# 一 数据库表的解读

系统中的Mysql数据库名字叫userServer,userServer中的每个表的解释都在下列文件中

[[金山文档] 用户服务器数据库解释.xls](https://kdocs.cn/l/coAw624EK2fc)

# **二 各个模块的基本流程**

## 邀请码模块

1. 邀请码模块是负责邀请码的创建和使用的模块，涉及到的实体有 邀请码模板，行为动作，邀请码。在后端代码中 所有的接口在com.mr.inviatecode.controller 包下面。

邀请码用于处理系统中的难以用代码直接描述的功能。运营模式为 后台管理员邀请码创建，创建之后将邀请码代理给代理商，代理商将邀请码分配给用户，用户在门户网站填写邀请码，后台识别邀请码后自动执行邀请码附带的功能。

除此外，邀请码还用于套装购买服务，该服务模式下，系统会定义许多组行为信息例如将用户拉入一个指定的空间。动作创建之后，将动作绑定到邀请码，最后将邀请码设置成系统的内置邀请码，并在配置中心中配置购买套装服务应该使用的邀请码。用户在购买套装后，后台会自动为用户使用内置邀请码，将用户加入到指定空间。

### 相关数据库解释

|  |  |
| --- | --- |
| 数据库表名 | 表存储的内容 |
| symbol | 邀请码的模板信息 |
| symbolactions | 邀请码模板和行为动作的关联信息，表示邀请码模板下有哪些动作 |
| actions | 动作的详细信息 |
| actionhistory | 邀请码的动作调用记录 |
| inviatation | 邀请码的详细信息 |
| inviatationhistory | 邀请码的状态变化过程 |

### **基本流程**

创建邀请码流程：

1. 人为的调用接口创建行为动作 (往actions表里插入数据)
2. 后台页面上创建邀请码模板 (往symbol表里插入数据)
3. 在后台页面上给邀请码模板绑定行为动作 (往symbolactions插入数据)
4. 在后台页面上创建邀请码并绑定一个模板。 (往inviatation插入数据，并将记录的symbol设置为模板对应的symbol)
5. 在后台页面上管理邀请码，冻结和解冻邀请码 。(修改inviatation的state 并添加邀请码状态变化记录到inviatationhistory)

使用邀请码流程：

1. 在后台页面上管理员分发邀请码或者在门户网站上用户购买邀请码
2. 在门户网站上用户填写邀请码之后点击使用，调用接口开始使用邀请码
3. 后端校验是否过期，是否有剩余可使用次数，是否有权限使用
4. 后端查询邀请码所绑定模板，并查询改模板绑定的行为动作
5. 后端执行这些行为动作（调用动作里指定的API）并记录调用日志

## **授权码模块**

授权码模块是负责授予激活权限的模块，核心的功能有创建授权码，激活机器，解绑机器，延长激活期限，查询代理人代理的授权码。涉及到的实体有 软件权限，授权码，激活记录。在后端代码中所有的接口在com.mr.authorizationcode.controller 包下面。

授权码的来源有两个： 一是后台管理员直接创建一条授权码，二是在复制用户的时候复制用户的授权码成为一条新的授权码。

系统中设定：一个用户目前只能拥有一个授权码。但一个授权码可以绑定多个软件的激活权限，一个授权码只能激活同一个软件（产品名，平台都相同的软件视为同一软件）一次。

授权码的使用流程是： 管理员创建授权码并将授权码分发给系统中还没有授权码的用户。用户在未被激活的机器上使用软件时，使用授权码激活机器。激活机器后显示过期时间，一段时间后，软件过期，管理员在后台上为用户续期，续期后，如果用户更换了设备，要想正常使用就需要将已经使用过的机器解除激活【解除激活目前的入口是门户网站的个人中心】

通常的激活流程是： 用户登录时找本地授权证书，证书不存在就试图从云端获取激活记录重新生成授权证书，如果还是不存在证书，就表名机器未被激活 ，客户端尝试使用登录者的授权码对机器进行激活，如果登录者没有激活权限（用户没有授权码，授权码上没有绑定软件的激活权限）则终止流程，如果有就直接激活机器，同时本地生成授权证书。激活后显示授权到期时间。

### **相关数据库解释**

|  |  |
| --- | --- |
| 数据库表名 | 表存储的内容 |
| activingathouity | 授权码可以附带的权限 |
| serialnumber\_authority | 存储授权码和权限的联系，表示授权码绑定了哪些权限 |
| serial\_number | 授权码的详细信息 |
| mashinecode\_user | 记录了系统中软件的激活记录 |

### **基本流程**

创建个人授权码：

1. 指定产品名称和权限类型，创建权限（往activingathouity插数据）
2. 后台管理界面创建个人授权码。选择权限、到期时间、公司、MRUI后，调用接口/post/serialnumber 创建。

创建企业授权码：

1. 指定产品名称和权限类型，创建权限（往activingathouity插数据）
2. 后台管理界面创建个人授权码。选择权限、到期时间、公司、MRUI后，调用接口serialnumberbycompany创建。

激活机器：1.客户端校验是否已被激活 ，调用（/user/can/authorization）

2.客户端发送激活请求，调用（/active/withoutSerialNumber）

3.客户端将授权码信息和激活信息写入本地

解绑机器：1.去门户网站的个人中心解除激活/客户端都调用（/unlock/machine）

## **广告模块**

广告模块是处理广告逻辑的模块，用于支持租赁业务中的霸屏客户端的广告显示，核心的功能有添加广告，制定广告策略，查询广告播放策略。涉及到的实体有广告、用户去广告记录、广告策略、广告设定。在后端代码中所有的接口在com.mr.ad.controller包下面。

