

# 1.环境配置

## rabbitmq && redis环境安装

```

Last login: Tue Dec 15 20:06:22 on ttys000
dujie@dujiedeMacBook-Pro ~ % cd /usr/local/Cellar/rabbitmq/3.8.3/sbin
dujie@dujiedeMacBook-Pro sbin % sudo rabbitmq-server
Password:

## ##      RabbitMQ 3.8.3
## ##
##### Copyright (c) 2007-2020 Pivotal Software, Inc.
##### ##
##### Licensed under the MPL 1.1. Website: https://rabbitmq.com

Doc guides: https://rabbitmq.com/documentation.html
Support:    https://rabbitmq.com/contact.html
Tutorials:  https://rabbitmq.com/getstarted.html
Monitoring: https://rabbitmq.com/monitoring.html

Logs: /usr/local/var/log/rabbitmq/rabbit@localhost.log
      /usr/local/var/log/rabbitmq/rabbit@localhost_upgrade.log

Config file(s): (none)

Starting broker... completed with 6 plugins.
```

```

dujie — redis-server 127.0.0.1:6379 ptr_munge= — 80x24

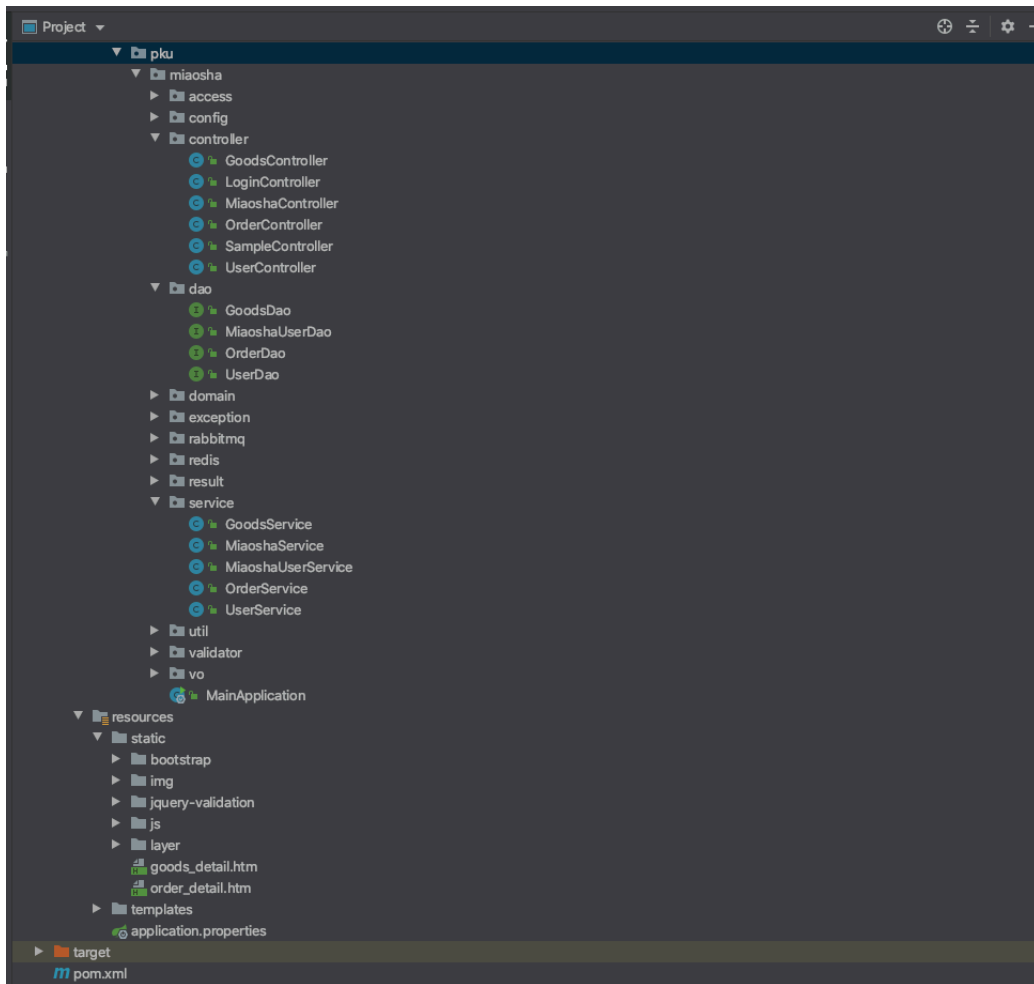
Redis 6.0.5 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 2768

http://redis.io

2768:M 15 Dec 2020 20:08:23.921 # Server initialized
2768:M 15 Dec 2020 20:08:23.921 * Loading RDB produced by version 6.0.5
2768:M 15 Dec 2020 20:08:23.921 * RDB age 7 seconds
2768:M 15 Dec 2020 20:08:23.921 * RDB memory usage when created 1.03 Mb
2768:M 15 Dec 2020 20:08:23.921 * DB loaded from disk: 0.000 seconds
2768:M 15 Dec 2020 20:08:23.921 * Ready to accept connections
```

