

**目 录**

#### [laychat系统说明](#_bookmark0) [workerman与Thinkphp5整合原理](#_bookmark1) [laychat系统的基础搭建](#_bookmark2)

[通用后台的搭建](#_bookmark3) [workerman的集成](#_bookmark4) [layIM的集成](#_bookmark5)

[管理员后台功能的实现](#_bookmark6) [后台登录退出的实现](#_bookmark7) [群组管理的设计与实现](#_bookmark8)

[群组基础配置](#_bookmark9) [群组管理](#_bookmark10)

[用户管理](#_bookmark11) [聊天设置](#_bookmark12) [用户的举报](#_bookmark13)

[聊天系统的设计与实现](#_bookmark14) [注册功能的实现](#_bookmark15) [登录功能的实现(1)](#_bookmark16) [登录功能的实现(2)](#_bookmark17) [登录功能的实现(3)](#_bookmark18) [登录功能的实现(4)](#_bookmark19) [修改个人资料的实现](#_bookmark20) [修改个性签名的实现](#_bookmark21) [切换在线状态的实现](#_bookmark22) [添加好友的实现](#_bookmark23) [消息盒子](#_bookmark24) [用意/拒绝好友(1)](#_bookmark25) [同意/拒绝好友(2)](#_bookmark26) [好友聊天的实现](#_bookmark27) [创建群组的实现](#_bookmark28) [按条件查找群组](#_bookmark29) [申请加入群组](#_bookmark30) [管理我的群组](#_bookmark31) [群组聊天](#_bookmark32) [右键退出群组](#_bookmark33) [发送语音](#_bookmark34)

[发送图片](#_bookmark35) [发送文件](#_bookmark36) [用户空间功能](#_bookmark37) [查看聊天记录](#_bookmark38)

[进入好友空间](#_bookmark39)

[将好友加入黑名单](#_bookmark40) [移动好友分组](#_bookmark41) [删除好友](#_bookmark42) [举报好友](#_bookmark43)

[数据字典](#_bookmark44) [全国区域数据](#_bookmark45) [系统整体结构说明](#_bookmark46) [Events.php全部代码](#_bookmark47) [main.js全部代码](#_bookmark48) [laychat系统源码](#_bookmark49)

# laychat系统说明

## ThinkPHP5+workerman+layIM打造聊天系统---laychat

#### ThinkPHP5 是ThinkPHP 3.X系列之后的重量级产品，以出色的架构设计，强劲的性能闻名。很适合我们快

速的开发成熟的商用系统。到目前为止，github上也出现了很多成熟的thinkphp5快速开发的通用后台产 品。在这里给自己的一个项目做个广告：[snake通用后台](https://github.com/nick-bai/snake)，算的上最早的一批tp5通用后台了，功能很简 单，便于新手快速的理解和使用tp5。到写作此文章开始，已经有135个star了。

Workerman是一款纯PHP开发的开源高性能的PHP socket 服务器框架。被广泛的用于手机app、移动通

讯，微信小程序，手游服务端、网络游戏、PHP聊天室、硬件通讯、智能家居、车联网、物联网等领域的 开发。 支持TCP长连接，支持Websocket、HTTP等协议，支持自定义协议。拥有异步Mysql、异步 Redis、异步Http、异步消息队列等众多高性能组件。它的作者是来自鹅厂的大神。本系统 laychat 采用的 是 Wokerman的 GatewayWorker3.X系列框架。文档可参考：[GetwayWorker手册](http://www.workerman.net/gatewaydoc/)

layIM是原阿里的大神 贤心倾力打造的一款Web即时通讯前端解决方案。界面优美，功能强悍，仿真度

高。官网：[layim官网](http://layim.layui.com/)，手册：[layim手册](http://www.layui.com/doc/modules/layim.html)

**聊天部分功能清单**

#### 聊天用户的注册

聊天用户的登录 修改个人资料 修改个性签名 切换在线状态 按条件搜索添加好友 系统消息盒子

添加好友申请与同意/拒绝好友 好友聊天

创建群组 按条件查找群 申请加入群组 管理我的群组 群组聊天 右键退出我的群组 聊天发送语音 聊天发送图片 聊天发送文件

空间的说说发表，评论功能 查看个人和群聊记录

好友右键功能

1. 进入好友空间

2. 将好友加入黑名单

3. 移动好友分组

4. 删除好友

5. 举报好友

**大致了解一下外观**

#### [了解外观](http://fly.layui.com/jie/6943.html)

**本系统用到的全部源码**

#### 我会在本套教程中，全部展示这些源码，展示源码的方式如：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller; use think\Controller;

class Chat extends Controller

{

public function index()

{

if(request()->isPost()){

$data = input('post.');

$data['file\_ext'] = trim($data['file\_ext']);

$data['img\_ext'] = trim($data['img\_ext']);

writeCtConfig($data);

return json(['code' => 1, 'data' => '', 'msg' => '配置成功']);

}

$config = readCtConfig();

empty($config) &&

$config = ['file\_size' => 2, 'file\_ext' => 'zip|rar', 'img\_size' => 2, 'img\_ext ' => 'png|jpg|gif'];

$this->assign([ 'config' => $config,

'up\_size' => ini\_get('upload\_max\_filesize')

]);

return $this->fetch();

}

}

#### 你可以跟着教程，一步一步搭建属于自己的 webIM系统。

此外，你也可以打开 《laychat系统源码》 这一章节，进行下载。

**本地环境的注意事项**

#### 本次系统演示，采用的 phpstudy 这个集成环境，php版本为 5.6.27。服务器 以apache 为例，请保持与我一 致，或者自己修改其中不兼容的部分。

**其他建议**

#### 1. 本套系统牵扯到的 websocket相关的知识，以及tcp协议的相关知识，不懂的可以百度做一些功课。也可 以通过我的博客：[我的博客](http://www.cnblogs.com/nickbai/)，了解一些相关的知识。

2. 本系统是基于windows平台的学习版本，当然你只需要下载linux版本的workerman即可轻松的实现linux平 台下的应用，为了演示和学习的方便，本系统基于windows平台下讲解。

3. 数据库不了解功能的 可以参考 《数据字典》 这一章节。

4. 由于讲解这个教程的时候，我的代码已经完成了，没办法一步一步的完全拆开介绍。因此，我会先全部放 出完成的js代码，然后顺着js的接口，一步一步的对php代码和workerman代码进行讲解。这些js代码，我 会首先放出，方便你在跟着教程的同时，有代码依据。

5. 有些workerman和layIM中的方法，可能没有讲到，如有不理解的，可以自行查找手册。手册写的都很明 确。

**郑重声明**

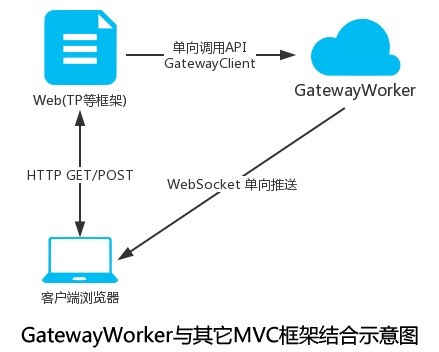
本系统用到的 layIM 为非开源程序。请务必到官网进行授权获得资源。本教程源码部分，不会放出 layim.js 这个源码。

# workerman与Thinkphp5整合原理

#### 看到一些人，可能不太明白workerman与thinkphp5是怎么工作的，这里，我引用workerman官方的文章来解 释一下，希望给你们一些参考。

**与ThinkPHP等框架结合**

使用GatewayWorker时开发者最关心的是如何与现有mvc框架(ThinkPHP Yii laravel等)整合，以下是官方推荐

的整合方式。见示意图：

**总体原则:**

现有mvc框架项目与GatewayWorker独立部署互不干扰

所有的业务逻辑都由网站页面post/get到mvc框架中完成

GatewayWorker不接受客户端发来的数据，即GatewayWorker不处理任何业务逻辑，GatewayWorker仅仅当 做一个单向的推送通道

仅当mvc框架需要向浏览器主动推送数据时才在mvc框架中调用Gateway的API(GatewayClient)完成推送

**具体实现步骤**

1、网站页面建立与GatewayWorker的websocket连接

2、GatewayWorker发现有页面发起连接时，将对应连接的client\_id发给网站页面 3、网站页面收到client\_id后触发一个ajax请求(假设是bind.php)将client\_id发到mvc后端

4、mvc后端bind.php收到client\_id后利用GatewayClient调用Gateway::bindUid($client\_id, $uid)将client\_id与 当前uid(用户id或者客户端唯一标识)绑定。如果有群组、群发功能，也可以利用

Gateway::joinGroup($client\_id, $group\_id)将client\_id加入到对应分组

5、页面发起的所有请求都直接post/get到mvc框架统一处理，包括发送消息

6、mvc框架处理业务过程中需要向某个uid或者某个群组发送数据时，直接调用GatewayClient的接口 Gateway::sendToUid Gateway::sendToGroup 等发送即可

**示例代码**

GatewayWorker中Events.php代码（只有个onConnect回调设置）

<?php

use \GatewayWorker\Lib\Gateway; class Events

{

// 当有客户端连接时，将client\_id返回，让mvc框架判断当前uid并执行绑定

public static function onConnect($client\_id)

{

Gateway::sendToClient($client\_id, json\_encode(array( 'type' => 'init',

'client\_id' => $client\_id

)));

}

// GatewayWorker建议不做任何业务逻辑，onMessage留空即可 public static function onMessage($client\_id, $message)

{

}

}

**网站页面js片段**

/\*\*

* 与GatewayWorker建立websocket连接，域名和端口改为你实际的域名端口，
* 其中端口为Gateway端口，即start\_gateway.php指定的端口。
* start\_gateway.php 中需要指定websocket协议，像这样
* $gateway = new Gateway(websocket://0.0.0.0:7272);

\*/

ws = new WebSocket("ws://your\_domain.com:7272");

// 服务端主动推送消息时会触发这里的onmessage ws.onmessage = function(e){

// json数据转换成js对象

var data = eval("("+e.data+")"); var type = data.type || ''; switch(type){

// Events.php中返回的init类型的消息，将client\_id发给后台进行uid绑定

case 'init':

// 利用jquery发起ajax请求，将client\_id发给后端进行uid绑定

$.post('./bind.php', {client\_id: data.client\_id}, function(data){}, 'json')

;

break;

// 当mvc框架调用GatewayClient发消息时直接alert出来

default :

alert(e.data);

}

};

**mvc后端uid绑定代码片段**

#### bind.php

<?php

//加载GatewayClient

require\_once '/your/path/GatewayClient/Gateway.php';

// GatewayClient 3.0.0版本开始要使用命名空间 use GatewayClient\Gateway;

// 设置GatewayWorker服务的Register服务ip和端口，请根据实际情况改成实际值

Gateway::$registerAddress = '127.0.0.1:1236';

// 假设用户已经登录，用户uid和群组id在session中

$uid = $\_SESSION['uid'];

$group\_id = $\_SESSION['group'];

// client\_id与uid绑定 Gateway::bindUid($client\_id, $uid);

// 加入某个群组（可调用多次加入多个群组）

Gateway::joinGroup($client\_id, $group\_id);

**mvc后端发消息代码片段**

send\_message.php

<?php

//加载GatewayClient

require\_once '/your/path/GatewayClient/Gateway.php';

// GatewayClient 3.0.0版本开始要使用命名空间 use GatewayClient\Gateway;

// 设置GatewayWorker服务的Register服务ip和端口，请根据实际情况改成实际值

Gateway::$registerAddress = '127.0.0.1:1236';

// 向任意uid的网站页面发送数据

Gateway::sendToUid($uid, $message);

// 向任意群组的网站页面发送数据

Gateway::sendToGroup($group, $message);

注意 以上仅是mvc框架与GatewayWorker官方推荐的结合方式，并不是强制使用此方式，开发者可以自由变化 选择结合方式以适应自己的业务需求。 当然也可以采用客户端与GatewayWorker直接双向通讯的方式完 成业务通讯。

**关于GatewayClient**

源码：

<https://github.com/walkor/GatewayClient>

注意：

如果GatewayClient和GatewayWorker不是在同一台服务器上，则需要先将start\_gateway.php中的lanIp改 成当前服务器的内网ip（如果不在一个内网可改成公网ip）。

如果GatewayClient和GatewayWorker在同一台服务器上运行，则不用做任何更改，直接按照示例使用

GatewayClient即可。 通过GatewayClient发送的数据不会经过Event.php，而是直接经由Gateway进程转发给客户端。 GatewayClient无法接收客户端发来的数据。

**客户端使用示例**

require\_once '/your/path/GatewayClient/Gateway.php';

/\*\*

\* gatewayClient 3.0.0及以上版本加了命名空间

\* 而3.0.0以下版本不需要use GatewayClient\Gateway;

\*\*/

use GatewayClient\Gateway;

/\*\*

\*====这个步骤是必须的====

\*这里填写Register服务的ip（通常是运行GatewayWorker的服务器ip）和端口

\*注意Register服务端口在start\_register.php中可以找到（chat默认是1236）

\*这里假设GatewayClient和Register服务都在一台服务器上，ip填写127.0.0.1

\*\*/

Gateway::$registerAddress = '127.0.0.1:1236';

// 以下是调用示例，接口与GatewayWorker环境的接口一致

// 接口具体使用方法见《Lib\Gateway类提供的接口》一章

// 注意除了不支持sendToCurrentClient和closeCurrentClient方法

// 其它方法都支持 Gateway::sendToAll($data); Gateway::sendToClient($client\_id, $data); Gateway::closeClient($client\_id); Gateway::isOnline($client\_id); Gateway::bindUid($client\_id, $uid); Gateway::isUidOnline($uid); Gateway::getClientIdByUid($uid); Gateway::unbindUid($client\_id, $uid); Gateway::sendToUid($uid, $data); Gateway::joinGroup($client\_id, $group); Gateway::sendToGroup($group, $data); Gateway::leaveGroup($client\_id, $group); Gateway::getClientCountByGroup($group);

Gateway::getClientSessionsByGroup($group); Gateway::getAllClientCount(); Gateway::getAllClientSessions(); Gateway::setSession($client\_id, $session); Gateway::updateSession($client\_id, $session); Gateway::getSession($client\_id);

...