当用户打开一个作品的时候，右下角会出现一些图标，这些图标要么是公司图标，产品图标，要么是一些自定义图标，要么是没有图标。这里的图标就是广告。

广告的播放与否和播放内容取决于云端的播放策略，目前系统里设置了只显示产品图标，只显示公司图标，轮流显示公司图标和产品图标，不显示图标四种广告播放策略。系统管理员可以在后台管理界面设置某角色的用户在某子服务器下面观看某个空间下的作品时采用的广告播放策略。用户在客户端点击去除广告的图标后，通过购买的手段使得一段时间内用户看作品时不再播放广告。此外用户自身的会员信息指定了默认的广告策略【白金会员附带的是不播放广告策略】。

此外，租赁服务会影响到广告的播放，如果是一个租赁作品，用户没有租赁作品的话，观看一段时间后就会被广告霸屏，用户在租赁期间是不会播放广告的。

### **相关数据库解释**

|  |  |
| --- | --- |
| 表名 | 简要描述 |
| ad | 存放广告的详细信息 |
| adstrategyinfo | 存放系统的广告设置规则（在expire之前，角色roleid 在浏览serverID号服务器下worksid号作品的时候，客户端需要采用sterategyid号广告策略来播放广告） |
| strategy | 广告策略的详细信息 |
| strategyad | 广告策略关联了哪些广告 |
| upadsstrategyapply | 特殊的广告策略设置 |

### **基本流程**

系统内置角色的默认广告策略：先在数据库中生成策略，随后写入Enum类。

创建广告设置规则: 后台管理界面上有广告管理模块，在该管理模块可以添加广告。

查询广告策略:遵循用户意愿>系统配置>角色默认的原则，先后从upadsstrategyapply，adstrategyinfo表中获取广告策略。

### **表的变动**

|  |  |
| --- | --- |
| 数据表 | 功能项 |
| Ad | 添加广告时插入数据，删除广告时删除数据 |
| adstrategyinfo | 添加一条配置规则时插入数据，删除一条配置规则时删除数据 |
| strategy | 人工的插入广告策略，后台管理也添加广告策略，暂无删除通道 |
| strategyad | 为策略绑定广告时插入数据，为策略解绑广告时删除数据 |
| upadsstrategyapply | 租赁作品，购买广告时插入数据，定时任务过期后修改数据 |

## **管理员模块**

管理员模块主要负责处理管理员的登录和注销，负责管理员的认证。涉及的实体有 admin主要的接口在com.mr.controller.AdminController.

管理员没有明显的添加渠道，如果要添加管理员只能编写SQL语句在数据库里执行。管理员登陆后台界面会查询数据表中的记录，如果符合就创建session之后返回登录成功。

### **数据表**

|  |  |
| --- | --- |
| 数据表 | 功能项 |
| admin  存放管理员信息记录 | 1. 人为的将管理员账号密码插入数据库 2. 管理员登录后台管理系统 |

## **管理员日志模块**

管理员在后台页面上的操作需要记录成日志。日志模块主要通过AOP切面编程完成功能，功能主要涉及adminlog表。

管理员日志主要用于溯源和追责，溯源就是查询一个变动的缘由，从哪个时间段的哪个值变成了另外一个时间端的另一个值，追责就是获得做出这个操作的管理员的账号。因此对于系统中管理员的修改操作，系统都应该将操作记录到数据库中。

### **数据表**

|  |  |
| --- | --- |
| 数据表 | 简要描述 |
| Adminlog | 存放管理员操作记录 |

### **实现方式**

有一个注解@AdminLog 和一个切面AdminLogAspect（这个切面封装了写入日志表的逻辑） ，被@AdminLog标记的方法会将AdminLogAspect 的横切逻辑走一遍。需要被监听的方法只需要用@AdminLog标注即可。

## **审计模块**

审计模块主要负责审核记录资金，涉及到实体表有assetlog.

系统做审计的原因是确保用户可使用的服务或者虚拟资产都是通过购买渠道获得的，并且用户的支付金额和流水和公司的流水保持一致。实现审计的前提有

1. 有一个记录用户属性的数据表
2. 用户有关资产的属性的变动都会被记录到掺进变动表中。

### **实现方式**

遵循四方对账，用户profile,流水记录，日志记录，总账记录四个方面能对比上。

1. 用户购买或退款会产生记录流水，交易记录。也会修改用户profile，
2. 通过日志计算出今日明细账
3. 通过昨日总账和今日明细账算出今日总账
4. 比对昨日总账和今日总账，看两者差别和明细账里的是否一致
5. 看明细账里的记录和用户profile的记录变动是否一致
6. 看明细帐的总计项和流水总计是否一致

## **云端内容收集模块**

云端作品模块负责对子服务器上的优秀作品和素材进行统计。子服务器会依据作品的下载量，作品的评分，素材的使用量来衡量作品的优秀程度。每日的下载量TOP10和评分TOP10的作品会视为优秀作品，会上报到云端。每日的引用量TOP10的素材也会上报云端，同时每日的作品日志记录，下载记录，评分记录也会上报到云端。手机模块主要涉及到的时实体有 works，material其主要接口在com.mr.InfoCollector.controller下。

云端收集内容的目的是为了以后建立公有素材库和公有作品库。为了防止两个公有库成为啥都存放的内容库，对于公有素材库和公有作品库里的内容，有一些要求：

素材库里的素材引用量或者下载量需要到达一定的数量

作品库里的作品下载量或者播放量需要到达一定的数量

素材库和作品库里的内容不会引起法律纷争

云端收集内容的方式是被动的，子服务器上运行了定时任务，会定时上报内容，对于不能连接外网的子服务器来说，这些定时任务就不会执行。

### **数据表**

|  |  |
| --- | --- |
| 表名 | 简要描述 |
| works | 优秀作品的详细信息 |
| works\_download\_record | 子服务器上的作品下载记录 |
| works\_score\_record | 子服务器上的作品评分记录 |
| workslog | 子服务器上的作品操作记录 |
| material | 素材的云端详细信息 |
| material\_log | 子服务器上的素材日志信息 |

