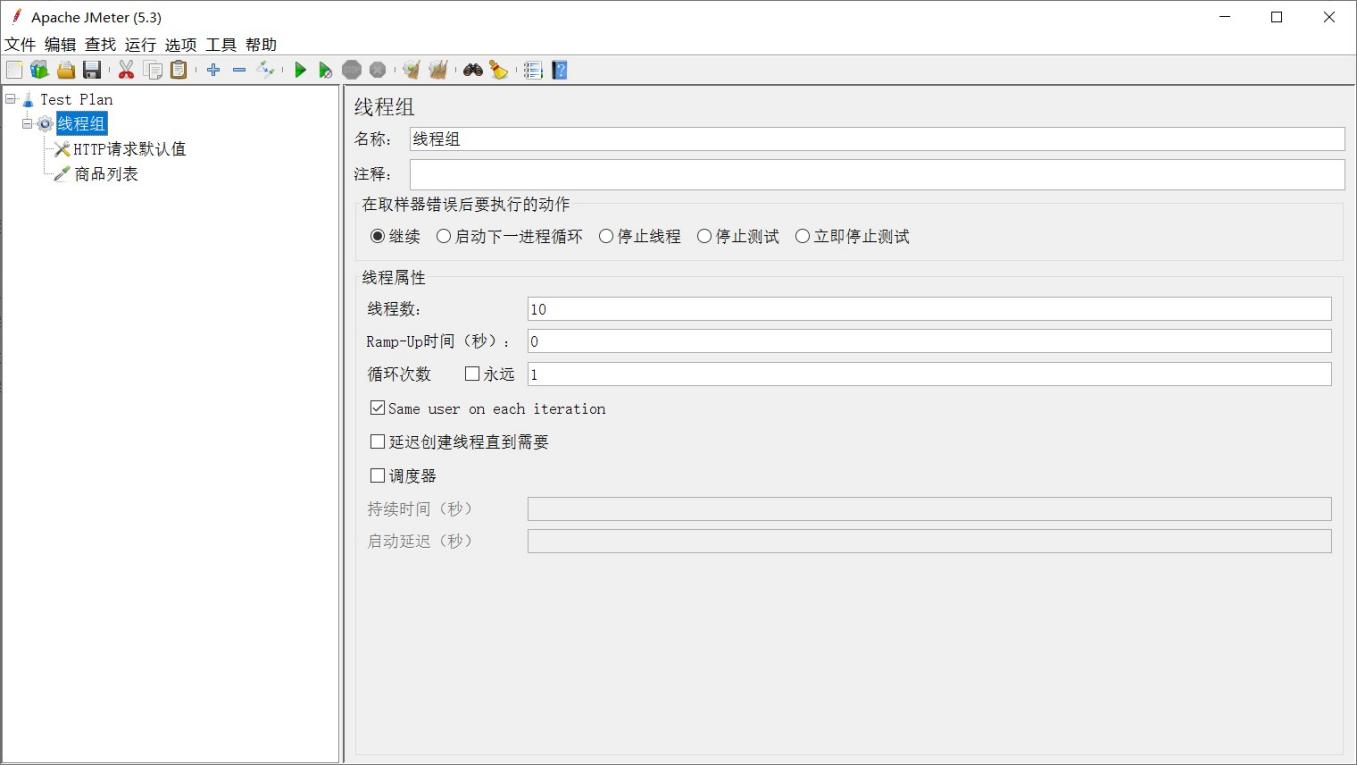
#### 修改中文

##### Options-->Choose Language-->Chinese(Simplified)



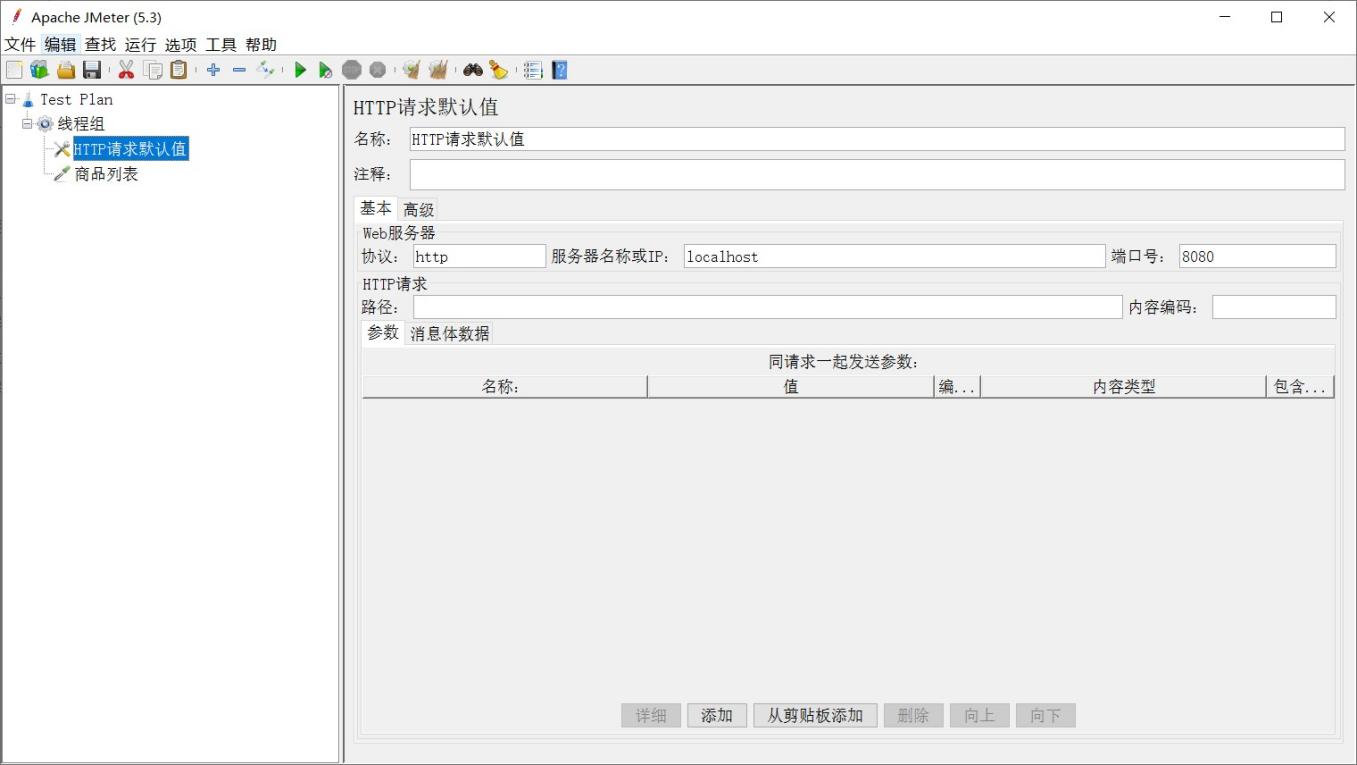
**简单使用**

我们先使用JMeter测试一下跳转商品列表页的接口。 首先创建线程组，步骤：添加--> 线程(用户) --> 线程组

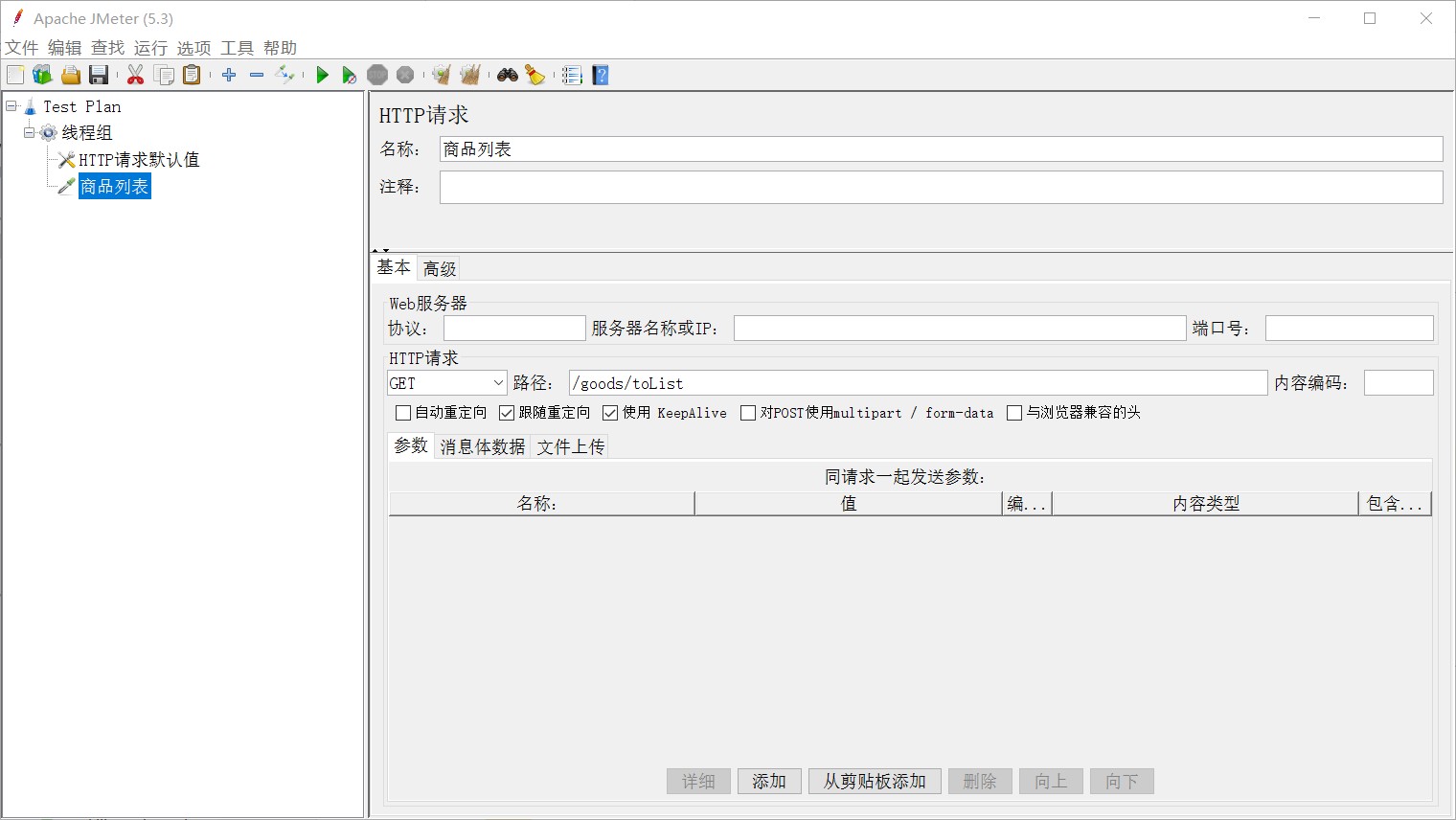


Ramp-up 指在几秒之内启动指定线程数

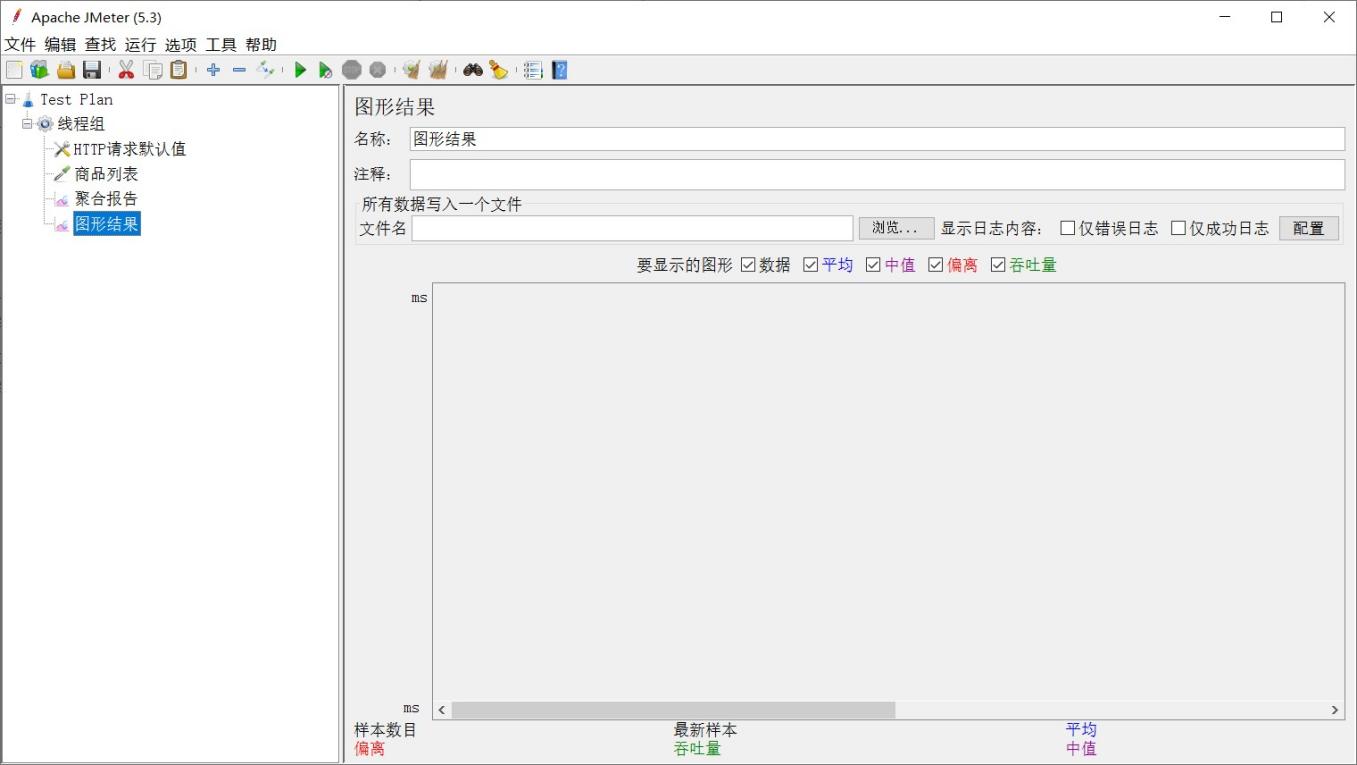
##### 创建HTTP请求默认值，步骤：添加--> 配置元件 --> HTTP请求默认值



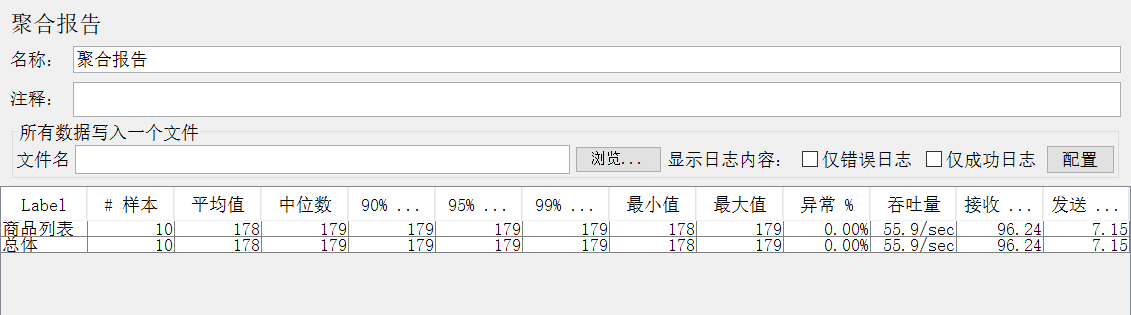
添加测试接口，步骤：添加 --> 取样器 --> HTTP请求

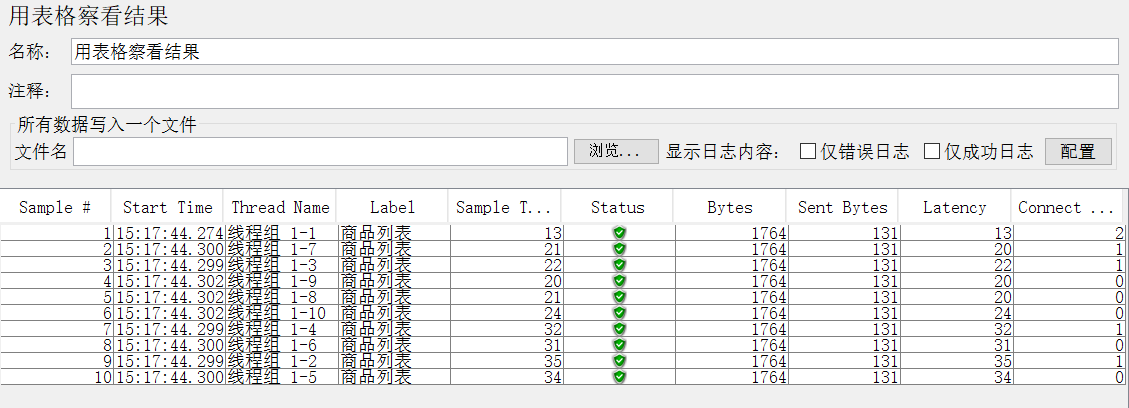


查看输出结果，步骤：添加 --> 监听器 --> 聚合报告/图形结果/用表格察看结果



启动即可在监听器看到对应的结果





**自定义变量**

**准备测试接口**

UserController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User; import com.xxxx.seckill.vo.RespBean;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/user") public class UserController {

/\*\*

* + 用户信息(测试)
  + @param user
  + @return

\*/

@RequestMapping("/info")

@ResponseBody

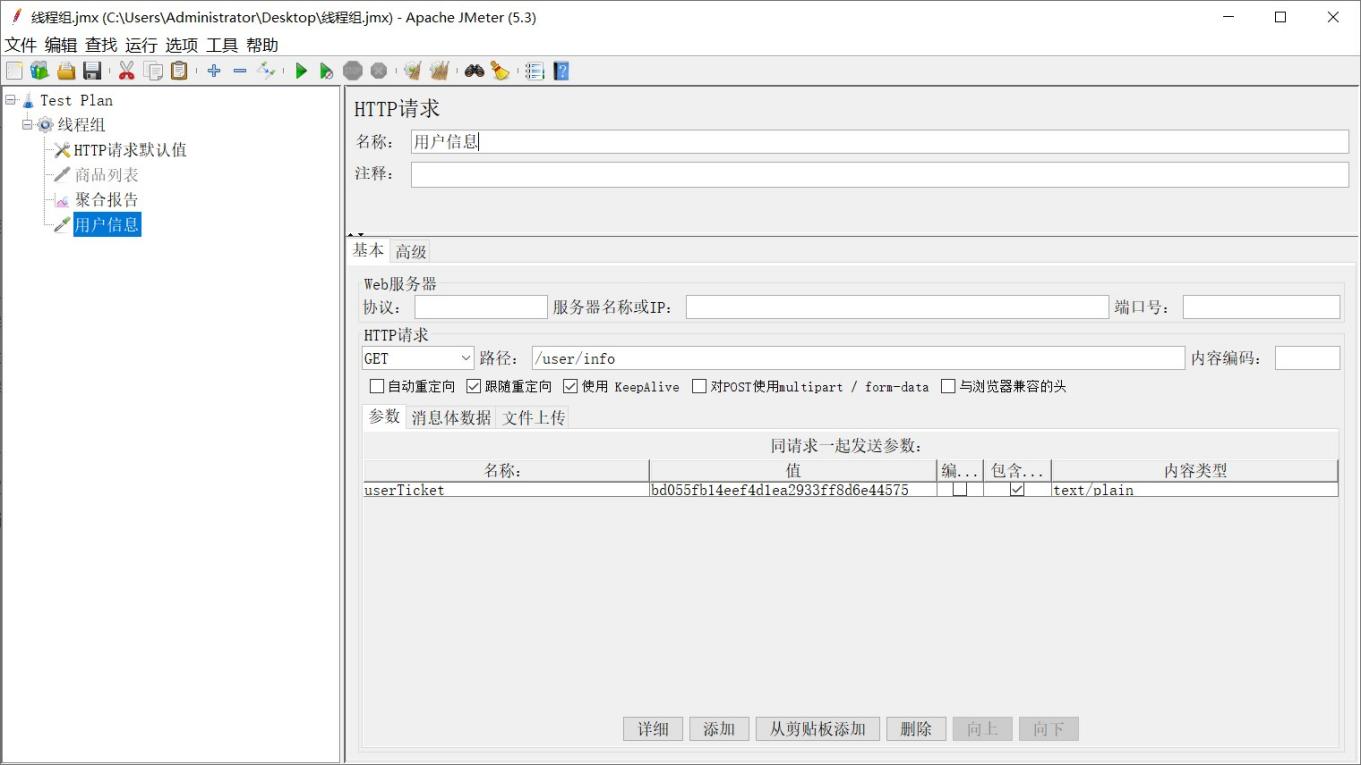
public RespBean info(User user){ return RespBean.success(user);