# laychat系统的基础搭建

#### 本章介绍如何快速的搭建这套系统的基础框架 搭建步骤

通后后台的搭建

workerman的集成 layIM的集成

**写在前面**

如果你看到了这个内容，那么感谢你对我的支持。这是我第一次写这样的文档，存在很多地方的不足。这套

系统是我利用空闲时间写的，当然作为学习用还是比较好的，可以帮助你理解wokerman的使用，以及 thinkphp5的使用，同时还能做一个看起来挺酷的webIM系统。 由于整理这个文档的时候，我早就完成了系统的开发。因此，想要完全的拆分出来，一步一步的讲，不太好 实现。因为一些功能互相之前有牵扯，但是我还是尽量的讲出自己的设计思路。

为了您更好的理解这套系统，我强烈建议 你先到 《laychat系统源码》 这一章节下载源码。另外，你还需 要自己阅读GetwayWoker的操作手册，thinkphp5的手册，layim的手册，laytpl手册，layer手册，laytree 手册，laypage手册。当然这些都很简单，相信你打开这个文档的时候，已经做了这部分的知识储备。

最后再次感谢您的支持！

# 通用后台的搭建

#### 你可以到github上去下载。推荐一个我的项目，github上 133 个star：[snake通用后台](https://github.com/nick-bai/snake)，虽然这个后台已经够 简洁了，已经集成的RBAC系统，会在我们开发时候带来一些小的阻碍（因为我们要不停的录入节点），如果 你觉得麻烦，那我们就从头打造一个不带RBAC的后台，来满足我们的开发。

下载初始化thinkphp5框架

去官网，点击thinkphp5.0.5完整版下载[thinkphp5.0.5完整版](http://www.thinkphp.cn/donate/download/id/870.html)，下载完毕之后，把他放到网站的根目录， 本人的系统根目录是 D:\WWW目录，你可以根据自己的实际情况，替换以下的相关配置。现在根目录下 新建一个文件加 lchat，将下载好的thinkphp5.0.5文件复制到D:\WWW\lchat文件夹下，解压缩到当前文 件夹。修改根目录下的 build.php文件：

<?php

// +----------------------------------------------------------------------

// | ThinkPHP [ WE CAN DO IT JUST THINK ]

// +----------------------------------------------------------------------

// | Copyright (c) 2006~2016 [http://thinkphp.cn](http://thinkphp.cn/) All rights reserved.

// +----------------------------------------------------------------------

// | Licensed ( <http://www.apache.org/licenses/LICENSE-2.0> )

// +----------------------------------------------------------------------

// | Author: liu21st <[liu21st@gmail.com](mailto:liu21st@gmail.com)>

// +----------------------------------------------------------------------

return [

// Th成应用公共文件

'**file**' => ['common.php', 'config.php', 'database.php'],

// 后台模块

'admin' => [

* file ' => ['common.php', 'config.php', 'database.php'],
* dir ' => ['behavior', 'controller', 'model', 'view'], 'controller' => ['Index'],

'model' => [],

'view' => ['index/index'],

],

// 前台模块

'index' => [

* file ' => ['common.php', 'config.php', 'database.php'],
* dir ' => ['behavior', 'controller', 'model', 'view'], 'controller' => ['Index'],

'model' => [],

'view' => ['index/index'],

]

// 其他更多的模块定义

];

添加 admin（后台）模块和 index ( 前台模块 )，打开 点击windows开始按键，输入 cmd，打开命令窗口。输入 命令：

D:

cd WWW\lchat

php think build --config build.php

等待，出现Successed，则表示初始化前后台文件成功，此时会在 application 文件加下看到我们需要的 admin 和 index 文件夹。此过程可参考手册：[自动生成目录结构](<http://www.kancloud.cn/manual/thinkphp5/> 118021)

\* 新建本地虚拟域名

打开 apache 的vhost-conf 文件，输入如下信息：

<VirtualHost \*:80>

DocumentRoot "D:\WWW\lchat\public" ServerName [www.lchat.com](http://www.lchat.com/)

\*注意\*： D:\WWW\lchat，根据你的实际情况改成自己的系统文件路径即可。 打开 windows 的hosts 文件，输入：

127.0.0.1 [www.lchat.com](http://www.lchat.com/)

重启 apache 打开 chrome浏览器，输入 [www.lchat.com](http://www.lchat.com/) 可以看到欢迎页面。输入： [www.lchat.com/admin](http://www.lchat.com/admin)

若显示：

No input file specified. 则将 public文件下的 .htacess 文件的最后一行，修改为：

RewriteRule ^(.\*)$ index.php?/$1 [QSA,PT,L]

在斜线 前面 加一个 ？号

\* 新建并配置 mysql数据库

用mysql操作软件 Navicat，新建一个数据库：laychat\_v3，并将编码选择为：utf8 -- UTF-8 Unicode，新建 一个 管理员表，方便我们后面来实现后台管理员的登录。

DROP TABLE IF EXISTS ;

v3\_admin

CREATE TABLE (

v3\_admin

int(11) NOT NULL AUTO\_INCREMENT,

id

varchar(155) NOT NULL, varchar(255) NOT NULL,

login\_name

password

PRIMARY KEY ( )

id

) ENGINE=MyISAM AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

// 数据库编码默认采用utf8

'charset'

// 数据库表前缀

'prefix'

=> 'utf8',

=> 'v3\_',

由于本系统前台 和 后台 采用的是同一套数据库，所以本次就不分开配置这些数据库的链接信息了。打开 applicat ion 文件夹下的，database.php，填写如下信息：

|  |  |  |
| --- | --- | --- |
| // 数据库类型 |  | |
| 'type'  // 服务器地址 | => | 'mysql', |
| 'hostname'  // 数据库名 | => | '127.0.0.1', |
| 'database'  // 用户名 | => | 'laychat\_v3', |
| 'username'  // 密码 | => | 'root', |
| 'password'  // 端口 | => | 'root', |
| 'hostport'  // 连接dsn | => | '3306', |
| 'dsn'  // 数据库连接参数 | => | '', |
| 'params' | => | [], |

\* 下载前后台ui资源

本次系统演示采用的后台UI是基于 layui实现的。具体的资源，我在这里就不做详细的下载介绍了，我将整理好的 静 态资源文件，放到这里：[static静态资源][(http://pan.baidu.com/s/1i48Rj8L)](http://pan.baidu.com/s/1i48Rj8L))，你只需要将这个下载下来

，放到 public/static，替换掉 static文件夹即可。

>[danger]注意： 在 static\common\js 下面，放置了所有的核心 业务代码，包括 main.js 和 find.js 等

，你在后面代码介绍中会频繁的看到讲解这部分代码。这也是前面注意事项的第4条。由于代码我完全写完了，无法完 全的拆分展示，所以通过先展示完整的js，然后倒过来讲解php和workerman的方式。

\* 配置静态资源

在根目录下 application/config.php 修改 视图输出字符串配置如下：

// 视图输出字符串内容替换

'view\_replace\_str' => [

'**CSS**' => '/static/admin/css', '**JS**' => '/static/admin/js',

'**COMMON**' => '/static/common'

#### ],

这样我们在模板中就可以用 这些变量来替换我们的资源路径了，便于我们后台的资源位置移动。

\* 编写后台的访问入口文件 index.php

找到 admin/controller/index.php 文件，输入如下的代码：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai [1902822973@qq.com](mailto:1902822973@qq.com)

// +----------------------------------------------------------------------

namespace app\admin\controller;

use think\Db;

use think\Controller;

class Index extends Controller

{

public function index()

{

return $this->fetch('/main');

}

public function homePage()

{

$version = Db::query('SELECT VERSION() AS ver');

$config = [

'url' => $\_SERVER['HTTP\_HOST'],

'document\_root' => $\_SERVER['DOCUMENT\_ROOT'], 'server\_os' => PHP\_OS,

'server\_port' => $\_SERVER['SERVER\_PORT'], 'server\_ip' => $\_SERVER['SERVER\_ADDR'], 'server\_soft' => $\_SERVER['SERVER\_SOFTWARE'], 'php\_version' => PHP\_VERSION, 'mysql\_version' => $version[0]['ver'],

'max\_upload\_size' => ini\_get('upload\_max\_filesize')

];

$this->assign([ 'config' => $config

]);

return $this->fetch('index');

}

#### }

找到 application/admin/view 文件夹，新建 main.html ,这个是 后台的基础框架，由于后台采用的是ifram e布局，这里我也采用了thinkphp5的模板布局，main.html的代码如下：

<!DOCTYPE html>

退出

你好！管理员, 欢迎登录

后台首页

群组管理 基础配置 群组列表 用户管理 用户列表 聊天设置 上传文件设置 用户的举报 举报列表

后台首页

2017 © Write by NickBai,Laychat-v3.0. 版权所有

此时再次访问 <http://www.lchat.com/admin> 后台的模样了。 是不是很简单呢，别忙，我们还有一个界面没有 处理，index.php这个文件中有一个 homePage方法，这个是后台访问的默认首页，我们在 application/admin/ view/index下面找到 index.html,把他替换成：

#### <!DOCTYPE html>

系统概览

|  |  |
| --- | --- |
| 网站域名 | {$config.url} |
| 网站目录 | {$config.document\_root} |
| 服务器操作系统 | {$config.server\_os} |
| 服务器端口 | {$config.server\_port} |
| 服务器IP | {$config.server\_ip} |
| 服务器环境 | {$config.server\_soft} |
| PHP版本 | {$config.php\_version} |
| MySQL版本 | {$config.mysql\_version} |
| 最大上传限制 | {$config.max\_upload\_size} |

关于作者

介绍

一个热爱技术的小菜鸟 邮箱：[1902822973@qq.com](mailto:1902822973@qq.com)

个人github

https://github.com/nick-bai/

个人博客

<http://www.cnblogs.com/nickbai/>

本项目

本项目是采用的layIM v3.0版+thinkphp5+workerman+layui,重构的第3版。因此叫 laychat-v3.0。 后台模板采用的是larryCMS

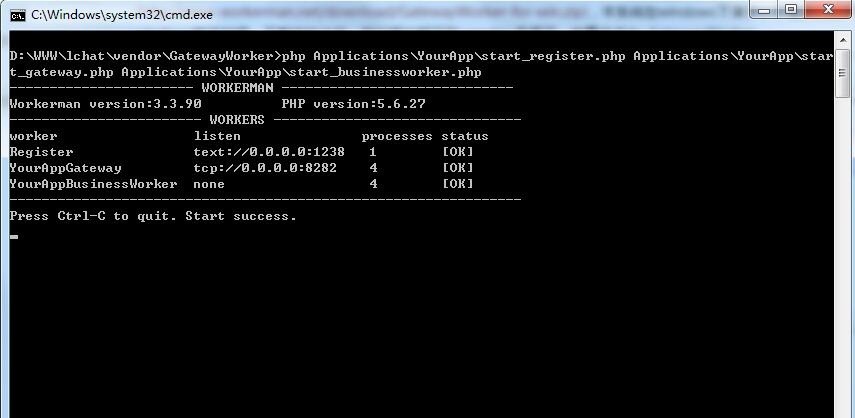
再次访问 <http://www.lchat.com/admin>

![](image/QQ截图20170313145836.jpg)