## **积分模块**

积分模块主要负责积分相关的业务，从功能上讲提供 增加用户积分，减少用户积分，查询用户积分三个核心功能。由于该模块涉及到用户财产，需要系统将积分变动记录写入日志，最后做审计。主要涉及到的实体有 userPoints 。该模块的接口在com.mr.controller.PointController中。

引入积分模块的目的有：1.通过赠送积分的形式，让用户免费使用那些原本需要付费的服务，便于初期推广。2.通过购买积分，将用户的现金流留在系统中。

积分的使用和获取变动都属于用户资产的变动，需要写入资产变动表，同时记录交易日志。积分原则上是不能赠送的，但是为了前期推广的需要，系统后台管理界面开了一个后门。积分的添加操作主要是对user\_points进行修改和对point\_record进行插入

积分的删除操作主要是对user\_points进行修改和对point\_record进行插入

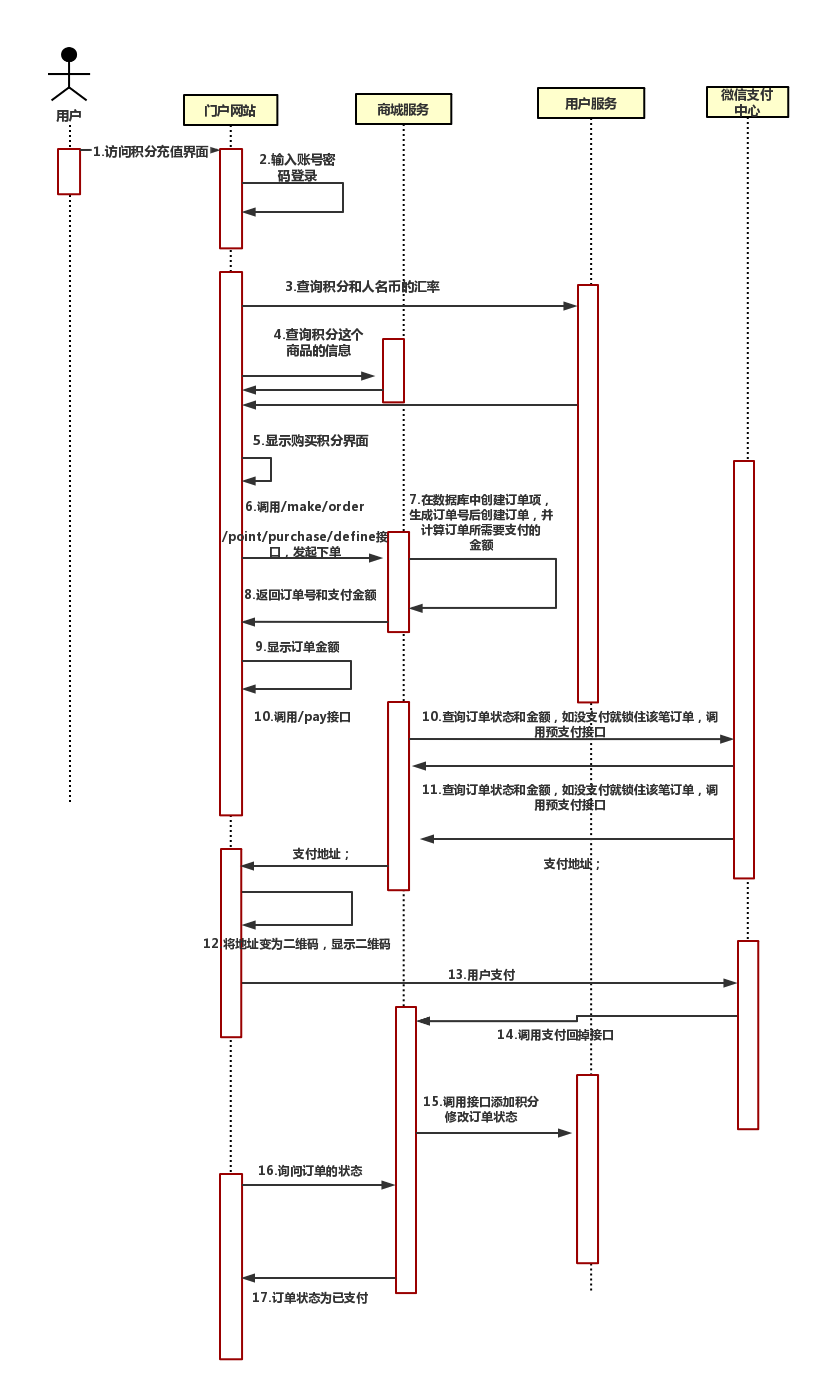
积分的添加和删除涉及到用户的资产变动，为了防止高并发操作引起修改错误，再修改之前采用悲观锁的方式将锁住，保证同一时间内只有一个线程可以修改用户的资产数据。当前使用的悲观锁采用数据库的for update 来构造。