## 项目模块框架



## 数据库设计

### 秒杀用户表

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	bigint	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
nickname	varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
password	varchar	32	0	<input type="checkbox"/>	<input type="checkbox"/>		
salt	varchar	10	0	<input type="checkbox"/>	<input type="checkbox"/>		
head	varchar	128	0	<input type="checkbox"/>	<input type="checkbox"/>		
register_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
last_login_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
login_count	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		

### 商品表

字段 索引 外键 触发器 选项 注释 SQL 预览								
名	类型	长度	小数点	不是 null	虚拟	键	注释	
id	bigint	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
goods_name	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_title	varchar	64	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_img	varchar	64	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_detail	longtext	0	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_price	decimal	10	2	<input type="checkbox"/>	<input type="checkbox"/>			
goods_stock	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>			


## 秒杀商品表

字段 索引 外键 触发器 选项 注释 SQL 预览								
名	类型	长度	小数点	不是 null	虚拟	键	注释	
id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
goods_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>			
miaosha_price	decimal	10	2	<input type="checkbox"/>	<input type="checkbox"/>			
stock_count	varchar	11	0	<input type="checkbox"/>	<input type="checkbox"/>			
start_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>			
end_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>			

## 秒杀订单表

字段 索引 外键 触发器 选项 注释 SQL 预览								
名	类型	长度	小数点	不是 null	虚拟	键	注释	
id	bigint	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
user_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			
order_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			

## 订单信息表

字段 索引 外键 触发器 选项 注释 SQL 预览								
名	类型	长度	小数点	不是 null	虚拟	键	注释	
id	bigint	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
user_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			
delivery_addr_id	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_name	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_count	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>			
goods_price	decimal	10	2	<input type="checkbox"/>	<input type="checkbox"/>			
order_channel	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>			
status	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>			
create_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>			
pay_date	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>			

## 2.登陆页面实现

# 实现效果&检验器

localhost

登录

用户登录

请输入手机号码

13000001001

请输入密码

.....|

最少要输入 6 个字符

重置

登录

## 技术要点

### 两次md5加密

采用两次md5加密保障用户密码安全性，密码传入表单及表单传入数据库均加密

```
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);
        System.out.println(str);
        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;
    }
}

```

## 数据库存储结果

id	nickname	password	salt	head	register_date	last_login_date	login_count
13000000	user0	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user1	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user2	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user3	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user4	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user5	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1
13000000	user6	b7797cce01b4b131b433b6acf4add449	1a2b3c	(NULL)	2020-12-09 22:1	(NULL)	1

## 参数校验

### 手机号码验证

```

public class ValidatorUtil {

    private static final Pattern mobile_pattern = Pattern.compile("1\\d{10}");

    public static boolean isMobile(String src) {
        if(StringUtils.isEmpty(src)) {
            return false;
        }
        Matcher m = mobile_pattern.matcher(src);
        return m.matches();
    }
}

```

最少要输入 11 个字符

## 异常处理

定义异常代码及全局异常处理器

```

public class CodeMsg {
    private int code;
    private String msg;
    //通用的错误码
    public static CodeMsg SUCCESS = new CodeMsg(0, "success");
    public static CodeMsg SERVER_ERROR = new CodeMsg(500100, "服务端异常");
    public static CodeMsg BIND_ERROR = new CodeMsg(500101, "参数校验异常: %s");
}