看到如下的页面，那么恭喜你，完美。后台搭建的初步效果已经达到。

# workerman的集成

#### 我们看到从官网下载的框架完整版，在 vendor 目录下面包含了一个workerman的文件。本系统采用的是基于 workerman开发的GetwayWorker框架，因此我们还需要自己去手动的下载这个框架。下载地 址：[GetwayWorker-for-win](http://www.workerman.net/download/GatewayWorker-for-win.zip)，本系统在windows下演示，于是我们就下载他的windows版本的吧。下载完毕之 后，我们把它解压到 vendor 目录下，并重命名为: GatewayWorker。

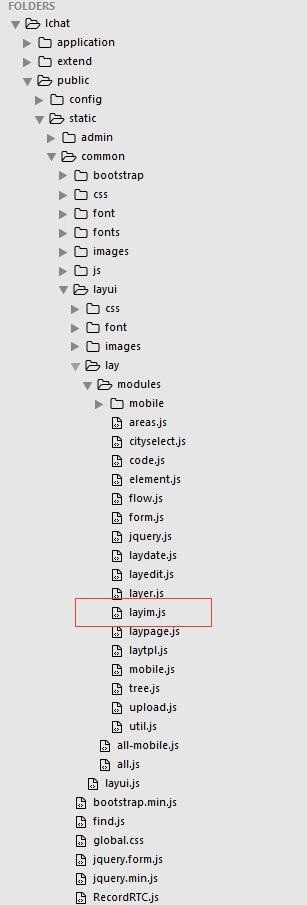
进入 vendor/GatewayWorker,双击 start\_for\_win.bat 出现如下的画面：

恭喜你，workerman的GatewayWorke框架，集成成功

# layIM的集成

#### layIM是需要授权才可以使用的资源，具体的授权价格，你可以去参考官网的规定，[layIM 官网](http://layim.layui.com/?from=layer)。由于版权问 题，我在这里不能放出 layim 的源代码。

你可以到官方进行授权获得源码之后，讲 layim.js 到 public\static\common\layui\lay\modules 下面，就像这 样：



本教程基于的是 layim v3.0.1 Pro 商用版

# 管理员后台功能的实现

#### 管理员后台功能如下：

基础登录退出功能

群组管理

1. 群组基础配置

2. 群组管理

用户管理

聊天设置 用户举报列表

# 后台登录退出的实现

#### 本章主要介绍登录功能的实现。

管理员数据库，我们已经在之前新建完成了。数据库为 v3\_admin表，手动加入一条记录，填写 用户名为 admin

登录密码为：21232f297a57a5a743894a0e4a801fc3

此处的密码，我们仅做简单的 md5加密，请在实际应用中采用更加安全的加密方式。密码的原文为 : admin 。数据库到此彻底准备完毕，那么我们开始实现功能吧。

**基类的建立**

所有的控制器，除了登录控制器之外的后台登录器，继承一个基类控制， Base.php，这样方便我们对登录状

态的控制。我们找到application\admin\controller文件夹。(注：后台的控制器都在此目录下新建) 新建 Base.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller; use think\Controller;

class Base extends Controller

{

public function \_initialize()

{

if(empty(cookie('username'))){

$this->redirect(url('login/index'));

}

$this->assign([

'username' => cookie('username')

]);

}

}

#### 修改 application\admin\controller 下的 Index.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

use think\Db;

class Index extends Base

{

public function index()

{

return $this->fetch('/main');

}

public function homePage()

{

$version = Db::query('SELECT VERSION() AS ver');

$config = [

'url' => $\_SERVER['HTTP\_HOST'],

'document\_root' => $\_SERVER['DOCUMENT\_ROOT'], 'server\_os' => PHP\_OS,

'server\_port' => $\_SERVER['SERVER\_PORT'], 'server\_ip' => $\_SERVER['SERVER\_ADDR'], 'server\_soft' => $\_SERVER['SERVER\_SOFTWARE'], 'php\_version' => PHP\_VERSION, 'mysql\_version' => $version[0]['ver'],

'max\_upload\_size' => ini\_get('upload\_max\_filesize')

];

$this->assign([ 'config' => $config

]);

return $this->fetch('index');

}

}

#### 让其继承 Base.php，而此时我们的 访问 [www.lchat.com/admin](http://www.lchat.com/admin) 的时候，回去默认访问 Index.php的index方

法，由于我们继承了 Base.php，而Base.php的构造方法中又进行了 cookie('username')的有无判断，从而确 认管理员是否登录。

我们在 application\admin 文件加下的 config.php 对后台的用户登录的 cookie有效期进行如下的配置：

<?php

//配置文件 return [

//cookie有效时长 'cookie\_save\_time' => 7200

];

**登录控制的建立以及功能的实现**

我们在 application\admin\controller 文件夹下新建一个 Login.php：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

use think\Controller;

class Login extends Controller

{

public function index()

{

return $this->fetch();

}

public function doLogin()

{

if(request()->isAjax()){

$uname = input('post.uname');

$pwd = input('post.pwd');

$admin = db('admin')->where('login\_name = "' . $uname . '"')->find(); if(empty($admin)){

return json(['code' => -1, 'data' => '', 'msg' => '管理员不存在']);

}

if( md5($pwd) != $admin['password'] ){

return json(['code' => -2, 'data' => '', 'msg' => '管理员或密码错误']);

}

//设置cookie

$time = config('cookie\_save\_time'); cookie('username', $uname, $time); cookie('uid', $admin['id'], $time);

return json(['code' => 1, 'data' => url('index/index'), 'msg' => '登录成功']

);

}

$this->error('非法访问');

}

public function loginOut()

{

cookie('username', null); cookie('uid', null);

$this->redirect(url('login/index'));

}

}

#### 然后在 admin\view 下新建一个 login文件夹，并在其下新建 index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>laychat-v3.0后台登录</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<!-- load css -->

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media= "all">

<link rel="stylesheet" type="text/css" href=" CSS /login.css" media="all">

</head>

<body>

<div class="layui-canvs"></div>

<div class="layui-layout layui-layout-login">

<h1>

<strong>Laychat-v3.0管理系统后台</strong>

</h1>

<div class="layui-user-icon larry-login">

<input type="text" placeholder="账号" class="login\_txtbx" id="uname"/>

</div>

<div class="layui-pwd-icon larry-login">

<input type="password" placeholder="密码" class="login\_txtbx" id="pwd"/>

</div>

<div class="layui-submit larry-login">

<input type="button" value="立即登陆" class="submit\_btn"/>

</div>

<div class="layui-login-text">

<p>© 2017-2018 NickBai 版权所有</p>

</div>

</div>

<script type="text/javascript" src=" COMMON /jquery.min.js"></script>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var url = "{:url('login/dologin')}";

</script>

<script type="text/javascript" src=" JS /login.js"></script>

</body>

</html>

#### 此时我们再次访问 [www.lchat.com/admin/](http://www.lchat.com/admin/) 则会自动跳转到登录页面，我们输入 用户名 admin 和 密码 : admin即可完成后台的登录。相信细心的你，肯定发现了，我们默认的框架页面 main.html 文件中，有一个：

<p>你好！管理员, 欢迎登录</p>

我们可以将此处的代码，换成：

<p>你好！{$username}, 欢迎登录</p>

因为我们在Base.php中渲染了 用户名，这样我们便可以实现，哪个管理员登录，显示哪个管理员的名称了。

**管理员退出**

我们在 main.html 中可以看到如下的代码：

<ul class="layui-nav larry-header-item">

<li class="layui-nav-item">

<a href="{:url('login/loginout')}">

<i class="iconfont icon-exit"></i>

退出</a>

</li>

</ul>

我在给出这个部分代码的时候，就已经给出了 退出的操作链接，这个链接会触发 login控制器的 loginOut方

法，从而清除登录标识，实现退出。

cookie('username', null); cookie('uid', null);

$this->redirect(url('login/index'));

# 群组管理的设计与实现

#### 后台群组管理的设计目的

给前台用户建立操作群组，设置标准，以及管理系统中存在的群组信息，便于对用户新建的群组进行有效的 管理。

主要功能有两块

群组的基础配置

系统群组列表

# 群组基础配置

## 数据以及功能的实现

#### 群组配置这一块，由于都是简答的配置项，我们为了简化设计，于是采用配置文件的形式，去实现这部分的

管理问题。在application\admin\controller文件夹下新建 Group.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Group extends Base

{

//群组设置

public function editGroup()

{

if(request()->isPost()){

$data = input('post.'); writeConfig($data);

return json(['code' => 1, 'data' => '', 'msg' => '配置成功']);

}

$config = readConfig();

empty($config) && $config = ['make' => 1, 'maxgroup' => 10, 'maxjoin' => 20, 'p ass' => -1];

$this->assign([ 'config' => $config

]);

return $this->fetch();

}

}

#### 为了方便文件的定位，我们在 public文件夹下的 index.php（入口文件）中，添加如下的配置：

define('APP\_ROOT', DIR )

在 public文件夹下新建一个 config文件夹放置我们的配置文件，在config下新建 一个空白的 group.conf文 件即可。

其中可见两个 方法，writeConfig和readConfig，这两个方法时我自定义的读写配置的方法，tp5的公用方法可

以写到 application下面的 common.php文件中

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

/\*\*

\* 写基础配置文件

\* @param $data

\*/

function writeConfig($data)

{

$path = APP\_ROOT . '/config/group.conf';

@file\_put\_contents($path, serialize($data)); return true;

}

//读配置文件

function readConfig()

{

$path = APP\_ROOT . '/config/group.conf';

$conf = file\_get\_contents($path); if(empty($conf))

return [];

return unserialize($conf);

}

#### 我将群组的配置序列化写入，group.conf中。

**视图部分的渲染**

application\admin\view新建一个 group文件夹，并在其下面新建一个 editgroup.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>基础配置</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<header class="larry-personal-tit">

<span>基础配置</span>

</header><!-- /header -->

<div class="larry-personal-body clearfix">

<form class="layui-form col-lg-5" id="commentForm" method="post" action="{: url('group/editgroup')}">

<div class="layui-form-item">

<label class="layui-form-label">创建群组</label>

<div class="layui-input-block">

<input type="radio" name="make" value="1" title="开启" {if condi tion="$config['make'] eq 1"}checked {/if}>

<div class="layui-unselect layui-form-radio layui-form-radioed"

>

<i class="layui-anim layui-icon"></i>

<span>开启</span>

</div>

<input type="radio" name="make" value="-1" title="关闭" {if cond

ition="$config['make'] eq -1"}checked {/if}>

<div class="layui-unselect layui-form-radio">

<i class="layui-anim layui-icon"></i>

<span>关闭</span>

</div>

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">是否审核</label>

<div class="layui-input-block">

<input type="radio" name="pass" value="1" title="是" {if condit ion="$config['pass'] eq 1"}checked {/if}>

<div class="layui-unselect layui-form-radio layui-form-radioed"

>

<i class="layui-anim layui-icon"></i>

<span>是 </span>

</div>

<input type="radio" name="pass" value="-1" title="否" {if condi

tion="$config['pass'] eq -1"}checked {/if}>

<div class="layui-unselect layui-form-radio">

<i class="layui-anim layui-icon"></i>

<span>否 </span>

</div>

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">最多创建群组数</label>

<div class="layui-input-block">

<input type="text" name="maxgroup" autocomplete="off" class="l ayui-input" lay-verify="number" value="{$config['maxgroup']}">

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">最多加入群组数</label>

<div class="layui-input-block">

<input type="text" name="maxjoin" autocomplete="off" class="la yui-input" lay-verify="number" value="{$config['maxjoin']}">

</div>

</div>

提交</button> utton>

<div class="layui-form-item">

<div class="layui-input-block">

<button class="layui-btn" lay-submit="" lay-filter="demo1">立即

<button type="reset" class="layui-btn layui-btn-primary">重置</b

</div>

</div>

</form>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /jquery.min.js"></script>