### **数据表**

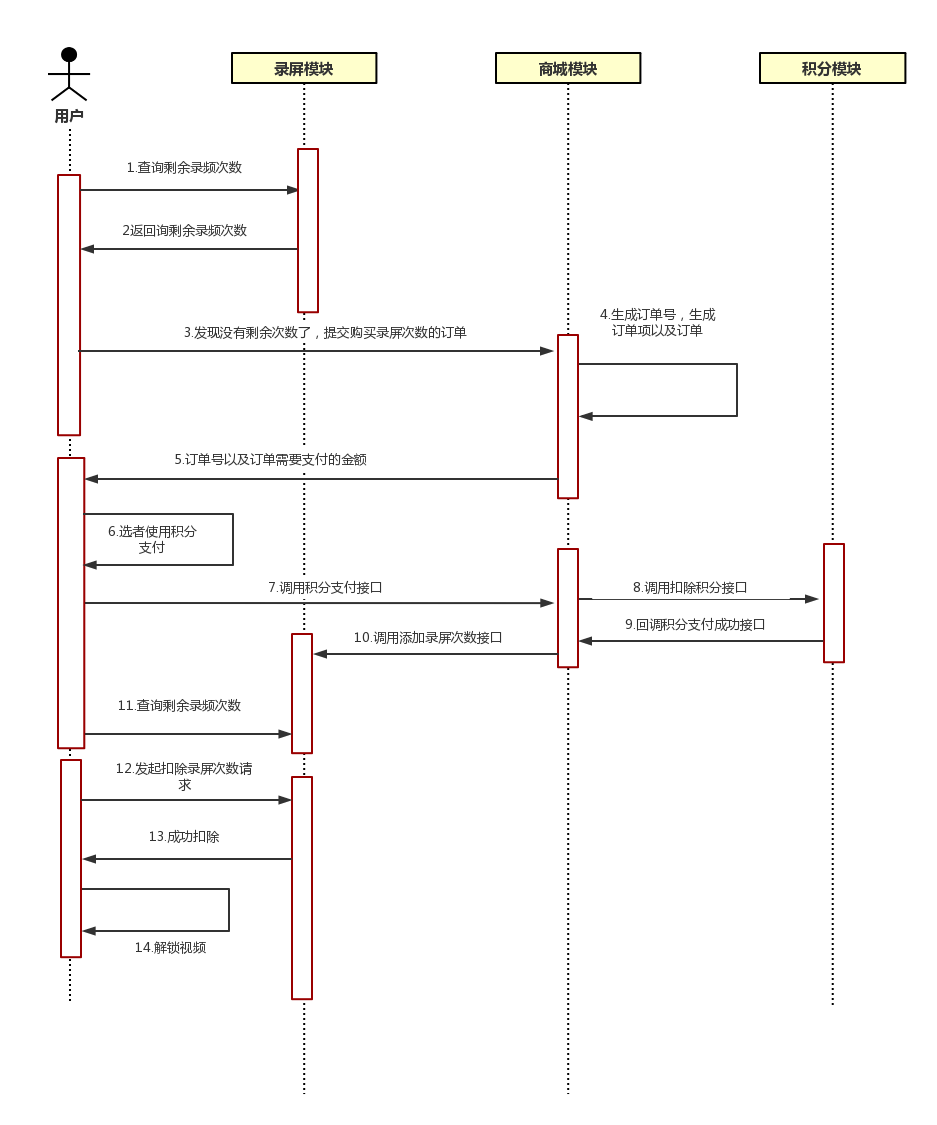
|  |  |
| --- | --- |
| 数据表 | 简要描述 |
| point\_record | 积分交易记录 |
| user\_points | 用户积分 |

