# 云计算 1607C 翟鸿业

# JAVA工程师

项目介绍

|  |
| --- |
| 项目背景是帮助运营方搭建网络借贷信息的中介平台，它的特点是可将社会中非常小的资金聚集起来，提高资金使用率。该P2P项目分为前台网站和后台管理系统，主要的模块分为三大模块，前台交易平台、后台业务支撑系统、个人用户中心。  项目中使用的是SpringMVC、Spring和MyBatis技术框架，RESTful接口设计风格，采用Maven聚合方式构建,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,权限管理使用的是Apache Shiro。  我主要负责的模块是 个人用户中心，我的账户管理、资金管理、投资管理、贷款管理模块。 |

1. 功能模块介绍

|  |
| --- |
| 我的账户管理：  我的主页（个人中心的首页，个人主要投资、借款、还款等信息和管理的入口。）  安全信息（密码找回、个人图像、联系方式、邮箱绑定、实名认证等）  账户信息（绑定银行卡、银行）  站内消息（收到的站内提醒） |
| **我的主页**  个人基本信息的展示,账户的基本信息的展示,图形化的资产分布,使用户更加直观的查看账户资金的流向,这里有一部分图形化的展示是通过HTML 5 Canvas来完成的,数据展示一般都用是Angular Js 数据双向绑定来打到数据展示的功能。投资、借款、还款等入口，就是A标签实现在网站内的功能跳转。  **安全信息**  里面涉及了个人用户的手机号修改(方便用户修改绑定的手机号还有密码的修改)、邮箱的绑定，实名认证等功能。  （不问不说）  邮箱绑定用到了（佑友第三方）邮箱服务器。将用户的信息和邮箱添加到数据库的同时添加一个UUID字段,然后通过spring MAil,来定义邮件的内容，将邮件的内容定义一个超链接，发送。用户收到包含UUID的链接后，和数据库中的UUID互相比对，如果成功验证绑定成功。  因为P2P行业的特殊性，为了防止他人用虚假的身份来借贷，实名认证功能比较繁琐，首先采用审核视频认证，用qq等第三方软件,验证成功后,后台人员再将用户的信息录入到能提供该服务的有全国公民身份证号码查询服务中心及某某通等，上传姓名、身份证号码要素，对方返回是否一致,认证周期一般在一到两天左右。  **账户信息**  绑定银行卡：因为我们的项目采用了银行管存的方式,绑定银行卡之前首先在银行的管存交易平台开一个二级账户,用户的银行卡,资金等信息,都是通过调用管存银行的接口,来实现投资,充值等功能,我们拿到银行的回调值后,将虚拟数据再添加到平台上。此银行管存平台支持很多大众银行卡的绑定。和我们对接的是花旗银行。 |
| **资金管理**  账户总资产分布展示,冻结金额、提现在途、投资后的资金等，详细划分各种资产的类型，将Mysql中，用户资产表的数据通过用户名查询出来，通过angular JS 展现在前台静态页面上（双向绑定），方便用户直观的查看账户总资产数据。月度账单的详细分析，将到账回报、回款本金、还有出借金额数据以条形统计图的形式展现出来，此功能只不过只显示本月的账单详细分析，点击的时候，后台只查询最近一个月的数据。（先从Redis中查询，如果没有再从数据库中查询）。 |
| **投资管理**  这模块要功能主要有用户回报总览,回报的详情,账户的充值、提现功能，交易详情。对用户的资产流向,收支明细、充值记录、提现记录数据表格的查询，资金的明细的数据量巨大，数据十分的重要，为了防止用户的资金明细数据丢失，必须定时备份（定时器)。  因为我们平台采用的是第三方+银行专户管理的支付方式,银行管存,用户在充值前首先在管存银行绑定银行卡，在绑卡的同时存管平台会给用户开启一个虚拟子账户，用户充值，提现等功能。 |
| **借贷管理**  我要贷款: 在页面填写相关借款信息，包括借款金额（不超过信用额度），利息，还款期限，最小投标数，借款描述信息等，其中利息和最小最大、借贷金额需要在跳转页面前从后台获取平台默认的最小最大利息值，页面填写好相关信息之后，参数通过表单传到后台，首先要去判断当前用户id是否是登录用户，资金是否正常，然后借款利息，金额，期限是否都是有效的参数，没有问题的话就在数据库生成标的记录，并将用户状态改为有一个借贷流程进行的状态，等待后台审核。  我要投资: |
| **风控系统（了解）**  1 贷前风控  对于信贷审批，一般流程是这样的：  借款人提交资料申请借款后，先是在平台风控系统中进行一次反欺诈过滤，对客户信息进行核查和预筛选。根据过滤后的各项数据评估借款人的信用资质。是否给用户授信以及相应的额度和费率。  然后审批人员再次对客户的借款资料、比如身份信息、收入证明、工作信息进行核查。主要资产证明以及其查证方法、信用报告及其他，并且在调查的时候这些资料还会相互交叉验证。  作为风控流程的第一步，必须严格核实所有资料的真伪，同时判断出各种资料及数据之间的关联，交叉验证逻辑一致，才能把好风控的第一道关卡，为后续的风险审核提供准确的评审和决策依据。  2 贷中管理  贷中管理，平台会安排专人，跟进借款人的贷前、贷中、贷后检查情况及进程。