}

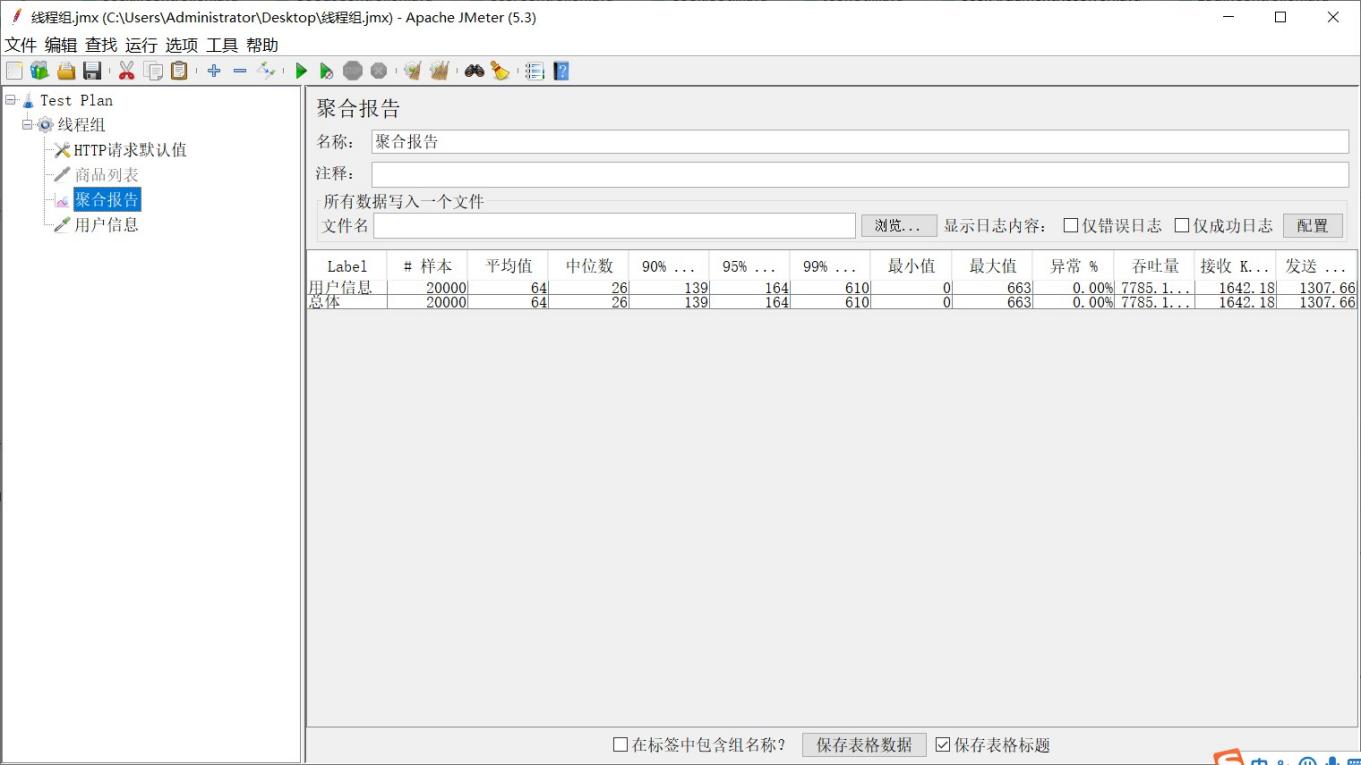
}

### 配置同一用户测试

##### 添加HTTP请求用户信息



查看聚合结果

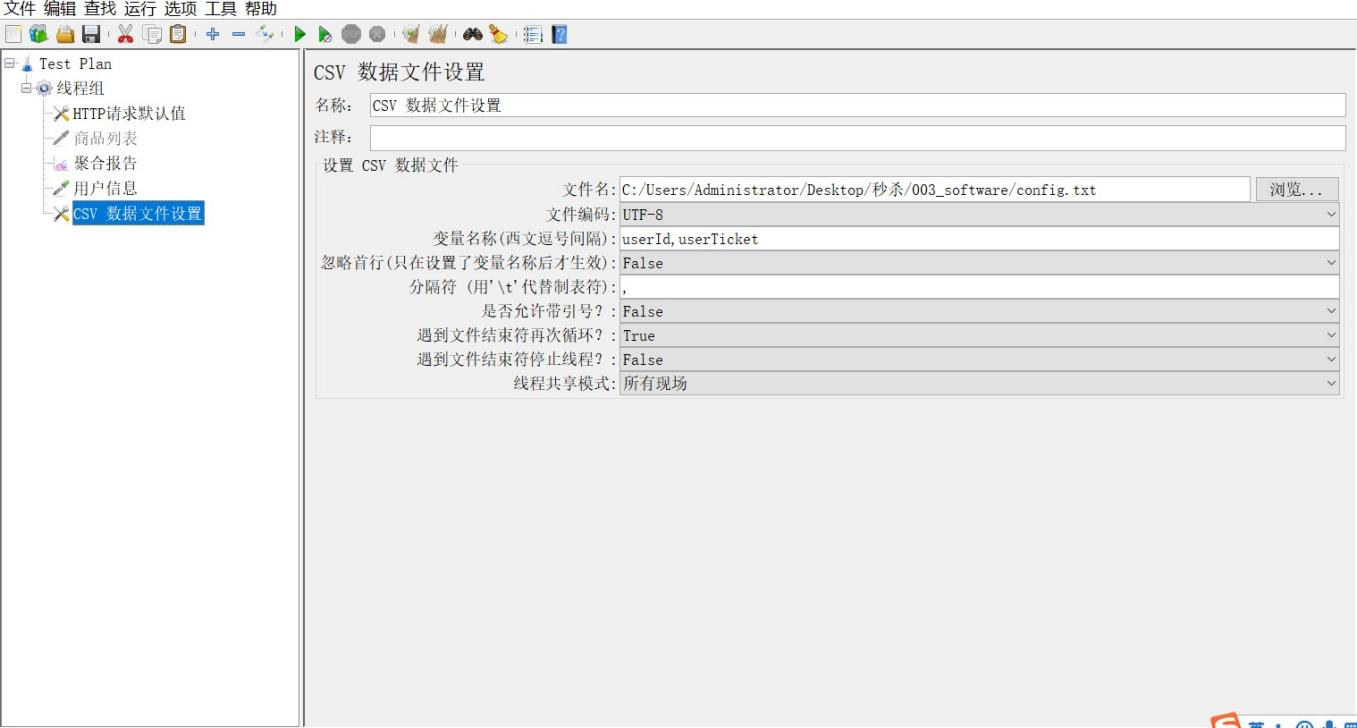


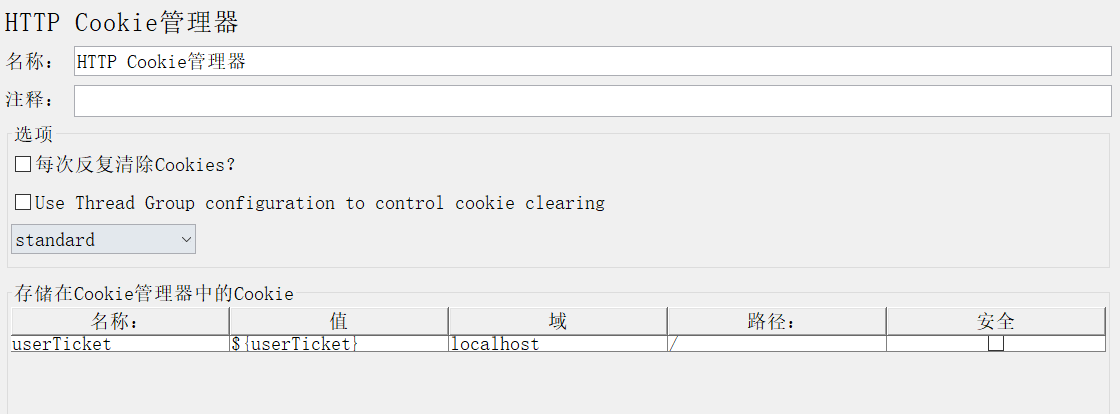
**配置不同用户测试**

准备配置文件config.txt

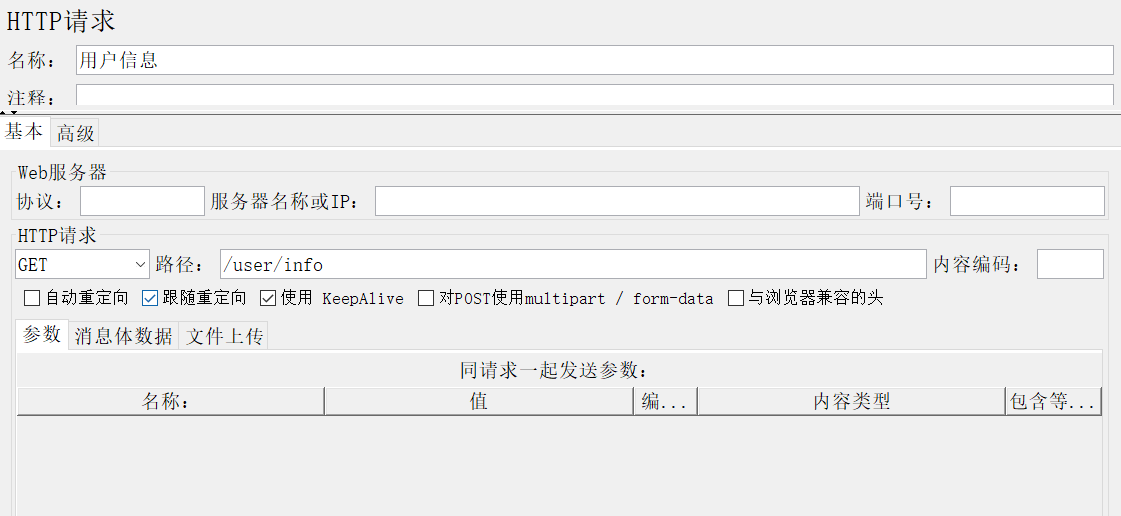
# 具 体 用 户 和 userTicket 18012345678,bd055fb14eef4d1ea2933ff8d6e44575

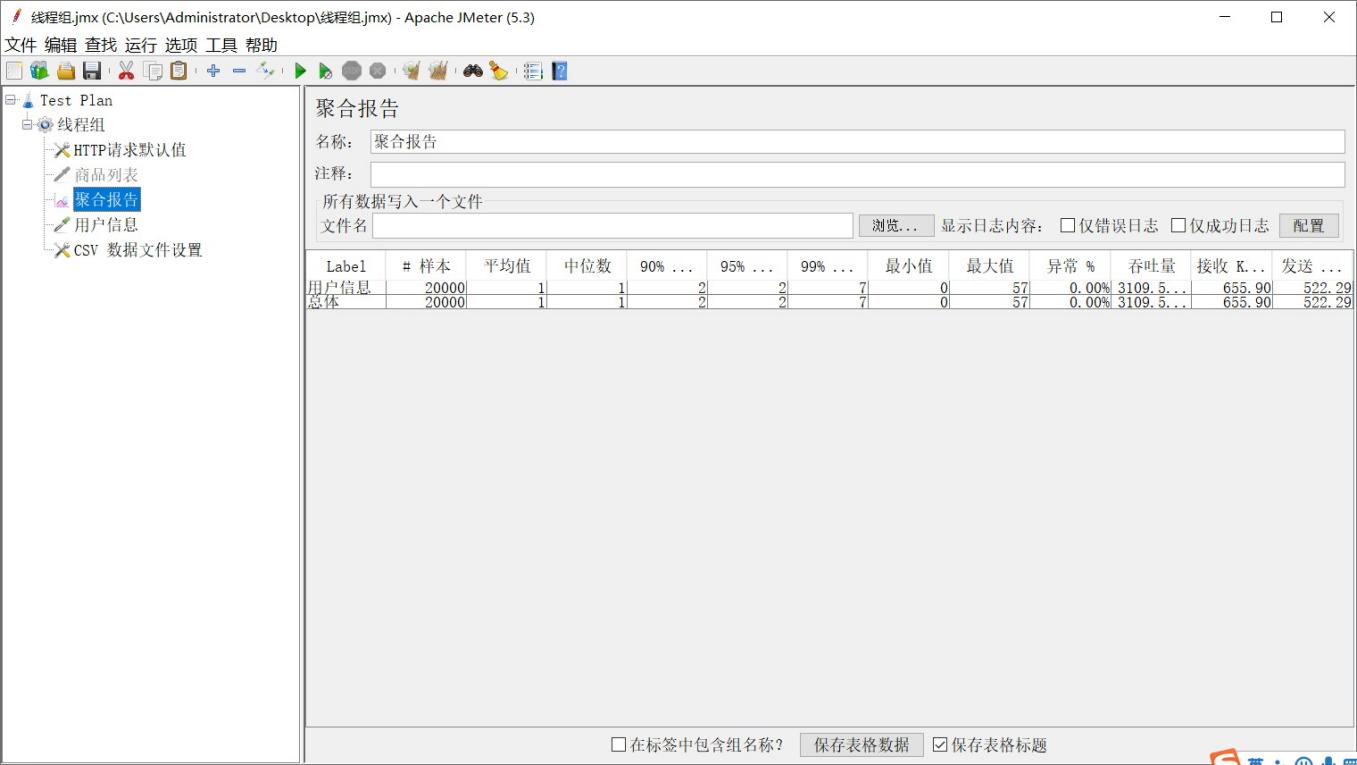
##### 添加 --> 配置元件 --> CSV Data Set Config



添加 --> 配置元件 --> HTTP Cookie管理器

修改HTTP请求用户信息

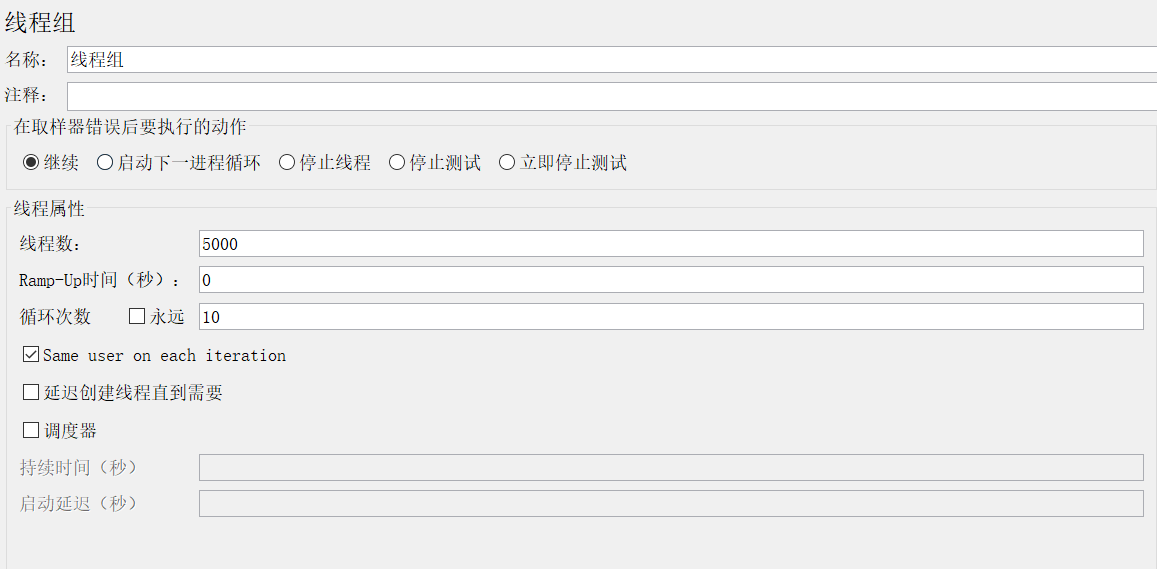


查看结果

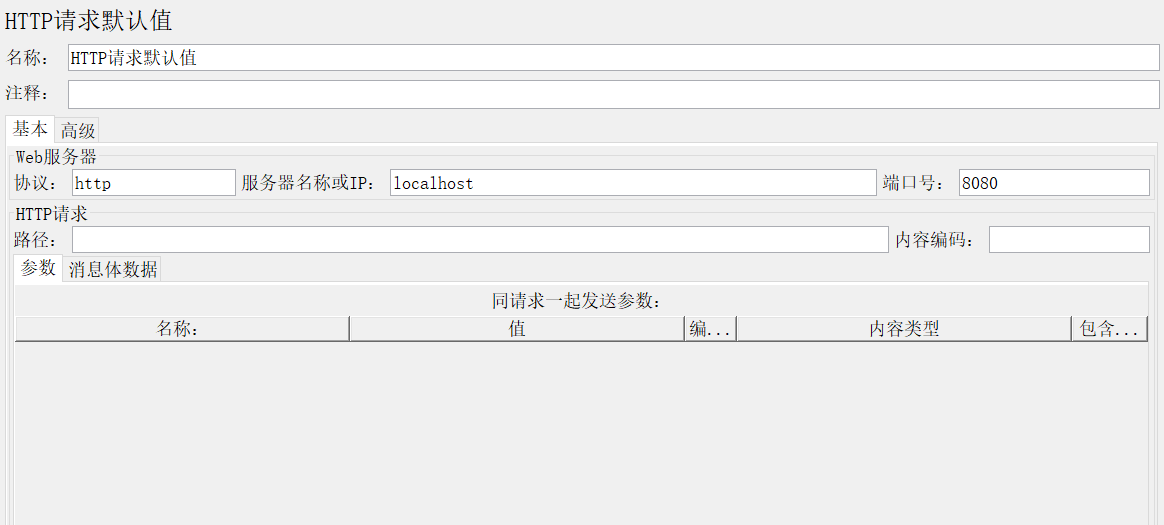
**正式压测**

**压测商品列表接口**

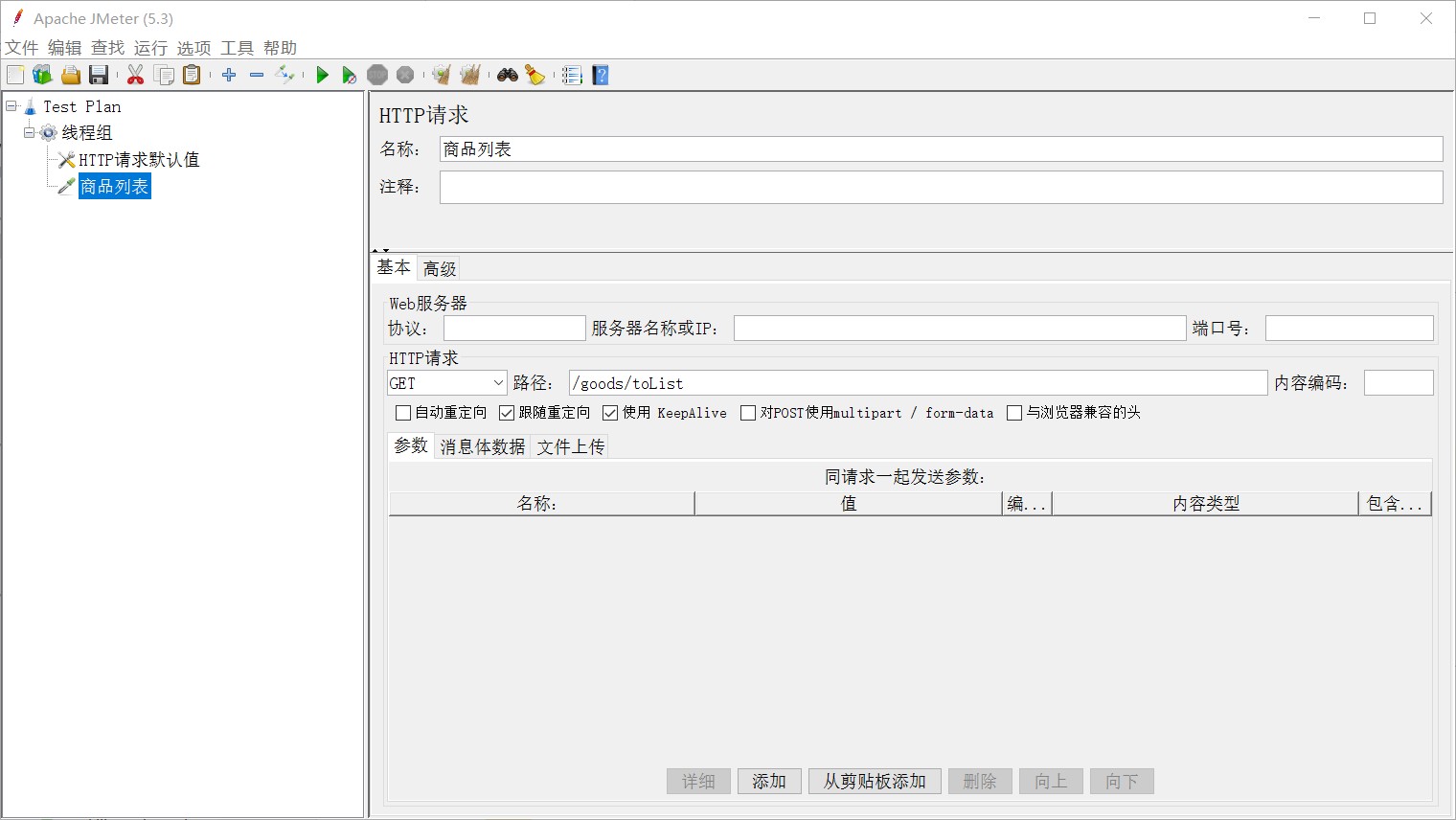
准备5000个线程，循环10次。压测商品列表接口，测试3次，查看结果。线程组



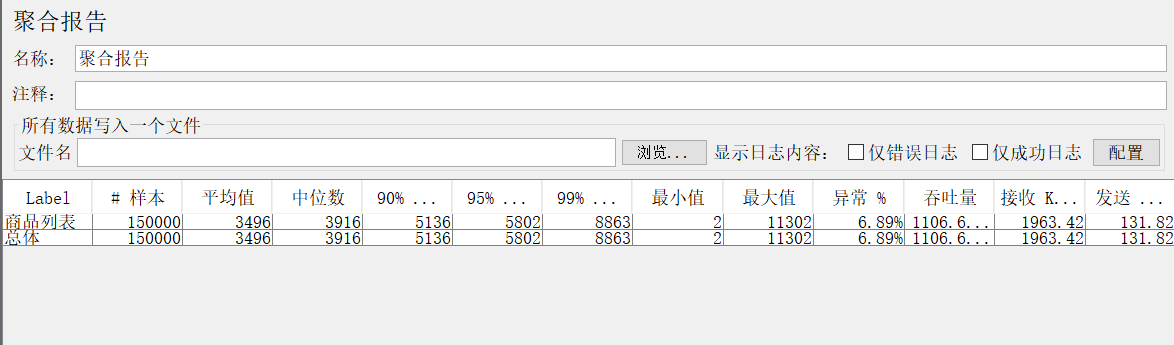
HTTP请求默认值



HTTP请求



结果



**压测秒杀接口**

**创建用户**

使用工具类往数据库插入5000用户，并且调用登录接口获取token，写入config.txt UserUtil.java

package com.xxxx.seckill.util;

import com.fasterxml.jackson.databind.ObjectMapper; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.vo.RespBean;

import java.io.ByteArrayOutputStream; import java.io.File;

import java.io.InputStream; import java.io.OutputStream; import java.io.RandomAccessFile;

import java.net.HttpURLConnection; import java.net.URL;

import java.sql.Connection; import java.sql.DriverManager;

import java.sql.PreparedStatement; import java.sql.Timestamp;

import java.util.ArrayList; import java.util.Date; import java.util.List;

