# 云计算 1607C 田常平

# JAVA工程师

## 项目介绍

|  |
| --- |
| 我们当时主要做的是一个融信网p2p平台，平台是由借贷双方自由竞价，撮合成交。投资人获取利息收益。贷款人到期偿还本金。平台不吸储不放贷，仅收取手续费，比如提现手续费、还款利息管理费等。通过该平台，让用户更安全同时也更快的进行资金的借款和还款。系统主要包括用户开户、登录、个人中心、我要借款、我要理财，我要充值、我要提现。主要采用dubbo+zookeeper进行服务之间的调用。数据库采用的是mysql。数据访问层采用mybatis，控制层采用的是springmvc，安全登录采用springcecurity，使用JWT解决用户跨域访问问题。前端技术采用的是freemarker模板引擎以及jQuary+bootstrap技术。 |

## **功能**

|  |
| --- |
| 用户开户 当用户完成平台注册之后，使用httpclient调用存管银行接口，会让用户在存管银行进行实名认证，需要用户填写要绑定的银行卡，真实姓名等相关信息。存管银行会为用户开通存管账户。之后的用户借款、投资、充值、提现都是通过存管银行进行资金流转。 我要借款 首先判断借款人在借款申请前基本资料里的必填字段是否齐全，如不齐全，提示需要补充的信息，并跳转至基本资料中进行补充。新建时，需要提交借款信息：借款金额，借款用途，借款期限，还款方式（等额本息和等额本金）等。用户提交借款申请成功后，提示工作人员将在1~3个工作日内审核并联系。工作人员审核通过后，在后台进行借款信息上传，发布。 我要充值 用户通过点击我要充值，首先会判断用户是否绑定银行卡，如果没有绑定，需要提示用户去开户页面绑定银行卡。进入充值页面，页面分为快捷支付，支付宝充值以及跨行转账，当时快捷充值金额0.15%收取手续费，当时考虑到成本问题，将快捷支付放到了最后一个。  当用户点击我要充值时，   1. 初始化阿里支付：根据支付宝的网关路径以及app\_id、商户私钥和支付宝公钥、编码类型等 2. 设置请求参数，也就是设置支付宝异步通知的路径nitify\_url和同步通知的return\_url   3.传递订单参数，包括使用雪花算法生成一个当前充值的订单id、充值的金额，对于充值的描述、最晚的付款时间，还有product\_code,仅支持FAST\_INSTANT\_TRADE\_PAY。通过setBizContent,以json字符串的形式存到请求里面。之情请求，返回支付宝的支付页面。用户在支付宝上完成充值之后通过之前的notify\_url回调平台，给平台返回参数，包括订单信息，签名以及支付宝订单号等。平台首先会对签名进行验签，防止伪造支付宝的响应参数。验签成功后。最后平台增加修改用户账户的可用金额。通知用户充值成功。 我要提现 用户点击我要提现时，会跳转到提现列表，列表上有用户的真实姓名，提现金额，实际到账金额，用户的手续费，用户完成提现申请后，后台会根据用户的提现申请进行审核，首先当前用户不能有多个提现申请，还有就是当前实际提现金额不能大于账户的可用余额，当审核通过后，会提示用户提现申请审核成功，跳转到存管银行的提现页面，页面上同样也有用户的真实姓名，提现金额，实际到账金额，用户的手续费以及银行卡号等。提现成功后，存管银行会将用户的存管账户钱直接打到用户提供的银行卡上，平台实时监控用户账户资金变化，通过存管银行反馈消息，修改用户在平台上的虚拟账户资金。 风控资料上传（不是自己负责的） 这个模块不是我做的，当时做借款模块时，也涉及到了对风控资料的判断。首先，借款之前完善个人资料，以方便平台进行审核，主要包括个人信息、婚姻状况（已婚、未婚）、子女状况、月收入范围，住房相关等信息。一旦实名认证通过，不能更改信息。所以要谨慎填写。  资料上传：本人身份证原件（正、反两面照片）、个人信用报告、最近连续3个月收入证明、房产证明、住址证明、车产证明、结婚证明、学历证明、个人头像。【注：资料上传时选择下拉列表内容，鼠标触发时展示提示信息，资料上传内容参见excel.】，上传图片使用的是uploadify.js插件上传。 我要还款（不是自己负责的） 还款模块：借款成功之后，会生成还款对象和回款对象。借款人根据还款对象来分期，首先判断当前用户账户余额是否大于还款金额，小于则提醒用户账户余额不足，去充值界面充值，假设账户金额大于还款金额，点击还款时，首先要冻结借款人的还款金额，银行会根据还款对象的还款金额转到各自的投资人的账户，还款成功后会减少借款人的账户金额，增加投资人的账户金额，增加平台手续费的账户金额。将还款成功的消息发送给平台，平台相应的改变账户金额，通知用户，还款成功。因为银行清算需要时间，所以要提醒借款人还款时间要早于最晚的标的还款时间，不然有可能造成因银行清算造成还款逾期。平台会对逾期的项目有个保证期，以15天为例，如果用户在保证期期间还款了，会支付相应的利息，也就是逾期罚息。如果在保证期间没有还款，就会进入线下催收。 |

