**聊天室文档**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编写人： | sphitx | 编写日期： | 2019-08-17 | 生效日期： | 2019-08-18 |
| 审批人： | -- | 审批日期： | -- |  |  |
| 第1次修订： | -- | 修订日期： | -- | 修订内容： | -- |
| 第2次修订： | -- | 修订日期： | -- | 修订内容： | -- |

配置安装

1. 安装swoole和reids ，安装完成后使用php -m ，查看安装模块列表有swoole和redis即代表安装成功。

2.添加配置文件web/customise/config/config.websocket.php

<?php

$conf['websocket'] = array(

'enable' => true,

'wsServerUrl' => 'wss://带证书的域名:811',

'server' => array(

'host' => '0.0.0.0',

'port' => 811,

),

);

?>

带证书域名指向**web 1**

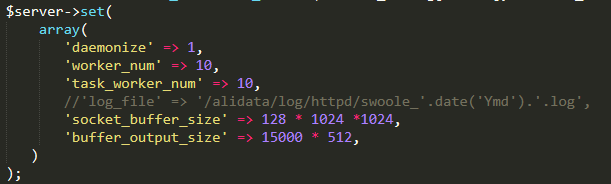
1. 启动server服务php /alidata/www/web/server.php，查看wss连通即代表启动成功。

聊天室swoole配置及常用事件

读取配置文件里的地址和端口创建swoole\_websocket服务



设置服务参数



daemonize

设置程序进入后台做为守护进程运行，长时间运行的服务器程序必须启用此项。如果不启用守护进程，当ssh终端退出后，程序将被终止运行。启用守护进程后，标准输入和输出会被重定向到log\_file，如果log\_file未设置，则所有输出会被丢弃。

Log\_file

指定日志文件路径，在swoole运行期间发生的异常信息会记录到这个文件中，开发调试阶段用来排查错误。

Woker\_num

指定启动的worker进程数，swoole是master->n\*woker的模式，开启woker进程数越多，

Server负载能力越大，但是相应的server占有内存也会更多。当woker进程数过多时，进程间切换带来的系统开销也会更大。因此建议开启的woker进程数为cpu进程数的1-4倍。

Task\_worker\_num

服务器开启的task进程数，设置此参数后，服务器会开启异步task功能，此时可以使用task方法投递异步任务。（必须要给swoole\_server设置ontask/onfinish两个回调函数，否则启动服务器会报错。）

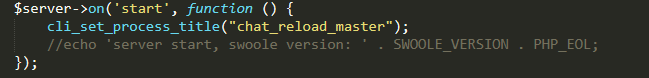
Socket\_buffer\_size

设置进程间通信的socket缓存区尺寸，默认为8M

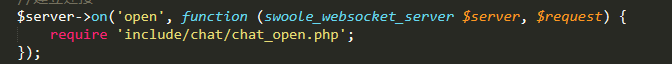
Buffer\_output\_size

配置发送输出缓存区内存尺寸，默认为2M

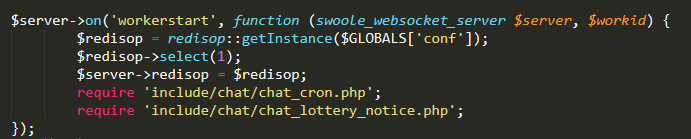
服务器启动的回调函数，仅允许echo、打印Log、修改进程名称。不得执行其他操作。



当websocket客户端与服务器建立连接并完成握手后会回调此函数，一般用于加载初始化消息，会员列表，历史消息等



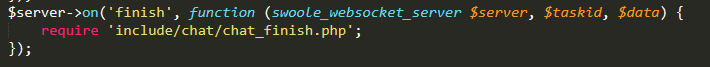
此事件在worker进程/task\_worker进程启动时发生，如果想使用swoole\_server\_reload实现代码重载，必须在workerstart中引入所需业务文件，在workerstart调用之前已包含的文件，不会被重新载入。多个进程间不能公用连接，必须每个进程单独创建redis连接。否则使用时会报错。一般在这里可以加入一些长期监听的事件，如获取计划消息，中奖公告等。



投递一个异步任务到task\_worker池中。此函数是非阻塞的，执行完毕立即返回。Worker进程可以继续处理新的请求。此处用于异步加载聊天历史消息，减少聊天室加载等待。