```

```

        public static CodeMsg REQUEST_ILLEGAL = new CodeMsg(500102, "请求非法");
        public static CodeMsg ACCESS_LIMIT_REACHED = new CodeMsg(500104, "访问太频繁!");
    };

    //登录模块 5002xx
    public static CodeMsg SESSION_ERROR = new CodeMsg(500210, "Session不存在或者已经失效");
    public static CodeMsg PASSWORD_EMPTY = new CodeMsg(500211, "登录密码不能为空");
    public static CodeMsg MOBILE_EMPTY = new CodeMsg(500212, "手机号不能为空");
    public static CodeMsg MOBILE_ERROR = new CodeMsg(500213, "手机号格式错误");
    public static CodeMsg MOBILE_NOT_EXIST = new CodeMsg(500214, "手机号不存在");
    public static CodeMsg PASSWORD_ERROR = new CodeMsg(500215, "密码错误");

    //商品模块 5003xx

    //订单模块 5004xx
    public static CodeMsg ORDER_NOT_EXIST = new CodeMsg(500400, "订单不存在");

    //秒杀模块 5005xx
    public static CodeMsg MIAO_SHA_OVER = new CodeMsg(500500, "商品已经秒杀完毕");
    public static CodeMsg REPEATE_MIAOSHA = new CodeMsg(500501, "不能重复秒杀");
    public static CodeMsg MIAOSHA_FAIL = new CodeMsg(500502, "秒杀失败");
}

@ControllerAdvice
@ResponseBody
public class GlobalExceptionHandler {
    @ExceptionHandler(value=Exception.class)
    public Result<String> exceptionHandler(HttpServletRequest request,
    Exception e){
        e.printStackTrace();
        if(e instanceof GlobalException) {
            GlobalException ex = (GlobalException)e;
            return Result.error(ex.getCm());
        }else if(e instanceof BindException) {
            BindException ex = (BindException)e;
            List<ObjectError> errors = ex.getAllErrors();
            ObjectError error = errors.get(0);
            String msg = error.getDefaultMessage();
            return Result.error(CodeMsg.BIND_ERROR.fillArgs(msg));
        }else {
            return Result.error(CodeMsg.SERVER_ERROR);
        }
    }
}

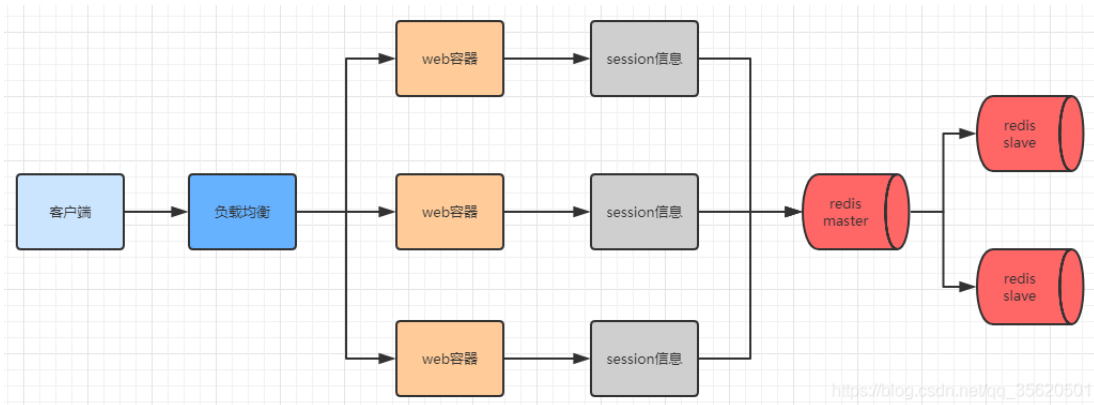
```

```
}
```

## 分布式session

客户端发送一个请求，经过负载均衡后该请求会被分配到服务器中的其中一个，由于不同服务器含有不同的web服务器(例如Tomcat)，不同的web服务器中并不能发现之前web服务器保存的session信息，就会再次生成一个JSESSIONID，之前的状态就会丢失。

解决方案：基于redis存储session



```
private void addCookie(HttpServletResponse response, String token, MiaoshaUser user) {
    redisService.set(MiaoshaUserKey.token, token, user);
    Cookie cookie = new Cookie(COOKI_NAME_TOKEN, token);
    cookie.setMaxAge(MiaoshaUserKey.token.expireSeconds());
    cookie.setPath("/");
    response.addCookie(cookie);
}

public MiaoshaUser getByToken(HttpServletResponse response, String token) {
    if(StringUtils.isEmpty(token)) {
        return null;
    }
    MiaoshaUser user = redisService.get(MiaoshaUserKey.token, token,
MiaoshaUser.class);
    //延长有效期
    if(user != null) {
        addCookie(response, token, user);
    }
    return user;
}
```

## Redis存储详情


```
[127.0.0.1:6379> keys *
1) "MiaoshaUserKey:id13000001001"
2) "OrderKey:moug13000000001_2"
3) "MiaoshaUserKey:tk289d5162729f403d941e3930a7d87ed2"
4) "OrderKey:moug13000001001_2"
5) "GoodsKey:gs1"
6) "GoodsKey:gs2"
7) "MiaoshaUserKey:id13000000001"
```