### **基本流程**

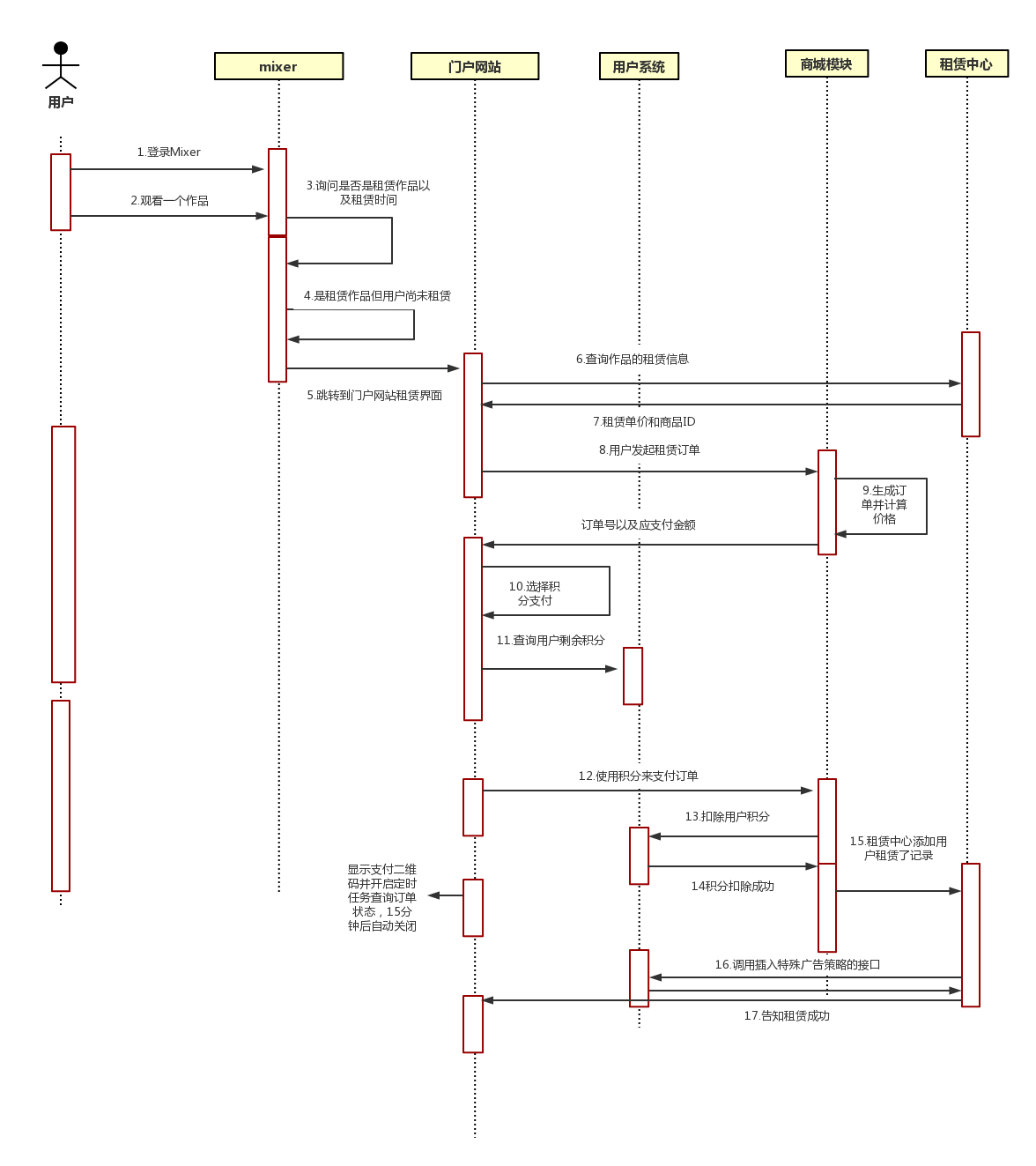
购买积分：



解锁录屏次数：



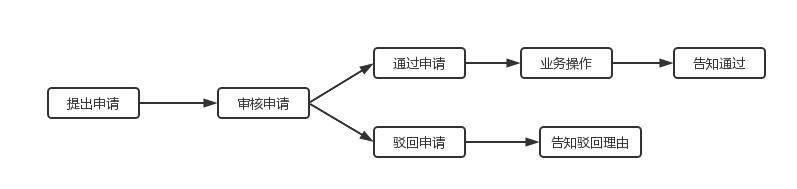
使用积分租赁作品：



## 申请处理模块

该模块对用户提交上来的各种申请进行处理；按照申请类型的不同，分为试用申请，租赁服务申请，合作申请。因此该模块提供各类申请的提交，审核，驳回的接口。接口集中位于com.mr.apply.controller.ApplyController。申请模块主要是对apply表的操作。

申请有一套标准的流程,如下图。不同类型的申请都会走这个流程，只有在通过申请后的业务操作部分有不同。

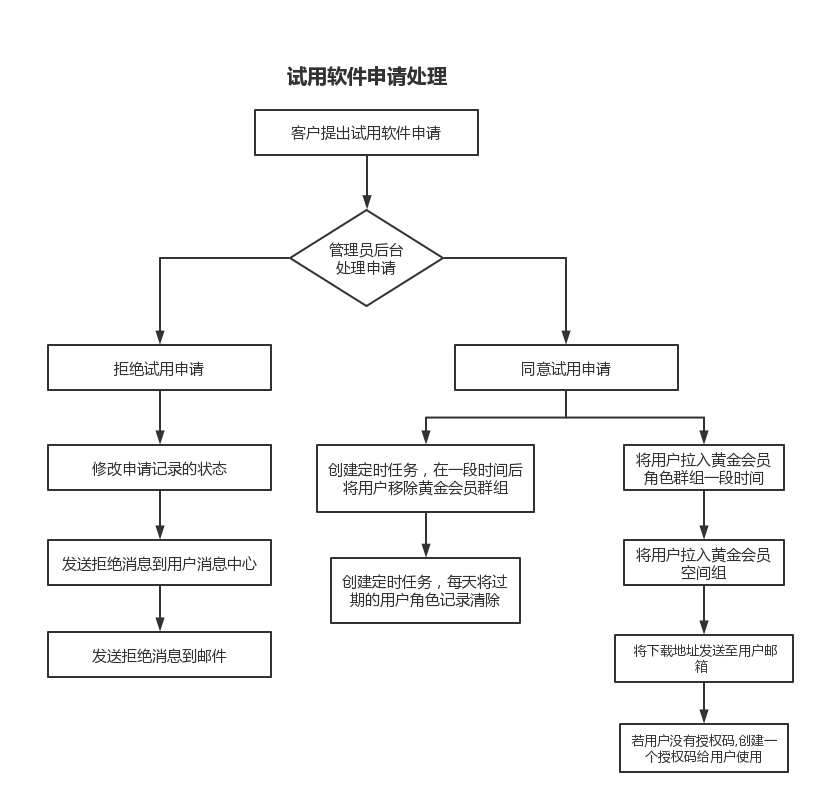


### **数据表**

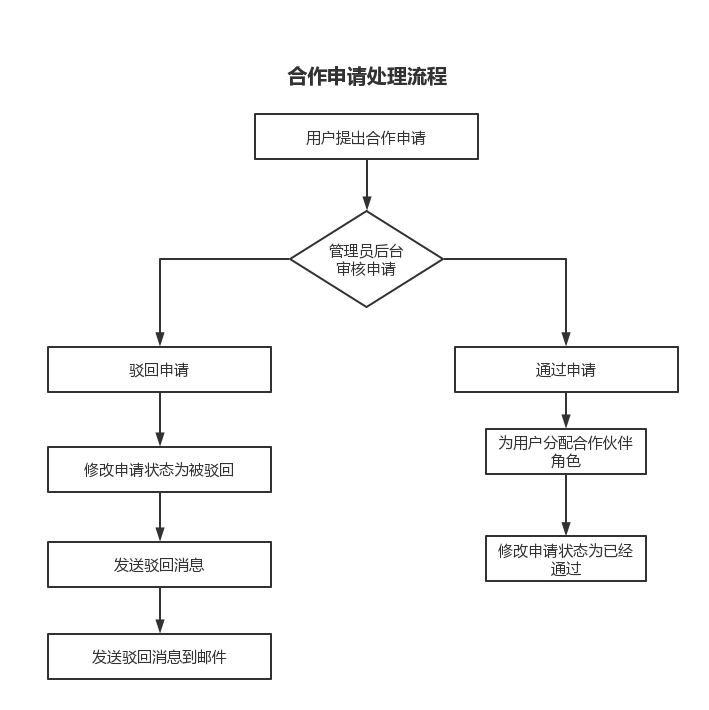
|  |  |
| --- | --- |
| 表名 | 表描述 |
| apply | 存放用户提出的各种申请 |