Task\_worker进程处理任务结束的回调，在此函数中会收到任务处理的结果，通知任务执行结束。如聊天历史消息已加载完毕。



当服务器收到来自客户端的数据时会回调此函数，$frame是swoole\_websocket\_frame对象，包含了客户端发来的数据消息，onmessage回调必须被设置，否则服务器将无法启动，但客户单发送的ping帧不会触发onmessage，底层会自动恢复pong包。此回调函数用于大部分客户端与服务器间的实时通信，如发送普通/图片消息，修改头像/昵称，禁言解禁等。



当websocket客户端与服务器连接关闭时执行的回调函数，可以由客户端发起也可以由服务器主动调用swoole\_server\_close来关闭。此处用户更新离线时会员列表，清除会员连接聊天室的个人信息。



启动服务器，监听所有TCP/UDP端口



SQL来源用途

1. 聊天室设置

CREATE TABLE `ssc\_chat\_settings` (  
 `name` varchar(64) NOT NULL,  
 `value` text NOT NULL,  
 PRIMARY KEY (`name`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

1. 聊天室房间

CREATE TABLE `ssc\_chat\_room` (  
 `id` int(10) unsigned NOT NULL AUTO\_INCREMENT,  
 `enable` tinyint(1) NOT NULL DEFAULT '1' COMMENT '显示开关：1 开启 0 关闭',  
 `isCron` tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否显示计划消息：1 显示 0 不显示',  
 `roomName` varchar(32) CHARACTER SET utf8 NOT NULL COMMENT '房间名称',  
 `levels` varchar(255) NOT NULL DEFAULT '' COMMENT '允许访问层级',  
 `typeId` int(11) NOT NULL DEFAULT '0' COMMENT '彩种ID：手机端进入该彩种时默认显示此房间',  
 `sorts` int(11) NOT NULL DEFAULT '0' COMMENT '排序',  
 `isDelete` tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否删除：1 删除 0 未删除',  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

1. 聊天室公告

CREATE TABLE `ssc\_chat\_notice` (  
 `id` int(11) NOT NULL AUTO\_INCREMENT,  
 `enable` tinyint(1) NOT NULL DEFAULT '1' COMMENT '是否显示：1 显示 0 不显示',  
 `title` varchar(32) NOT NULL COMMENT '公告标题',  
 `content` text NOT NULL COMMENT '公告内容',  
 `sorts` int(11) NOT NULL DEFAULT '0' COMMENT '排序',  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

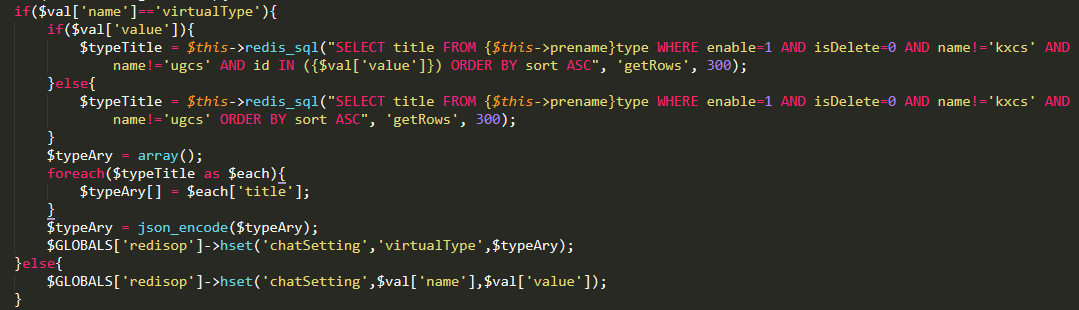
1. 聊天室会员信息（暂存昵称）

CREATE TABLE `ssc\_chat\_members` (  
 `uid` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT '会员ID',  
 `nickname` varchar(16) NOT NULL COMMENT '会员昵称',  
 PRIMARY KEY (`uid`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