<script type="text/javascript" src=" COMMON /jquery.form.js"></script>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<script type="text/javascript"> layui.use(['form'],function(){

var form = layui.form();

});

$(function(){

var options = { beforeSubmit:showStart, success:showSuccess

};

$('#commentForm').submit(function(){

$(this).ajaxSubmit(options); return false;

});

});

var loading = ''; function showStart(){

layui.use(['layer'],function(){ let layer = layui.layer;

loading = layer.load(0, {shade: false});

});

return true;

}

function showSuccess(data){ layui.use(['layer'],function(){

let layer = layui.layer; layer.close( loading ); if( 1 == data.code ){

layer.msg(data.msg, {'time' : 2000});

//layer.alert( data.msg, {'icon': 1, 'closeBtn': 0, 'title': '友情提示'}

);

}else{

//layer.alert( data.msg, {'icon': 2, 'closeBtn': 0, 'title': '友情提示'}

);

layer.msg(data.msg, {'time' : 2000});

}

});

}

</script>

</body>

</html>

# 群组管理

群组管理分为：群组列表以及基于其上的审核操作。 所有用户新建的群组，在后台均有展示，方便管理员进行监管。同时，如果在基础配置中打开了，新建群 组需要审核，此时用户新建的群组默认是不可以使用的，也需要后台管理的审核方可Th效。

## 群组列表的实现

#### 新建一个群组的数据库，后台的群组管理主要读取这张表的数据，群组表主要功能是记录前台新建的群组。

DROP TABLE IF EXISTS `v3\_chatgroup`; CREATE TABLE `v3\_chatgroup` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '群组id',

`group\_name` varchar(155) NOT NULL COMMENT '群组名称',

`avatar` varchar(155) DEFAULT NULL COMMENT '群组头像',

`owner\_name` varchar(155) DEFAULT NULL COMMENT '群主名称',

`owner\_id` int(11) DEFAULT NULL COMMENT '群主id',

`owner\_avatar` varchar(155) DEFAULT NULL COMMENT '群主头像',

`owner\_sign` varchar(155) DEFAULT NULL COMMENT '群主签名',

`addtime` int(10) DEFAULT NULL,

`status` tinyint(1) DEFAULT NULL COMMENT '1审核通过 -1待审核 -2审核不通过', PRIMARY KEY (`id`)

) ENGINE=MyISAM AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

application\admin\controller 下面的 Group.php文件：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Group extends Base

{

//群组列表

public function index()

{

$where = '1=1';

$groupName = input('param.group\_name'); if(!empty($groupName)){

$where .= ' and group\_name = "' . $groupName . '"';

}

$list = db('chatgroup')->where($where)->order('addtime desc')->paginate(10);

$this->assign([ 'list' => $list,

'gname' => empty($groupName) ? '' : $groupName, 'total' => $list->total(),

'status' => config('group\_status')

]);

return $this->fetch();

}

}

#### 加入一个 index方法，来展示用户新建的群组信息。视图部分，在application\admin\view\group 下面新建

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>群组列表</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<div class="layui-tab">

<blockquote class="layui-elem-quote mylog-info-tit">

<form class="layui-form layui-form-pane" action="{:url('group/index')}"

method="get">

<div class="layui-inline">

<div class="layui-input-inline">

<input name="group\_name" value="{$gname}" placeholder="请输

入群组名称" class="layui-input" type="text">

</div>

</div>

<div class="layui-inline">

<button class="layui-btn" style="font-size:14px">查询</button>

</div>

</form>

</blockquote>

<div class="larry-separate"></div>

<div class="layui-tab-content larry-personal-body clearfix mylog-info-box">

<!-- 操作日志 -->

<div class="layui-tab-item layui-field-box layui-show">

<table class="layui-table table-hover" lay-even="" lay-skin="nob">

<thead>

<tr>

<th></th>

<th>群组ID</th>

<th>群组名称</th>

<th>群组管理员</th>

<th>群组头像</th>

<th>创建时间</th>

<th>状态</th>

<th>操作</th>

</tr>

</thead>

<tbody>

{if !empty($list)}

{foreach name="list" item="vo"}

<tr>

<td></td>

<td>{$vo.id}</td>

<td>{$vo.group\_name}</td>

<td>{$vo.owner\_name}</td>

<td><img src="{$vo.avatar}" width="40px" height="40px"></td

>

<td>{$vo.addtime|date="Y-m-d H:i:s", ###}</td>

<td>

{if condition="$vo['status'] eq -2"}

<span style="color: red">{$status[$vo['status']]}</span

>

{else/}

{$status[$vo['status']]}

{/if}

</td>

<td>

{if condition="$vo['status'] eq 1"}

-

{else /}

<button class="layui-btn layui-btn-small" style="font-s

ize:14px" onclick="pass({$vo.id})">审核</button>

{/if}

</td>

</tr>

{/foreach}

{/if}

</tbody>

</table>

<span style="float: left;margin-top:34px">共 {$total} 条</span>

<div style="float: right">{$list->render()}</div>

</div>

</div>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var pass\_url = "{:url('group/pass')}"; function pass(id){

layui.use(['layer', 'jquery'], function(){ var layer = layui.layer,

$ = layui.jquery; layer.ready(function(){

//询问框

layer.confirm('是否通过审核？', { closeBtn: 0,

icon: 3,

title: '友情提示',

btn: ['通过','不通过'] //按钮

}, function(){

$.post(pass\_url, {'id' : id, 'status' : 1}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:2000}); setTimeout(function(){

window.location.reload();

}, 1500);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json');

}, function(){

$.post(pass\_url, {'id' : id, 'status' : -2}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:2000}); setTimeout(function(){

window.location.reload();

}, 1500);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json');

});

});

}) ;

}

</script>

</body>

</html>

#### 其中的js也实现了，基于列表展示的 审核功能。

**审核功能的实现**

application\admin\controller 下面的 Group.php文件：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Group extends Base

{

//审核群组

public function pass()

{

if(request()->isAjax()){

$id = input('post.id');

$status = input('post.status');

$flag = db('chatgroup')->where('id', $id)->setField('status', $status);

if(false === $flag){

return json(['code' => -1, 'data' => '', 'msg' => '审核失败']);

}

return json(['code' => 1, 'data' => '', 'msg' => '审核成功']);

}

}

}

#### 改变数据库中的相应的群组 status 即可。

最终完整的 Group.php，如下：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Group extends Base

{

//群组列表

public function index()

{

$where = '1=1';

$groupName = input('param.group\_name'); if(!empty($groupName)){

$where .= ' and group\_name = "' . $groupName . '"';

}

$list = db('chatgroup')->where($where)->order('addtime desc')->paginate(10);

$this->assign([ 'list' => $list,

'gname' => empty($groupName) ? '' : $groupName, 'total' => $list->total(),

'status' => config('group\_status')

]);

return $this->fetch();

}

//群组设置

public function editGroup()

{

if(request()->isPost()){

$data = input('post.'); writeConfig($data);

return json(['code' => 1, 'data' => '', 'msg' => '配置成功']);

}

$config = readConfig();

empty($config) && $config = ['make' => 1, 'maxgroup' => 10, 'maxjoin' => 20, 'p ass' => -1];

$this->assign([ 'config' => $config

]);

return $this->fetch();

}

//审核群组

public function pass()

{

if(request()->isAjax()){

$id = input('post.id');

$status = input('post.status');

$flag = db('chatgroup')->where('id', $id)->setField('status', $status);

if(false === $flag){

return json(['code' => -1, 'data' => '', 'msg' => '审核失败']);

}

return json(['code' => 1, 'data' => '', 'msg' => '审核成功']);

}

}

}

# 用户管理

用户管理，方便管理员查看系统一共注册了多少位用户，对他们的信息有基础的了解。

## 前台用户表数据库的设计

#### 为了兼容layim的用户格式，我们将注册的前台用户信息按照如下的数据库格式存储：

DROP TABLE IF EXISTS `v3\_chatuser`;

CREATE TABLE `v3\_chatuser` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(155) DEFAULT NULL,

`pwd` varchar(155) DEFAULT NULL COMMENT '密码',

`sign` varchar(255) DEFAULT NULL,

`avatar` varchar(255) DEFAULT NULL,

`sex`

`age`

`pid`

`cid`

`aid`

tinyint(1) DEFAULT '1' COMMENT '性别 1男 -1女',

int(3) DEFAULT '18' COMMENT '年龄',

int(10)

int(10) int(10)

DEFAULT

DEFAULT DEFAULT

'110000'

'110000'

'110101'

COMMENT '所在省份id',

COMMENT '所在城市id', COMMENT '所在区id',

`area` varchar(255) DEFAULT '北京-北京市-东城区' COMMENT '所在区域描述',

`status` tinyint(1) DEFAULT '0' COMMENT '0下线 1在线', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

**后台管理员查看用户列表**

在application\admin\controller新建 User.php文件

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class User extends Base

{

//用户列表

public function index()

{

$where = '1=1';

$userName = input('param.user\_name'); if(!empty($userName)){

$where .= ' and user\_name = "' . $userName . '"';

}

$list = db('chatuser')->where($where)->paginate(10);

$this->assign([ 'list' => $list,

'uname' => empty($userName) ? '' : $userName, 'total' => $list->total(),

'sex' => ['1' => '男', '-1' => '女']

]);

return $this->fetch();

}

}

#### 视图渲染部分：在application\admin\view下新建 user文件加，并在其下新建 index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>群组列表</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<div class="layui-tab">

<blockquote class="layui-elem-quote mylog-info-tit">

<form class="layui-form layui-form-pane" action="{:url('user/index')}"

method="get">

<div class="layui-inline">

<div class="layui-input-inline">

<input name="user\_name" value="{$uname}" placeholder="请输入

用户名" class="layui-input" type="text">

</div>

</div>

<div class="layui-inline">

<button class="layui-btn" style="font-size:14px">查询</button>

</div>

</form>

</blockquote>

<div class="larry-separate"></div>

<div class="layui-tab-content larry-personal-body clearfix mylog-info-box">

<!-- 操作日志 -->

<div class="layui-tab-item layui-field-box layui-show">

<table class="layui-table table-hover" lay-even="" lay-skin="nob">

<thead>

<tr>

<th></th>

<th>用户ID</th>

<th>用户名称</th>

<th>用户头像</th>

<th>性别</th>

<th>年龄</th>

<th>所在地区</th>

</tr>

</thead>

<tbody>

{if !empty($list)}

{foreach name="list" item="vo"}

<tr>

<td></td>

<td>{$vo.id}</td>

<td>{$vo.user\_name}</td>

<td><img src="{$vo.avatar}" width="40px" height="40px"></td

>

<td>{$sex[$vo.sex]}</td>

<td>{$vo.age}</td>

<td>{$vo.area}</td>

</tr>

{/foreach}

{/if}

</tbody>

</table>

<span style="float: left;margin-top:34px">共 {$total} 条</span>

<div style="float: right">{$list->render()}</div>

</div>

</div>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

</body>

</html>

#### 即可完成对前台注册用户的查看。

# 聊天设置

聊天设置主要是对前台用户聊天中上传图片的大小和后缀等信息做出配置，防止用户上传不合法的图片。

## 配置文件的存储方式设计

#### 由于配置的项目比较少，此次依旧是采用文件的存储方式。在public\config 文件夹下新建 chat.conf。

**配置方法的实现**

在application\admin\controller文件家下 新建 Chat.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Chat extends Base

{

public function index()

{

if(request()->isPost()){

$data = input('post.');

$data['file\_ext'] = trim($data['file\_ext']);

$data['img\_ext'] = trim($data['img\_ext']);

writeCtConfig($data);

return json(['code' => 1, 'data' => '', 'msg' => '配置成功']);

}

$config = readCtConfig();

empty($config) &&

$config = ['file\_size' => 2, 'file\_ext' => 'zip|rar', 'img\_size' => 2, 'img\_ext ' => 'png|jpg|gif'];

$this->assign([ 'config' => $config,

'up\_size' => ini\_get('upload\_max\_filesize')

]);

return $this->fetch();

}

}

#### 在 application 文件下的 common.php 公用方法中添加两个读写配置文件方法：

//写聊天配置

function writeCtConfig($data)

{

$path = APP\_ROOT . '/config/chat.conf';

@file\_put\_contents($path, serialize($data)); return true;

}

//读聊天配置文件

function readCtConfig()