/\*\*

* 生成用户工具类

\*

* @author zhoubin

\* @since 1.0.0

\*/

public class UserUtil {

private static void createUser(int count) throws Exception { List<User> users = new ArrayList<>(count);

//生成用户

for (int i = 0; i < count; i++) { User user = new User(); user.setId(13000000000L + i); user.setLoginCount(1); user.setUsername("user" + i); user.setRegisterDate(new Date()); user.setSalt("1a2b3c");

user.setPassword(MD5Util.inputPassToDbPass("123456", user.getSalt())); users.add(user);

}

System.out.println("create user");

//插入数据库

Connection conn = getConn();

String sql = "insert into t\_user(login\_count, username, register\_date, salt, password, id)values(?,?,?,?,?,?)";

PreparedStatement pstmt = conn.prepareStatement(sql); for (int i = 0; i < users.size(); i++) {

User user = users.get(i); pstmt.setInt(1, user.getLoginCount()); pstmt.setString(2, user.getUsername());

pstmt.setTimestamp(3, new Timestamp(user.getRegisterDate().getTime())); pstmt.setString(4, user.getSalt());

pstmt.setString(5, user.getPassword()); pstmt.setLong(6, user.getId()); pstmt.addBatch();

}

pstmt.executeBatch(); pstmt.close(); conn.close();

System.out.println("insert to db");

//登录，生成token

String urlString = "http://localhost:8080/login/doLogin";

File file = new File("C:\\Users\\Administrator\\Desktop\\config.txt"); if (file.exists()) {

file.delete();

}

RandomAccessFile raf = new RandomAccessFile(file, "rw"); file.createNewFile();

raf.seek(0);

for (int i = 0; i < users.size(); i++) { User user = users.get(i);

URL url = new URL(urlString);

HttpURLConnection co = (HttpURLConnection) url.openConnection(); co.setRequestMethod("POST");

co.setDoOutput(true);

OutputStream out = co.getOutputStream();

String params = "mobile=" + user.getId() + "&password=" + MD5Util.inputPassToFormPass("123456");

out.write(params.getBytes()); out.flush();

InputStream inputStream = co.getInputStream(); ByteArrayOutputStream bout = new ByteArrayOutputStream(); byte buff[] = new byte[1024];

int len = 0;

while ((len = inputStream.read(buff)) >= 0) { bout.write(buff, 0, len);

}

inputStream.close();

bout.close();

String response = new String(bout.toByteArray()); ObjectMapper mapper = new ObjectMapper();

RespBean respBean = mapper.readValue(response, RespBean.class); String userTicket = ((String) respBean.getObj()); System.out.println("create userTicket : " + user.getId());

String row = user.getId() + "," + userTicket; raf.seek(raf.length()); raf.write(row.getBytes()); raf.write("\r\n".getBytes());

System.out.println("write to file : " + user.getId());

}

raf.close();

System.out.println("over");

}

private static Connection getConn() throws Exception { String url = "jdbc:mysql://localhost:3306/seckill?

useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai"; String username = "root";

String password = "root";

String driver = "com.mysql.cj.jdbc.Driver"; Class.forName(driver);

return DriverManager.getConnection(url, username, password);

}

public static void main(String[] args) throws Exception { createUser(5000);

}

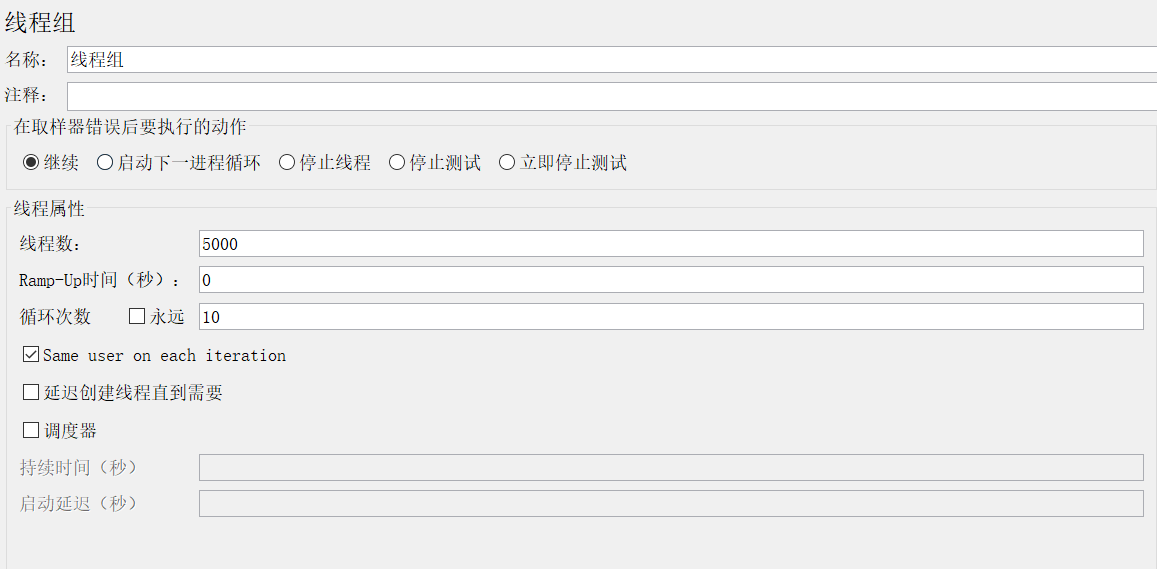
}

##### config.txt

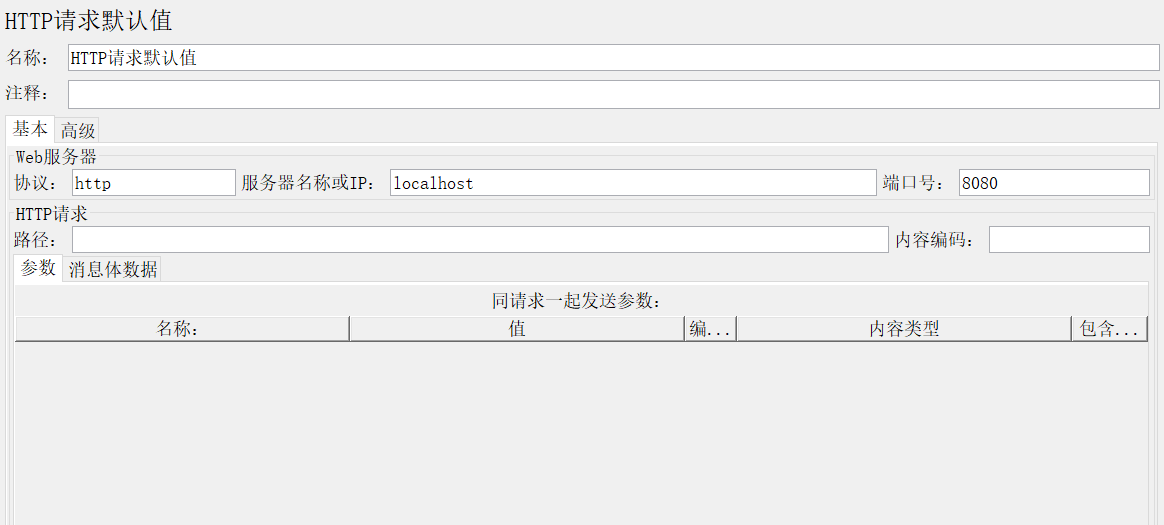


**配置秒杀接口测试**

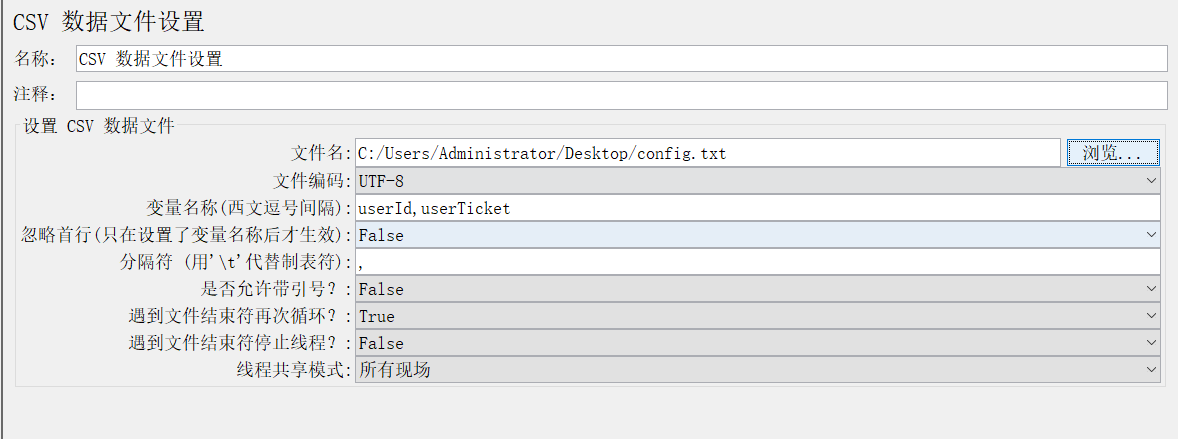
线程组



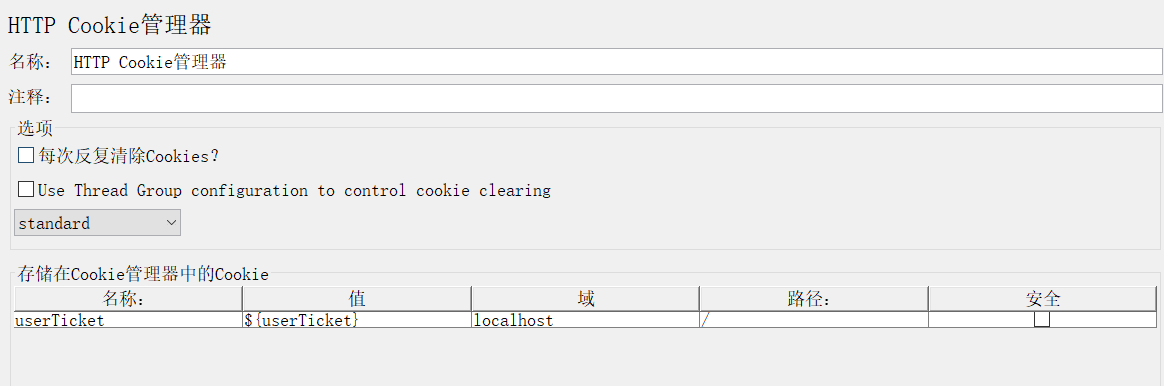
HTTP请求默认值



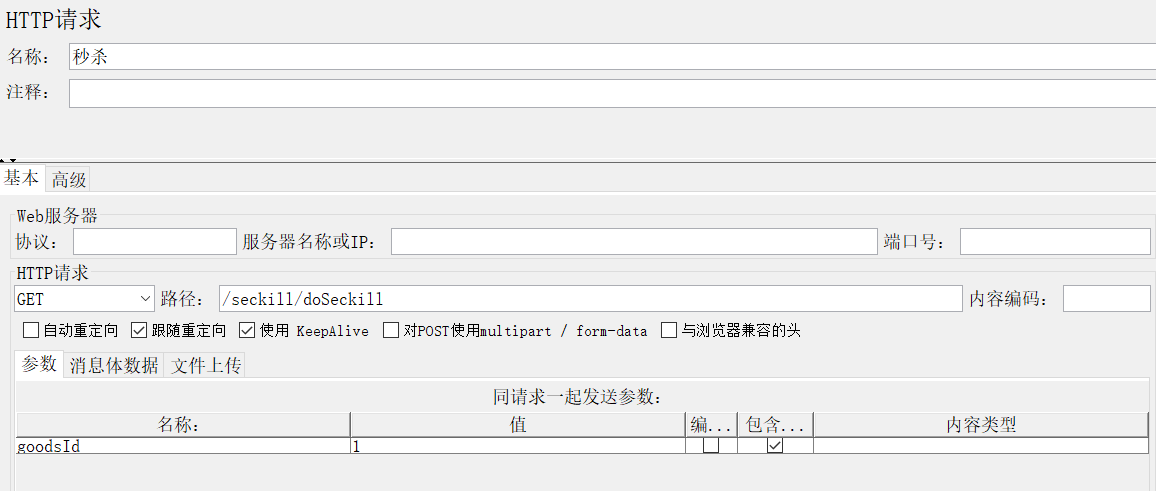
CVS数据文件设置



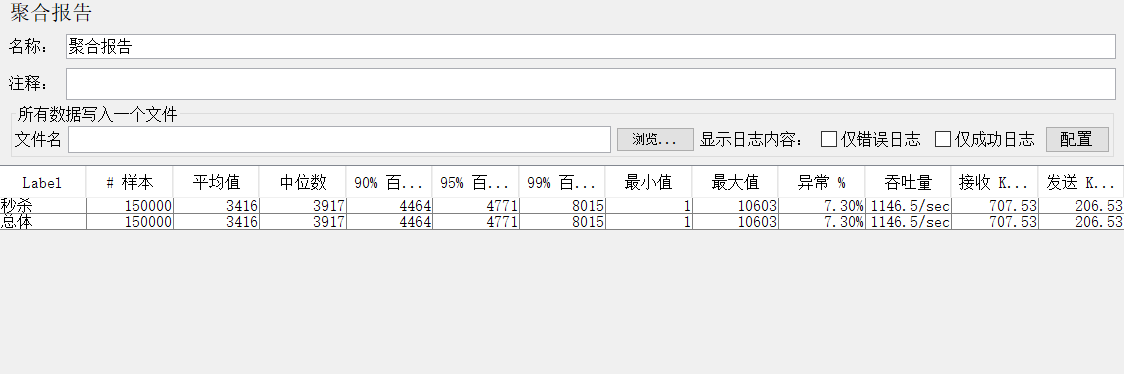
HTTP Cookie管理器



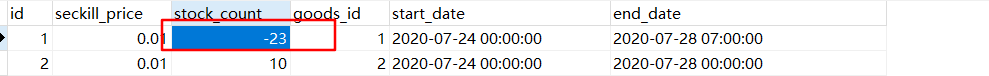
HTTP请求



结果



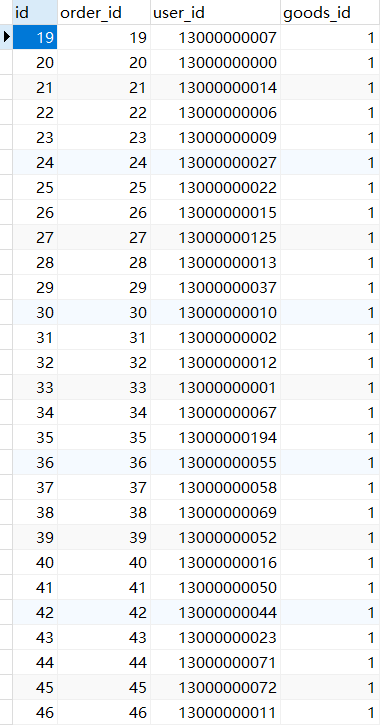
可以看出已经出现了库存超卖的情况



id use, ,d goods\_id delive,y \_add『\_id goods \_name goods \_co unt goods \_p, ice o,de 『 \_channel status c『eate\_date pay\_dat

-► 13000000007 0 iphone11 0.01 0 2020-07-27 21:;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | 13000000000 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 21 | 13000000014 | 0 iphone11 | 0 01 | 0 2020-07-27 2, 1 |
| 22 | 13000000006 | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 23 | 13000000009 | 0 iphone11 | 001 | 0 2020~07~27 2, 1 |
| 24 | 13000000027 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 25 | 13000000022 | 0 iphone11 | 001 | a 2020 -01 -21 2,1 |
| 26 | 13 0000000 1 S | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 27 | 13000000125 | 0 iphone11 | 0 01 | 0 2020-07-27 21, |
| 28 | 13000000013 | 0 iphonell | 001 | 0 2020-07-272, 1 |
| 29 | 13000000037 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 30 | 13000000010 | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 31 | 13000000002 | 0 iphone11 | 0 01 | 0 2020-07-27 2, 1 |
| 32 | 13000000012 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 33 | 13000000001 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 34 | 13000000067 | 0 iphone11 | 0 01 | 0 2020-07-27 2, 1 |
| 35 | 13000000194 | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 36 | 13000000055 | 0 iphone11 | 001 | 0 2020~07~27 2, 1 |
| 37 | 130000000 58 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 38 | 13000000069 | 0 iphone11 | 001 | a 2020 -01 -21 2,1 |
| 39 | 13 00000005 2 | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 40 | 13000000016 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 41 | 13000000050 | 0 iphonell | 001 | 0 2020-07-27 2, 1 |
| 42 | 13000000044 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 43 | 13000000023 | 0 iphone11 | 0 01 | 0 2020-07-27 21; |
| 44 | 13000000071 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 45 | 13000000072 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |
| 46 | 13000000011 | 0 iphone11 | 001 | 0 2020-07-27 2, 1 |



# 页面优化

## 缓存

### 页面缓存

##### GoodsController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.vo.GoodsVo;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.data.redis.core.ValueOperations; import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.ResponseBody; import org.thymeleaf.context.WebContext;

import org.thymeleaf.spring5.view.ThymeleafViewResolver;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.util.Date;

import java.util.concurrent.TimeUnit;

/\*\*

* 商品

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/goods") public class GoodsController {

@Autowired

private IGoodsService goodsService;

@Autowired

private RedisTemplate redisTemplate;

@Autowired

private ThymeleafViewResolver thymeleafViewResolver;

/\*\*

* + 跳转商品列表页

\*

* + @return

\*/

@RequestMapping(value = "/toList", produces = "text/html;charset=utf-8")

@ResponseBody

public String toLogin(HttpServletRequest request, HttpServletResponse response, Model model, User user) {

ValueOperations valueOperations = redisTemplate.opsForValue();

//Redis中获取页面，如果不为空，直接返回页面

String html = (String) valueOperations.get("goodsList"); if (!StringUtils.isEmpty(html)) {

return html;

}

model.addAttribute("user", user); model.addAttribute("goodsList", goodsService.findGoodsVo());

// return "goodsList";