读取聊天室设置信息（此表数据量不大，所以用\*来读取所有设置）



读取聊天室设置的虚拟彩种，并取出对应的彩种名称



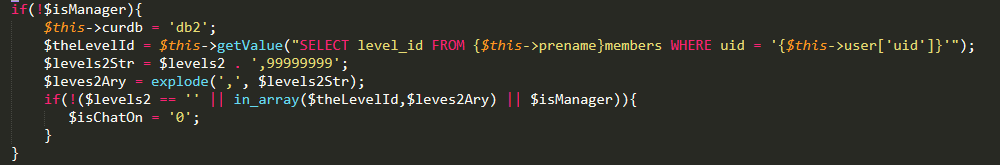
读取聊天室设置的可用房间



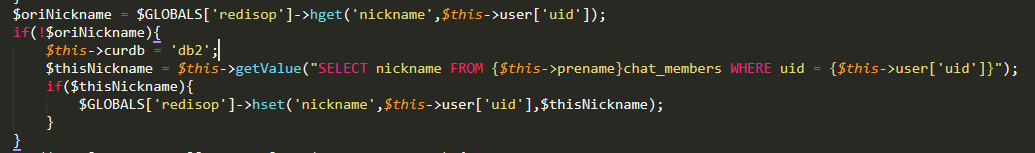
读取检查该会员是否是聊天室管理员



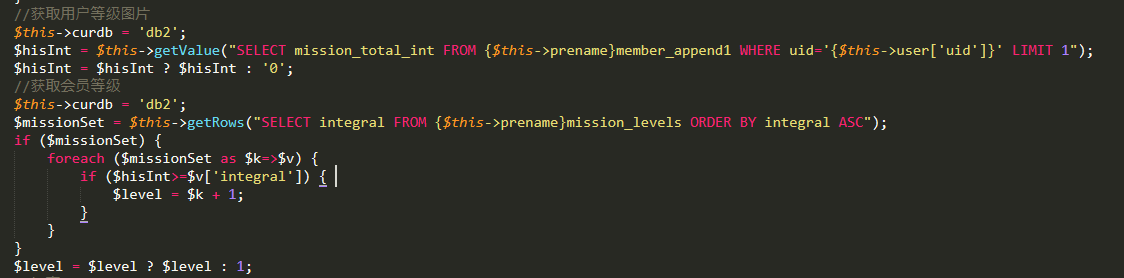
读取检查会员是否满足聊天室层级查看条件



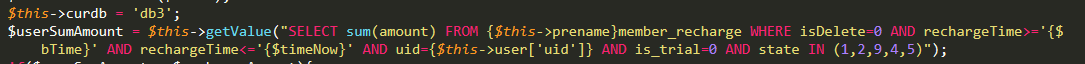
缓存里没昵称时从昵称表更新昵称并重新缓存，防止昵称丢失



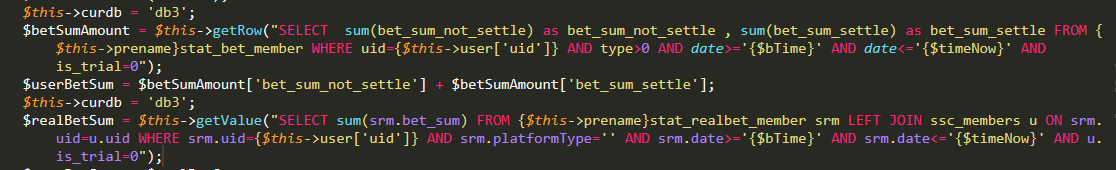
读取会员积分并计算对应的等级（匹配等级图片）



检查在设置时间内存款是否满足发言条件



检查在设置时间内打码量是否满足发言条件



前后端code交互

|  |  |  |  |
| --- | --- | --- | --- |
| code | 描述 | 数据参考 | 响应(code) |
| 0001 | 发送普通消息（图片，跟注，分享战绩） | {"code":"0001","msg":"一条消息","roomId":"0","chat\_type":0,"chatUid":"","chatName":"","level":"1","t":"1566106339","username":"acheng500","ip":"118.143.206.6","token":"ffbe994a0536a83908acbd397232fe2a"} | 0001 |
| 0003 | 修改头像 | {"code":"0003","data\_type":"image","t":"1566107752","msg":"data:image/jpeg;base64","username":"acheng500","token":"6519f00be7d44bc2ac21a25b93a46d48"} | 0014，0024 |
| 0004 | 修改昵称 | {"code":"0004","data\_type":"text","t":"1566107752","msg":"昵称","username":"acheng500","token":"3f0fda3f566504e56eb2add1c1f86a5f"} | 0015，0025 |
| 0005 | 禁言 | {"code":"0005","t":"1566108282","username":"acheng001","chatUid":401566107025,"talkPri":1,"time":10,"token":"a6cd3ddffb8b2c602f48a6fcc9ddab5a"} | 1005，0026/  2005 |
| 0008 | 禁言IP | {"code":"0008","t":"1566108282","username":"acheng001","chatUid":31566108410,"ip":"118.143.206.6","talkPri":1,"time":10,"token":"a6cd3ddffb8b2c602f48a6fcc9ddab5a"} | 1008，2008 |
| 0007 | 禁用昵称 | {"code":"0007","t":"1566108282","username":"acheng001","chatUid":31566108410,"nicknamePri":1,"time":10,"token":"a6cd3ddffb8b2c602f48a6fcc9ddab5a"} | 1007,  0025，  2007 |
| 0002 | 搜索在线用户 | {"code":"0002","t":"1566108282","username":"acheng001","condition":"username","value":"acheng500","token":"a6cd3ddffb8b2c602f48a6fcc9ddab5a"} | 0022 |
