# 云计算 1607C 吕光

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 紫枫信贷是一个p2p金融平台，由借贷双方自由竞价，撮合成交。我们平台主要包括借款人、投资人和平台管理人员，资金借出人获取利息收益，并承担风险；资金借款人到期偿还本金，平台管理人员通过审核相关流程完成借款、投资等业务，平台从中收取中介服务费。这个平台主要分为前台后台两个系统。前台系统主要包括开户登录，个人中心，借款资质申请，我要借款，充值，投标，提现等功能。后台系统主要包括用户管理、安全管理、审核管理、平台管理等，其中最重要的是审核管理，它是对借款人资质审核，以及对账户资金变动的相关审核，比如发标前审核、满标一审、满标二审等功能，满足消费金融公司快速、高效、便捷的借款业务需求。项目中使用的是SpringMVC、Spring和MyBatis技术框架，MySQL存储数据，使用Dubbo和Zookeeper实现服务之间的调用，前端使用jQuery、FreeMaker和Bootstrap实现页面展示，使用Redis对数据进行缓存处理。 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| **负责模块：**   1. **我要借款：**   如果我们的用户想要借款，首先会通过前台页面点击我要借款跳转到我要借款首页，首先我们会判断用户是否登录，若是用户在登录的状态下跳转到我要借款首页，那么，我们会先判断用户是否满足申请借款的条件，借款条件有是否填写个人资料、是否进行实名认证、风控资料分数是否达到可借款分数，以及是否进行视频认证，如果有任何一个条件没有满足，点击申请贷款都没有任何反应，在没有满足的条件后面都会有个×，并且点击相应的连接，就能跳转到做这个认证的界面；如果都满足了这些申请条件，用户就可以申请借款，如果当前用户已经有一个借款申请还在申请流程中，那么就不能再次申请借款。填写完借款相关信息之后将数据传到我们的后台，比如借款金额、借款利息、借款期限、还款方式、最小投标等一些信息，那么我们的后台会再次判断借款人是否满足我们的借款条件和当前没有借款在审核流程中，然后再判断借款申请信息中的借款金额是否大于平台最小借款金额和是否小于剩余信用额度；借款利息是否在5%和20%之间；最小投标金额是否大于平台规定的最小投标金额等。如果这些条件都满足的话，那么会生成一个借款对象保存到数据库中，然后再为当前借款人添加借款状态码，代表当前用户在借款中。这时我们的后台管理人员登录后台管理系统点击发标前审核就会出现申请借款的相关信息，后台管理人员通过审核借款人的申请信息进行审核通过或者审核拒绝操作。如果审核通过修改该借款信息状态为招标中，在设置相应的风控意见，前台投标页面就会展示这个标的信息，如果审核拒绝，修改标的状态为发标前审核拒绝，然后再去掉用户借款状态码，这是在用户登录的状态下。若是没有登录，展现给用户的是个静态页面，点击申请借款没有任何效果，这就是借款发标前相关操作。   1. **我要投标：**   如果投资人想投资的话，在前台客户系统点击我要投资就会跳转到我要投资主页，在该页面可以看到正在招标中的招标信息，投资人可以点击查看按钮去查看该标的详细信息，包括借款金额、年化利率、可得利息、借款人信息、材料信息、还款情况等信息，投资人可以根据标的信息选择合适的标进行投标。如果投资人和该标的借款人是同一人，那么在标的详情页面展示的是撤销借款按钮而不是马上投标按钮，我们平台是不允许自己给自己投标的，当投资人想要投资的时候就可以在相应的投资框输入投资金额进行投标，我们平台规定每个投资人最多对一个标的投标次数不超过5次，并且对一个标的总投标金额不超过该标的总金额的20%，这也是为了让其他投资人进行投标并获得收益。投资人输入投标金额后前台会对投标金额进行校验，如果投标金额大于该标所需的金额或者小于最小投标金额，那么会有提示用户投标金额超过借款标金额或者投标金额小于最小投标金额等信息。