## 2.商品秒杀功能

### 商品列表页

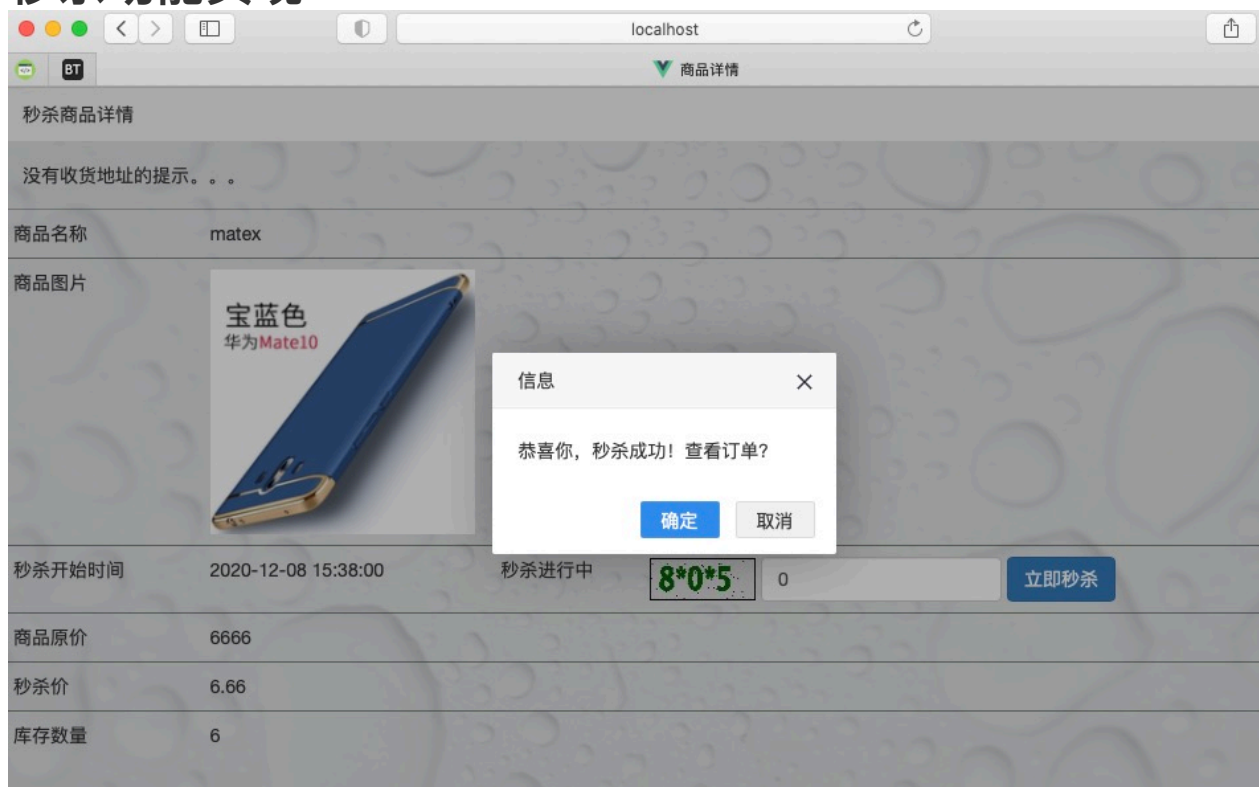
商品名称	商品图片	商品原价	秒杀价	库存数量	详情
iphonex		8888.0	9.99	4	<a href="#">详情</a>
matex		6666.0	6.66	5	<a href="#">详情</a>

### 商品详情页

秒杀商品详情			
没有收货地址的提示。。。			
商品名称	iphonex		
商品图片			
秒杀开始时间	2020-12-22 15:18:00	秒杀倒计时: 582327秒	<a href="#">立即秒杀</a>
商品原价	8888		
秒杀价	9.99		
库存数量	4		



# 秒杀功能实现



## 技术要点

### 秒杀倒计时

```
function countdown(){
    var remainSeconds = $("#remainSeconds").val();
    var timeout;
    if(remainSeconds > 0){//秒杀还没开始, 倒计时
        $("#buyButton").attr("disabled", true);
        $("#miaoshaTip").html("秒杀倒计时: "+remainSeconds+"秒");
        timeout = setTimeout(function(){
            $("#countDown").text(remainSeconds - 1);
            $("#remainSeconds").val(remainSeconds - 1);
            countdown();
        },1000);
    }else if(remainSeconds == 0){//秒杀进行中
        $("#buyButton").attr("disabled", false);
        if(timeout){
            clearTimeout(timeout);
        }
        $("#miaoshaTip").html("秒杀进行中");
        $("#verifyCodeImg").attr("src", "/miaosha/verifyCode?
goodsId="+$("#goodsId").val());
        $("#verifyCodeImg").show();
        $("#verifyCode").show();
    }else{//秒杀已经结束
```

```

        $("#buyButton").attr("disabled", true);
        $("#miaoshaTip").html("秒杀已经结束");
        $("#verifyCodeImg").hide();
        $("#verifyCode").hide();
    }
}