确保出借人的借款资金能按期回款，并及时发现和反馈的项目潜在的风险。  对于项目存在逾期风险的情况，平台在评估分析后，会根据风险分析结果，结合风险发生的原因选择风险应对方案：规避风险、减少风险或分担风险、接受风险。  3 贷后催收  无论一个P2P平台号称自己的风控多牛，也不可能保证每一位客户都不逾期，也不能保证没有任何坏账。平台把钱借出去，不可避免会有一部分借款人出现逾期，这就需要催收。  具有一定规模的P2P平台一般都会有自己的催收团队，遇到实在催收不回来的单子（比如M3）就会外包第三方。外包费用大概按催收金额的10%-50%不等，价钱因素是由逾期时间，欠款金额决定。  以上三个环节就是一个大致的风控流程，涉及到贷前、贷中、贷后三个阶段。  **P2P平台的风险保障模式一般有这么3类：1.大数据构建风控模型；2、第三方担保保障机制、3.保险公司履约险保障。**  1 大数据构建风控模型：  Ⅰ. 反欺诈模型：适用于个人信贷，因为小额分散，借款人还款能力不是核心问题，主要风险是还款意愿（比如老赖）。因此大多数的P2P平台，大数据风控90%的价值在于反欺诈。  Ⅱ. 评分卡模型：评分卡模型的核心价值是量化定价，包括授信额度、贷款期限、利率等。主要工具就是评分卡，先给客户信用评分定级，然后不同级别不同利率。  2 第三方担保保障机制：平台跟第三方担保公司进行合作，为平台的借款项目提供担保服务，在平台的项目逾期时，由第三方担保机构为投资人的本息提供全部或者部分赔付。  3 保险公司履约险保障：所谓履约保证保险，是指保险公司向履约保证保险的受益人(出借人)承诺，如果债务人（借款人）不按照合同约定或法律规定履行还款义务，则由该保险公司按照保单约定赔付出借人本息。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| **分布式事物的解决,如何保证业务事物的一致性**  布式系统的核心就是处理各种异常情况，这也是分布式系统复杂的地方，因为分布式的网络环境很复杂，这种“断电”故障要比单机多很多，所以我们在做分布式系统的时候，最先考虑的就是这种情况。这些异常可能有 机器宕机、网络异常、消息丢失、消息乱序、数据错误、不可靠的TCP、存储数据丢失、其他异常等等... 解决方法：**分布式事务解决方案CAP** CAP定理是由加州大学伯克利分校Eric Brewer教授提出来的，他指出WEB服务无法同时满足一下3个属性：  一致性(Consistency) ： 客户端知道一系列的操作都会同时发生(生效)  可用性(Availability) ： 每个操作都必须以可预期的响应结束  分区容错性(Partition tolerance) ： 即使出现单个组件无法可用,操作依然可以完成  CAP 是一个在分布式系统中（SOA，MicroService）实现事件总线及最终一致性（分布式事务）的一个开源的 C# 库，她具有轻量级，高性能，易使用等特点。 **2.消费端弄丢了数据** RabbitMQ 如果丢失了数据，主要是因为你消费的时候，刚消费到，还没处理，结果进程挂了，比如重启了，那么就尴尬了，RabbitMQ 认为你都消费了，这数据就丢了。  这个时候得用 RabbitMQ 提供的 ack 机制，简单来说，就是你必须关闭 RabbitMQ 的自动 ack，可以通过一个 api 来调用就行，然后每次你自己代码里确保处理完的时候，再在程序里 ack 一把。这样的话，如果你还没处理完，不就没有 ack了？那 RabbitMQ 就认为你还没处理完，这个时候 RabbitMQ 会把这个消费分配给别的 consumer 去处理，消息是不会丢的。   1. ****大量的使用缓存，对于缓存服务器，也有很大的压力，有时候Redis 压力比mysql还要大很多，思考如何减少Redis的访问？****   一般抢标，标的数量也少，标在100单左右，但是并发量可能达到几千。  在Redis预减库存的时候，内存中维护一个isOvermap作为一个标记，当没有库存的时候，将其置为true。每次抢标业务 访问Redis之前，查一下map,true说明没有库存，就直接返回No\_stock。  4.****大量的使用了缓存，那么就存在缓存的过期时间控制以及缓存击穿以及缓存雪崩等问题？****  **解决：首先针对不同的缓存设置不同的过期时间，比如session缓存，在userKey这个前缀中，设置是30分钟过期，并且加入一层再登陆增加缓存时间的机制。这样每次取session,都会延长30分钟，相对来说，就减少了缓存过期的几率。**  **针对热点数据，比如优惠、高反、汇报高的标详情信息，热点标由于考虑到是一般抢标10分钟内几乎抢完，于是就设置为10分钟的缓存。**  **针对热点数据的缓存击穿问题，万一一波一波的抢标，（高反标）这种，某个时间点万一大量并发，刚好我的这个标的缓存时间过了，去访问数据库。对于这种热点数据，我将过期时间一起存入缓存中，取出来的时候，比对一下过期时间和当前时间，少于1分钟，我就更新一下缓存，防止他过期。** |