| 0006 | 删除留言 | {"code":"0006","t":"1566108282","username":"acheng001","historyId":"31566108410","token":"a6cd3ddffb8b2c602f48a6fcc9ddab5a"} | 0016，  0026 |

Redis变量用途说明

聊天室redis存储key前缀均以 站点编号\_ 开头，下面已省略

|  |  |
| --- | --- |
| 发言时间间隔 | |
| key | value |
| chatLot | 整型时间秒 |

|  |  |
| --- | --- |
| 用户等级（hash） | |
| key | value |
| chatUserLevel , uid | 正整数 |

|  |  |
| --- | --- |
| 用户聊天IP（hash） | |
| key | value |
| chatUserIP , uid | 127.0.0.1 |

|  |  |
| --- | --- |
| 历史聊天室记录 | |
| key | value |
| 'chatHistory'.$user->uid.$t | $response（消息数据） |

|  |  |
| --- | --- |
| 禁言用户（hash） | |
| key | value |
| chatDisabled , uid | 1 |

|  |  |
| --- | --- |
| 会员个人信息 | |
| key | value |
| 'chatRoom'.$request->fd | Json\_encode($this->user) |

|  |  |
| --- | --- |
| 禁用昵称（hash） | |
| key | value |
| chatNicknameDisabled,uid | 1 |

|  |  |
| --- | --- |
| 会员昵称 | |
| key | value |
| nickname | 昵称 |

|  |  |
| --- | --- |
| 禁用IP发言（hash） | |
| key | value |
| chatIPDisabled,ip | 1 |

|  |  |
| --- | --- |
| 某个会员的连接号（hash） | |
| key | value |
| chatRoomFd,uid | $request->fd（连接号） |

|  |  |
| --- | --- |
| 在线server所有连接号 | |
| key | value |
| chatFd | $chatFdAry（所有连接号数组） |

|  |  |
| --- | --- |
| 在线server所有连接号 | |
| key | value |
| chatFd | $chatFdAry（所有连接号数组） |

|  |  |
| --- | --- |
| 聊天室计划API头像 | |
| key | value |
| userImageCronApi | 静态服务器图片地址 |

|  |  |
| --- | --- |
| 聊天室前台等级图片 | |
| key | value |
| chatLevel.$level | 静态服务器图片地址 |

|  |  |
| --- | --- |
| 聊天室后台等级图片 | |
| key | value |
| chatLevelImg.$level | 静态服务器图片地址 |

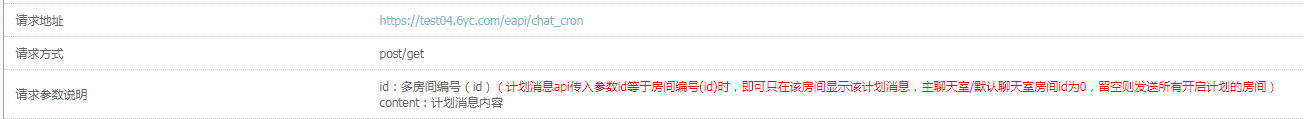
|  |  |
| --- | --- |
| 聊天室设置（hash） | |
| key | value |
| chatSetting,name | 聊天室设置的值 |

|  |  |
| --- | --- |
| 聊天室发言标记1 / 存款条件（hash） | |
| key | value |
| chatUserFlag1,uid | 1 / 0 |