```

## 秒杀业务逻辑


//判断库存

```

GoodsVo goods = goodsService.getGoodsVoByGoodsId(goodsId); //10个商品, req1
req2
int stock = goods.getStockCount();
if(stock <= 0) {
    return Result.error(CodeMsg.MIAO_SHA_OVER);
}
//判断是否已经秒杀到了
MiaoshaOrder order =
orderService.getMiaoshaOrderByUserIdGoodsId(user.getId(), goodsId);
if(order != null) {
    return Result.error(CodeMsg.REPEATE_MIAOSHA);
}
//减库存 下订单 写入秒杀订单
OrderInfo orderInfo = miaoshaService.miaosha(user, goods);
return Result.success(orderInfo);
*/

```

## 查看订单详情

秒杀订单详情	
商品名称	matex
商品图片	
订单价格	6.66
下单时间	2020-12-15 20:11:35
订单状态	未支付
	立即支付
收货人	XXX 18812341234
收货地址	北京市昌平区回龙观龙博一区

## 3.缓存优化

### Redis页面缓存技术

```
@RequestMapping(value="/to_list", produces="text/html")
@ResponseBody
public String list(HttpServletRequest request, HttpServletResponse
response, Model model,MiaoshaUser user){
    model.addAttribute("user", user);
    //取缓存
    String html = redisService.get(GoodsKey.getGoodsList, "",
String.class);
    if(!StringUtils.isEmpty(html)) {
        return html;
    }
    List<GoodsVo> goodsList = goodsService.listGoodsVo();
    model.addAttribute("goodsList", goodsList);
    //    return "goods_list";
    SpringWebContext ctx = new SpringWebContext(request,response,
        request.getServletContext(),request.getLocale(),
model.asMap(), applicationContext );
    //手动渲染
    html = thymeleafViewResolver.getTemplateEngine().process("goods_list",
ctx);
    if(!StringUtils.isEmpty(html)) {
        redisService.set(GoodsKey.getGoodsList, "", html);
    }
    return html;
}
```

### Redis缓存页面详情

```
[127.0.0.1:6379> get GoodsKey:gl
"<!DOCTYPE HTML>\n\n<html>\n\n<head>\n    <title>\xe5\x95\x86\xe5\x93\x81\xe5\x88\x97\xe8\xa1\xa8</title>\n    <meta http-equiv=\n    \"Content-Type\" content=\"text/html; charset=UTF-8\" />\n    <!-- jquery -->\n    <script type=\"text/javascript\" src=\"/js/jq\n    uery.min.js\"></script>\n    <!-- bootstrap -->\n    <link rel=\"stylesheet\" type=\"text/css\" href=\"/bootstrap/css/bootstrap\n    .min.css\" />\n    <script type=\"text/javascript\" src=\"/bootstrap/js/bootstrap.min.js\"></script>\n    <!-- jquery-validator\n    -->\n    <script type=\"text/javascript\" src=\"/jquery-validation/jquery.validate.min.js\"></script>\n    <script type=\"text\n    /javascript\" src=\"/jquery-validation/localization/messages_zh.min.js\"></script>\n    <!-- layer -->\n    <script type=\"text\n    /javascript\" src=\"/layer/layer.js\"></script>\n    <!-- md5.js -->\n    <script type=\"text/javascript\" src=\"/js/md5.min.js\n    \"></script>\n    <!-- common.js -->\n    <script type=\"text/javascript\" src=\"/js/common.js\"></script>\n\n</head>\n\n<body>\n\n<div class=\"panel panel-default\">\n    <div class=\"panel-heading\">\xe7\xa7\x92\xe6\x9d\x80\xe5\x95\x86\xe5\x93\x81\xe5\x88\n    \x97\xe8\xa1\xa8</div>\n    <table class=\"table\" id=\"goodslist\">\n        <tr><td>\xe5\x95\x86\xe5\x93\x81\xe5\x9b\xbe\xe7\x89\x87</td><td>\xe5\x95\x86\xe5\x93\x81\xe5\x8e\x9f\xe4\xbb\xbb</td><t\n    d>\xe7\xa7\x92\xe6\x9d\x80\xe4\xbb\xbb</td><td>\xe5\xba\x93\xe5\xad\x98\xe6\x95\xb0\xe9\x87\x8f</td><td>\xe8\xaf\xa6\xe6\x83\x8\n    5</td></tr>\n        <tr>\n            <td>\xe8\x88.0</td>\n            <td>\xe9.99</td>\n            <td>\xe4</td>\n            <td>\xe8\xaf\xa6\xe6\x83\x85</td>\n        </tr>\n        <tr>\n            <td>\xe8\xaf\xa6\xe6\x83\x85</td>\n            <td>\xe8\xaf\xa6\xe6\x83\x85</td>\n            <td>\xe8\xaf\xa6\xe6\x83\x85</td>\n            <td>\xe8\xaf\xa6\xe6\x83\x85</td>\n        </tr>\n    </table>\n\n</div>\n\n</body>\n\n</html>\n\n"
```