1. 项目技术亮点分析

|  |
| --- |
| **利用缓存减少数据库的压力，以及读取缓存的速度远远快于数据库（网络时延+IO）。** 本项目大量的利用了缓存技术,包括用户信息缓存（分布式session），标信息的缓存，数据字典的缓存，登录日志的缓存等，减少了对数据库服务器的访问。  **1.用户信息缓存引出：分布式session。**  我们知道当服务器集群的时候，若用户第一个请求在第一台服务器上，第二个请求在其他服务器上，会出现session的丢失的情况，丢失用户信息。而且在这种高并发场景下，一定是很多服务器同步工作，所以如何解决session分布式的问题是一个重点。  本项目采用：利用redis缓存的方法，另外布置一个Redis服务器专门用于存放用户的session信息。这样就不会出现用户session丢失的情况。（每次需要session，从缓存中取即可）  这种方式的优点：相对其他的分布式方式，   1. 服务器文件同步（不建议使用，这样会造成文件重复，资源浪费） 2. session存数据库（不建议用，会加大数据库压力） 3. 使用cookie（不建议用，cookie不太安全）   对大量的缓存引用也出现了一个问题，如何识别不同模块中的缓存（key值重复，如何辨别是不同模块的key）。 引出：通用缓存key封装  **2.通用缓存key封装**  利用一个抽象类，定义BaseKey（前缀）,定义了缓存的String prefix(前缀) 以及缓存的过期时间。让不同模块继承它。这样每次存入一个模块的缓存的时候，加上这个缓存特定的前缀，以及可以统一制定不同的过期时间。  **3.页面静态化以及前后端分离**  页面静态化的主要目的是为了加快页面的加载速度。做法：将标的详情页面做成静态HTML，放在CDN（减少了服务端的压力）上做为静态数据发送给用户端，而数据信息通过前端ajax 异步发送请求来获取。只获取动态数据信息部分，加载速度可以达到全部渲染的2倍。 |

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 |  |