{

$path = APP\_ROOT . '/config/chat.conf';

$conf = file\_get\_contents($path); if(empty($conf))

return [];

return unserialize($conf);

}

视图部分：

在application\admin\view文件夹下新建 chat文件夹，然后在chat下新建 index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>上传文件配置</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<header class="larry-personal-tit">

<span>上传文件配置</span>

</header><!-- /header -->

<div class="larry-personal-body clearfix">

<form class="layui-form col-lg-5" id="commentForm" method="post" action="{: url('chat/index')}">

<div class="layui-form-item">

<label class="layui-form-label">最大文件大小</label>

<div class="layui-input-block">

<input type="text" name="file\_size" autocomplete="off" class=" layui-input col-lg-5" lay-verify="number" value="{$config.file\_size}">

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">文件后缀</label>

<div class="layui-input-block">

<input type="text" name="file\_ext" autocomplete="off" class="l ayui-input col-lg-5" lay-verify="required" value="{$config.file\_ext}">

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">图片大小</label>

<div class="layui-input-block">

<input type="text" name="img\_size" autocomplete="off" class="l ayui-input col-lg-5" lay-verify="number" value="{$config.img\_size}">

</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">图片后缀</label>

<div class="layui-input-block">

<input type="text" name="img\_ext" autocomplete="off" class="la yui-input col-lg-5" lay-verify="required" value="{$config.img\_ext}">

</div>

</div>

<div class="layui-form-item">

<div class="layui-input-block">

<button class="layui-btn" lay-submit="" lay-filter="demo1">立即

提交</button>

utton>

。<br/>

<button type="reset" class="layui-btn layui-btn-primary">重置</b

</div>

</div>

<div class="layui-form-item fly-form-app">

注：1、当前php允许的上传文件大小为{$up\_size}。<br/> 2、填写的文件和图片大小的单位是M，只需填写数字即可，不要超过php允许的最大值

3、文件和图片后缀用 英文的 | 来隔开，不允许出现空格。

</div>

</form>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /jquery.min.js"></script>

<script type="text/javascript" src=" COMMON /jquery.form.js"></script>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<script type="text/javascript"> layui.use(['form'],function(){

var form = layui.form();

});

$(function(){

var options = { beforeSubmit:showStart, success:showSuccess

};

$('#commentForm').submit(function(){

$(this).ajaxSubmit(options); return false;

});

});

var loading = ''; function showStart(){

layui.use(['layer'],function(){ let layer = layui.layer;

loading = layer.load(0, {shade: false});

});

return true;

}

function showSuccess(data){ layui.use(['layer'],function(){

let layer = layui.layer;

layer.close( loading );

if( 1 == data.code ){ layer.msg(data.msg, {'time' : 2000});

//layer.alert( data.msg, {'icon': 1, 'closeBtn': 0, 'title': '友情提示'}

);

}else{

//layer.alert( data.msg, {'icon': 2, 'closeBtn': 0, 'title': '友情提示'}

);

layer.msg(data.msg, {'time' : 2000});

}

});

}

</script>

</body>

</html>

# 用户的举报

用户的举报，主要的是让管理员了解哪些用户被举报了，并且可以查看这些用户被举报的证据，从而做出 是否限制被举报用户的操作。

## 数据库设计

DROP TABLE IF EXISTS `v3\_report`; CREATE TABLE `v3\_report` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`report\_uid` int(11) NOT NULL COMMENT '举报人id',

`report\_user` varchar(155) NOT NULL COMMENT '举报人名',

`reported\_uid` int(11) NOT NULL COMMENT '被举报人id',

`reported\_user` varchar(155) NOT NULL COMMENT '被举报人名',

`addtime` int(11) NOT NULL COMMENT '举报时间',

`content` text COMMENT '举报证据',

`report\_type` varchar(155) NOT NULL COMMENT '举报类型大类',

`report\_detail` varchar(155) NOT NULL COMMENT '举报类型小类', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

**功能的实现**

#### 在application\admin\controller下新建，Report.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\admin\controller;

class Report extends Base

{

//举报信息

public function index()

{

$where = '1=1';

$userName = input('param.user\_name'); if(!empty($userName)){

$where .= ' and report\_user = "' . $userName . '"';

}

$list = db('report')->field('id,report\_uid,report\_user,reported\_uid,reported\_us er,report\_type,report\_detail')

->where($where)->order('addtime desc')->paginate(10);

$this->assign([ 'list' => $list,

'uname' => empty($userName) ? '' : $userName, 'total' => $list->total()

]);

return $this->fetch();

}

//查看证据

public function seeDetail()

{

$id = input('id');

$detail = db('report')->field('content')->where('id', $id)->find(); if(!empty($detail)){

$detail = unserialize($detail['content']);

}

$this->assign([ 'detail' => $detail

]);

return $this->fetch();

}

}

#### 视图部分的渲染：

在 application\admin\view下新建 report文件夹，并在report新建 index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>举报列表</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<div class="layui-tab">

<blockquote class="layui-elem-quote mylog-info-tit">

<form class="layui-form layui-form-pane" action="{:url('report/index')}

" method="get">

<div class="layui-inline">

<div class="layui-input-inline">

<input name="user\_name" value="{$uname}" placeholder="请输入

举报人姓名" class="layui-input" type="text">

</div>

</div>

<div class="layui-inline">

<button class="layui-btn" style="font-size:14px">查询</button>

</div>

</form>

</blockquote>

<div class="larry-separate"></div>

<div class="layui-tab-content larry-personal-body clearfix mylog-info-box">

<!-- 操作日志 -->

<div class="layui-tab-item layui-field-box layui-show">

<table class="layui-table table-hover" lay-even="" lay-skin="nob">

<thead>

<tr>

<th></th>

<th>举报人ID</th>

<th>举报人名称</th>

<th>被举报人ID</th>

<th>被举报人名称</th>

<th>举报类型</th>

<th>举报详情</th>

<th>操作</th>

</tr>

</thead>

<tbody>

{if !empty($list)}

{foreach name="list" item="vo"}

<tr>

<td></td>

<td>{$vo.report\_uid}</td>

<td>{$vo.report\_user}</td>

<td>{$vo.reported\_uid}</td>

<td>{$vo.reported\_user}</td>

<td>{$vo.report\_type}</td>

<td>{$vo.report\_detail}</td>

<td>

<button class="layui-btn layui-btn-small" style="font-s ize:12px" onclick="show\_tips({$vo.id})">查看证据</button>

</td>

</tr>

{/foreach}

{/if}

</tbody>

</table>

<span style="float: left;margin-top:34px">共 {$total} 条</span>

<div style="float: right">{$list->render()}</div>

</div>

</div>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<script type="text/javascript"> function show\_tips(id){

layui.use(['layer', 'jquery'], function(){ var layer = layui.layer,

$ = layui.jquery; layer.open({

type: 2,

title: '查看举报证据', shadeClose: true, shade: 0.2,

skin: 'layui-layer-molv', area: ['750px', '500px'],

content: '/admin/report/seeDetail/id/' + id //iframe的url

});

});

}

</script>

</body>

</html>

#### 以及查看详情的页面seedetail.html：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>查看证据</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<div class="layui-tab">

<blockquote class="layui-elem-quote mylog-info-tit" style="font-size: 14px"

>

被举报人 -> 给举报人&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;最近发出的消息

</blockquote>

<div class="larry-separate"></div>

<div class="layui-tab-content larry-personal-body clearfix mylog-info-box">

<!-- 操作日志 -->

<div class="layui-tab-item layui-field-box layui-show">

<table class="layui-table table-hover" lay-even="" lay-skin="nob">

<tbody>

{if !empty($detail)}

{foreach name="detail" item="vo"}

<blockquote class="layui-elem-quote layui-quote-nm" style="font

-size: 14px;margin-top: 5px">

{$vo.content}

</blockquote>

{/foreach}

{else/}

<span style="color: red;margin-left:37%;font-size: 14px">暂无聊

天记录作为证据</span>

{/if}

</tbody>

</table>

</div>

</div>

</div>

</div>

</section>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

</body>

</html>

# 聊天系统的设计与实现

#### laychat v3.0 实现了如下的功能

聊天用户的注册

聊天用户的登录 修改个人资料 修改个性签名 切换在线状态 按条件搜索添加好友 系统消息盒子

添加好友申请与同意/拒绝好友 好友聊天

创建群组 按条件查找群 申请加入群组 管理我的群组 群组聊天 右键退出我的群组 聊天发送语音 聊天发送图片 聊天发送文件

空间的说说发表，评论功能 查看个人和群聊记录 好友右键功能

1. 进入好友空间

2. 将好友加入黑名单

3. 移动好友分组

4. 删除好友

5. 举报好友

# 注册功能的实现

#### 本套webIM系统跟传统的qq一样，需要用户注册才能进入系统。前面我们搭建的thinkphp5的通用系统，给前 台设计了index模块。详情可以参考 系统结构说明。

**数据库说明**

用户注册进入用户聊天表 v3\_chatuser

DROP TABLE IF EXISTS `v3\_chatuser`;

CREATE TABLE `v3\_chatuser` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(155) DEFAULT NULL,

`pwd` varchar(155) DEFAULT NULL COMMENT '密码',

`sign` varchar(255) DEFAULT NULL,

`avatar` varchar(255) DEFAULT NULL,

`sex`

`age`

`pid`

`cid`

`aid`

tinyint(1) DEFAULT '1' COMMENT '性别 1男 -1女',

int(3) DEFAULT '18' COMMENT '年龄',

int(10)

int(10) int(10)

DEFAULT

DEFAULT DEFAULT

'110000'

'110000'

'110101'

COMMENT '所在省份id',

COMMENT '所在城市id', COMMENT '所在区id',

`area` varchar(255) DEFAULT '北京-北京市-东城区' COMMENT '所在区域描述',

`status` tinyint(1) DEFAULT '0' COMMENT '0下线 1在线', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

**系统设计**

用户登录聊天系统的前提是必须要登录系统，因此我们在index模块也设计一个Base.php控制器，所有的功能

控制器继承这个基类控制器，从而实现用户所有的操作都在登录的前提下。在 application\index\controller 新建 Base.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

use think\Controller; use session\Session;

class Base extends Controller

{

public function \_initialize()

{

if(empty(session('f\_user\_name'))){

$this->redirect(url('login/index'));

}

}

}

#### 为了区分与管理员后台的登录存储状态，我们在这里采用了 session 来存储登录的状态。Base类的功能很简

洁，也很容易理解。就是判断当前用户是否处于登录状态。否则强制用户登录。

当用户访问 <http://www.lchat.com/> 时，由于我们的默认配置，系统会自动寻找 index 模块下的 index 控制器 的 index 方法，如果你不想让系统默认访问这个控制器，你可以在config.php 下面的

// 默认模块名

'default\_module'

// 默认控制器名

'default\_controller'

// 默认操作名

'default\_action'

=> 'index',

=>

'Index',

=>

'index',

在这里，由于默认的配置符合我的需求，就不做这方面的修改了。我们在 index模块的 controller 文件夹下新

建 Index.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller; use session\Session;

class Index extends Base

{

public function index()

{

//聊天用户

$userInfo = [

'id' => session('f\_user\_id'), 'username' => session('f\_user\_name'), 'avatar' => session('f\_user\_avatar'), 'sign' => session('f\_user\_sign')

];

$this->assign([

'uinfo' => $userInfo

]);

return $this->fetch();

}

}

#### 可以看到，Index的index方法很简单，如果用户登录的话，则会渲染index.html页面，否则在Base.php的构造

方法，就会强制用户去登录。那问题来了，不是说本节讲的是注册的吗？怎么全是在讲登录啊。别急，注册 没有专门的在一个入口让用户去访问注册，而是在登录的页面，设置了一个 没有账号 ？去注册，这样的机 制。类已于 qq，引导用户前去注册。



以上就是我们的登录页面，以及注册页面的入口。我们将注册和登录合并在一个控制器中去完成，下面 我们

在 index模块的 controller 文件夹下新建 Login.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller; use think\Controller;

class Login extends Controller

{

//注册页面

public function register()

{

$ip = request()->ip();

$taobaoUrl = '[http://ip.taobao.com/service/getIpInfo.php?ip='](http://ip.taobao.com/service/getIpInfo.php?ip=%27) . $ip;