### 对象缓存技术

## 减少从数据库查询次数

```
public MiaoshaUser getById(long id) {
    //取缓存
    MiaoshaUser user = redisService.get(MiaoshaUserKey.getById, ""+id,
MiaoshaUser.class);
    if(user != null) {
        return user;
    }
    //取数据库
    user = miaoshaUserDao.getById(id);
    if(user != null) {
        redisService.set(MiaoshaUserKey.getById, ""+id, user);
    }
    return user;
}

public boolean updatePassword(String token, long id, String formPass) {
    //取user
    MiaoshaUser user = getById(id);
    if(user == null) {
        throw new GlobalException(CodeMsg.MOBILE_NOT_EXIST);
    }
    //更新数据库
    MiaoshaUser toBeUpdate = new MiaoshaUser();
    toBeUpdate.setId(id);
    toBeUpdate.setPassword(MD5Util.formPassToDBPass(formPass,
user.getSalt()));
    miaoshaUserDao.update(toBeUpdate);
    //处理缓存
    redisService.delete(MiaoshaUserKey.getById, ""+id);
    user.setPassword(toBeUpdate.getPassword());
    redisService.set(MiaoshaUserKey.token, token, user);
    return true;
}
```

## Redis对象缓存详情

```
[127.0.0.1:6379> keys *
1) "MiaoshaUserKey:id13000001001"
2) "OrderKey:moug13000000001_2"
3) "MiaoshaUserKey:tk289d5162729f403d941e3930a7d87ed2"
4) "OrderKey:moug13000001001_2"
5) "GoodsKey:gs1"
6) "GoodsKey:gs2"
7) "MiaoshaUserKey:id13000000001"
```

## 4.rabbitmq处理订单

## mq代码

```
public Queue miaoShaQueue(){
    return new Queue(MIAOSHA_QUEUE,true);
}

public void sendMiaoshaMessage(MiaoshaMessage mm) {
    String msg = RedisService.beanToString(mm);
    log.info("send message:"+msg);
    amqpTemplate.convertAndSend(MQConfig.MIAOSHA_QUEUE, msg);
}

public void receive(String message) {
    log.info("receive message:"+message);
    MiaoshaMessage mm = RedisService.stringToBean(message,
MiaoshaMessage.class);
    MiaoshaUser user = mm.getUser();
    long goodsId = mm.getGoodsId();

    GoodsVo goods = goodsService.getGoodsVoByGoodsId(goodsId);
    int stock = goods.getStockCount();
    if(stock <= 0) {
        return;
    }
}
```

## 业务逻辑代码

```
public Result<Integer> miaosha(Model model,MiaoshaUser user,
                                @RequestParam("goodsId")long goodsId,
                                @PathVariable("path") String path) {
    model.addAttribute("user", user);
    if(user == null) {
        return Result.error(CodeMsg.SESSION_ERROR);
    }

    //内存标记, 减少redis访问
    boolean over = localOverMap.get(goodsId);
    if(over) {
        return Result.error(CodeMsg.MIAO_SHA_OVER);
    }
    //预减库存
    long stock = redisService.decr(GoodsKey.getMiaoshaGoodsStock,
""+goodsId);//10
    if(stock < 0) {
        localOverMap.put(goodsId, true);
        return Result.error(CodeMsg.MIAO_SHA_OVER);
    }
}
```

```

        //判断是否已经秒杀到了
        MiaoshaOrder order =
orderService.getMiaoshaOrderByUserIdGoodsId(user.getId(), goodsId);
        if(order != null) {
            return Result.error(CodeMsg.REPEATE_MIAOSHA);
        }
        //入队
        MiaoshaMessage mm = new MiaoshaMessage();
        mm.setUser(user);
        mm.setGoodsId(goodsId);
        sender.sendMiaoshaMessage(mm);
        return Result.success(0); //排队中
    }

    public Result<Long> miaoshaResult(Model model, MiaoshaUser user,
                                     @RequestParam("goodsId") long goodsId) {
        model.addAttribute("user", user);
        if(user == null) {
            return Result.error(CodeMsg.SESSION_ERROR);
        }
        long result = miaoshaService.getMiaoshaResult(user.getId(), goodsId);
        return Result.success(result);
    }

```

## 秒杀队列



Overview Connections Channels Exchanges **Queues** Admin

### Queue miaosha.queue

▼ Overview

Queued messages last minute ?