## 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 借款信息是怎么展示的 |
| 我们在项目中首页展示投标信息时用到了redis。因为投标信息比较多，属于热数据，所以我们当时考虑使用redis来存储。key是string类型的bidrequest字符串，value是一个hash，key为投标信息的id，value为投标信息对象。通过redisTemplate.opsForHash().put()方法存储。当用户投标信息审核成功，准备发布时，会将投标信息存储到redis里面。当在前台页面展示时，会首先redis里面读取，如果redis里面没有，则从mysql里面读取，如果有数据，则将数据存到redis里面并返回，如果没有数据，会在redis里面给对应的key对应的value设置一个null值 |
| 用户的安全登录时怎么实现的 |
| 我们使用springsecurity实现用户的安全登录。编写配置文件spring-security.xml里面。我记得有个http的标签，可以对静态资源放行，并设置登陆成功的跳转页面，默认的登录页面。当用户发送登录请求时，通过配置文件。我们自定义一个认证类，实现UserDetailsService接口。重写loadUserByUserName方法，然后创建角色列表，将在配置文件里面配置的角色放到列表里面，根据用户名去数据库查询是否存在该用户，如果存在将用户的密码与输入的密码进行比对，返回userDetails。这样就完成了安全登录。 |
| 验证码的消息是怎么获取到的 |
| 我们把用户的手机号，随机生成的四位验证码和在阿里云注册时的签名、模板等参数存到map里面，通过rabbittemplate存到RabbitMQ里面，消费者通过RabbitListener注解监听生产者，一旦监听到消息，获得到参数，就调用阿里大于的短信接口，完成发送短信的功能。 |
| 你们是怎么防止别人恶意调用短信接口的 |
| 我们项目上线之后，线上用的短信接口被攻击了,我们的用户就收到了很多条垃圾短信，然后就被用户投诉,当时借鉴了大多数网站的设计，我们在点击获取验证码按钮之前，需要填写一个图片验证码，来防止脚本恶意攻击。 |
| 手机发送验证码 |
| 当用户点击我要注册时，会跳转到注册页面，页面上需要输入用户的手机号，获取验证码，我们使用了阿里大于和云通讯两个短信平台来实现发送验证码。就比如说阿里大于，我们调用阿里大于接口，传递手机号，随机的四位验证码，和在阿里云注册时的签名、模板等参数，来完成验证码的发送。为了防止用户多次点击获取验证码，我们把验证码以string类型形式存储到redis中，key是用户的手机号，value是获取的验证码。存储redis有四个参数，key、value、过期时间和时间类型（小时、分钟、秒等）我们当时设置的验证码过期时间为60秒。在60秒之内用户不可以对同一个手机号重复发送验证码。当用户输入验证码之后，会向后台发送请求，判断验证码是否正确。因为当前验证码存到了redis里面，所以我们从redis里面获取验证码，同时设置了过期时间，所以也能防止用户输入过期的验证码。 |
| JWT token验证 |
| 为了保证接口的安全性，我们使用JWT实现跨域访问，当用户登录成功后，后台生成一个token，就是一个String字符串，这个token包含一些信息，比如当前用户的唯一标识id，用户的手机号，Token的签发时间以及token的过期时间。过期时间我们设置为7200秒。  如果登录成功，将这个登陆成功状态信息和token一起返回给前台。前台获取到这个token之后，每次请求接口时，都在请求头携带者token。我们在后台通过拦截器拦截到前台的请求，解析token，如果解析成功，将token重新放到request里面，当调用后台接口时，会首先判断request里面有没有token，如果token存在，可以做接下来的操作，如果不存在，则向前台响应对应的状态码以及错误提示。 |
| 在集群模式下怎么保证定时任务不会重复执行？ |
| 我们项目里面没有用到定时任务，我就说一下我的看法，我们在定时任务执行之后，我们会项目的名称，和ip以key-value形式存到redis里面，并设置合理的过期时间，这样当其他定时任务触发之前，就先要从redis里面判断，是否存在当前的key-value，如果存在，就不让定时任务执行。这样就可以解决重复执行的问题 |
| 分布式的架构下怎么保证id是唯一的 |
| 我们在项目里面使用的是雪花算法。使用的是idword工具类创建id，如果要求不高的话，可以使用时间戳来定义id。 |
| redis解决缓存失效和缓存穿透问题 |
| 首先先说下缓存失效，在同一时间有大量的key同时失效，瞬间加大了mysql的压力。解决办法就是存储key时失效时间要错开。  缓存穿透：他指的是用户访问的数据从redis里面查不到，从mysql里面查，如果重复访问，会造成mysql的压力加大。所以就要要求我们第一次从mysql里面查的时候，如果没有数据，就在redis把对应的value设置为null值。 |