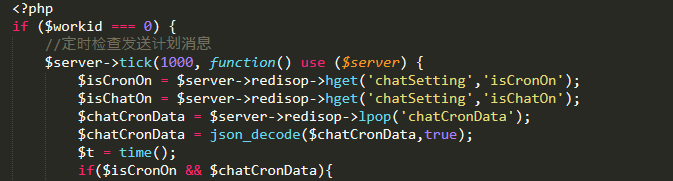
|  |  |
| --- | --- |
| 聊天室发言标记2 / 打码量条件（hash） | |
| key | value |
| chatUserFlag2,uid | 1 / 0 |

计划消息

第三方通过计划API往系统发送计划消息，以为队列形式存放于redis，参数房间id错误时会发送到默认主聊天室。

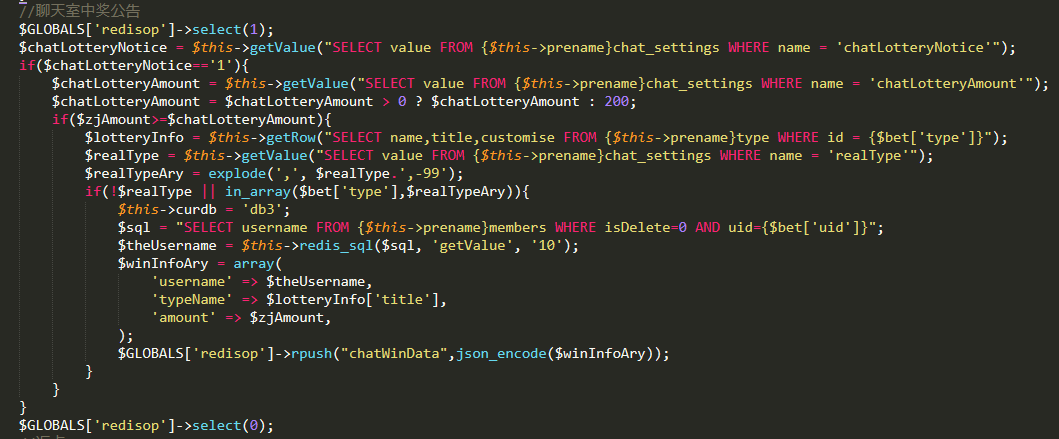


在上述swoole常用事件workstart里调用workerid进程号为0时开启server定时器来检测系统收到的计划消息并发送到聊天室

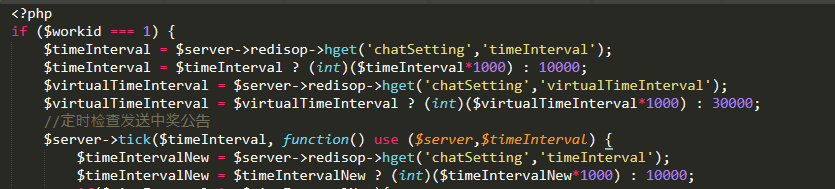


中奖公告

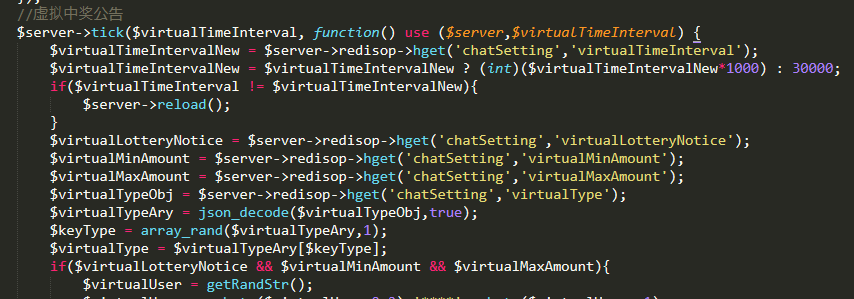
原理和计划消息类似，真实中奖公告在kanjiang结算方法将满足条件的中奖信息以队列形式存放与redis