//如果为空，手动渲染，存入Redis并返回

WebContext context = new WebContext(request, response, request.getServletContext(), request.getLocale(),

model.asMap());

html = thymeleafViewResolver.getTemplateEngine().process("goodsList", context);

if (!StringUtils.isEmpty(html)) { valueOperations.set("goodsList", html, 60, TimeUnit.SECONDS);

}

return html;

}

/\*\*

* + 跳转商品详情页

\*

* + @param model
  + @param user
  + @param goodsId
  + @return

\*/

@RequestMapping(value = "/toDetail/{goodsId}", produces = "text/html;charset=utf-8")

@ResponseBody

public String toDetail(HttpServletRequest request, HttpServletResponse response, Model model, User user,

@PathVariable Long goodsId) { ValueOperations valueOperations = redisTemplate.opsForValue();

//Redis中获取页面，如果不为空，直接返回页面

String html = (String) valueOperations.get("goodsDetail:" + goodsId); if (!StringUtils.isEmpty(html)) {

return html;

1000);

}

model.addAttribute("user", user);

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId); model.addAttribute("goods", goods);

Date startDate = goods.getStartDate(); Date endDate = goods.getEndDate(); Date nowDate = new Date();

//秒杀状态

int secKillStatus = 0;

//剩余开始时间

int remainSeconds = 0;

//秒杀还未开始

if (nowDate.before(startDate)) {

remainSeconds = (int) ((startDate.getTime() - nowDate.getTime()) /

// 秒杀已结束

} else if (nowDate.after(endDate)) { secKillStatus = 2;

remainSeconds = -1;

// 秒杀中

} else {

secKillStatus = 1;

remainSeconds = 0;

}

model.addAttribute("secKillStatus", secKillStatus); model.addAttribute("remainSeconds", remainSeconds);

// return "goodsDetail";

//如果为空，手动渲染，存入Redis并返回

WebContext context = new WebContext(request, response, request.getServletContext(), request.getLocale(),

model.asMap());

html = thymeleafViewResolver.getTemplateEngine().process("goodsDetail", context);

if (!StringUtils.isEmpty(html)) { valueOperations.set("goodsDetail:" + goodsId, html, 60,

TimeUnit.SECONDS);

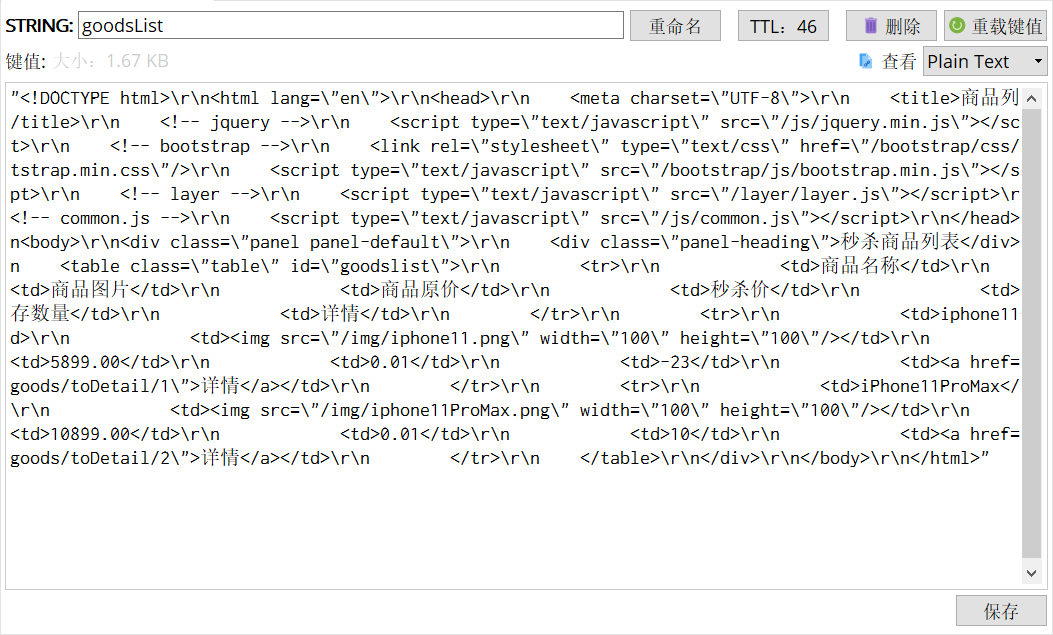
}

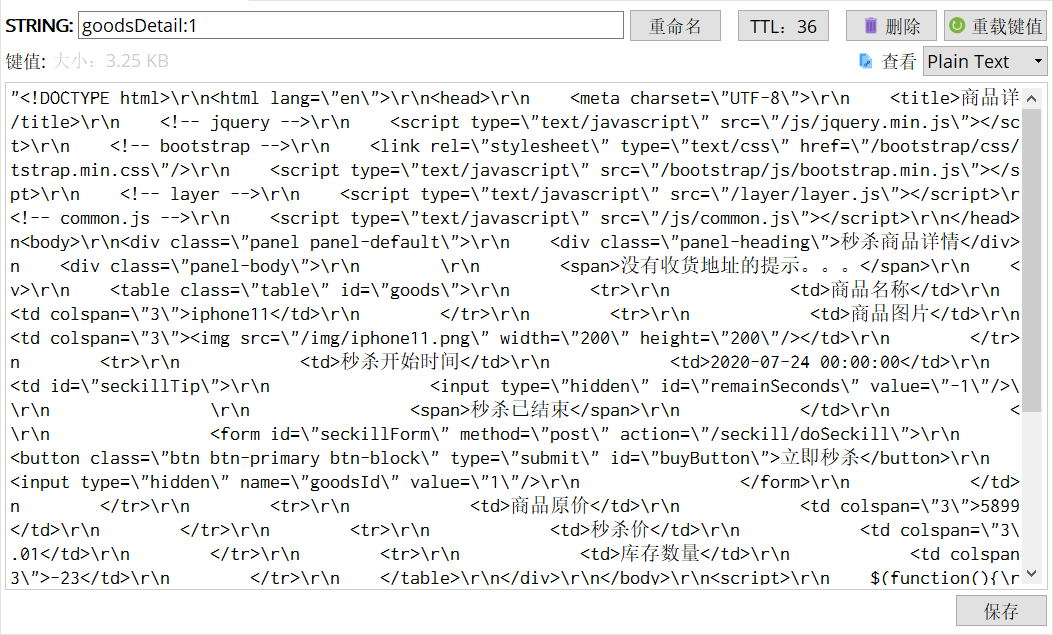
return html;

}

}

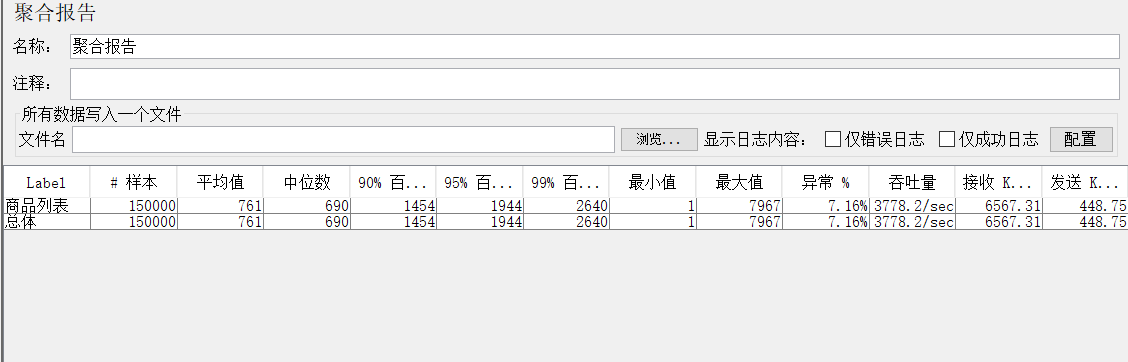
##### 重新运行项目查看效果







测试，可以发现对比之前QPS提升明显



**对象缓存**

RespBeanEnum.java

MOBILE\_NOT\_EXIST(500213, "手机号码不存在"), PASSWORD\_UPDATE\_FAIL(500214, "密码更新失败"),

##### IUserService.java

/\*\*

* 更新密码
* @param userTicket
* @param id
* @param password
* @return

\*/

RespBean updatePassword(String userTicket,Long id,String password);

##### UserServiceImpl.java

/\*\*

* 更新密码

\*

* @param userTicket
* @param id
* @param password
* @return

\*/

@Override

public RespBean updatePassword(String userTicket, Long id, String password) { User user = userMapper.selectById(id);

if (user == null) {

throw new GlobalException(RespBeanEnum.MOBILE\_NOT\_EXIST);

}

user.setPassword(MD5Util.inputPassToDbPass(password, user.getSalt())); int result = userMapper.updateById(user);

if (1 == result) {

//删除Redis

redisTemplate.delete("user:" + userTicket); return RespBean.success();

}

return RespBean.error(RespBeanEnum.PASSWORD\_UPDATE\_FAIL);

}

## 页面静态化

### 商品详情静态化

##### DetailVo.java

package com.xxxx.seckill.vo;

import com.xxxx.seckill.pojo.User; import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor public class DetailVo {

private User user; private GoodsVo goodsVo;

private int secKillStatus; private int remainSeconds;

}

##### GoodsController.java

/\*\*

* 跳转商品详情页

\*

* @param model
* @param user
* @param goodsId
* @return

\*/

@RequestMapping(value = "/detail/{goodsId}")

@ResponseBody

public RespBean toDetail(HttpServletRequest request, HttpServletResponse response, Model model, User user,

@PathVariable Long goodsId) { GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId); Date startDate = goods.getStartDate();

Date endDate = goods.getEndDate(); Date nowDate = new Date();

//秒杀状态

int secKillStatus = 0;

//剩余开始时间

int remainSeconds = 0;

//秒杀还未开始

if (nowDate.before(startDate)) {

remainSeconds = (int) ((startDate.getTime() - nowDate.getTime()) / 1000);

// 秒杀已结束

} else if (nowDate.after(endDate)) { secKillStatus = 2;

remainSeconds = -1;

// 秒杀中

} else {

secKillStatus = 1;

remainSeconds = 0;

}

DetailVo detailVo = new DetailVo(); detailVo.setGoodsVo(goods); detailVo.setUser(user); detailVo.setRemainSeconds(remainSeconds); detailVo.setSecKillStatus(secKillStatus); return RespBean.success(detailVo);

}

##### common.js

// 获取url参数

function g\_getQueryString(name) {

var reg = new RegExp("(^|&)" + name + "=([^&]\*)(&|$)"); var r = window.location.search.substr(1).match(reg); if(r != null) return unescape(r[2]);

return null;

};

//设定时间格式化函数，使用new Date().format("yyyy-MM-dd HH:mm:ss"); Date.prototype.format = function (format) {

var args = {

"M+": this.getMonth() + 1,

"d+": this.getDate(),

"H+": this.getHours(),

"m+": this.getMinutes(),

"s+": this.getSeconds(),

};

if (/(y+)/.test(format))

format = format.replace(RegExp.$1, (this.getFullYear() + "").substr(4 - RegExp.$1.length));

for (var i in args) { var n = args[i];

if (new RegExp("(" + i + ")").test(format))

format = format.replace(RegExp.$1, RegExp.$1.length == 1 ? n : ("00" + n).substr(("" + n).length));

}

return format;

};

##### goodsDetail.htm

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>商品详情</title>

<!-- jquery -->

<script type="text/javascript" src="/js/jquery.min.js"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" href="/bootstrap/css/bootstrap.min.css"/>

<script type="text/javascript" src="/bootstrap/js/bootstrap.min.js">

</script>

<!-- layer -->

<script type="text/javascript" src="/layer/layer.js"></script>

<!-- common.js -->

<script type="text/javascript" src="/js/common.js"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀商品详情</div>

<div class="panel-body">

<span id="userTip"> 您还没有登录，请登陆后再操作<br/></span>

<span>没有收货地址的提示。。。</span>

</div>

<table class="table" id="goods">

<tr>

<td>商品名称</td>

<td colspan="3" id="goodsName"></td>

</tr>

<tr>

<td>商品图片</td>

<td colspan="3"><img id="goodsImg" width="200" height="200"/></td>

</tr>

<tr>

<td>秒杀开始时间</td>

<td id="startTime"></td>

<td>

id="countDown"

</span>秒</span>

<input type="hidden" id="remainSeconds"/>

<!-- <span if="secKillStatus eq 0">秒杀倒计时：<span

text="remainSeconds">

<span if="secKillStatus eq 1">秒杀进行中</span>

<span if="secKillStatus eq 2">秒杀已结束</span>-->

<span id="seckillTip"></span>

</td>

<td>

<form id="seckillForm" method="post" action="/seckill/doSeckill">

<button class="btn btn-primary btn-block" type="submit"

id="buyButton">立即秒杀</button>

<input type="hidden" name="goodsId" id="goodsId"/>

</form>

</td>

</tr>

<tr>

<td>商品原价</td>

<td colspan="3" id="goodsPrice"></td>

</tr>

<tr>

<td>秒杀价</td>

<td colspan="3" id="seckillPrice"></td>

</tr>

<tr>

<td>库存数量</td>

<td colspan="3" id="stockCount"></td>

</tr>

</table>

</div>

</body>

<script>

$(function () {

// countDown(); getDetails();

}

);

function getDetails() {

var goodsId = g\_getQueryString("goodsId");

$.ajax({

url: "/goods/detail/" + goodsId, type: "GET",

success: function (data) { if (data.code == 200) {

render(data.obj);

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误");

}

})

}

function render(detail) { var user = detail.user;

var goods = detail.goodsVo;

var remainSeconds = detail.remainSeconds; if (user) {

$("#userTip").hide();

}

$("#goodsName").text(goods.goodsName);

$("#goodsImg").attr("src", goods.goodsImg);

$("#startTime").text(new Date(goods.startDate).format("yyyy-MM-dd HH:mm:ss"));

$("#remainSeconds").val(remainSeconds);

$("#goodsId").val(goods.id);

$("#goodsPrice").text(goods.goodsPrice);

$("#seckillPrice").text(goods.seckillPrice);

$("#stockCount").text(goods.stockCount); countDown();

}

function countDown() {

var remainSeconds = $("#remainSeconds").val(); var timeout;

//秒杀还没开始，倒计时

if (remainSeconds > 0) {

$("#buyButton").attr("disabled", true);

$("#seckillTip").html("秒杀倒计时：" + remainSeconds + "秒"); timeout = setTimeout(function () {

// $("#countDown").text(remainSeconds - 1);

$("#remainSeconds").val(remainSeconds - 1); countDown();

}, 1000

);

}

//秒杀进行中

else if (remainSeconds == 0) {

$("#buyButton").attr("disabled", false); if (timeout) {

clearTimeout(timeout);

}

$("#seckillTip").html("秒杀进行中");

//秒杀已经结束

} else {

$("#buyButton").attr("disabled", true);

$("#seckillTip").html("秒杀已经结束");

}

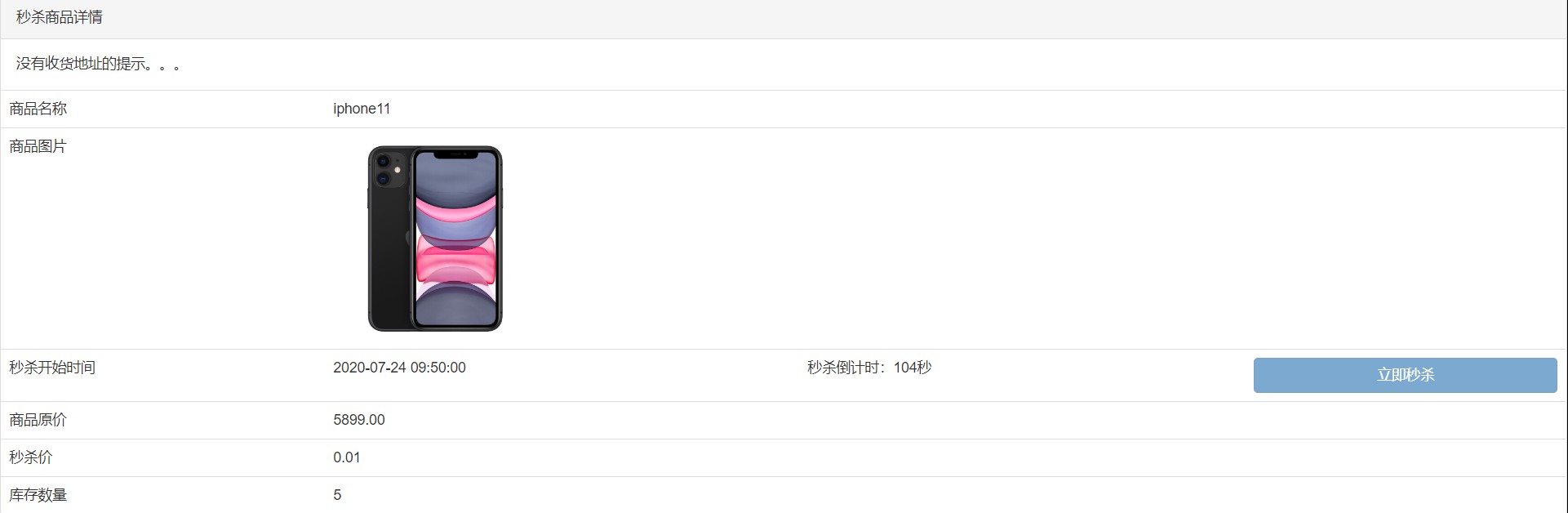
}

</script>

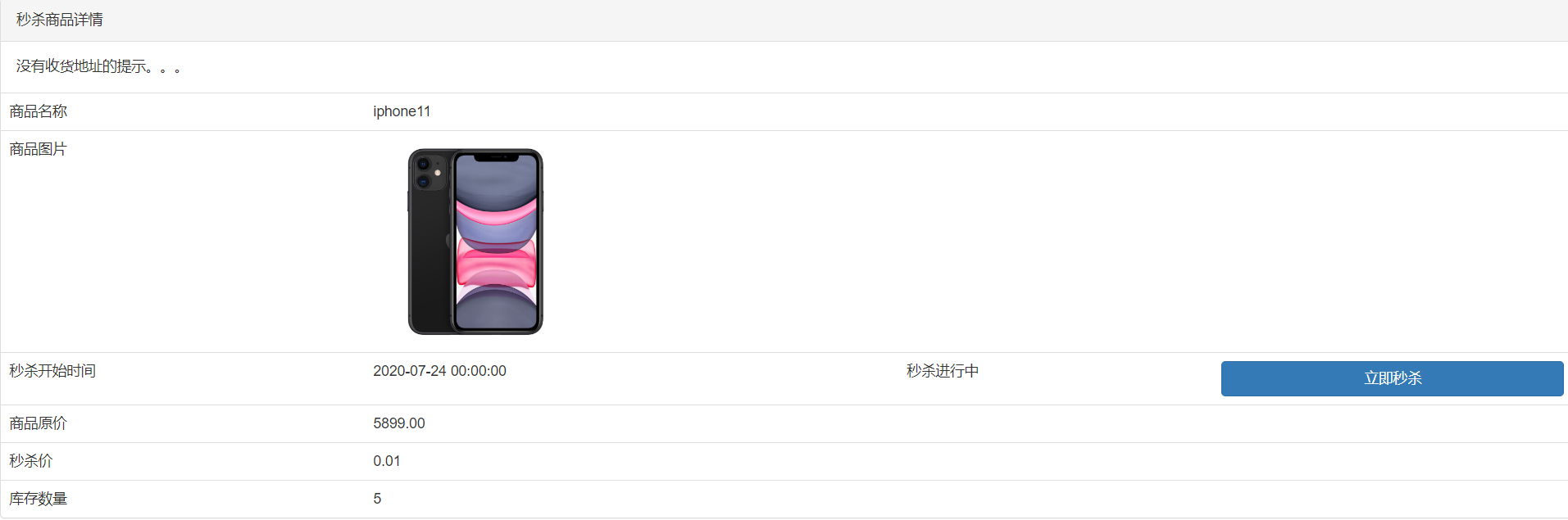
</html>

**测试**

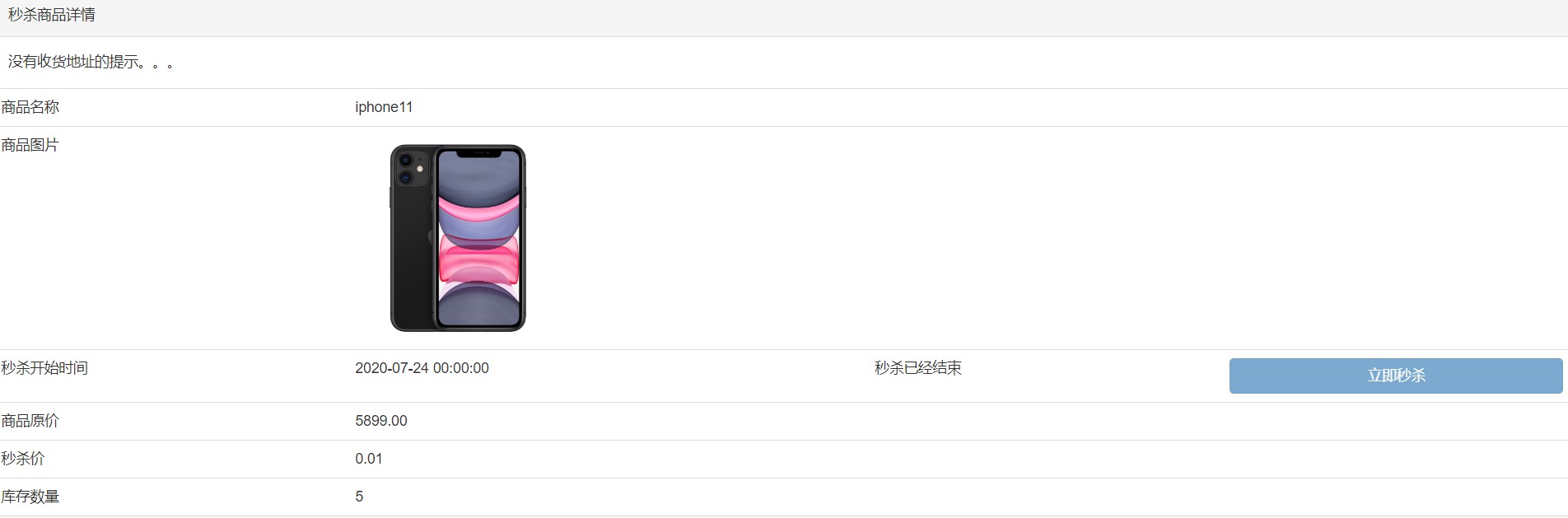
##### 秒杀未开始



秒杀进行中



秒杀已结束



**秒杀静态化**

SeckillController.java

package com.xxxx.seckill.controller;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.xxxx.seckill.pojo.Order;

import com.xxxx.seckill.pojo.SeckillOrder; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.vo.GoodsVo;

import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill") public class SeckillController {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@RequestMapping(value = "/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//判断是否重复抢购

SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id",

user.getId()).eq( "goods\_id", goodsId));

if (seckillOrder != null) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

Order order = orderService.seckill(user, goods); return RespBean.success(order);

}

}

##### goodsDetail.htm

<td>

<!--<form id="seckillForm" method="post" action="/seckill/doSeckill">

<button class="btn btn-primary btn-block" type="submit" id="buyButton">

立即秒杀</button>

<input type="hidden" name="goodsId" id="goodsId"/>

</form>-->

<button class="btn btn-primary btn-block" type="button" id="buyButton"

onclick="doSeckill()">立即秒杀

</button>

<input type="hidden" name="goodsId" id="goodsId"/>

</td>

<script>

function doSeckill() {

$.ajax({

url: "/seckill/doSeckill", type: "POST",

data: {

goodsId: $("#goodsId").val(),

data.obj.id;

},

success: function (data) { if (data.code == 200) {

window.location.href = "/orderDetail.htm?orderId=" +

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误");

}

})

}

</script>

##### orderDetail.htm

<!DOCTYPE HTML>

<html>

<head>

<title>订单详情</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<!-- jquery -->

<script type="text/javascript" src="/js/jquery.min.js"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" href="/bootstrap/css/bootstrap.min.css" />

<script type="text/javascript" src="/bootstrap/js/bootstrap.min.js">

</script>

<!-- layer -->

<script type="text/javascript" src="/layer/layer.js"></script>

<!-- common.js -->

<script type="text/javascript" src="/js/common.js"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀订单详情</div>

<table class="table" id="order">

<tr>

<td>商品名称</td>

<td id="goodsName" colspan="3"></td>

</tr>

<tr>

<td>商品图片</td>

<td colspan="2"><img id="goodsImg" width="200" height="200" /></td>

</tr>

<tr>

<td>订单价格</td>

<td colspan="2" id="goodsPrice"></td>

</tr>

<tr>

<td>下单时间</td>

<td id="createDate" colspan="2"></td>

</tr>

<tr>

<td>订单状态</td>

<td id="status">

<!--<span if="order.status eq 0">未支付</span>

<span if="order.status eq 1">待发货</span>

<span if="order.status eq 2">已发货</span>

<span if="order.status eq 3">已收货</span>

<span if="order.status eq 4">已退款</span>

<span if="order.status eq 5">已完成</span>-->

</td>

<td>

<button class="btn btn-primary btn-block" type="submit"

id="payButton">立即支付</button>

</td>

</tr>

<tr>

<td>收货人</td>

<td colspan="2">XXX 18012345678</td>

</tr>

<tr>

<td>收货地址</td>

<td colspan="2">上海市浦东区世纪大道</td>

</tr>

</table>

</div>

</body>

</html>

##### applictaion.yml

spring:

#静态资源处理resources:

#启用默认静态资源处理，默认启用add-mappings: true

cache:

cachecontrol:

#缓存响应时间，单位秒max-age: 3600

chain:

#资源链中启用缓存，默认启用cache: true

#启用资源链，默认禁用enabled: true

#启用压缩资源(gzip,brotli)解析,默认禁用compressed: true

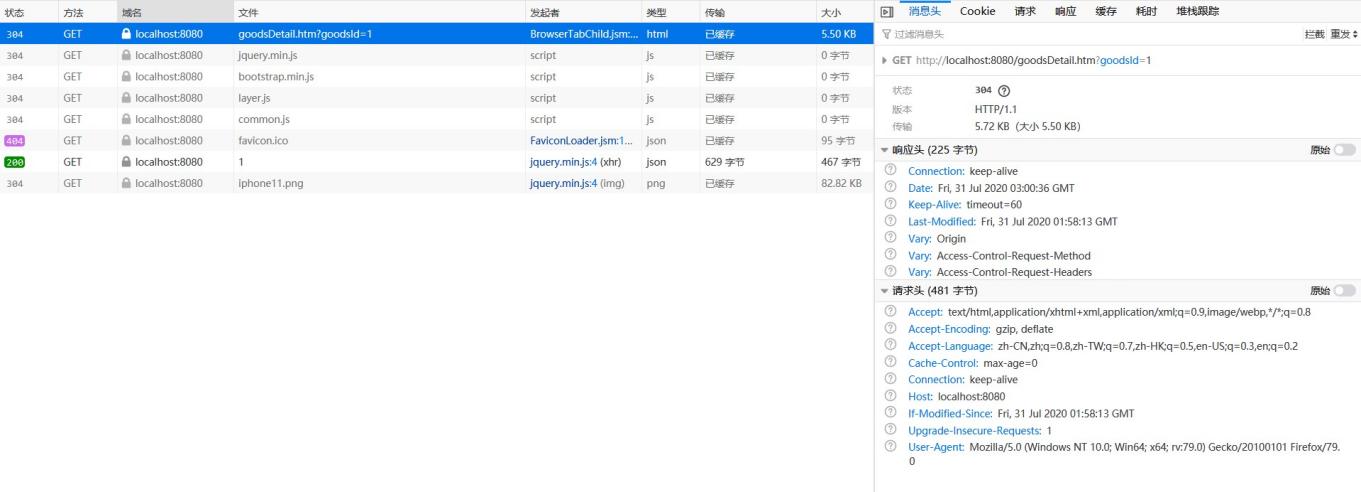
#启用H5应用缓存，默认禁用

html-application-cache: true

#静态资源位置

static-locations: classpath:/static/

#### 测试



**订单详情静态化**

**订单详情静态化**

##### OrderController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IOrderService; import com.xxxx.seckill.vo.OrderDetailVo; import com.xxxx.seckill.vo.RespBean;

import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/order") public class OrderController {

@Autowired

private IOrderService orderService;

/\*\*

* + 订单详情
  + @param user
  + @param orderId
  + @return

\*/

@RequestMapping("/detail")

@ResponseBody

public RespBean detail(User user,Long orderId){ if (null==user){

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

OrderDetailVo detail = orderService.detail(orderId); return RespBean.success(detail);

}

}

##### IOrderService.java

/\*\*

* 订单详情
* @param orderId
* @return

\*/

OrderDetailVo detail(Long orderId);

##### OrderServiceImpl.java

/\*\*

* 订单详情
* @param orderId
* @return

\*/

@Override

public OrderDetailVo detail(Long orderId) { if (null==orderId){

throw new GlobalException(RespBeanEnum.ORDER\_NOT\_EXIST);

}

Order order = orderMapper.selectById(orderId);

GoodsVo goodsVo = goodsService.findGoodsVoByGoodsId(order.getGoodsId()); OrderDetailVo detail = new OrderDetailVo();

detail.setGoodsVo(goodsVo); detail.setOrder(order); return detail;

}

##### OrderDetailVo.java

package com.xxxx.seckill.vo;

import com.xxxx.seckill.pojo.Order; import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

public class OrderDetailVo { private Order order; private GoodsVo goodsVo;

}

##### orderDetail.htm

<!DOCTYPE HTML>

<html>

<head>

<title>订单详情</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<!-- jquery -->

<script type="text/javascript" src="/js/jquery.min.js"></script>

<!-- bootstrap -->

<link rel="stylesheet" type="text/css" href="/bootstrap/css/bootstrap.min.css"/>

<script type="text/javascript" src="/bootstrap/js/bootstrap.min.js">

</script>

<!-- layer -->

<script type="text/javascript" src="/layer/layer.js"></script>

<!-- common.js -->

<script type="text/javascript" src="/js/common.js"></script>

</head>

<body>

<div class="panel panel-default">

<div class="panel-heading">秒杀订单详情</div>

<table class="table" id="order">

<tr>

<td>商品名称</td>

<td id="goodsName" colspan="3"></td>

</tr>

<tr>

<td>商品图片</td>

<td colspan="2"><img id="goodsImg" width="200" height="200"/></td>

</tr>

<tr>

<td>订单价格</td>

<td colspan="2" id="goodsPrice"></td>

</tr>

<tr>

<td>下单时间</td>

<td id="createDate" colspan="2"></td>

</tr>

<tr>

<td>订单状态</td>

<td id="status">

<!--<span if="order.status eq 0">未支付</span>

<span if="order.status eq 1">待发货</span>

<span if="order.status eq 2">已发货</span>

<span if="order.status eq 3">已收货</span>

<span if="order.status eq 4">已退款</span>

<span if="order.status eq 5">已完成</span>-->

</td>

<td>

<button class="btn btn-primary btn-block" type="submit"

id="payButton">立即支付</button>

</td>

</tr>

<tr>

<td>收货人</td>

<td colspan="2">XXX 18012345678</td>

</tr>

<tr>

<td>收货地址</td>

<td colspan="2">上海市浦东区世纪大道</td>

</tr>

</table>

</div>

<script>

$(function () { getOrderDetail();

});

function getOrderDetail() {

var orderId = g\_getQueryString("orderId");

$.ajax({

url: "/order/detail", type: "GET",

data: {

orderId: orderId

},

success: function (data) { if (data.code == 200) {

render(data.obj);

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误")

}

})

}

function render(detail) {

var goods = detail.goodsVo; var order = detail.order;

$("#goodsName").text(goods.goodsName);

$("#goodsImg").attr("src", goods.goodsImg);

$("#goodsPrice").text(order.goodsPrice);

$("#createDate").text(new Date(order.createDate).format("yyyy-MM-dd HH:mm:ss"));

var status = order.status; var statusText = "" switch (status) {

case 0:

statusText = "未支付"; break;

case 1:

statusText = "待发货"; break;

case 2:

statusText = "已发货"; break;

case 3:

statusText = "已收货"; break;

case 4:

statusText = "已退款"; break;

case 5:

statusText = "已完成"; break;

}

$("#status").text(statusText);

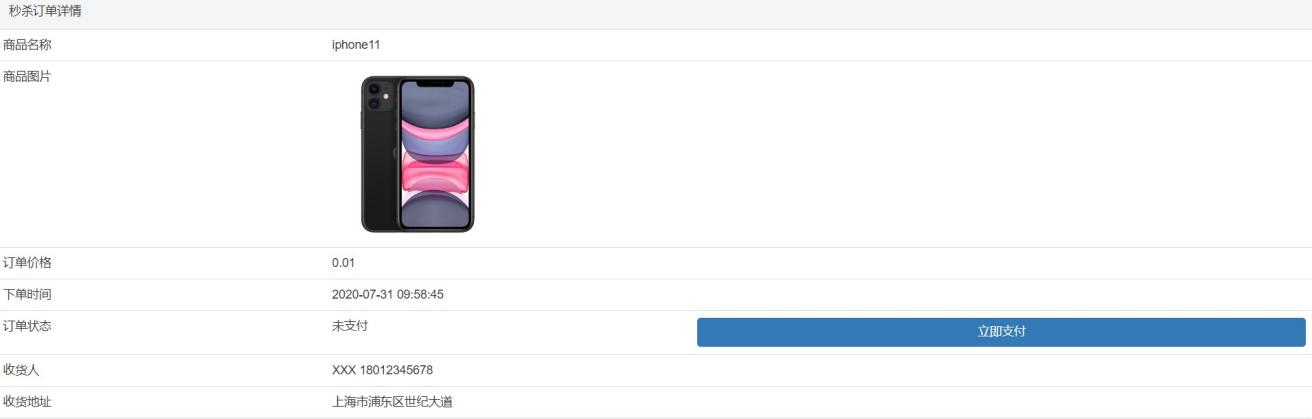
}

</script>

</body>

</html>

##### 效果



**解决库存超卖**

减库存时判断库存是否足够

OrderServiceImpl.java

//秒杀商品表减库存

SeckillGoods seckillGoods = seckillGoodsService.getOne(new QueryWrapper<SeckillGoods>().eq("goods\_id",

goods.getId())); seckillGoods.setStockCount(seckillGoods.getStockCount() - 1);

seckillGoodsService.update(new UpdateWrapper<SeckillGoods>().set("stock\_count", seckillGoods.getStockCount()).eq("id", seckillGoods.getId()).gt("stock\_count", 0));

// seckillGoodsService.updateById(seckillGoods);

##### 解决同一用户同时秒杀多件商品。可以通过数据库建立唯一索引避免



将秒杀订单信息存入Redis，方便判断是否重复抢购时进行查询

OrderServiceImpl.java

/\*\*

* 秒杀

\*

* @param user
* @param goods
* @return

\*/

@Override

@Transactional

public Order seckill(User user, GoodsVo goods) {

//秒杀商品表减库存

SeckillGoods seckillGoods = seckillGoodsService.getOne(new QueryWrapper<SeckillGoods>().eq("goods\_id",

goods.getId())); seckillGoods.setStockCount(seckillGoods.getStockCount() - 1); boolean seckillGoodsResult = seckillGoodsService.update(new

UpdateWrapper<SeckillGoods>().set("stock\_count",

seckillGoods.getStockCount()).eq("id", seckillGoods.getId()).gt("stock\_count", 0));

// seckillGoodsService.updateById(seckillGoods); if (!(goodsResult&&seckillGoodsResult)){

return null;

}

//生成订单

Order order = new Order(); order.setUserId(user.getId()); order.setGoodsId(goods.getId()); order.setDeliveryAddrId(0L); order.setGoodsName(goods.getGoodsName()); order.setGoodsCount(1); order.setGoodsPrice(seckillGoods.getSeckillPrice()); order.setOrderChannel(1);

order.setStatus(0); order.setCreateDate(new Date()); orderMapper.insert(order);

//生成秒杀订单

SeckillOrder seckillOrder = new SeckillOrder(); seckillOrder.setOrderId(order.getId()); seckillOrder.setUserId(user.getId()); seckillOrder.setGoodsId(goods.getId()); seckillOrderService.save(seckillOrder); redisTemplate.opsForValue().set("order:" + user.getId() + ":" +

goods.getId(),

JsonUtil.object2JsonStr(seckillOrder)); return order;

}

##### seckillController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.Order; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.vo.GoodsVo;

import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.stereotype.Controller;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill") public class SeckillController {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@Autowired

private RedisTemplate redisTemplate;

@RequestMapping(value = "/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//判断是否重复抢购

// SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id",

// user.getId()).eq(

// "goods\_id",

// goodsId));

String seckillOrderJson = (String) redisTemplate.opsForValue().get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

Order order = orderService.seckill(user, goods); if (null != order) {

return RespBean.success(order);

}

return RespBean.error(RespBeanEnum.ERROR);

}

}

##### SeckillOrder.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.Order; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService;

import com.xxxx.seckill.service.IOrderService; import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.vo.GoodsVo;

import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.stereotype.Controller;

import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill") public class SeckillController {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@Autowired

private RedisTemplate redisTemplate;

@RequestMapping(value = "/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//判断是否重复抢购

// SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id",

// user.getId()).eq(

// "goods\_id",

// goodsId));

String seckillOrderJson = (String) redisTemplate.opsForValue().get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

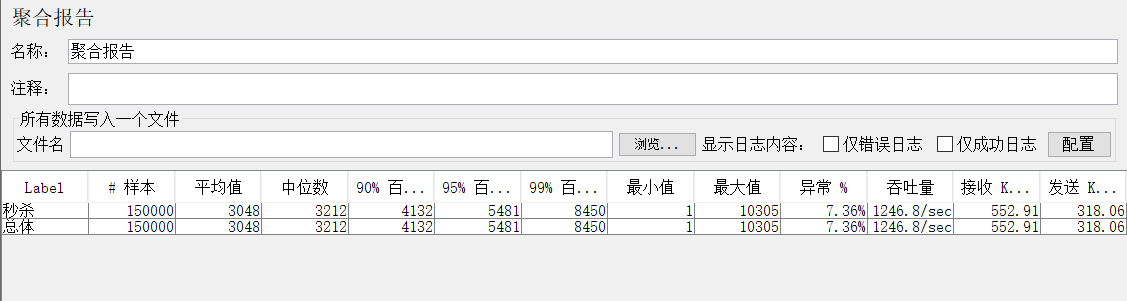
Order order = orderService.seckill(user, goods); return RespBean.success(order);

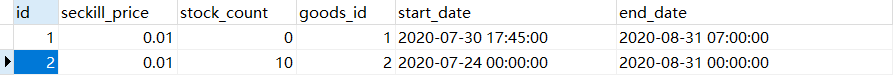
}

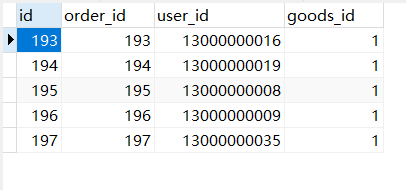
}

#### 测试

##### QPS提升并不明显，重点在于是否出现库存超卖现象







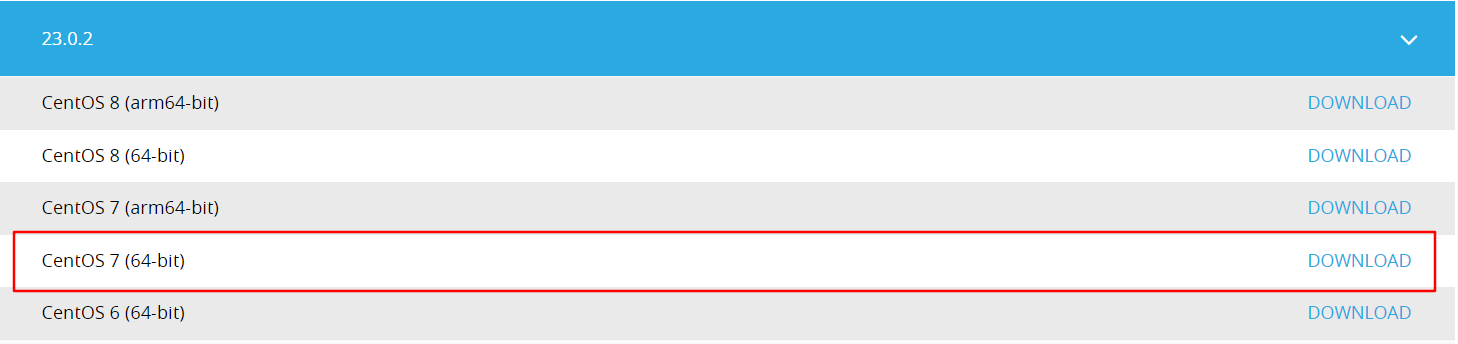
**服务优化**

**RabbitMQ入门**

**安装**

**安装Erlang**

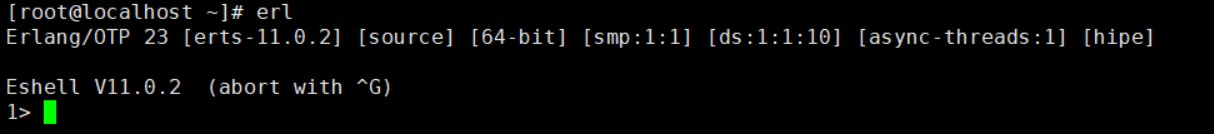
官网提示：<https://www.erlang-solutions.com/resources/download.html>