### **基本流程：**

1. 试用申请的处理流程



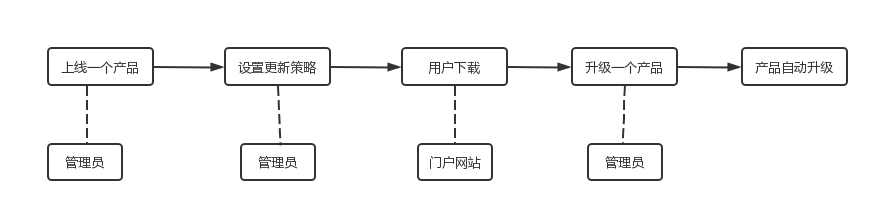
1. 合作申请的处理流程



## **产品模块**

产品模块封装了产品相关的业务。系统中有哪些产品，哪些产品处于开放状态，哪些产品处于测试阶段，产品附带了哪些空间、MRUI 等问题都通过产品模块进行解决。该模块涉及到的实体有 product . 与之相关的模块有server ,MRUI，space ，client.主要接口在com.mr.productVersion.controller 下。

产品的使用流程如下：



上线一个新产品时需要

1. 校验数据库里是否已经有产品名称和平台完全相同的记录
2. 需要在配置表（Systemconfig）里对产品默认的附带空间进行设置
3. 需要在合适的目录下放置软件安装包
4. 敲定该软件可以使用哪些子服务器
5. 敲定软件的运维时间以及是否强制更新

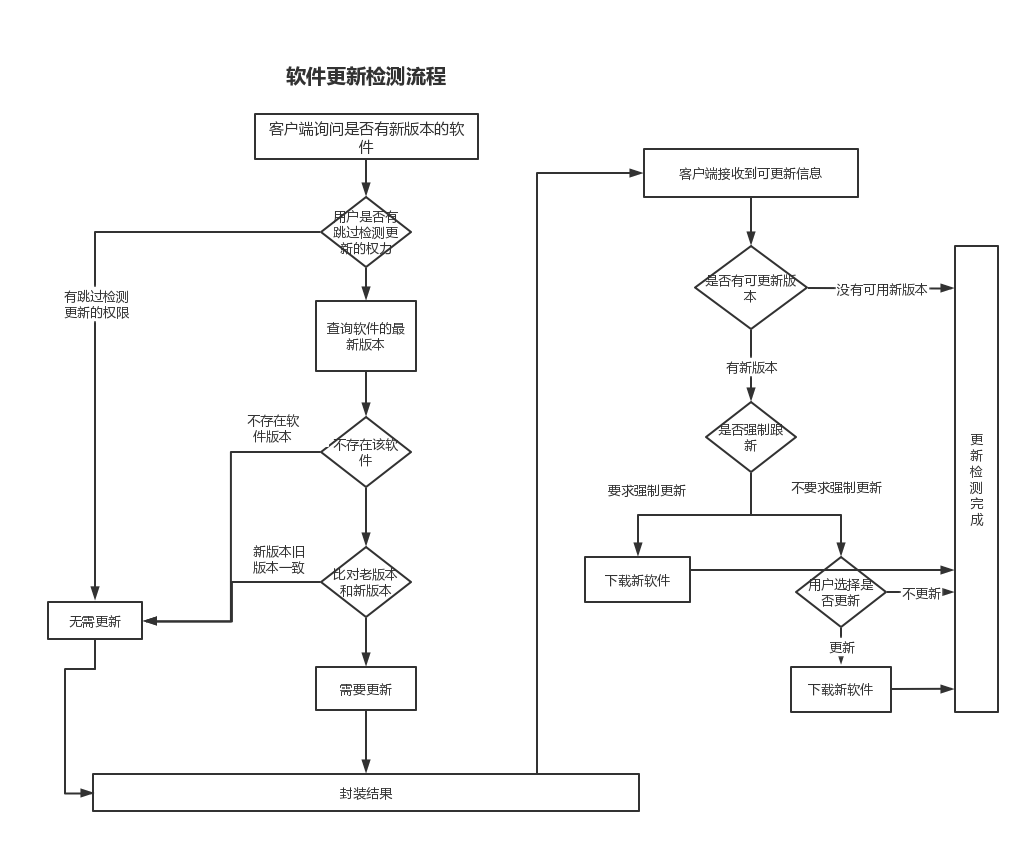
设置更新策略中有几个设置项

1.更新策略表示的是用户在检测到新版本时会不会跳过用户选择升级软件还是取消升级这个步骤。2.是否对所有用户开放，表示的是所有用户都能检测到这个软件版本而非一部分用户3.运维时间段，表示的是只有在系统指定的运维时间段，用户才能够检测到新版本4.是否暂停新版本更新，表示这个版本就是最终版本，不论是否有更高级的版本，这个版本也不会更新到更高版本，通常用于交付一个软件。

### **数据表**

|  |  |
| --- | --- |
| 表名 | 表的描述 |
| client | 运行的客户端记录 |
| product\_version | 记录系统中的软件详细信息 |
| product\_version\_role | 记录哪些角色可以试用这些软件 |
| productserver | 记录一个服务器支持哪些产品，用于从子服务器的后台管理页面下载软件。 |
| product\_space\_access | 记录产品的空间访问规则（可以访问那些空间，禁止访问哪些空间） |

### **基本流程**



## **服务器模块**

服务器模块主要管控子服务器，负责管理后台系统的上线；子服务器的上线；子服务器的配置。主要的功能接口在com.mr.servers.controller包和com.mr.controller.ServerController类中。涉及到的相关实体有：server,materialConfiguration,serverConfiguration 。