在输入符合的投标金额后点击马上投标按钮，弹出提示框让投资人输入交易密码进行投资，之后我们的后台会对传入的投标金额进行判断，判断该标的借款状态是否在招标中，当前用户的账户金额余额是否大于等于投标金额，投标金额是否大于等于最小投标金额，投标金额是否小于等于借款剩余投标金额，还有当前用户不是借款的借款人，在满足这些条件后调用第三方存管银行（宜宾银行）提供的接口，在投资人账户中冻结本次投资金额，减少账户可用余额，操作完成后通知我们平台操作成功，然后我们平台会生成一个投标对象保存到数据库当中，修改投资人的账户信息（账户可用余额减少，冻结金额增加）并且生成这次投标的一条投标流水，对该次投标做一次投标记录，接着修改借款相关信息（以投标次数，投标总金额）之后在判断该标是否已经投满，如果没投满产生流标，退还之前投资人的投资金额，如果已经投满，我们将该标的借款状态改为满标一审状态，等待后台管理员的满一审核。   1. **满标一审：**   对于借款主要包括几个重要的对象：bidrequest：借款信息，bid：投资人每次的投标记录，Accountflow：账户流水；比如说您想要在我这里借款，我是投资人，当我投资的金额达到您的借款金额时，就进入了满标一审的状态，在满标一审审核时，首先后台要对您的信息再次核对，是否存在当前借款信息，是否处于满标一审状态，如果满足条件，同时生成审核历史对象。如果不满足条件，前台提示用户不满足条件，进行资质审核如果都没有问题，当管理员点击审核通过按钮时，借款对象(bidrequest)的state状态就变成了满标二审状态。当管理员点击审核拒绝时，说明该借款进入了满标拒绝的状态，需要退还投资人的钱，因为涉及到了钱的变动，我们在这里有一个账户的流水对象accountflow，这个对象包含当前投资流水金额，当前流水的状态以及流水的备注信息。用于记录退还投资人的投资的流水，同时该借款人的个人信息中的借款状态移除。借款人可以再次发起借款，投资人也可以再次投资。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| **（1）难点：**借款人应还的利息和多位投资人应收利息不能达成一致？  在当时遇到比较难以解决的一个问题就是当我们后期测试的时候，发现当我们成功开标后，多个人同时投标，看标满之后的收益。结果发现借款人应还的利息和多位投资人应收利息不能达成一致，差了几分钱，当时也是用了很多方法，都没能解决，最后才发现是因为投资方和借款方调用算法的不同而造成的，这个问题也是告诉了我，我们在开始做项目之前，一定要把各方面都考虑一下，已免后期出现问题，影响效率。  **（2）难点：**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。  解决方法：  1.增加图形验证：即当用户进行“获取动态短信” 操作前，弹出图片验证码，要求用户输入验证码（通过后台随机生成图片\字符验证），然后在调用短信接口。  2.限制发送时长：建议采用限制重复发送动态短信的间隔时长，即当单个用户请求发送一次动态短信之后，服务器端限制只有在一定时长之后（此处一般为60秒），才能进行第二次动态短信请求。该功能可进一步保障用户体验，并避免包含手工攻击恶意发送垃圾验证短信。用户手机号发送到后台后，先在redis中判断手机号key是否存在，存在则不送。  3.单IP请求次数限制：对该用户在规定时间内如30分钟，访问次数是否超值做一个判断（如值为10次），使用redis.incr（key）自增判断实现，一旦用户请求次数（包括失败请求次数）超出设定的阈值，则暂停对该IP一段时间的请求;若情节特别严重，可以将IP加入黑名单，禁止该IP的访问请求。该措施能限制一个 IP地址的大量请求，避免攻击者通过同一个IP对大量用户进行攻击。 |