安装erlang

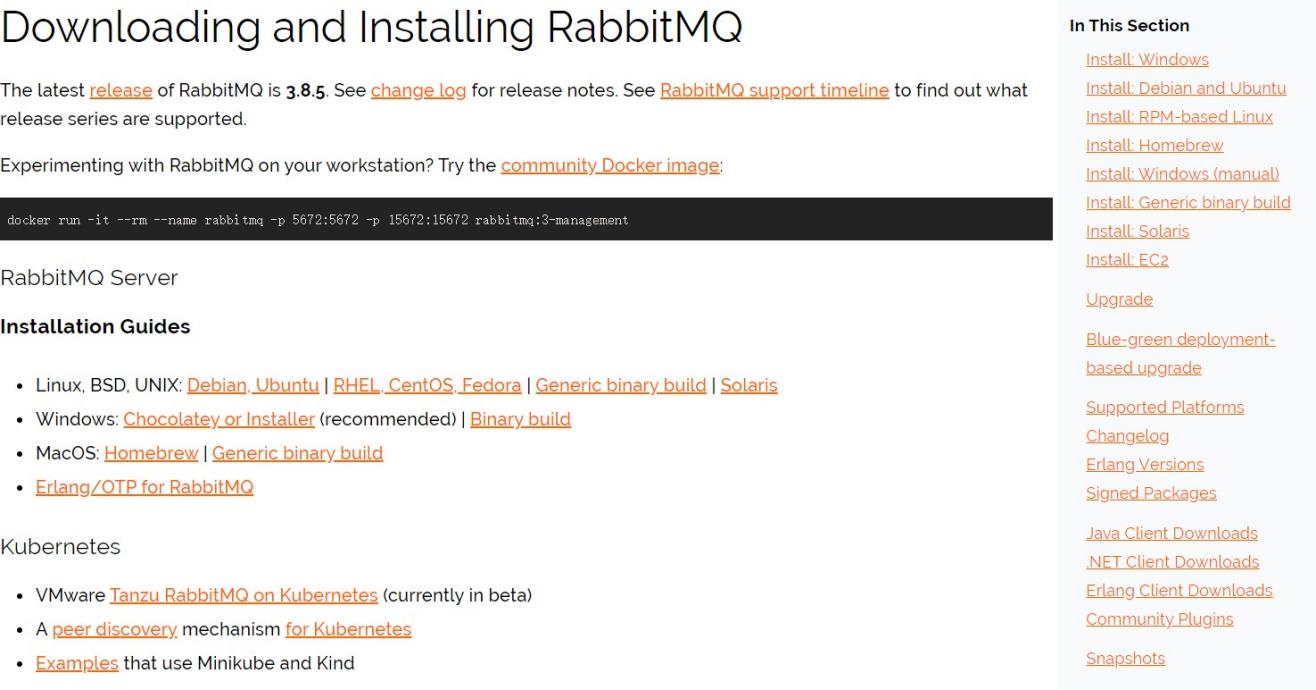
yum -y install esl-erlang\_23.0.2-1\_centos\_7\_amd64.rpm

##### 检测erlang



**安装RabbitMQ**

官网下载地址：<http://www.rabbitmq.com/download.html>

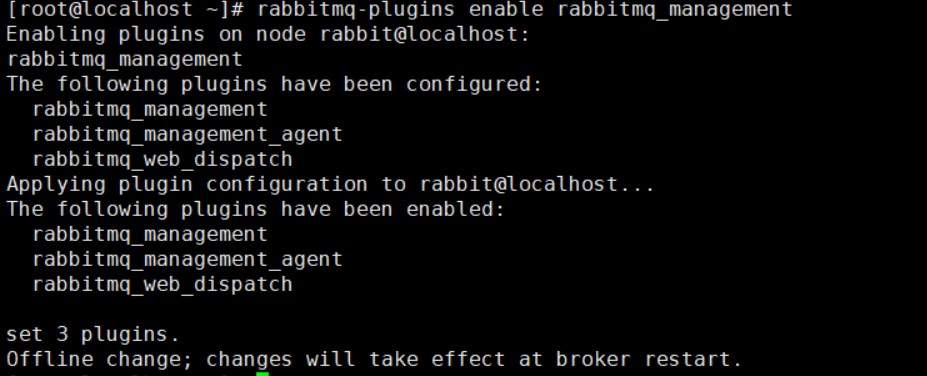


安装rabbitmq

yum -y install rabbitmq-server-3.8.5-1.el7.noarch.rpm

##### 安装UI插件

rabbitmq-plugins enable rabbitmq\_management

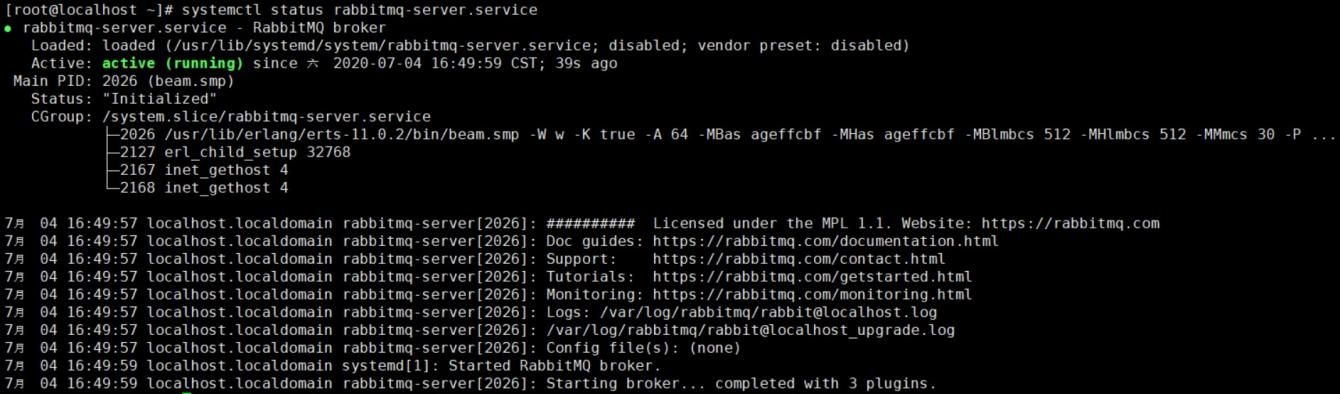


##### 启用rabbitmq服务

systemctl start rabbitmq-server.service

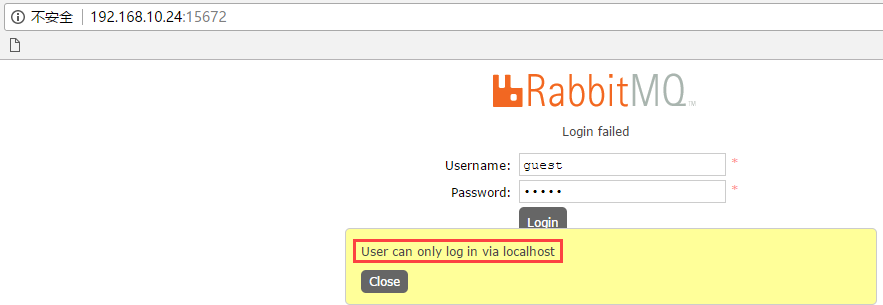
##### 检测服务

systemctl status rabbitmq-server.service



##### 访问

guest用户默认只可以localhost(本机)访问



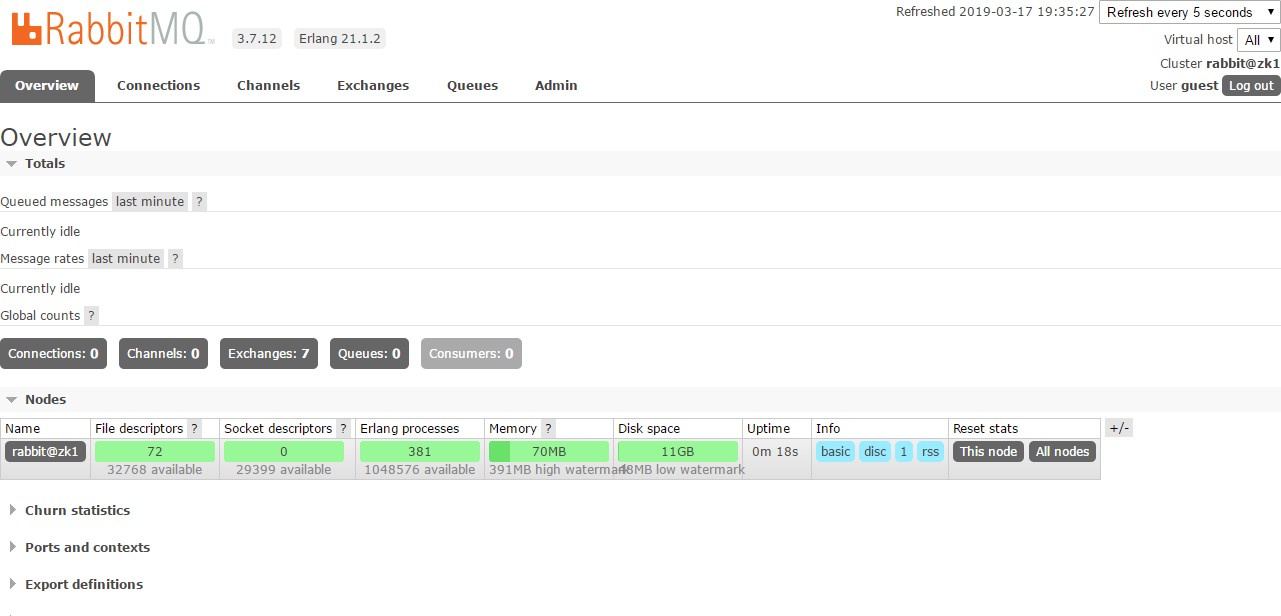
在rabbitmq的配置文件目录下（默认为：/etc/rabbitmq）创建一个**rabbitmq.config**文件。 文件中添加如下配置（请不要忘记那个“**.**”）：

[{rabbit, [{loopback\_users, []}]}].

##### 重启rabbitmq服务

systemctl restart rabbitmq-server.service

##### 重新访问



**使用**

依赖

<!-- AMQP依赖 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-amqp</artifactId>

</dependency>

##### 配置

application.yml

spring:

#RabbitMQ rabbitmq:

#服务器地址

host: 192.168.10.100

#用户名username: guest

#密码

password: guest

#虚拟主机virtual-host: /

# 端 口 port: 5672 listener:

simple:

#消费者最小数量concurrency: 10

#消费者最大数量

max-concurrency: 10

#限制消费者每次只处理一条消息，处理完再继续下一条消息prefetch: 1

#启动时是否默认启动容器，默认true auto-startup: true

#被拒绝时重新进入队列

default-requeue-rejected: true template:

retry:

#发布重试，默认false enabled: true

#重试时间 默认1000ms initial-interval: 1000

#重试最大次数，默认3次max-attempts: 3

#重试最大间隔时间，默认10000ms max-interval: 10000

#重试间隔的乘数。比如配2.0 第一次等10s，第二次等20s，第三次等40s multiplier: 1.0

##### RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Queue;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

@Bean

public Queue queue(){

return new Queue("queue",true);

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void send(Object msg) {

log.info(" 发 送 消 息 ："+msg); rabbitTemplate.convertAndSend("queue", msg);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@RabbitListener(queues = "queue") public void receive(Object msg) {

log.info("接受消息：" + msg);

}

}

##### UserController.java

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq")

@ResponseBody public void mq() {

mqSender.send("Hello");

}

#### 结果



**RabbitMQ交换机**

**Fanout模式**

##### 不处理路由键，只需要简单的将队里绑定到交换机上

发送到交换机的消息都会被转发到与该交换机绑定的所有队列上

Fanout交换机转发消息是最快的

**实现代码**

RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Binding; import org.springframework.amqp.core.BindingBuilder; import org.springframework.amqp.core.FanoutExchange; import org.springframework.amqp.core.Queue;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

private static final String QUEUE01 = "queue\_fanout01"; private static final String QUEUE02 = "queue\_fanout02"; private static final String EXCHANGE = "fanoutExchange";

@Bean

public Queue queue01(){ return new Queue(QUEUE01);

}

@Bean

public Queue queue02(){ return new Queue(QUEUE02);

}

@Bean

public FanoutExchange fanoutExchange(){ return new FanoutExchange(EXCHANGE);

}

@Bean

public Binding binding01(){

return BindingBuilder.bind(queue01()).to(fanoutExchange());

}

@Bean

public Binding binding02(){

return BindingBuilder.bind(queue02()).to(fanoutExchange());

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void send(Object msg) {

log.info(" 发 送 消 息 ："+msg); rabbitTemplate.convertAndSend("fanoutExchange","",msg);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@RabbitListener(queues = "queue\_fanout01") public void receive01(Object msg) {

log.info("QUEUE01接受消息：" + msg);

}

@RabbitListener(queues = "queue\_fanout02") public void receive02(Object msg) {

log.info("QUEUE02接受消息：" + msg);

}

}

##### UserController.java

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/fanout")

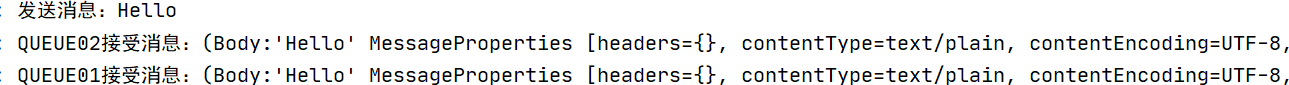
@ResponseBody public void mq() {

mqSender.send("Hello");

}

#### 测试

调用 mq/direct01 接口，消息经由交换机转发到绑定该交换机的所有队列



### **Direct**模式

##### 所有发送到Direct Exchange的消息被转发到RouteKey中指定的Queue

注意：Direct模式可以使用RabbitMQ自带的Exchange：default Exchange,所以不需要将

Exchange进行任何绑定(binding)操作，消息传递时，RouteKey必须完全匹配才会被队列接收，否 则该消息会被抛弃。

重点：routing key与队列queues 的key保持一致，即可以路由到对应的queue中。

**代码实现**

RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Binding; import org.springframework.amqp.core.BindingBuilder; import org.springframework.amqp.core.DirectExchange; import org.springframework.amqp.core.Queue;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

private static final String QUEUE01 = "queue\_direct01"; private static final String QUEUE02 = "queue\_direct02"; private static final String EXCHANGE = "directExchange"; private static final String ROUTINGKEY01 = "queue.red"; private static final String ROUTINGKEY02 = "queue.green";

@Bean

public Queue queue01(){ return new Queue(QUEUE01);

}

@Bean

public Queue queue02(){ return new Queue(QUEUE02);

}

@Bean

public DirectExchange directExchange(){

return new DirectExchange(EXCHANGE);

}

@Bean

public Binding binding01(){ return

BindingBuilder.bind(queue01()).to(directExchange()).with(ROUTINGKEY01);

}

@Bean

public Binding binding02(){ return

BindingBuilder.bind(queue02()).to(directExchange()).with(ROUTINGKEY02);

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void send01(Object msg) { log.info(" 发 送 red 消 息 ："+msg); rabbitTemplate.convertAndSend("directExchange","queue.red",msg);

}

public void send02(Object msg) {

log.info(" 发 送 green 消 息 ："+msg); rabbitTemplate.convertAndSend("directExchange","queue.green",msg);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@RabbitListener(queues = "queue\_direct01") public void receive01(Object msg) {

log.info("QUEUE01接受消息：" + msg);

}

@RabbitListener(queues = "queue\_direct02") public void receive02(Object msg) {

log.info("QUEUE02接受消息：" + msg);

}

}

##### UserController.java

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/direct01")

@ResponseBody public void mq01() {

mqSender.send01("Hello,Red");

}

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/direct02")

@ResponseBody public void mq02() {

mqSender.send02("Hello,Green");

}

**测试**

调用 mq/direct01 接口，消息经由交换机绑定的 queue.red RoutingKey 转发到 queue\_direct01 队列



调用 mq/direct02 接口，消息经由交换机绑定的 queue.green RoutingKey 转发到 queue\_direct02

##### 队列



**Topic模式**

所有发送到Topic Exchange的消息被转发到所有管线RouteKey中指定Topic的Queue上Exchange将RouteKey和某Topic进行模糊匹配,此时队列需要绑定一个Topic

对于routing key匹配模式定义规则举例如下:

routing key为一个句点号 . 分隔的字符串（我们将被句点号 . 分隔开的每一段独立的字符串称为一个单词），如“stock.usd.nyse”、“nyse.vmw”、“quick.orange.rabbit”

routing key中可以存在两种特殊字符 \* 与 # ，用于做模糊匹配，其中 \* 用于匹配一个单词， # 用

于匹配多个单词（可以是零个）

**代码实现**

RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Binding; import org.springframework.amqp.core.BindingBuilder; import org.springframework.amqp.core.Queue;

import org.springframework.amqp.core.TopicExchange; import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

private static final String QUEUE01 = "queue\_topic01"; private static final String QUEUE02 = "queue\_topic02"; private static final String EXCHANGE = "topicExchange"; private static final String ROUTINGKEY01 = "#.queue.#"; private static final String ROUTINGKEY02 = "\*.queue.#";

@Bean

public Queue queue01(){ return new Queue(QUEUE01);

}

@Bean

public Queue queue02(){

return new Queue(QUEUE02);

}

@Bean

public TopicExchange topicExchange(){ return new TopicExchange(EXCHANGE);

}

@Bean

public Binding binding01(){ return

BindingBuilder.bind(queue01()).to(topicExchange()).with(ROUTINGKEY01);

}

@Bean

public Binding binding02(){ return

BindingBuilder.bind(queue02()).to(topicExchange()).with(ROUTINGKEY02);

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void send01(Object msg) { log.info(" 发 送 消 息 ( 被 01 队 列 接 受 )："+msg); rabbitTemplate.convertAndSend("topicExchange","queue.red.message",msg);

}

public void send02(Object msg) { log.info("发送消息(被两个queue接受)："+msg);

rabbitTemplate.convertAndSend("topicExchange","message.queue.green.abc",msg);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@RabbitListener(queues = "queue\_topic01") public void receive01(Object msg) {

log.info("QUEUE01接受消息：" + msg);

}

@RabbitListener(queues = "queue\_topic02") public void receive02(Object msg) {

log.info("QUEUE02接受消息：" + msg);

}

}

##### UserController.java

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/topic01")

@ResponseBody public void mq01() {

mqSender.send01("Hello,Red");

}

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/topic02")

@ResponseBody public void mq02() {

mqSender.send02("Hello,Green");

}

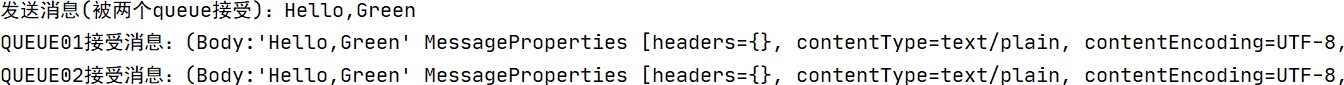
#### 测试

调用 mq/topic01 接口，消息经由交换机绑定的 #.queue.# RoutingKey 转发到 queue\_topic01 队列



调用 mq/topic02 接口，消息经由交换机绑定的 \*.queue.# 和 #.queue.# RoutingKey 转发到

queue\_topic01 和 queue\_topic02 队列



### **Headers**模式

##### 不依赖routingkey，使用发送消息时basicProperties对象中的headers匹配队列

headers是一个键值对类型，键值对的值可以是任何类型

在队列绑定交换机时用x-match来指定，all代表定义的多个键值对都要满足，any则代表只要满足 一个可以了

**代码实现**

RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Binding; import org.springframework.amqp.core.BindingBuilder;

import org.springframework.amqp.core.HeadersExchange; import org.springframework.amqp.core.Queue;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import java.util.HashMap; import java.util.Map;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

private static final String QUEUE01 = "queue\_header01"; private static final String QUEUE02 = "queue\_header02"; private static final String EXCHANGE = "headersExchange";

@Bean

public Queue queue01(){

return new Queue(QUEUE01);

}

@Bean

public Queue queue02(){ return new Queue(QUEUE02);

}

@Bean

public HeadersExchange headersExchange(){ return new HeadersExchange(EXCHANGE);

}

@Bean

public Binding binding01(){ Map<String,Object> map = new HashMap<>(); map.put("color","red");

map.put("speed","low"); return

BindingBuilder.bind(queue01()).to(headersExchange()).whereAny(map).match();

}

@Bean

public Binding binding02(){ Map<String,Object> map = new HashMap<>(); map.put("color","red");

map.put("speed","fast"); return

BindingBuilder.bind(queue02()).to(headersExchange()).whereAll(map).match();