Message rates last minute ?

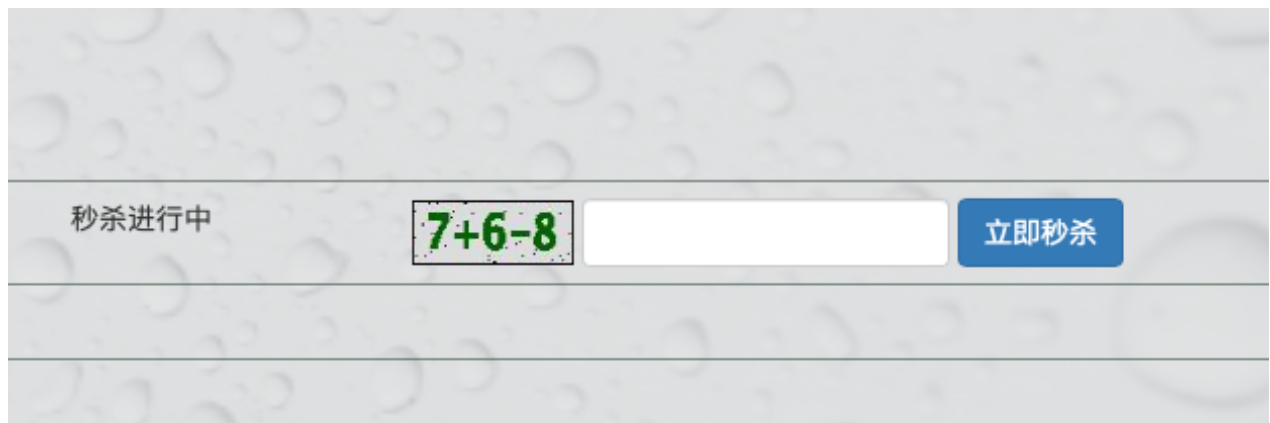


## 5.图像验证码及接口处理

### 图形验证码

目的是降低秒杀瞬间的高并发

### 实现效果



### 实现代码

```
public BufferedImage createVerifyCode(MiaoshaUser user, long goodsId) {
    if(user == null || goodsId <=0) {
        return null;
    }
    int width = 100;
    int height = 32;
    //create the image
    BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();
    // set the background color
    g.setColor(new Color(0xDCDCDC));
    g.fillRect(0, 0, width, height);
    // draw the border
    g.setColor(Color.black);
    g.drawRect(0, 0, width - 1, height - 1);
    // create a random instance to generate the codes
    Random rdm = new Random();
    // make some confusion
    for (int i = 0; i < 50; i++) {
        int x = rdm.nextInt(width);
        int y = rdm.nextInt(height);
        g.drawOval(x, y, 0, 0);
    }
    // generate a random code
```

```

        String verifyCode = generateVerifyCode(rdm);
        g.setColor(new Color(0, 100, 0));
        g.setFont(new Font("Candara", Font.BOLD, 24));
        g.drawString(verifyCode, 8, 24);
        g.dispose();
        //把验证码存到redis中
        int rnd = calc(verifyCode);
        redisService.set(MiaoshaKey.getMiaoshaVerifyCode,
            user.getId()+"."+goodsId, rnd);
        //输出图片
        return image;
    }

    public boolean checkVerifyCode(MiaoshaUser user, long goodsId, int
        verifyCode) {
        if(user == null || goodsId <=0) {
            return false;
        }
        Integer codeOld = redisService.get(MiaoshaKey.getMiaoshaVerifyCode,
            user.getId()+"."+goodsId, Integer.class);
        if(codeOld == null || codeOld - verifyCode != 0 ) {
            return false;
        }
        redisService.delete(MiaoshaKey.getMiaoshaVerifyCode,
            user.getId()+"."+goodsId);
        return true;
    }

    private static int calc(String exp) {
        try {
            ScriptEngineManager manager = new ScriptEngineManager();
            ScriptEngine engine = manager.getEngineByName("JavaScript");
            return (Integer)engine.eval(exp);
        }catch(Exception e) {
            e.printStackTrace();
            return 0;
        }
    }

    private static char[] ops = new char[] {'+', '-', '*'};
    /**
     * + - *
     * */
    private String generateVerifyCode(Random rdm) {
        int num1 = rdm.nextInt(10);
        int num2 = rdm.nextInt(10);
        int num3 = rdm.nextInt(10);
        char op1 = ops[rdm.nextInt(3)];
        char op2 = ops[rdm.nextInt(3)];
    }

```



```

        String exp = "" + num1 + op1 + num2 + op2 + num3;
        return exp;
    }
}