在上述swoole常用事件workstart里调用workerid进程号为1时开启server定时器来检测系统存储的中奖公告并发送到聊天室



虚拟中奖公告，完全由读取后台设置来生成拼装成相应中奖公告并发送至聊天室



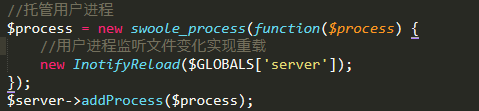
监控聊天室文件变动自动重启

实现基础：依赖PHP扩展inotify实时监控文件变化及swoole自带reload方法重启所有worker/task进程实现

在server入口引入inotify处理类



将监听聊天室文件变化加入监听进程



初始化调用监听方法，再重启一次swoole服务监听就会一直生效



后台管理介绍

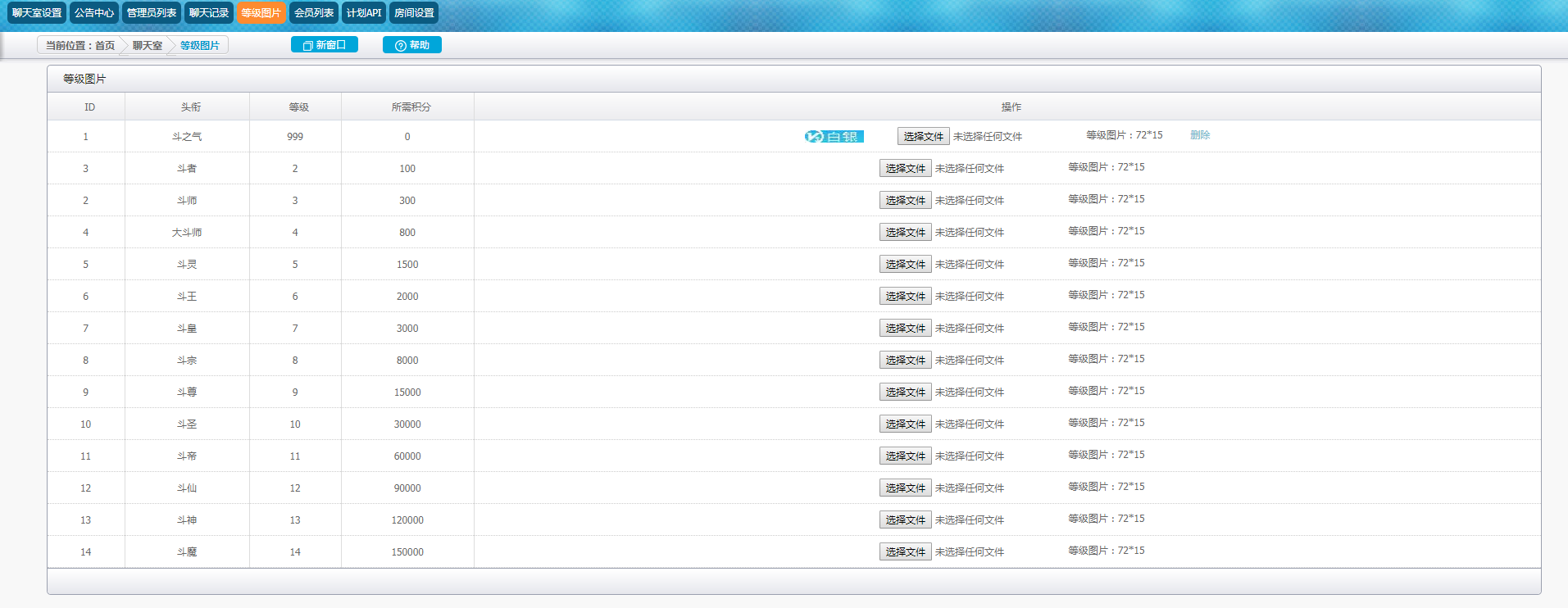
管理聊天室的一些基本设置，数据存放于ssc\_chat\_settings，由于聊天室服务没有连接数据库，会同时会存放一份到redis供聊天室服务使用。



聊天室记录管理，整体存放2小时，前台初始化加载历史记录30条，有时会有一条消息出现多次的情况，是因为发送到了不同房间，可以通过房间id来查看。



管理聊天室等级图标，等级和站点任务系统等级挂钩，现有等级图片上传碍于上传图片插件压缩限制，导致图片质量不太清晰，后续考虑换一套上传插件或做一套精练的等级图标来使用。



会员列表，这里可以查看聊天室在线正式会员，下方总数为在线会员总数，可以进行一些禁言解禁类操作。



计划API设置，初次使用需要配置API调用白名单，否则会被拦截无法使用。这里可以设置聊天室API名称和头像。