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.core.Message;

import org.springframework.amqp.core.MessageProperties; import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void send01(String msg) {

log.info(" 发 送 消 息 ( 被 两 个 queue 接 受 )：" + msg); MessageProperties properties = new MessageProperties(); properties.setHeader("color", "red"); properties.setHeader("speed", "fast");

Message message = new Message(msg.getBytes(), properties); rabbitTemplate.convertAndSend("headersExchange", "", message);

}

public void send02(String msg) {

log.info(" 发 送 消 息 ( 被 01 队 列 接 受 )：" + msg); MessageProperties properties = new MessageProperties(); properties.setHeader("color", "red"); properties.setHeader("speed", "normal");

Message message = new Message(msg.getBytes(), properties); rabbitTemplate.convertAndSend("headersExchange", "", message);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.core.Message;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@RabbitListener(queues = "queue\_header01") public void receive01(Message message) {

log.info("QUEUE01接受Message对象：" + message);

log.info("QUEUE01接受消息：" + new String(message.getBody()));

}

@RabbitListener(queues = "queue\_header02") public void receive02(Message message) {

log.info("QUEUE02接受Message对象：" + message);

log.info("QUEUE02接受消息：" + new String(message.getBody()));

}

}

##### UserController.java

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/header01")

@ResponseBody public void mq01() {

mqSender.send01("Hello,header01");

}

/\*\*

* 测试发送RabbitMQ消息

\*/

@RequestMapping("/mq/header02")

@ResponseBody public void mq02() {

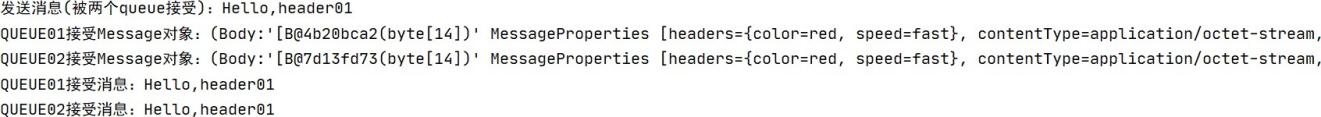
mqSender.send02("Hello,header02");

}

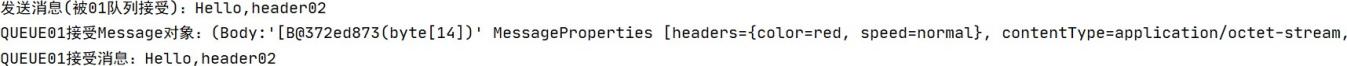
**测试**

queue\_header01 设置x-match为any， queue\_header02 设置x-match为all。因此调用 mq/header01

##### 接口，可以匹配两个队列



调用 mq/header02 接口，只能匹配 queue\_header01 队列



## 接口优化

##### 思路：减少数据库访问

1. 系统初始化，把商品库存数量加载到Redis
2. 收到请求，Redis预减库存。库存不足，直接返回。否则进入第3步
3. 请求入队，立即返回排队中
4. 请求出队，生成订单，减少库存
5. 客户端轮询，是否秒杀成功

**Redis操作库存**

SeckillController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.rabbitmq.MQSender; import com.xxxx.seckill.rabbitmq.SeckillMessage; import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillOrderService; import com.xxxx.seckill.util.JsonUtil;

import com.xxxx.seckill.vo.GoodsVo; import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.InitializingBean; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.data.redis.core.ValueOperations; import org.springframework.stereotype.Controller;

import org.springframework.util.CollectionUtils; import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;

import java.util.HashMap; import java.util.List; import java.util.Map;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill")

public class SeckillController implements InitializingBean {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@Autowired

private RedisTemplate redisTemplate;

@Autowired

private MQSender mqSender;

private Map<Long, Boolean> EmptyStockMap = new HashMap<>();

/\*\*

* + 秒杀

\*

* + @param user
  + @param goodsId
  + @return

\*/

@RequestMapping(value = "/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

/\*GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//判断是否重复抢购

// SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id",

// user.getId()).eq(

// "goods\_id",

// goodsId));

String seckillOrderJson = (String) redisTemplate.opsForValue().get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

Order order = orderService.seckill(user, goods); if (null != order) {

return RespBean.success(order);

}\*/

ValueOperations valueOperations = redisTemplate.opsForValue();

//判断是否重复抢购

String seckillOrderJson = (String) valueOperations.get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

//内存标记,减少Redis访问

if (EmptyStockMap.get(goodsId)) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//预减库存

Long stock = valueOperations.decrement("seckillGoods:" + goodsId); if (stock < 0) {

EmptyStockMap.put(goodsId,true); valueOperations.increment("seckillGoods:" + goodsId); return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

/\*\*

}

// 请求入队，立即返回排队中

SeckillMessage message = new SeckillMessage(user, goodsId); mqSender.sendsecKillMessage(JsonUtil.object2JsonStr(message)); return RespBean.success(0);

* + 系统初始化，把商品库存数量加载到Redis

\*

* + @throws Exception

\*/

@Override

public void afterPropertiesSet() throws Exception { List<GoodsVo> list = goodsService.findGoodsVo(); if (CollectionUtils.isEmpty(list)) {

return;

}

list.forEach(goodsVo -> { redisTemplate.opsForValue().set("seckillGoods:" + goodsVo.getId(),

goodsVo.getStockCount());

EmptyStockMap.put(goodsVo.getId(), false);

});

}

}

**RabbitMQ秒杀**

##### SeckillMessage.java

package com.xxxx.seckill.rabbitmq;

import com.xxxx.seckill.pojo.User; import lombok.AllArgsConstructor; import lombok.Data;

import lombok.NoArgsConstructor;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

public class SeckillMessage { private User user;

private Long goodsId;

}

##### RabbitMQConfig.java

package com.xxxx.seckill.config;

import org.springframework.amqp.core.Binding; import org.springframework.amqp.core.BindingBuilder; import org.springframework.amqp.core.Queue;

import org.springframework.amqp.core.TopicExchange; import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Configuration

public class RabbitMQConfig {

private static final String QUEUE = "seckillQueue"; private static final String EXCHANGE = "seckillExchange";

@Bean

public Queue queue(){ return new Queue(QUEUE);

}

@Bean

public TopicExchange topicExchange(){ return new TopicExchange(EXCHANGE);

}

@Bean

public Binding binding01(){

return BindingBuilder.bind(queue()).to(topicExchange()).with("seckill.#");

}

}

##### MQSender.java

package com.xxxx.seckill.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.core.RabbitTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQSender {

@Autowired

private RabbitTemplate rabbitTemplate;

public void sendsecKillMessage(String message) {

log.info(" 发 送 消 息 ：" + message); rabbitTemplate.convertAndSend("seckillExchange", "seckill.msg", message);

}

}

##### MQReceiver.java

package com.xxxx.seckill.rabbitmq;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService; import com.xxxx.seckill.util.JsonUtil;

import com.xxxx.seckill.vo.GoodsVo; import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.stereotype.Service; import org.springframework.util.StringUtils;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

@Slf4j

public class MQReceiver {

@Autowired

private IGoodsService goodsService;

@Autowired

private RedisTemplate redisTemplate;

@Autowired

private IOrderService orderService;

@RabbitListener(queues = "seckillQueue") public void receive(String msg) {

log.info("QUEUE接受消息：" + msg);

SeckillMessage message = JsonUtil.jsonStr2Object(msg, SeckillMessage.class);

Long goodsId = message.getGoodsId(); User user = message.getUser();

GoodsVo goods = goodsService.findGoodsVoByGoodsId(goodsId);

//判断库存

if (goods.getStockCount() < 1) { return;

}

//判断是否重复抢购

// SeckillOrder seckillOrder = seckillOrderService.getOne(new QueryWrapper<SeckillOrder>().eq("user\_id",

// user.getId()).eq(

// "goods\_id",

// goodsId));

String seckillOrderJson = (String) redisTemplate.opsForValue().get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) { return;

}

orderService.seckill(user, goods);

}

}

### 客户端轮询秒杀结果

##### SeckillController.java

/\*\*

* 获取秒杀结果

\*

* @param user
* @param goodsId
* @return orderId:成功，-1：秒杀失败，0：排队中

\*/

@RequestMapping(value = "/result", method = RequestMethod.GET)

@ResponseBody

public RespBean getResult(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

Long orderId = seckillOrderService.getResult(user, goodsId); return RespBean.success(orderId);

}

##### ISeckillOrderService.java

package com.xxxx.seckill.service;

import com.baomidou.mybatisplus.extension.service.IService; import com.xxxx.seckill.pojo.SeckillOrder;

import com.xxxx.seckill.pojo.User;

/\*\*

\* <p>

* 服务类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

public interface ISeckillOrderService extends IService<SeckillOrder> {

/\*\*

* + 获取秒杀结果
  + @param user
  + @param goodsId
  + @return

\*/

Long getResult(User user, Long goodsId);

}

##### SeckillOrderServiceImpl.java

package com.xxxx.seckill.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.xxxx.seckill.mapper.SeckillOrderMapper;

import com.xxxx.seckill.pojo.SeckillOrder; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.ISeckillOrderService;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.stereotype.Service;

/\*\*

\* <p>

* 服务实现类

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Service

public class SeckillOrderServiceImpl extends ServiceImpl<SeckillOrderMapper, SeckillOrder> implements ISeckillOrderService {

@Autowired

private SeckillOrderMapper seckillOrderMapper;

@Autowired

private RedisTemplate redisTemplate;

/\*\*

* + 获取秒杀结果

\*

* + @param user
  + @param goodsId
  + @return

\*/

@Override

public Long getResult(User user, Long goodsId) {

SeckillOrder seckillOrder = seckillOrderMapper.selectOne(new QueryWrapper<SeckillOrder>().eq("user\_id", user.getId()).eq("goods\_id", goodsId));

if (null != seckillOrder) { return seckillOrder.getId();

} else {

if (redisTemplate.hasKey("isStockEmpty:" + goodsId)) { return -1L;

}else {

return 0L;

}

}

}

}

##### OrderServiceImpl.java

/\*\*

* 秒杀

\*

* @param user
* @param goods
* @return

\*/

@Override

@Transactional

public Order seckill(User user, GoodsVo goods) {

ValueOperations valueOperations = redisTemplate.opsForValue();

//秒杀商品表减库存

SeckillGoods seckillGoods = seckillGoodsService.getOne(new QueryWrapper<SeckillGoods>().eq("goods\_id", goods.getId()));

boolean seckillGoodsResult = seckillGoodsService.update(

new UpdateWrapper<SeckillGoods>().setSql("stock\_count = stock\_count- 1").eq("goods\_id", goods.getId()).gt("stock\_count", 0));

// seckillGoodsService.updateById(seckillGoods); if (seckillGoods.getStockCount() < 1) {

//判断是否还有库存

valueOperations.set("isStockEmpty:" + goods.getId(), "0"); return null;

}

//生成订单

Order order = new Order(); order.setUserId(user.getId()); order.setGoodsId(goods.getId()); order.setDeliveryAddrId(0L); order.setGoodsName(goods.getGoodsName()); order.setGoodsCount(1); order.setGoodsPrice(seckillGoods.getSeckillPrice()); order.setOrderChannel(1);

order.setStatus(0); order.setCreateDate(new Date()); orderMapper.insert(order);

//生成秒杀订单

SeckillOrder seckillOrder = new SeckillOrder(); seckillOrder.setOrderId(order.getId()); seckillOrder.setUserId(user.getId()); seckillOrder.setGoodsId(goods.getId()); seckillOrderService.save(seckillOrder); valueOperations.set("order:" + user.getId() + ":" + goods.getId(),

JsonUtil.object2JsonStr(seckillOrder)); return order;

}

##### goodsDetail.htm

function doSeckill() {

$.ajax({

url: "/seckill/doSeckill", type: "POST",

data: {

goodsId: $("#goodsId").val(),

},

success: function (data) { if (data.code == 200) {

// window.location.href = "/orderDetail.htm?orderId=" +

data.obj.id;

getResult($("#goodsId").val());

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误");

}

})

}

function getResult(goodsId) { g\_showLoading();

$.ajax({

url: "/seckill/result", type: "GET",

data: {

goodsId: goodsId,

消"]},

result;

},

success: function (data) { if (data.code == 200) {

var result = data.obj; if (result < 0) {

layer.msg("对不起，秒杀失败！");

} else if (result == 0) { setTimeout(function () {

getResult(goodsId);

}, 50);

} else {

layer.confirm("恭喜你，秒杀成功！查看订单？", {btn: ["确定", "取

function () {

window.location.href = "/orderDetail.htm?orderId=" +

},

function () {

layer.close();

});

}

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误");

}

})

}

### 测试

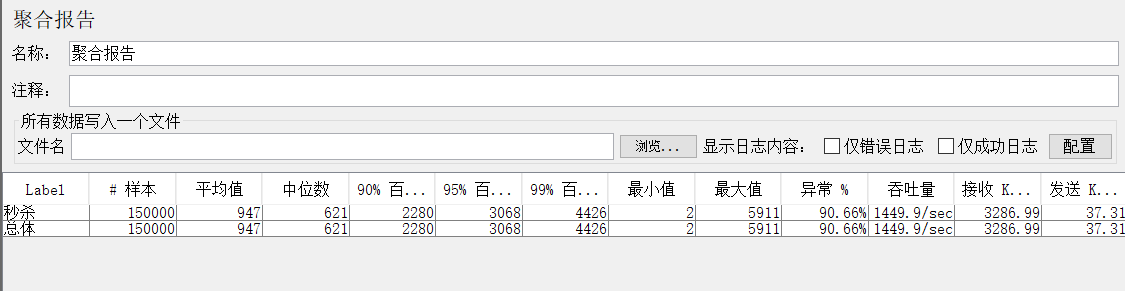
##### 项目启动，Redis预加载库存



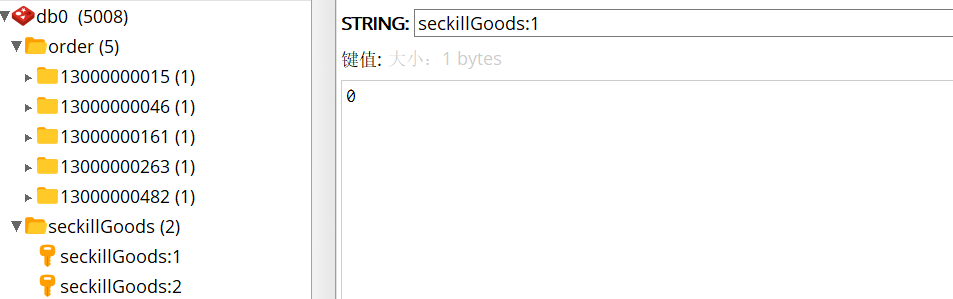
秒杀成功，数据库及Redis库存数量正确

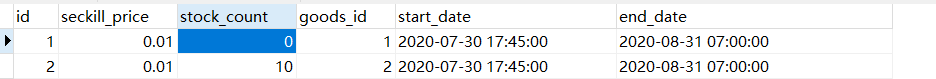


**压测秒杀**

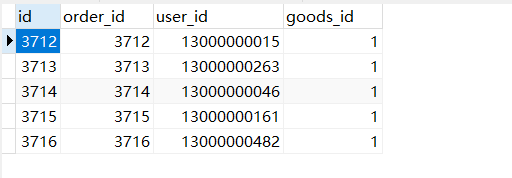
QPS相比之前有一定提升

数据库以及Redis库存数量和订单都正确









**优化Redis操作库存**

上面代码实际演示会发现Redis的库存有问题，原因在于Redis没有做到原子性。我们采用锁去解决

**分布式锁**

进来一个线程先占位，当别的线程进来操作时，发现已经有人占位了，就会放弃或者稍后再试 线程操作执行完成后，需要调用del指令释放位子

package com.xxxx.seckill;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.data.redis.core.RedisTemplate; import org.springframework.data.redis.core.ValueOperations;

@SpringBootTest

class SeckillApplicationTests {

@Autowired

private RedisTemplate redisTemplate;

@Test

public void testLock01(){

ValueOperations valueOperations = redisTemplate.opsForValue(); Boolean isLock = valueOperations.setIfAbsent("k1", "v1");

if (isLock){

valueOperations.set("name","xxxx");

String name = (String) valueOperations.get("name"); System.out.println(name); redisTemplate.delete("k1");

}else {

System.out.println("有线程在使用，请稍后");

}

}

}

##### 为了防止业务执行过程中抛异常或者挂机导致del指定没法调用形成死锁，可以添加超时时间

@Test

public void testLock02(){

ValueOperations valueOperations = redisTemplate.opsForValue();

Boolean isLock = valueOperations.setIfAbsent("k1","v1",5, TimeUnit.SECONDS); if (isLock){

valueOperations.set("name","xxxx");

String name = (String) valueOperations.get("name"); System.out.println(name); redisTemplate.delete("k1");

}else {

System.out.println("有线程在使用，请稍后");

}

}

##### 上面例子，如果业务非常耗时会紊乱。举例：第一个线程首先获得锁，然后执行业务代码，但是业务代 码耗时8秒，这样会在第一个线程的任务还未执行成功锁就会被释放，这时第二个线程会获取到锁开始 执行，在第二个线程开执行了3秒，第一个线程也执行完了，此时第一个线程会释放锁，但是注意，他 释放的第二个现成的锁，释放之后，第三个线程进来。

解决方案：

尽量避免在获取锁之后，执行耗时操作

将锁的value设置为一个随机字符串，每次释放锁的时候，都去比较随机字符串是否一致，如果一 致，再去释放，否则不释放。

释放锁时要去查看所得value，比较value是否正确，释放锁总共三个步骤，这三个步骤不具 备原子性。

**Lua脚本**

Lua脚本优势：

使用方便，Redis内置了对Lua脚本的支持

Lua脚本可以在Rdis服务端原子的执行多个Redis命令

由于网络在很大程度上会影响到Redis性能，使用Lua脚本可以让多个命令一次执行，可以有 效解决网络给Redis带来的性能问题

使用Lua脚本思路：

提前在Redis服务端写好Lua脚本，然后在java客户端去调用脚本

可以在java客户端写Lua脚本，写好之后，去执行。需要执行时，每次将脚本发送到Redis上 去执行

创建Lua脚本(放在resources目录下) lock.lua

if redis.call("get",KEYS[1]) == ARGV[1] then return redis.call("del",KEYS[1])

else

return 0

end

##### 调用脚本

RedisConfig.java

@Bean

public DefaultRedisScript<Boolean> script() { DefaultRedisScript<Boolean> redisScript = new DefaultRedisScript<>();

//放在和application.yml 同层目录下

redisScript.setLocation(new ClassPathResource("lock.lua")); redisScript.setResultType(Boolean.class);

return redisScript;

}

##### Test.java

@Test

public void testLock03(){

ValueOperations valueOperations = redisTemplate.opsForValue(); String value = UUID.randomUUID().toString();

//给锁添加一个过期时间，防止应用在运行过程中抛出异常导致锁无法及时得到释放

Boolean isLock = valueOperations.setIfAbsent("k1",value,5, TimeUnit.SECONDS);

//没人占位

if (isLock){ valueOperations.set("name","xxxx");

String name = (String) valueOperations.get("name"); System.out.println(name); System.out.println(valueOperations.get("k1"));

//释放锁

Boolean result = (Boolean) redisTemplate.execute(script, Collections.singletonList("k1"), value);

System.out.println(result);

}else {

//有人占位，停止/暂缓 操作

System.out.println("有线程在使用，请稍后");

}

}

**优化Redis预减库存**

##### stock.lua

if (redis.call('exists', KEYS[1]) == 1) then

local stock = tonumber(redis.call('get', KEYS[1])); if (stock > 0) then

redis.call('incrby', KEYS[1], -1); return stock;

end; return 0;

end;

##### RedisConfig.java

@Bean

public DefaultRedisScript<Long> script() {

DefaultRedisScript<Long> redisScript = new DefaultRedisScript<>();

//放在和application.yml 同层目录下

redisScript.setLocation(new ClassPathResource("stock.lua")); redisScript.setResultType(Long.class);

return redisScript;

}

##### SeckillController.java

/\*\*

* 秒杀

\*

* @param user
* @param goodsId
* @return

\*/

@RequestMapping(value = "/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

ValueOperations valueOperations = redisTemplate.opsForValue();

//判断是否重复抢购

String seckillOrderJson = (String) valueOperations.get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

//内存标记,减少Redis访问

if (EmptyStockMap.get(goodsId)) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//预减库存

Long stock = (Long) redisTemplate.execute(script, Collections.singletonList("seckillGoods:" + goodsId), Collections.EMPTY\_LIST);

if (stock < 0) { EmptyStockMap.put(goodsId,true);

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

// 请求入队，立即返回排队中

SeckillMessage message = new SeckillMessage(user, goodsId); mqSender.sendsecKillMessage(JsonUtil.object2JsonStr(message)); return RespBean.success(0);