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $taobaoUrl); curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt ( $ch, CURLOPT\_NOSIGNAL,true);//支持毫秒级别超时设置

curl\_setopt($ch, CURLOPT\_TIMEOUT, 1200); //1.2秒未获取到信息，视为定位失败

$myCity = curl\_exec($ch); curl\_close($ch);

$myCity = json\_decode($myCity, true); if('中国' != $myCity['data']['country']){

$local = ['110000', '110100', '110101']; //默认定位北京东城

}else{

$local = [$myCity['data']['region\_id'], $myCity['data']['city\_id'], 0];

}

unset($myCity);

$this->assign([ 'local' => $local

]);

return $this->fetch();

}

}

#### 在展示用户注册页面的时候，我们首先根据用户的ip，给用户定位所在位置，这里采用的是淘宝的第三方接

口，这里定位的目的，是为了后面，我们加好友的时候，方便按照地区查找这个功能。定位失败的话，我们 会给一个默认的 北京东城区的定位。控制器有了，我们来看一下页面吧。在 application\index\view 下新建 一个 login文件夹，然后在其下面 新建 register.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<title>laychat-v3.0 - 注册</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

</head>

<body>

<div class="main layui-clear" style="width:40%;margin:0 auto;box-shadow: 2px 10px 15px 8px #888888;padding: 30px;">

<blockquote class="layui-elem-quote"> laychat-v3.0 注册 账号

</blockquote>

<div class="layui-form layui-form-pane">

<form method="post" id="register\_form">

<br/>

<h2 class="page-title layui-elem-quote layui-quote-nm">账号信息</h2>

<hr/>

<div class="layui-form-item">

<label for="user\_name" class="layui-form-label">用户名</label>

<div class="layui-input-inline">

<input type="text" id="user\_name" name="user\_name" lay-verify="user name" autocomplete="off"

class="layui-input">

</div>

<div class="layui-form-mid layui-word-aux">必填，将会成为您唯一的登入名</div

>

</div>

<div class="layui-form-item">

<label for="pwd" class="layui-form-label">密码</label>

<div class="layui-input-inline">

<input type="password" id="pwd" name="pwd" lay-verify="pass" autoco

mplete="off" class="layui-input">

</div>

<div class="layui-form-mid layui-word-aux">必填，6到16个字符</div>

</div>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">重复密码</label>

<div class="layui-input-inline">

<input type="password" id="repwd" name="repwd" lay-verify="pass" au

tocomplete="off"

class="layui-input">

</div>

<div class="layui-form-mid layui-word-aux">必填，6到16个字符</div>

</div>

<br/>

<h2 class="page-title layui-elem-quote layui-quote-nm">个人信息</h2>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">性别</label>

<div class="layui-input-block">

<input name="sex" value="1" title="男" checked="" type="radio">

<input name="sex" value="-1" title="女" type="radio">

</div>

</div>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">年龄</label>

<div class="layui-input-inline">

<input name="age" lay-verify="number" autocomplete="off" class="lay ui-input" type="number"

value="18">

</div>

<div class="layui-form-mid layui-word-aux">限数字</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">居住地区</label>

<div class="layui-input-inline">

<select name="province" lay-filter="province">

<option value="">请选择省</option>

</select>

</div>

<div class="layui-input-inline">

<select name="city" lay-filter="city">

<option value="">请选择市</option>

</select>

</div>

<div class="layui-input-inline">

<select name="area" lay-filter="area">

<option value="">请选择县/区</option>

</select>

</div>

</div>

<div class="layui-form-item">

<button class="layui-btn" lay-filter="\*" lay-submit>立即注册</button>

</div>

<div class="layui-form-item fly-form-app"> 注：所在区域不选择,系统默认为根据ip定位的所在地,只精确到市。<br/> 若无法定位，则默认选中 北京-东城区

</div>

</form>

</div>

</div>

<script src=" COMMON /layui/layui.js"></script>

<script src=" COMMON /jquery.min.js"></script>

<script src=" COMMON /layui/lay/modules/cityselect.js"></script>

<script type="text/javascript">

var area\_url = "{:url('findgroup/getArea')}";

var do\_register\_url = "{:url('login/doregister')}"; var p\_code = "{$local['0']}";

var c\_code = "{$local['1']}";

var a\_code = "{$local['2']}";

</script>

<script type="text/javascript" src=" COMMON /js/register.js"></script>

</body>

</html>

#### 访问 <http://www.lchat.com/index/login/register.html> 即可看到 我们的注册页面：

整体注册部分的js代码 register.js 在下载的静态资源中可以找到。这里就不做代码的详情展示了。用户填写相 应的内容，提交注册，即可完成账号的注册。处理注册的代码我们也写在 Login.php中：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller; use think\Controller;

class Login extends Controller

{

//注册页面

public function register()

{

$ip = request()->ip();

$taobaoUrl = '[http://ip.taobao.com/service/getIpInfo.php?ip='](http://ip.taobao.com/service/getIpInfo.php?ip=%27) . $ip;

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $taobaoUrl); curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt ( $ch, CURLOPT\_NOSIGNAL,true);//支持毫秒级别超时设置

curl\_setopt($ch, CURLOPT\_TIMEOUT, 1200); //1.2秒未获取到信息，视为定位失败

$myCity = curl\_exec($ch); curl\_close($ch);

$myCity = json\_decode($myCity, true); if('中国' != $myCity['data']['country']){

$local = ['110000', '110100', '110101']; //默认定位北京东城

}else{

$local = [$myCity['data']['region\_id'], $myCity['data']['city\_id'], 0];

}

unset($myCity);

$this->assign([ 'local' => $local

]);

return $this->fetch();

}

//处理注册

public function doRegister()

{

if(request()->isAjax()){

$param = input('post.');

//TODO 理论上应该对所有的传入参数做正则校验,此处为了节省时间，暂时未做

if($param['pwd'] != $param['repwd']){

return json(['code' => -1, 'data' => '', 'msg' => '两次密码输入不一致']);

}

//查询获得区域描述

$where = 'id =' . $param['province']; if(!empty($param['city'])){

$where .= ' or id=' . $param['city'];

}else{

$param['city'] = 0;

}

if(!empty($param['area'])){

$where .= ' or id=' . $param['area'];

}else{

$param['area'] = 0;

select();

}

$area = db('area')->field('area\_name')->where($where)->order('level asc')->

$areaStr = ''; if(!empty($area)){

foreach($area as $key=>$vo){

$areaStr .= $vo['area\_name'] . '-';

}

$areaStr = rtrim($areaStr, '-');

}else{

$areaStr = '北京-北京市-东城区';

}

unset($area);

$insertData = [

'user\_name' => trim($param['user\_name']), 'pwd' => md5($param['pwd']),

'sign' => '暂无',

'avatar' => config('avatar'), 'sex' => $param['sex'],

'age' => $param['age'], 'pid' => $param['province'], 'cid' => $param['city'],

'aid' => $param['area'], 'area' => $areaStr, 'status' => 0

];

unset($param);

']);

db('chatuser')->insert($insertData);

return json(['code' => '1', 'data' => url('login/index'), 'msg' => '注册成功

}

$this->error('非法访问');

}

}

# 登录功能的实现(1)

#### 用户完成注册之后，就可以登录到聊天系统了。登录聊天系统，不仅仅是像我们的后台那么简单，记录用户 的登录状态之后，跳转到主页面即可。这里是本系统的核心重点，也是关系到后面的功能能否实现的第一 步。

因此，这里我要分两部分来讲解：

第一部分，我会快速的展示即讲解，普通登录所用到的操作。相信大家看了，系统后台登录功能的实现肯 定很快就能完成这部分的登录功能。

第二部分 我将会将重点放在，登录之后，如何与workerman建立链接，实现客户端与服务端的通信的建立 这部分内容。

**普通登录的是实现**

基本与后台的登录的功能类似，一个展示登录页面的功能，一个处理登录的功能。依旧将代码放在 Login.php

中

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller; use think\Controller;

class Login extends Controller

{

//展示登录页面

public function index()

{

return $this->fetch();

}

//处理登录

public function doLogin()

{

if(request()->isAjax()){

$userName = input('post.user\_name'); if(empty($userName)){

return json(['code' => -3, 'data' => '', 'msg' => '用户名不能为空']);

}

$pwd = input('post.pwd'); if(empty($pwd)){

return json(['code' => -4, 'data' => '', 'msg' => '密码不能为空']);

}

$user = db('chatuser')->field('id,user\_name,pwd,sign,avatar')

->where('user\_name = "' . $userName . '"')->find();

if(empty($user)){

return json(['code' => -1, 'data' => '', 'msg' => '用户不存在']);

}

if( md5($pwd) != $user['pwd'] ){

return json(['code' => -2, 'data' => '', 'msg' => '密码错误']);

}

//设置用户登录

db('chatuser')->where('id', $user['id'])->setField('status', 1);

//设置session标识状态

session('f\_user\_name', $user['user\_name']); session('f\_user\_id', $user['id']); session('f\_user\_sign', $user['sign']); session('f\_user\_avatar', $user['avatar']);

return json(['code' => 1, 'data' => url('index/index'), 'msg' => '登录成功']

);

}

$this->error('非法访问');

}

//注册页面

public function register()

{

$ip = request()->ip();

$taobaoUrl = '[http://ip.taobao.com/service/getIpInfo.php?ip='](http://ip.taobao.com/service/getIpInfo.php?ip=%27) . $ip;

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $taobaoUrl); curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt ( $ch, CURLOPT\_NOSIGNAL,true);//支持毫秒级别超时设置

curl\_setopt($ch, CURLOPT\_TIMEOUT, 1200); //1.2秒未获取到信息，视为定位失败

$myCity = curl\_exec($ch); curl\_close($ch);

$myCity = json\_decode($myCity, true); if('中国' != $myCity['data']['country']){

$local = ['110000', '110100', '110101']; //默认定位北京东城

}else{

$local = [$myCity['data']['region\_id'], $myCity['data']['city\_id'], 0];

}

unset($myCity);

$this->assign([ 'local' => $local

]);

return $this->fetch();

}

//处理注册

public function doRegister()

{

if(request()->isAjax()){

$param = input('post.');

//TODO 理论上应该对所有的传入参数做正则校验,此处为了节省时间，暂时未做

if($param['pwd'] != $param['repwd']){

return json(['code' => -1, 'data' => '', 'msg' => '两次密码输入不一致']);

}

//查询获得区域描述

$where = 'id =' . $param['province']; if(!empty($param['city'])){

$where .= ' or id=' . $param['city'];

}else{

$param['city'] = 0;

}

if(!empty($param['area'])){

$where .= ' or id=' . $param['area'];

}else{

$param['area'] = 0;

select();

}

$area = db('area')->field('area\_name')->where($where)->order('level asc')->

$areaStr = ''; if(!empty($area)){

foreach($area as $key=>$vo){

$areaStr .= $vo['area\_name'] . '-';

}

$areaStr = rtrim($areaStr, '-');

}else{

$areaStr = '北京-北京市-东城区';

}

unset($area);

$insertData = [

'user\_name' => trim($param['user\_name']), 'pwd' => md5($param['pwd']),

'sign' => '暂无',

'avatar' => config('avatar'), 'sex' => $param['sex'],

'age' => $param['age'], 'pid' => $param['province'], 'cid' => $param['city'],

'aid' => $param['area'], 'area' => $areaStr, 'status' => 0

];

unset($param);

']);

db('chatuser')->insert($insertData);

return json(['code' => '1', 'data' => url('login/index'), 'msg' => '注册成功

}

$this->error('非法访问');

}

}

#### 在这里，将用户常用的一些热点信息，放在session中，方便我们后面的读取。在 application\index\view\login 文件夹下，新建index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale

=1.0">

<title>Laychat-v3.0 - 登录</title>

<link href=" COMMON /layui/css/layui.css" rel="stylesheet">

<link href=" COMMON /css/bootstrap.min.css" rel="stylesheet">

<link href=" COMMON /css/font-awesome.min.css?v=4.4.0" rel="stylesheet">

<link href=" COMMON /css/animate.min.css" rel="stylesheet">

<link href=" COMMON /css/style.min.css" rel="stylesheet">

<link href=" COMMON /css/login.min.css" rel="stylesheet">

<!--[if lt IE 9]>

<meta http-equiv="refresh" content="0;ie.html" />

<![endif]-->

<script>

if(window.top!==window.self){window.top.location=window.location};