1. 项目技术亮点分析

|  |
| --- |
| **一．Dubbo+Zookeeper技术：**  **（1）Dubbo介绍：**  Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用，说白了就是个远程服务调用的分布式框架。告别Web Service模式中的wsdl,以服务者与消费者的方式在dubbo 上注册)。  其核心部分包含:  1.远程通讯，提供对多种基于长连接的NiO框架抽象封装，包括多种线程模型，序列化，以及“请求一响应”模式的信息交换方式。  2.集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持。以  及负载均衡，失败容错，地址路由，动态配置等集群支持。  3.自动发现:基于注册中心目录服务，使用服务消费能动态查找服务提供方,使地址透明,使用服务提供方可以平滑增加或减少服务器。  **（2）Zookeeper介绍：**  Zookeeper是一个分布式开源框架，提供了协调分布式应用的基本服务，它向外部应用暴露一组通用服务——分布式同步（Distributed Synchronization）、命名服务（Naming Service）、集群维护（Group Maintenance）等，简化分布式应用协调及其管理的难度，提供高性能的分布式服务。ZooKeeper本身可以以单机模式安装运行，不过它的长处在于通过分布式ZooKeeper集群（一个Leader，多个Follower），基于一定的策略来保证ZooKeeper集群的稳定性和可用性，从而实现分布式应用的可靠性。  **（3）Dubbo+Zookeeper工作原理：**  首先服务Container在启动时，它会将相关的服务注册到Zookeeper下，zookeeper注册中心具有服务的注册与发现的功能，本质上讲，他维护了一个服务的白名单，服务消费者在启动的时候,会向zookeeper注册中心订阅自己的服务，此时，注册中心返回服务提供者提供的地址列表给消费者.服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。所有提供者在zookeeper上创建的节点都是临时节点,利用的是临时节点的生命周期和客户端会话相关的特性,因此一旦提供者所在的机器出现故障导致该提供者无法对外提供服务时,该临时节点就会自动从zookeeper上删除,zookeeper会把最新的服务推送给消费者.消费者每次调用服务提供方是不经过zookeeper的,消费者只是从zookeeper那里获取服务提供方地址列表,存入内存中,所以当zookeeper宕机后,不会影响消费者调用服务提供者,影响的是zookeeper宕机之后,如果提供者有变动,增加或者减少,无法把最新的服务提供者地址列表推送给消费者。  服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。  **（4）Zookeeper选举机制：**  假设有五台服务器组成的zookeeper集群,它们的id从1-5,同时它们都是最新启动的,也就是没有历史数据,在存放数据量这一点上,都是一样的.假设这些服务器依序启动：  1) 服务器1启动,此时只有它一台服务器启动了,它发出去的报没有任何响应,所以它的选举状态一直是LOOKING状态。  2) 服务器2启动,它与最开始启动的服务器1进行通信,互相交换自己的选举结果,由于两者都没有历史数据,所以id值较大的服务器2胜出,但是由于没有达到超 过半数以上的服务器都同意选举它(这个例子中的半数以上是3),所以服务器1,2还是继续保持LOOKING状态。  3) 服务器3启动,根据前面的理论分析,服务器3成为服务器1,2,3中的老大,而与上面不同的是,此时有三台服务器选举了它,所以它成为了这次选举的leader。  4) 服务器4启动,根据前面的分析,理论上服务器4应该是服务器1,2,3,4中最大的,但是由于前面已经有半数以上的服务器选举了服务器3,所以它只能接受当小弟的命了。  5) 服务器5启动,同4一样,当小弟。  **（5）项目中如何使用Dubbo+Zookeeper：**  要想实现dubboServer和dubboClient之间完成调用，首先服务端和客户端需要一个接口来公用。可以有两种方式：  1)接口写在一个应用中，服务端和客户端都对这个应用进行引用  2)更多的时候，两个应用是不太可能引用同一个应用。这个时候应该把接口以jar包的形式给Server和Client共同使用，然后，Server对jar包中的接口进行实现，并且向dubbo进行注册，而Client端调用jar包中的接口中的方法名，dubbo会去自动的寻找这个由Server端注册的方法的实现（如果使用zookeeper则通过spring的<dubbo:registry address=”\*\*”/>标签，将服务发布，客户端也使用该标签实现订阅）。   1. **Redis技术：** 2. **Redis简介：**   redis是一种内存型、非关系型数据库；redis的数据会存储到硬盘上；它保存数据一种采用的是rdb(快照的形式进行数据的保存，是redis默认开启的方案，它将内存中的数据都保存到磁盘上，放到rdb文件中，重新启动服务器是将rdb的内容重新取出来,再回复到内存中。还有一种是aof默认是不开启的，因为他会频繁的操作磁盘，每秒钟同步一次，所以默认是不开启的。开启之后数据的恢复就是用aof。  **（2）Redis五种数据类型：**  String、List、Hash、Set、Zset  **（3）Redis持久化方式：**  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：使用单独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。  **（4）Redis分布式集群：**  Redis集群中至少应该有三个节点。为了实现容错机制（所有的redis节点彼此互联(PING-PONG机制)每个节点都有一个FAIL，因为是通过ping-pong机制来进行互联，所以当一个节点没有返回pong的时候，标记为pfail,通过集群中超过半数的节点检测失效时才生效变为PFAIL,就代表本节点失效），要保证集群的高可用，需要每个节点有一个备份机。Redis 集群中内置了 16384（0-16383个哈希槽（slot），每个节点维护部分槽及槽所映射的键值数据，集群所搭建的服务器平均分配了这16384个hash槽，所以每当我们添加一个key-value 时，redis 先对 key 使用 crc16 算法算出一个结果，然后把结果对 16384 求余数，这样每个 key 都会对应一个编号在 0-16383 之间的哈希槽（例：一个集群里面有M1-S1、M2-S2、M3-S3六个主从节点，其中节点 M1包含 0 到 5460号哈希槽，节点M2包含5461 到 10922 号哈希槽，节点M3包含10923 到 16383号哈希槽。如果是M2宕掉，集群便会选举S2为新节点继续服务，整个集群还会正常运行。当M2、S2都宕掉了，这时候集群就不可用了），然后重定向到包含这个hash槽的节点，继续添加的时候，又会算相应的hash槽，概率基本上是每添加一个key跳转到不同的节点，数据均匀的分布在不同的节点上，这样实现了每个节点都有一定数量的数据，如果主节点挂了那从节点就顶上来，这样就实现了高可用。  **（5）Redis在项目中应用：**  发送手机验证码：  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，因为设置了过期时间，所以也能防止用户输入过期的验证码。 |
|  |

1. 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 |

1. 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 |  |