## 项目技术亮点分析

|  |
| --- |
| **Dubbo+Zookepper：**首先融信网是一个分布式项目，根据需求拆分了对各模块，web层要调用service层的服务，那么就需要远程服务调用，dubbo就是分布式服务框架，是远程服务调用（RPC）的一种方案，就像调用远程方法一样，优点很多，比如说配置简单，只需在spring加载Dubbo的配置，软负载均衡及容错机制zookepper适用于分布式中一致性处理的框架  1.生产者在启动时向注册中心注册自己提供的服务  2.服务消费在启动时，向注册中心订阅自己所需的服务  3.注册中心返回服务提供者地址表给消费者  4.服务消费者从服务者地址列表，基于软负载均衡算法，选一个提供者进行调用，如果调用失败，在选一台调用  5.服务消费者和提供者在内存中累计次数和调用时间，定时每分钟发送一次统计数据到监控中心  **Redis**：我们在项目里面主要把redis作为缓存使用，查询数据更快，能够提高用户的体验，更大程度上减轻了mysql数据库的压力。  **RDB持久化**  RDB 是在某个时间点将数据写入一个临时文件，持久化结束后，用这个临时文件替换上次持久化的文件，达到数据恢复。  优点：使用fork出子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 redis 的高性能  缺点：RDB 是间隔一段时间进行持久化，如果持久化之间 redis 发生故障，会发生数据丢失。所以这种方式更适合数据要求不严谨的时候。  **AOF持久化**  Append-only file，将“操作 + 数据”以格式化指令的方式追加到操作日志文件的尾部，在 append 操作返回后(已经写入到文件或者即将写入)，才进行实际的数据变更，“日志文件”保存了历史所有的操作过程；当 server 需要数据恢复时，可以直接 replay 此日志文件，即可还原所有的操作过程。AOF 相对可靠，它和 mysql 中 bin.log、apache.log、zookeeper 中 txn-log 简直异曲同工。AOF 文件内容是字符串，非常容易阅读和解析。  优点：可以保持更高的数据完整性，如果设置追加 file 的时间是 1s，如果 redis 发生故障，最多会丢失 1s 的数据；且如果日志写入不完整支持 redis-check-aof 来进行日志修复；AOF 文件没被 rewrite 之前（文件过大时会对命令进行合并重写），可以删除其中的某些命令（比如误操作的 flushall）。  缺点：AOF 文件比 RDB 文件大，且恢复速度慢。  **RabbitMq：**是一套开源的消息中间件，通过rabbitmq在项目中实现了消息的生产和消费，有三种模式，包括直接模式、分裂模式以及topic模式。项目中采用的是直接模式。他的主要功能是削峰、异步、解耦我们之前会在代码里面直接调用阿里云接口发送短信，这样会造成程序的耦合性非常高。而将消息放到rabbitmq里面，消费者监听消息，调用阿里云接口发送验证码。能够将程序解耦，更容易维护。 |

## 项目答辩中亮点简历审核评定标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考核内容** | **简历样式是否规范** | **项目内容是否完整** | **项目中是否有亮点** | **技术难点解决办法是否高效** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |
| 简历是否合格 | | | | 合格 | 不合格 |

## **项目答辩成绩评定标准**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考核内容** | **亮点简历与项目描述是否匹配** | **项目流程是否清晰** | **学生讲解是否流畅** | **项目介绍是否到位** | **技术亮点考核是否合格** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |  |
| 项目答辩是否合格 | | | | 合格 | 不合格 |  |