</script>

</head>

<body class="signin">

<div class="signinpanel">

<div class="row">

<div class="col-sm-7">

<div class="signin-info">

<div class="logopanel m-b">

<h1>[ Laychat-v3.0 ]</h1>

</div>

<div class="m-b"></div>

<h4>欢迎使用 <strong>Laychat-v3.0</strong></h4>

<strong>还没有账号？ <a href="{:url('login/register')}" target="\_blank">

立即注册&raquo;</a></strong>

</div>

</div>

<div class="col-sm-5">

<form method="post" action="{:url('login/dologin')}" id="login\_form">

<h4 class="no-margins">登录：</h4>

<p class="m-t-md">登录到Laychat-v3.0聊天页面</p>

<input type="text" class="form-control uname" placeholder="用户名" name= "user\_name" required/>

<input type="password" class="form-control pword m-b" placeholder="密码" name="pwd" required/>

<button class="btn btn-success btn-block" id="login\_btn">登录</button>

</form>

</div>

</div>

<div class="signup-footer">

<div class="pull-left">

&copy; 2017 All Rights Reserved. Laychat-v3.0

</div>

</div>

</div>

<script src=" COMMON /jquery.min.js"></script>

<script src=" COMMON /bootstrap.min.js?v=3.3.6"></script>

<script src=" COMMON /layui/layui.js"></script>

<script src=" COMMON /jquery.form.js"></script>

<script type="text/javascript">

$(function(){

var options = { beforeSubmit:showStart, success:showSuccess

};

$('#login\_form').submit(function(){

$(this).ajaxSubmit(options); return false;

});

function showStart(){ layui.use(['layer'], function(){

var layer = layui.layer; layer.ready(function(){

layer.load(0, {shade:false, time:100});

});

});

return true;

}

function showSuccess(data){ layui.use(['layer'], function(){

var layer = layui.layer; layer.ready(function(){

if( 1 == data.code ){ layer.msg(data.msg, {'time' : 2000}); window.location.href = data.data;

}else{

layer.msg(data.msg, {'time' : 2000});

}

});

});

}

});

</script>

</body>

</html>

#### 此时在没有登录的前提下，访问 <http://www.lchat.com/> 则会自动跳转到登录页面，用户输入自己的账号密

码，即可完成登录。

# 登录功能的实现(2)

#### 当用户完成登录之后，会自动跳转到 index 模块下的 index.php的index方法，这个方法也是渲染主页面的方 法。前面我门已经展示了这部分的代码，如果忘记的话，这里我再次展示 index.php里面的代码：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller; use session\Session;

class Index extends Base

{

public function index()

{

//聊天用户

$userInfo = [

'id' => session('f\_user\_id'), 'username' => session('f\_user\_name'), 'avatar' => session('f\_user\_avatar'), 'sign' => session('f\_user\_sign')

];

$this->assign([

'uinfo' => $userInfo

]);

return $this->fetch();

}

}

#### 功能很是简单，渲染用户的热点信息以及展示 页面。我们在 application\index\view 下面新建一个 index 文

件夹，并在其中新建一个 index.html

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Laychat-v3.0</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css" media="all">

<style>.layui-layim-user{cursor: pointer}</style>

<script src=" COMMON /jquery.min.js"></script>

<script src=" COMMON /RecordRTC.js"></script>

</head>

<body>

<input type="hidden" id="audio\_src"/>

<!-- audio box -->

<div class="wrapper wrapper-content animated" id="audio\_box" style="display: none;margi n-top: 10px">

<div class="row">

<span id="tips" style="margin-left:26%; color:red;"></span>

ic"/>

<div class="col-sm-12" style="text-align: center;margin-top:10px">

<img src=" COMMON /images/audio.png" width="100px" height="100px" id="a\_p

</div>

</div>

<div class="col-sm-4 col-sm-offset-4" style="margin:10px 0px 0px 20px">

<button class="layui-btn layui-btn-small" id="say">开始说话</button>

<button class="layui-btn layui-btn-small layui-btn-danger layui-btn-disabled" i d="over—say">结束发送</button>

</div>

</div>

<!-- audio box -->

<script type="text/javascript"> var my\_events;

var user\_list\_url = "{:url('index/getList')}";

var member\_list\_url = "{:url('index/getMembers')}"; var upload\_img\_url = "{:url('upload/uploadimg')}"; var upload\_file\_url = "{:url('upload/uploadFile')}"; var msg\_box\_url = "{:url('Msgbox/index')}";

var find\_url = "{:url('findgroup/index')}"; var chatlog\_url = "{:url('Chatlog/index')}";

var change\_sign\_url = "{:url('chatuser/changeSign')}"; var chat\_user\_url = "{:url('chatuser/index')}";

var save\_audio\_url = "{:url('tools/saveAudio')}"; var get\_noread\_url = "{:url('msgbox/getNoRead')}"; var uid = "{$uinfo.id}";

var uname = "{$uinfo.username}"; var avatar = "{$uinfo.avatar}"; var sign = "{$uinfo.sign}";

var join\_group\_url = "{:url('findgroup/joinDetail')}";

</script>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript" src=" COMMON /js/audio.js"></script>

<script type="text/javascript" src=" COMMON /js/main.js"></script>

</body>

</html>

#### 这里能看到，我定义了大量的接口地址，现在我们不用关心，这部分是用来干嘛的，后面都会讲到，有全部 能用到。我们主要来看看 main.js，这部分代码在 static 静态文件中，我都给出了，详情可以参考： main.js全部代码这一章节或者自己打开本地的代码进行查看，这里就不再罗列全部的代码了。

同样的，不用担心这里面的大量代码。每一个功能，我都会在后面的内容中讲解到。首先我们来看一下，基 础的配置

//基础配置 layim.config({

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

//查看群员接口

,members: {

url: member\_list\_url

,data: {}

}

,uploadImage: {

url: upload\_img\_url

}

,uploadFile: {

url: upload\_file\_url

}

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

}

]

,initSkin: '3.jpg' //1-5 设置初始背景

,msgbox: msg\_box\_url//消息盒子页面地址，若不开启，剔除该项即可

,find: find\_url //发现页面地址，若不开启，剔除该项即可

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

});

具体的每个接口地址有什么作用，我这里有都有注释，这个也是按照 layim的手册进行配置的，不熟悉的，可

以自行查找layim手册。[layim手册](http://www.layui.com/doc/modules/layim.html)，初次登陆，不使用其他功能的话，主要影响的是

init: {

url: user\_list\_url

,data: {}

}

初始化的接口，这里主要是调取当前登录用户的好友列表。从接口的地址：

var user\_list\_url = "{:url('index/getList')}";

我们可以看到，我是在index控制器中有一个 getList方法：

//获取列表

public function getList()

{

//查询自己的信息

$uid = session('f\_user\_id');

$mine = db('chatuser')->where('id', $uid)->find();

//查询当前用户的所处的群组

$groupArr = db('groupdetail')->alias('j')->field('c.group\_name groupname,c.id,c

.avatar')

->join('v3\_chatgroup c', 'j.group\_id = c.id')->where('j.user\_id', $uid)

->group('j.group\_id')->select();

$online = 0;

$group = []; //记录分组信息

$userGroup = config('user\_group');

$list = []; //群组成员信息

$i = 0;

$j = 0;

//查询该用户的好友

$friends = db('friends')->alias('f')->field('c.user\_name,c.id,c.avatar,c.sign,c

.status,f.group\_id')

->join('v3\_chatuser c', 'c.id = f.friend\_id')

->where('f.user\_id', $uid)->select();

foreach( $userGroup as $key=>$vo ){

$group[$i] = [ 'groupname' => $vo, 'id' => $key, 'online' => 0, 'list' => []

];

$i++;

}

unset( $userGroup );

foreach( $group as $key=>$vo ){ foreach( $friends as $k=>$v ) {

if ($vo['id'] == $v['group\_id']) {

$list[$j]['username'] = $v['user\_name'];

$list[$j]['id'] = $v['id'];

$list[$j]['avatar'] = $v['avatar'];

$list[$j]['sign'] = $v['sign'];

$list[$j]['status'] = empty($v['status']) ? 'offline' : 'online';

if (1 == $v['status']) {

$online++;

}

$group[$key]['online'] = $online;

$group[$key]['list'] = $list;

$j++;

}

}

$j = 0;

$online = 0; unset($list);

}

//print\_r($group);die; unset( $friends );

$return = [

'code' => 0, 'msg'=> '', 'data' => [

'mine' => [

'username' => $mine['user\_name'], 'id' => $mine['id'],

'status' => 'online', 'sign' => $mine['sign'], 'avatar' => $mine['avatar']

],

'friend' => $group, 'group' => $groupArr

],

];

return json( $return );

}

#### 为什么要按照这种格式返回数据，这里都可以从layim的手册中，找到答案。调取用户好友的时候，我用到了

一张 friends表：

DROP TABLE IF EXISTS `v3\_friends`; CREATE TABLE `v3\_friends` (

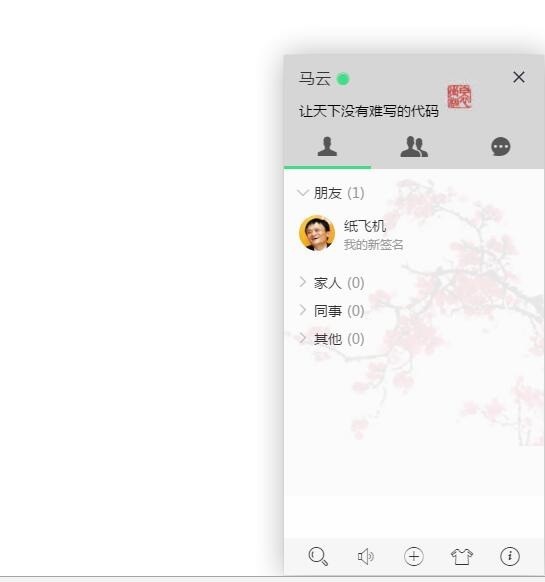
`user\_id` int(11) NOT NULL COMMENT '用户id',

`friend\_id` int(11) NOT NULL COMMENT '朋友id',

`group\_id` int(11) NOT NULL COMMENT '朋友所属组别id', KEY `user\_id` (`user\_id`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

这里是记录用户之间的好友关系。注释很明确，我就不做过多的解释了。

此时登录，并刷新页面就可以看到聊天的页面了。

那此时是不是就成功了呢？当然不是了，你现在看到的仅仅是一个静态页面，没有任何的功能，我们需要将 他与workerman通过websocket进行链接，这样所有的客户端都与workerman进行链接，就可以实现，客户端

#### 之间的通信了。

为什么客户端和客户端之间可以实现通信呢？ 每一个登录的客户端都与服务器通过tcp实现长链接，服务器将会记录每个链接与客户端之间的关系，这样 就可以区分链接来自哪个客户端。当一个客户端向指定的另一个客户端发送信息的时候，会先将信息推送 给服务器，并在信息中说明，我要将这个数据发送给谁，服务器通过自己内部的关系表，将数据准确的推 送给接收的客户端。

# 登录功能的实现(3)

#### 在实现客户端与workerman建立链接之前，我们先将workerman服务器配置完成。

**wokerman的配置**

项目的建立

我们将 vendor\GatewayWorker\Applications 下面的 YourApp 更名为 Laychat，并在其下，新建一个 Config文件夹，其下新建一个 Db.php 用来放数据库链接资源：

<?php

namespace Config;

/\*\*

* mysql配置
* @author walkor

\*/ class Db

{

/\*\*

\* 数据库的一个实例配置，则使用时像下面这样使用

* $user\_array = Db::instance('db1')->select('name,age')->from('users')->where('age> 12')->query();

\* 等价于

* $user\_array = Db::instance('db1')->query('SELECT `name`,`age` FROM `users` WHERE

`age`>12');

* @var array

\*/

public static $db1 = array( 'host' => '127.0.0.1',

'port' => 3306, 'user' => 'root', 'password' => 'root',

'dbname' => 'laychat\_v3', 'charset' => 'utf8'

);

}

#### Events.php 是我们放项目逻辑的主要地方 start\_businessworker.php 修改成如下的配置：

<?php

/\*\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

This file is part of workerman.

Licensed under The

For full copyright Redistributions of

MIT License

and license information, please see the MIT-LICENSE.txt files must retain the above copyright notice.