### **数据表**

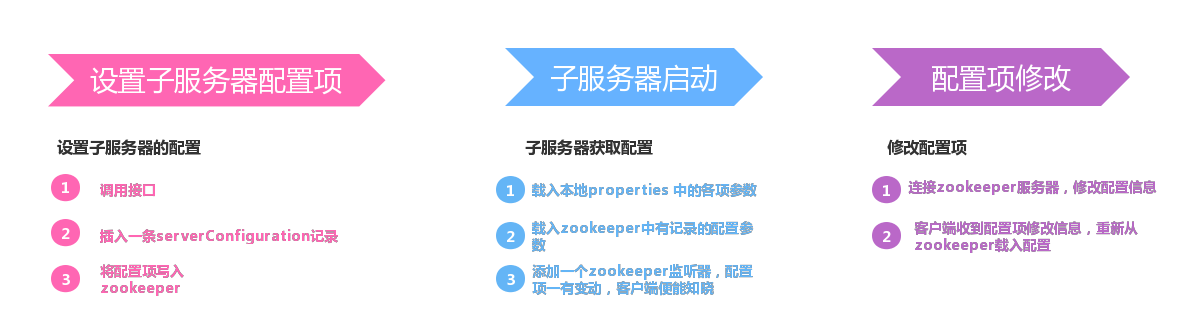
|  |  |
| --- | --- |
| 表名 | 表的详细描述 |
| company | 存储客户记录（代理企业记录） |
| managementconfig | 存储素材子系统管理界面的配置记录 |
| server | 子服务器的详细信息 |
| serverconfiguration | 子服务器的配置信息 |
| productserver | 记录一个服务器支持哪些产品 |

### **基本流程**

系统管理页面上线：



子服务器的服务配置



## **用户角色模块**

用户角色模块有两个核心功能身份认证和用户鉴权。与之相关的实体有 function,role,user。其中一个user对应多个role,一个role对应多个function 。用户角色模块的相关数据实体有

Function,role，user。相关的接口在 com.mr.roleFunction.controller

身份验证：用户在使用软件过程中需要保证使用者是系统的用户，通常是在使用软件前强制用户使用账号密码登录系统，如果登录通过就算身份识别成功。

用户鉴权：不同角色的用户可以访问不同的功能，客户端向云端请求用户能够使用的功能，按照服务器返回的结果渲染不同的功能项。

### **数据表**

|  |  |
| --- | --- |
| 数据表 | 简要描述 |
| function | 存储系统可管控的功能的详细记录 |
| role | 记录角色的详细信息 |
| role\_function | 记录一个角色拥有哪些功能 |
| user\_role | 记录用户有哪些角色，分别在哪段时间内有效 |
| user | 储存用户的详细信息 |

## **反馈模块**

反馈模块的主要作用是为了让公司及时知晓系统运行过程中出现了哪些异常。

反馈主要来源于两方面，一是用户产生的，在使用软件过程中，会出现以下情况：

软件不能满足用户需求，软件在一些常规操作之后发生了错误，软件的一些功能需要完善。出现这些情况时，需要将这些系统需要改进的地方反馈到云端。针对用户提出的反馈，管理员在后台查看反馈后，可以将反馈指派给一个内部人员进行处理，内部人员处理之后将发送已解决信息给反馈者。

是2是系统执行定时任务时遇到了意料之外的错误。这部分错误是定时任务出现错误后，需要将错误上报到反馈中心，后台人员通过页面了解错误后能够协助解决系统中的错误。

## **MRUI模块**

MRUI模块是为代理商定制的多款UI界面，不同的代理商下的用户登入MRMixer后就会使用各自定制的MRUI。MRUI通常是由MRUI制作者制作，然后在MRUI上传界面将MRUI上传到云端，每一个MRUI有代理商，版本，下载地址，名字等核心属性。在用户登入MRMixer后，如果没有MRUI，mixer会根据机器激活的信息向服务端请求mixer会显示的UI，如果有则询问是否可以升级【请求有没有UI和询问是否能升级是同一个接口】。

注意：MRUI 在上传过程中是直接上传到了灵镜世界公司运营的子服务器上的一个空间。一个MRUI的代理商还会附带一部分空间，在使用MRUI时，用户能看到的空间需要补上MRUI所附带的空间。

MRUI模块涉及到的实体有mrui，mruigroup。相关接口在com.mr.screen.controller包下面。

### **数据表**

|  |  |
| --- | --- |
| 表名 | 简要描述 |
| screen | 存储mrui记录 |
| mruigroup | 存储销售渠道（代理商）的记录 |
| mruigroup\_space | 记录销售渠道（代理商）的附带的空间信息 |

用户模块

用户模块主要对用户的各类信息进行管理。

数据表

|  |  |
| --- | --- |
| 数据表 | 简要描述 |
| user | 存储用户的基本信息 |
| user\_business | 用户的虚拟资产信息【除积分】 |
| user\_local | 用户在子服务器上的信息【用户是否在子服务器上】 |
| user\_mruigroup | 用户挂在代理商名下使用了Mixer |
| user\_role | 用户角色信息 |
| user\_points | 用户积分 |
| user\_social | 申请使用Mixer的用户社会信息 |
| Usercompany | 用户属于哪个企业 |
| Userforbidden | 表示冻结一个子服务器下的一个用户一段时间【暂时无用】 |
| userprofile | 用户头像 |
| userprofilehistory | 用户历史头像 |
| userspace | 用户在哪个时间段内可以访问哪个子服务器下面的哪个空间 |

# **三 项目上线流程**

## **项目的整体架构**

项目使用了tomcat做服务容器，使用spring + springMVC 做请求的分发，使用redis集群做缓存处理，使用mysql做存储层，使用文件系统存储文件。使用tomcat的虚拟目录来映射文件地址。

|  |  |
| --- | --- |
| 环境 | 架构描述 |
| Alpha | Tomcat 使用了8080端口 位于/userServer；redis集群端口是为10086，10087，10088 位于/usr/local/redis-cluster；mysql是单机；使用/ljManage 作为存放文件的根目录；项目需要部署在alpha用户服务器上 |
| beta | Tomcat 使用了7980端口 位于/userTomcat；redis集群端口是为10086，10087，10088 位于/usr/redis-cluster；mysql是单机；使用/nas/ljManage 作为存放文件的根目录，该目录是一块NAS存储盘挂在的目录；项目需要部署在beta服务器上。 |
| gamma | Tomcat 使用了7980端口 位于/userTomcat；redis集群端口是为10086，10087，10088 位于/usr/local/redis/；mysql是单机；使用/nas/ljManage 作为存放文件的根目录，该目录是一块NAS存储盘挂在的目录；项目需要部署在gamma用户服务器上。 |