```

## 接口访问次数限制

```

    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler)
        throws Exception {
        if(handler instanceof HandlerMethod) {
            MiaoshaUser user = getUser(request, response);
            UserContext.setUser(user);
            HandlerMethod hm = (HandlerMethod)handler;
            AccessLimit accessLimit =
hm.getMethodAnnotation(AccessLimit.class);
            if(accessLimit == null) {
                return true;
            }
            int seconds = accessLimit.seconds();
            int maxCount = accessLimit.maxCount();
            boolean needLogin = accessLimit.needLogin();
            String key = request.getRequestURI();
            if(needLogin) {
                if(user == null) {
                    render(response, CodeMsg.SESSION_ERROR);
                    return false;
                }
                key += "_" + user.getId();
            }else {
                //do nothing
            }
            AccessKey ak = AccessKey.withExpire(seconds);
            Integer count = redisService.get(ak, key, Integer.class);
            if(count == null) {
                redisService.set(ak, key, 1);
            }else if(count < maxCount) {
                redisService.incr(ak, key);
            }else {
                render(response, CodeMsg.ACCESS_LIMIT_REACHED);
                return false;
            }
        }
        return true;
    }
}

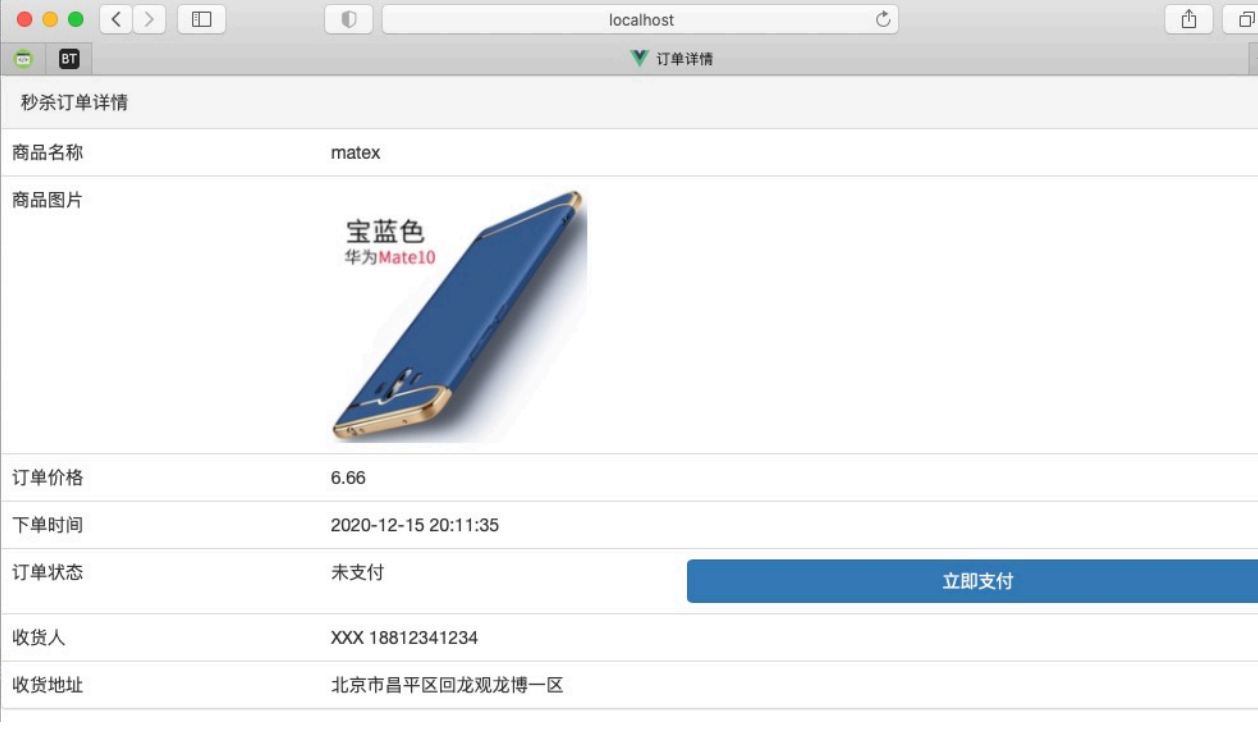
```

# 实现效果



# 项目总结

## 秒杀成功



## 数据库信息

id	user_id	order_id	goods_id
355407	13000000	355408	2
355408	13000001	355409	2

id	user_id	goods_id	delivery_addr_id	goods_name	goods_count	goods_price	order_channel	status	create_date	pay_date	
355408	13000000	2	(NULL)	matex	1	6.66		1	0 2020-12-11 23:	(NULL)	
355409	13000001	2	(NULL)	matex	1	6.66		1	0 2020-12-15 20	(NULL)	