}

# 安全优化

## 秒杀接口地址隐藏

##### 秒杀开始之前，先去请求接口获取秒杀地址

SeckillController.java

package com.xxxx.seckill.controller;

import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.rabbitmq.MQSender; import com.xxxx.seckill.rabbitmq.SeckillMessage; import com.xxxx.seckill.service.IGoodsService; import com.xxxx.seckill.service.IOrderService;

import com.xxxx.seckill.service.ISeckillOrderService;

import com.xxxx.seckill.util.JsonUtil; import com.xxxx.seckill.vo.GoodsVo; import com.xxxx.seckill.vo.RespBean; import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.InitializingBean; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.data.redis.core.ValueOperations;

import org.springframework.data.redis.core.script.DefaultRedisScript; import org.springframework.stereotype.Controller;

import org.springframework.util.CollectionUtils; import org.springframework.util.StringUtils;

import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;

import java.util.Collections; import java.util.HashMap; import java.util.List; import java.util.Map;

/\*\*

\* <p>

* 前端控制器

\* </p>

\*

* @author zhoubin

\* @since 1.0.0

\*/

@Controller

@RequestMapping("/seckill")

public class SeckillController implements InitializingBean {

@Autowired

private IGoodsService goodsService;

@Autowired

private ISeckillOrderService seckillOrderService;

@Autowired

private IOrderService orderService;

@Autowired

private RedisTemplate redisTemplate;

@Autowired

private MQSender mqSender;

@Autowired

private DefaultRedisScript<Long> script;

private Map<Long, Boolean> EmptyStockMap = new HashMap<>();

/\*\*

* + 秒杀

\*

* + @param user
  + @param goodsId
  + @return

\*/

@RequestMapping(value = "/{path}/doSeckill", method = RequestMethod.POST)

@ResponseBody

public RespBean doSeckill(@PathVariable String path, User user, Long goodsId)

{

if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

ValueOperations valueOperations = redisTemplate.opsForValue(); boolean check = orderService.checkPath(user,goodsId,path);

if (!check){

return RespBean.error(RespBeanEnum.REQUEST\_ILLEGAL);

}

//判断是否重复抢购

String seckillOrderJson = (String) valueOperations.get("order:" + user.getId() + ":" + goodsId);

if (!StringUtils.isEmpty(seckillOrderJson)) {

return RespBean.error(RespBeanEnum.REPEATE\_ERROR);

}

//内存标记,减少Redis访问

if (EmptyStockMap.get(goodsId)) {

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

//预减库存

// Long stock = valueOperations.decrement("seckillGoods:" + goodsId);

// if (stock < 0) {

// EmptyStockMap.put(goodsId,true);

// valueOperations.increment("seckillGoods:" + goodsId);

// return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

// }

Long stock = (Long) redisTemplate.execute(script, Collections.singletonList("seckillGoods:" + goodsId),

Collections.EMPTY\_LIST); if (stock <= 0) {

EmptyStockMap.put(goodsId, true);

return RespBean.error(RespBeanEnum.EMPTY\_STOCK);

}

// 请求入队，立即返回排队中

SeckillMessage message = new SeckillMessage(user, goodsId); mqSender.sendsecKillMessage(JsonUtil.object2JsonStr(message)); return RespBean.success(0);

}

/\*\*

* + 获取秒杀地址

\*

* + @param user
  + @param goodsId
  + @return

\*/

@RequestMapping(value = "/path", method = RequestMethod.GET)

@ResponseBody

public RespBean getPath(User user, Long goodsId) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

String str = orderService.createPath(user,goodsId);

return RespBean.success(str);

}

}

##### IOrderService.java

/\*\*

* 验证秒杀地址
* @param user
* @param goodsId
* @param path
* @return

\*/

boolean checkPath(User user, Long goodsId, String path);

/\*\*

* 生成秒杀地址
* @param user
* @param goodsId
* @return

\*/

String createPath(User user, Long goodsId);

##### OrderServiceImpl.java

/\*\*

* 验证请求地址

\*

* @param user
* @param goodsId
* @param path
* @return

\*/

@Override

public boolean checkPath(User user, Long goodsId, String path) { if (user==null|| StringUtils.isEmpty(path)){

return false;

}

String redisPath = (String) redisTemplate.opsForValue().get("seckillPath:" + user.getId() + ":" + goodsId);

return path.equals(redisPath);

}

/\*\*

* 生成秒杀地址

\*

* @param user
* @param goodsId
* @return

\*/

@Override

public String createPath(User user, Long goodsId) { String str = MD5Util.md5(UUIDUtil.uuid() + "123456");

redisTemplate.opsForValue().set("seckillPath:" + user.getId() + ":" + goodsId, str, 60, TimeUnit.SECONDS);

return str;

}

##### goodsDetail.htm

function getSeckillPath() {

var goodsId = $("#goodsId").val(); g\_showLoading();

$.ajax({

url: "/seckill/path", type: "GET",

data: {

goodsId: goodsId,

},

success: function (data) { if (data.code == 200) {

var path = data.obj; doSeckill(path);

} else {

layer.msg(data.message);

}

}

,

error: function () {

layer.msg("客户端请求错误");

}

})

}

function doSeckill(path) {

$.ajax({

url: "/seckill/" + path + "/doSeckill", type: "POST",

data: {

goodsId: $("#goodsId").val(),

},

success: function (data) { if (data.code == 200) {

// window.location.href = "/orderDetail.htm?orderId=" +

data.obj.id;

getResult($("#goodsId").val());

} else {

layer.msg(data.message);

}

},

error: function () {

layer.msg("客户端请求错误");

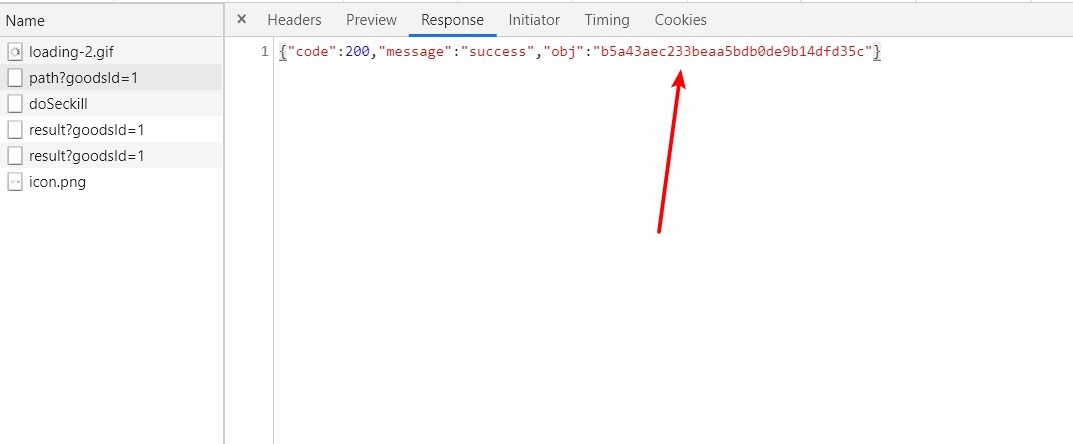
}

})

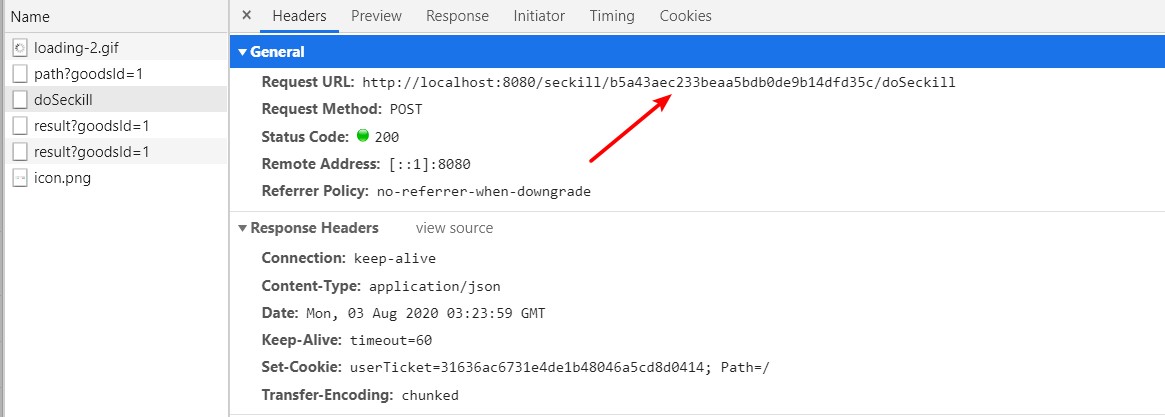
}

**测试**

##### 先去请求接口获取秒杀地址



秒杀真正地址



**图形验证码**

点击秒杀开始前，先输入验证码，分散用户的请求

**生成验证码**

引入依赖pom.xml

<!-- 验证码 -->

<dependency>

<groupId>com.github.whvcse</groupId>

<artifactId>easy-captcha</artifactId>

<version>1.6.2</version>

</dependency>

##### SeckillController.java

/\*\*

* 验证码

\*

* @param user
* @param goodsId
* @return

\*/

@RequestMapping(value = "/captcha", method = RequestMethod.GET)

public void verifyCode(User user, Long goodsId, HttpServletResponse response) { if (null==user||goodsId<0){

throw new GlobalException(RespBeanEnum.REQUEST\_ILLEGAL);

}

// 设置请求头为输出图片类型response.setContentType("image/jpg"); response.setHeader("Pragma", "No-cache"); response.setHeader("Cache-Control", "no-cache"); response.setDateHeader("Expires", 0);

//生成验证码，将结果放入redis

ArithmeticCaptcha captcha = new ArithmeticCaptcha(130, 32, 3);

redisTemplate.opsForValue().set("captcha:"+user.getId()+":"+goodsId,captcha.text (),300, TimeUnit.SECONDS);

try {

captcha.out(response.getOutputStream());

} catch (IOException e) {

log.error("验证码生成失败",e.getMessage());

}

}

##### goodsDetail.htm

<div class="row">

<div class="form-inline">

<img id="captchaImg" width="130" height="32" style="display: none" onclick="refreshCaptcha()"/>

<input id="captcha" class="form-control" style="display: none"/>

<button class="btn btn-primary" type="button" id="buyButton"

onclick="getSeckillPath()">立即秒杀

</button>

</div>

</div>

<script>

function refreshCaptcha() {

$("#captchaImg").attr("src", "/seckill/captcha?goodsId=" +

$("#goodsId").val() + "&time=" + new Date())

}

function countDown() {

var remainSeconds = $("#remainSeconds").val(); var timeout;

//秒杀还没开始，倒计时

if (remainSeconds > 0) {

$("#buyButton").attr("disabled", true);

$("#seckillTip").html("秒杀倒计时：" + remainSeconds + "秒"); timeout = setTimeout(function () {

// $("#countDown").text(remainSeconds - 1);

$("#remainSeconds").val(remainSeconds - 1); countDown();

}, 1000

);

}

//秒杀进行中

else if (remainSeconds == 0) {

$("#buyButton").attr("disabled", false); if (timeout) {

clearTimeout(timeout);

}

$("#seckillTip").html("秒杀进行中");

$("#captchaImg").attr("src", "/seckill/captcha?goodsId=" +

$("#goodsId").val())

$("#captchaImg").show();

$("#captcha").show();

//秒杀已经结束

} else {

$("#buyButton").attr("disabled", true);

$("#seckillTip").html("秒杀已经结束");

$("#captchaImg").hide();

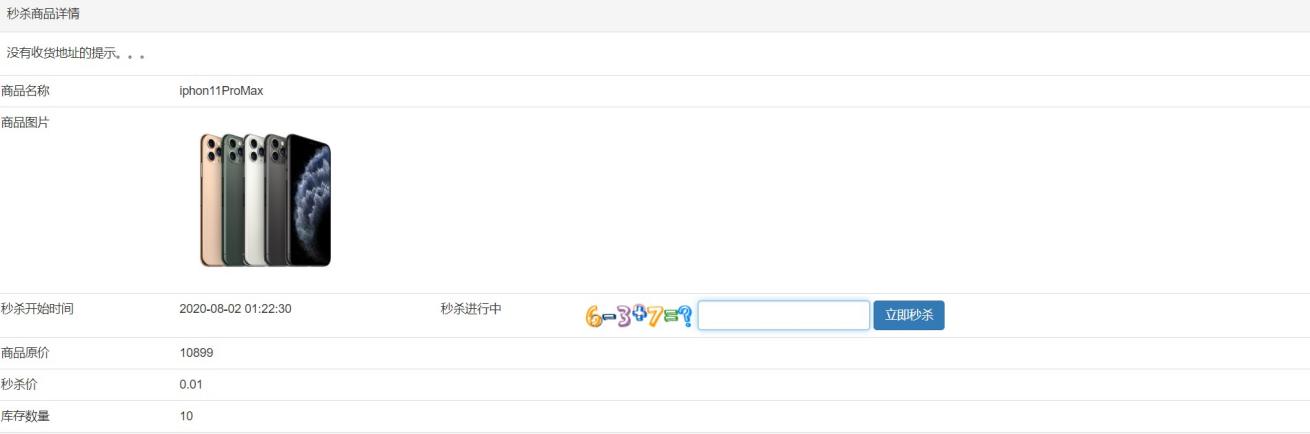
$("#captcha").hide();

}

}

</script>

#### 测试



**验证验证码**

##### SeckillController.java

/\*\*

* 获取秒杀地址

\*

* @param user
* @param goodsId
* @return

\*/

@RequestMapping(value = "/path", method = RequestMethod.GET)

@ResponseBody

public RespBean getPath(User user, Long goodsId,String captcha) { if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

boolean check = orderService.checkCaptcha(user, goodsId, captcha); if (!check){

return RespBean.error(RespBeanEnum.ERROR\_CAPTCHA);

}

String str = orderService.createPath(user, goodsId);

return RespBean.success(str);

}

##### IOrderService.java

/\*\*

* 校验验证码

\*

* @param user
* @param goodsId
* @param captcha
* @return

\*/

boolean checkCaptcha(User user, Long goodsId, String captcha);

##### OrderServiceImpl.java

/\*\*

* 校验验证码

\*

* @param user
* @param goodsId
* @param captcha
* @return

\*/

@Override

public boolean checkCaptcha(User user, Long goodsId, String captcha) { if (StringUtils.isEmpty(captcha)||null==user||goodsId<0){

return false;

}

String redisCaptcha = (String) redisTemplate.opsForValue().get("captcha:" + user.getId() + ":" + goodsId);

return redisCaptcha.equals(captcha);

}

##### goodsDetail.htm

function getSeckillPath() {

var goodsId = $("#goodsId").val(); var captcha = $("#captcha").val(); g\_showLoading();

$.ajax({

url: "/seckill/path", type: "GET",

data: {

goodsId: goodsId, captcha: captcha

},

success: function (data) { if (data.code == 200) {

var path = data.obj; doSeckill(path);

} else {

layer.msg(data.message);

}

}

,

error: function () { layer.msg("客户端请求错误");

}

})

}

#### 测试

##### 输入错误验证码，提示错误并且无法秒杀



输入正确验证码，正常秒杀



**接口限流**

**简单接口限流**

SeckillController.java

/\*\*

* 获取秒杀地址

\*

* @param user
* @param goodsId
* @return

\*/

@RequestMapping(value = "/path", method = RequestMethod.GET)

@ResponseBody

public RespBean getPath(User user, Long goodsId, String captcha, HttpServletRequest request) {

if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

ValueOperations valueOperations = redisTemplate.opsForValue();

//限制访问次数，5秒内访问5次

String uri = request.getRequestURI();

//方便测试captcha = "0";

Integer count = (Integer) valueOperations.get(uri + ":" + user.getId()); if (count==null){

valueOperations.set(uri + ":" + user.getId(),1,5,TimeUnit.SECONDS);

}else if (count<5){

valueOperations.increment(uri + ":" + user.getId());

}else {

return RespBean.error(RespBeanEnum.ACCESS\_LIMIT\_REACHED);

}

boolean check = orderService.checkCaptcha(user, goodsId, captcha); if (!check){

return RespBean.error(RespBeanEnum.ERROR\_CAPTCHA);

}

String str = orderService.createPath(user, goodsId); return RespBean.success(str);

}

#### 测试



**通用接口限流**

##### UserContext.java

package com.xxxx.seckill.config;

import com.xxxx.seckill.pojo.User;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

public class UserContext {

private static ThreadLocal<User> userHolder = new ThreadLocal<>();

public static void setUser(User user) { userHolder.set(user);

}

public static User getUser() { return userHolder.get();

}

}

##### UserArgumentResolver.java

@Override

public Object resolveArgument(MethodParameter parameter, ModelAndViewContainer mavContainer,

NativeWebRequest webRequest, WebDataBinderFactory binderFactory) throws Exception {

return UserContext.getUser();

}

##### AccessInterceptor.java

package com.xxxx.seckill.config;

import com.fasterxml.jackson.databind.ObjectMapper; import com.xxxx.seckill.pojo.User;

import com.xxxx.seckill.service.IUserService; import com.xxxx.seckill.util.CookieUtil; import com.xxxx.seckill.vo.RespBean;

import com.xxxx.seckill.vo.RespBeanEnum;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.redis.core.RedisTemplate; import org.springframework.data.redis.core.ValueOperations; import org.springframework.stereotype.Component;

import org.springframework.util.StringUtils;

import org.springframework.web.method.HandlerMethod; import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException;

import java.io.PrintWriter;

import java.util.concurrent.TimeUnit;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Component

public class AccessInterceptor implements HandlerInterceptor {

@Autowired

private IUserService userService;

@Autowired

private RedisTemplate redisTemplate;

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

if (handler instanceof HandlerMethod){ User user = getUser(request,response); UserContext.setUser(user);

HandlerMethod hm = (HandlerMethod) handler;

AccessLimit accessLimit = hm.getMethodAnnotation(AccessLimit.class); if (accessLimit==null){

return true;

}

int second = accessLimit.second(); int maxCount = accessLimit.maxCount();

boolean needLogin = accessLimit.needLogin(); String key = request.getRequestURI();

if (needLogin){

if (user==null){ render(response,RespBeanEnum.SESSION\_ERROR); return false;

}

key += ":"+user.getId();

}

ValueOperations valueOperations = redisTemplate.opsForValue(); Integer count = (Integer) valueOperations.get(key);

if (count == null) {

valueOperations.set(key, 1, second, TimeUnit.SECONDS);

} else if (count < maxCount) { valueOperations.increment(key);

} else {

render(response,RespBeanEnum.ACCESS\_LIMIT\_REACHED); return false;

}

}

return true;

}

private void render(HttpServletResponse response, RespBeanEnum respBeanEnum) throws IOException {

response.setCharacterEncoding("UTF-8"); response.setContentType("application/json"); PrintWriter out = response.getWriter(); RespBean bean = RespBean.error(respBeanEnum);

out.write(new ObjectMapper().writeValueAsString(bean)); out.flush();

out.close();

}

private User getUser(HttpServletRequest request, HttpServletResponse response) {

String ticket = CookieUtil.getCookieValue(request, "userTicket"); if (StringUtils.isEmpty(ticket)) {

return null;

}

return userService.getByUserTicket(ticket, request, response);

}

}

##### WebConfig.java

@Override

public void addInterceptors(InterceptorRegistry registry) { registry.addInterceptor(accessInterceptor);

}

##### AccessLimit.java

package com.xxxx.seckill.config;

import java.lang.annotation.ElementType; import java.lang.annotation.Retention; import java.lang.annotation.RetentionPolicy; import java.lang.annotation.Target;

/\*\*

* @author zhoubin

\* @since 1.0.0

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.METHOD) public @interface AccessLimit {

int second();

int maxCount();

boolean needLogin() default true;

}

##### SeckillController.java

/\*\*

* 获取秒杀地址

\*

* @param user
* @param goodsId
* @return

\*/

@AccessLimit(second = 5, maxCount = 5, needLogin = true)

@RequestMapping(value = "/path", method = RequestMethod.GET)

@ResponseBody

public RespBean getPath(User user, Long goodsId, String captcha, HttpServletRequest request) {

if (user == null) {

return RespBean.error(RespBeanEnum.SESSION\_ERROR);

}

//方便测试captcha = "0";

boolean check = orderService.checkCaptcha(user, goodsId, captcha); if (!check) {

return RespBean.error(RespBeanEnum.ERROR\_CAPTCHA);

}

String str = orderService.createPath(user, goodsId); return RespBean.success(str);

}

#### 测试

秒杀商品详悄

没有收货地址的促示．．

商品名称 iphoa11P「oMa<

商品图片

＿

|  |  |  |
| --- | --- | --- |
| 秒杀开始时间 | 2020-08-02 01,22,30 | 秒杀进行中 |
| 商品原价 | 10899 |  |
| 秒杀价 | 001 |  |
| 库存数量 |  |  |