## **项目的技术点**

### **RPC远程调用**

RPC远程调用主要功能为服务的注册、服务的发现、服务的续约，服务掉线检测。该功能实现代码在com.mr.Remote和com.mr.remoteMoudel两个包中。 项目中使用的RPC结构为RMI+zookeeper结构，并自定义了@RemoteService和@RemoteClient来标识需要注册到注册中心的服务和需要从注册中心拉取的服务。

服务的注册：在项目启动时会创建RMI注册中心，并将要发布的服务注册到RMI注册中心，注册完成之后将注册中心的ip和port以json的格式放入zookeeper中的指定节点下【例如 /beta ；/alpha】。

服务的发现：项目启动后，将需要调用的远程服务对象生成，首先通过服务的名字和环境，定位到目标zookeeper节点，该节点存放了该服务的RMI注册中心信息。使用获取到的RMI注册中心信息，创建一个连接到该注册中心。最后从注册中心拿到该服务。如果获取过程中发生异常（没有找到节点或者没能连接到服务中心）则使用动态代理的方式抛出一个异常。创建出来的对象会缓存到本地，方便直接调用。此外，项目还创建了一个监听器，监听zookeeper服务目录节点，当新增一个服务或者下线一个服务时，能够实时的监听到服务的变动，最后依据zookeeper返回的目录变动，重新生成本地对象的缓存。

服务的续约：项目中使用zookeeper客户端连接zookeeper服务器，他们之间建立的是长连接，不需要续约。

服务的更新检测： 注册服务时，创建一个zookeeper临时节点，根据zookeeper临时节点的生命周期特性，服务掉线了，zookeeper连接会断开，临时节点就会被删除。随后触发了服务消费端的监听器。

### **悲观锁**

悲观锁主要运用在并发场景，在用户的积分模块中，为了防止并发请求引起积分的变动错误，使用了mysql提供的悲观锁。 首先查询出用户积分，使用可重复读的隔离模式请求mysql中的用户积分记录，同时结尾加上 for update 锁住这条记录。在事务提交前，其他修改语句是不能够修改被锁定了的用户积分记录的。然后在事务中修改用户积分记录，最后提交事务释放锁。

### **缓存的使用**

缓存主要存储变动不是很大的数据。在项目中的读取服务器配置项和软件能够使用的子服务器中都使用了缓存。缓存采取旁路缓存策略，在查询时先查询数据库中有没有相关记录，如果有则返回，如果没有就从数据库中读取数据，将读出来的数据放入缓存，最后返回结果。在删除或修改数据时，先删除缓存中的相关记录，在修改数据库。由于在更新数据库记录的过程中，需要删除记录相关的缓存。所以项目中缓存通常都用在哪些一个serviceImpl中就将增上改查操作做完且没有和其他数据表联和修改的地方。缓存主要依赖于自定义的两个注解@PutCacheIfAbsent和@RemoveCache。被@PutCacheIfAbsent注解的方法会执行切面PutCacheIfAbsentAspect将数据写入缓存；被@RemoveCache（key=””）注解的方法会将 key相关的缓存清除。

### **异常处理**

不同的异常代表不同的错误，抛出不同的异常信息有助于快速定位到错误。异常处理基于Spring的全局异常处理器，项目中自定义了BaseException，这个异常类表明是服务过程中出现了不被业务流程认可的错误。BaseException中包含了错误状态吗，每一个错误对应一个状态码，当业务出错时，抛出的BaseException异常会被拦截，在全局处理器中可以处理该异常，比如将HTTP状态码设置为200并放回错误状态码。

## **项目的上线流程**

### **上线前**

1.确认上线环境是alpha,beta,gamma中的哪一个。修改配置文件/src/main/webapp/WEB-INF/web.xml中的配置文件名称。2.修改项目的跨域信息，全局搜索@CrossOrigin 标识并进行修改 3. 比对gitlab上当前环境的上一个版本和本地版本有那些差异，查看新上线的功能是否齐全 4.打包war包 。

### **上线中**

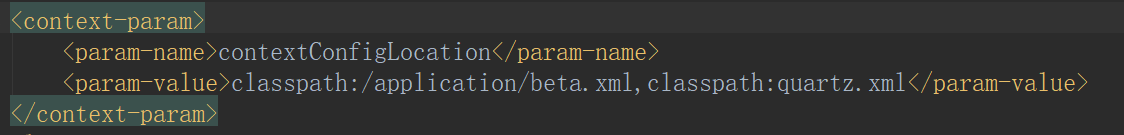
1. 停止tomcat容器（以kill-15的形式关掉进程），2.备份当前war包，3.备份当前mysql表数据 4.在云端备份磁盘快照5.将war包上传到webapps目录下，6.删除tomcat根目录下的/webapps/ljManage 目录 7.运行 删除tomcat根目录下的/bin/startup.sh 文件启动

### **上线后**

1.测试最新版本是否正常，测试过程中如果需要对某些数据进行修改则记录日志。2.如果测试通过，则停止tomcat容器，将mysql数据复原并将测试过程中做的修改重试一次，3.清空redis集群里的内容。4.如果有文件修改，则将文件复原。5.重新启动tomcat容器。6.将代码提交到gitlab中。

## **项目的启动流程**

1.项目的启动入口在/src/main/webapp/WEB-INF/web.xml中，该文件中有如下配置：



该配置告知了tomcat以beta.xml为基准创建SpringIOC容器，并将beta.xml中定义的bean对象和被@Component注解的类实例化。

2. com.mr.remoteMoudel.RemoteServiceConfiguration实例化之后，项目中的服务注册与服务发现流程开始

3.系统中的定时任务启动 ，被@Scheduled标注了的方法都是一个定时任务。

4.web.xml中配置了如下servelet.表示所有的接口请求到达服务器后都会转到一个特定的类org.springframework.web.servlet.DispatcherServlet类，该类按照springmvc.xml里面设定的内容初始化MVC框架，并设置拦截器。