@author walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>

@copyright walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>

@link <http://www.workerman.net/>

@license <http://www.opensource.org/licenses/mit-license.php> MIT License

\*/

use \Workerman\Worker; use \Workerman\WebServer;

use \GatewayWorker\Gateway;

use \GatewayWorker\BusinessWorker; use \Workerman\Autoloader;

// 自动加载类

require\_once DIR . '/../../vendor/autoload.php';

// bussinessWorker 进程

$worker = new BusinessWorker();

// worker名称

$worker->name = 'LaychatBusinessWorker';

// bussinessWorker进程数量

$worker->count = 4;

// 服务注册地址

$worker->registerAddress = '127.0.0.1:1238';

// 如果不是在根目录启动，则运行runAll方法

if(!defined('GLOBAL\_START'))

{

Worker::runAll();

}

start\_gateway.php 修改成如下的配置：

<?php

/\*\*

* This file is part of workerman.

\*

* Licensed under The MIT License
* For full copyright and license information, please see the MIT-LICENSE.txt
* Redistributions of files must retain the above copyright notice.

\*

* @author walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>
* @copyright walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>
* @link <http://www.workerman.net/>
* @license <http://www.opensource.org/licenses/mit-license.php> MIT License

\*/

use \Workerman\Worker; use \Workerman\WebServer;

use \GatewayWorker\Gateway;

use \GatewayWorker\BusinessWorker; use \Workerman\Autoloader;

// 自动加载类

require\_once DIR . '/../../vendor/autoload.php';

// gateway 进程，Websocket协议实现webIM

$gateway = new Gateway("Websocket://0.0.0.0:8282");

// gateway名称，status方便查看

$gateway->name = 'LaychatAppGateway';

// gateway进程数

$gateway->count = 4;

// 本机ip，分布式部署时使用内网ip

$gateway->lanIp = '127.0.0.1';

// 内部通讯起始端口，假如$gateway->count=4，起始端口为4000

// 则一般会使用4000 4001 4002 4003 4个端口作为内部通讯端口

$gateway->startPort = 2900;

// 服务注册地址

$gateway->registerAddress = '127.0.0.1:1238';

// 心跳间隔

$gateway->pingInterval = 10;

// 连续次数

$gateway->pingNotResponseLimit = 2;

// 心跳数据

$gateway->pingData = '{"message\_type":"ping"}';

/\*

// 当客户端连接上来时，设置连接的onWebSocketConnect，即在websocket握手时的回调

$gateway->onConnect = function($connection)

{

$connection->onWebSocketConnect = function($connection , $http\_header)

{

// 可以在这里判断连接来源是否合法，不合法就关掉连接

// $\_SERVER['HTTP\_ORIGIN']标识来自哪个站点的页面发起的websocket链接 if($\_SERVER['HTTP\_ORIGIN'] != 'http://kedou.workerman.net')

{

$connection->close();

}

// onWebSocketConnect 里面$\_GET $\_SERVER是可用的

// var\_dump($\_GET, $\_SERVER);

};

};

\*/

// 如果不是在根目录启动，则运行runAll方法

if(!defined('GLOBAL\_START'))

{

Worker::runAll();

}

#### 注意：这里的重点是 $gateway = new Gateway("Websocket://0.0.0.0:8282"); 用websocket协议 start\_register.php修改成如下的配置：

<?php

/\*\*

* This file is part of workerman.

\*

* Licensed under The MIT License
* For full copyright and license information, please see the MIT-LICENSE.txt
* Redistributions of files must retain the above copyright notice.

\*

* @author walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>
* @copyright walkor<[walkor@workerman.net](mailto:walkor@workerman.net)>
* @link <http://www.workerman.net/>
* @license <http://www.opensource.org/licenses/mit-license.php> MIT License

\*/

use \Workerman\Worker;

use \GatewayWorker\Register;

// 自动加载类

require\_once DIR . '/../../vendor/autoload.php';

// register 必须是text协议

$register = new Register('text://0.0.0.0:1238');

// 如果不是在根目录启动，则运行runAll方法

if(!defined('GLOBAL\_START'))

{

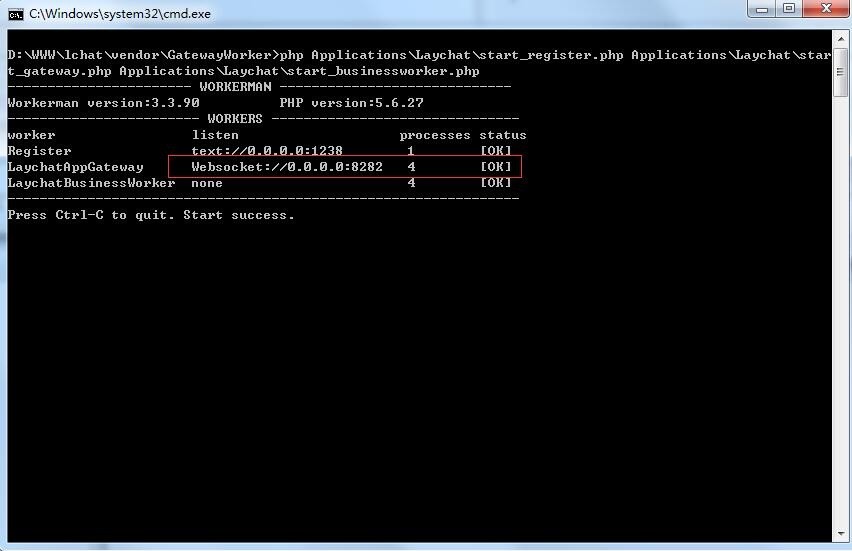
Worker::runAll();

}

#### 然后我们修改启动程序 start\_for\_win.bat

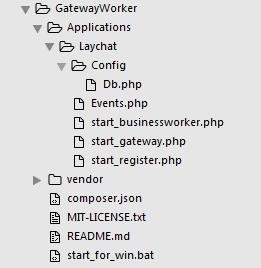
php Applications\Laychat\start\_register.php Applications\Laychat\start\_gateway.php Appl ications\Laychat\start\_businessworker.php

pause

双击 start\_for\_win.bat 出现如下画面：

表示 workerman服务器，配置完成。我们可以看到 此时websocket监听的是 8282端口。

**GetwayWoker扩展目录最终结构**



# 登录功能的实现(4)

#### 前面的workerman服务器已经配置完成，并成功的启动了。此时，保持启动窗口不要关闭。我们来实现客户 端与服务器的连接。workerman的websocket监听的是 8282 端口，因此，我们在进行 websocket链接的时 候，链接这个端口即可。

**web页面配置**

如何让web页面链接上workerman服务器呢？

socket = new WebSocket('ws://127.0.0.1:8282');

//连接成功时触发 socket.onopen = function(){

// 登录

var login\_data = '{"type":"init","id":"' + uid + '", "username":"' + uname + '"

, "avatar":"' + avatar + '", "sign":"' + sign + '"}'; socket.send( login\_data );

console.log("websocket握手成功!");

};

//监听收到的消息

socket.onmessage = function(res){}

通过h5的websocket接口，由于服务器是在本地，因此我们链接 127.0.0.1:8282端口，如果服务器建立链接之

后，就会触发 socket.onopen这个方法，可以看到我，我在服务器链接成功之后，发送了一段信息。这里，我 规定了一个数据格式：

{"type":"init","id":"' + uid + '", "username":"' + uname + '", "avatar":"' + avatar + ' ", "sign":"' + sign + '"}

type 标识消息的类型，便于workerman接收之后识别消息的类型，然后可以看到的是当前用户的基本信息。

为什么要发送这些消息呢？这里主要是为了方便通知我的好友，我上线了，而通知上线的接口，layim需要这 些数据，因此，我在此处将这些数据发送给workerman方便workerman推送上线消息，而不要再到数据库中 去查询，浪费时间。

注：如果你在外网或者局域网链接workerman，只要修改ws://127.0.0.1:8282 链接的ip和端口即可。

socket.onmessage 这个方法，主要是监听一些服务器推送过来的消息，然后做出相应的动作。例如，我们马

上要讲到的，某个用户登录之后，workerman会推送上线消息：

//监听收到的消息

socket.onmessage = function(res){

//console.log(res.data);

var data = eval("("+res.data+")"); switch(data['message\_type']){

// 在线

case 'online':

layim.setFriendStatus(data.id, 'online'); break;

}

}

接收到消息之后，按照 layim的设置上线的功能，直接调用即可。

**workerman的业务逻辑**

workerman的所有的业务逻辑，都写在 Events.php 文件中：

<?php

use \GatewayWorker\Lib\Gateway; use \GatewayWorker\Lib\Db;

/\*\*

\* 主逻辑

* 主要是处理 onConnect onMessage onClose 三个方法
* onConnect 和 onClose 如果不需要可以不用实现并删除

\*/

class Events

{

/\*\*

\* 当客户端发来消息时触发

* @param int $client\_id 连接id
* @param mixed $message 具体消息

\*/

public static function onMessage($client\_id, $data)

{

$message = json\_decode($data, true);

$message\_type = $message['type']; switch ($message\_type) {

case 'init':

// uid

$uid = $message['id'];

// 设置session

$\_SESSION = array(

'username' => $message['username'], 'avatar' => $message['avatar'], 'id' => $uid,

'sign' => $message['sign']

);

// 将当前链接与uid绑定

Gateway::bindUid($client\_id, $uid);

$db1 = Db::instance('db1'); //数据库链接

//查询最近1周有无需要推送的离线信息

$time = time() - 7 \* 3600 \* 24;

$resMsg = $db1->select('id,from\_id,from\_name,from\_avatar,timeline,conte nt')->from('v3\_chatlog')

->where("to\_id= {$uid} and timeline > {$time} and type = 'friend' a

nd need\_send = 1" )

->query();

//var\_export($resMsg); if (!empty($resMsg)) {

foreach ($resMsg as $key => $vo) {

$log\_message = [

'message\_type' => 'logMessage', 'data' => [

'username' => $vo['from\_name'], 'avatar' => $vo['from\_avatar'], 'id' => $vo['from\_id'],

'type' => 'friend',

'content' => htmlspecialchars($vo['content']), 'timestamp' => $vo['timeline'] \* 1000,

]

];

Gateway::sendToUid($uid, json\_encode($log\_message));

=" . $vo['id']);

}

}

//设置推送状态为已经推送

$db1->query("UPDATE `v3\_chatlog` SET `need\_send` = '0' WHERE id

// 通知所有该用户好友，此用户上线，将此用户头像变亮

$friends = $db1->query("select `user\_id` from `v3\_friends` where `frien d\_id` = " . $uid);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [ 'message\_type' => 'online', 'id' => $uid,

line\_message));

}

}

}

];

Gateway::sendToClient($user\_client\_id['0'], json\_encode($on

//查询当前的用户是在哪个分组中,将当前的链接加入该分组

$ret = $db1->query("select `group\_id` from `v3\_groupdetail` where `user

\_id` = {$uid} group by `group\_id`");

if (!empty($ret)) {

foreach ($ret as $key => $vo) {

Gateway::joinGroup($client\_id, $vo['group\_id']); //将登录用户加

入群组

}

}

unset($ret);

//设置用户为登录状态

$db1->query("update `v3\_chatuser` set status=1 where id=" . $uid);

return;

}

}

/\*\*

\* 当用户断开连接时触发

\* @param int $client\_id 连接id

\*/

public static function onClose($client\_id)

{

//维持连接的worker退出才触发

if(!empty($\_SESSION['id'])){

$db1 = Db::instance('db1'); //数据库链接

//通知该用户的好友，该用户下线

$friends = $db1->query("select `user\_id` from `v3\_friends` where `friend\_id

` = " . $\_SESSION['id']);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [ 'message\_type' => 'logout', 'id' => $\_SESSION['id'],

\_message));

}

];

Gateway::sendToClient($user\_client\_id['0'], json\_encode($online

}

}

['id']);

}

//设置用户为退出状态

$db1->query("update `v3\_chatuser` set `status` = 0 where id = " . $\_SESSION

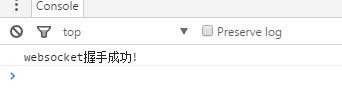
}

}

#### 我们只要关心，onMessage这个方法即可。用户登录之后，发送了 一个 init 类型的消息，链接成功

workerman之后，会触发 onMessage方法，我们用switch分之结构来区分这些消息类型，做出相应的处理。

**建立链接**

保持workerman的启动窗口不关闭，登录聊天系统。打开F12,在控制台能看到

那么接下来，我们所有的功能都能基于搭建好的系统愉快的写业务逻辑了。

**关于用户退出**

相信看了登录这部分代码，可以看到，我没有给登录用户设置身份过期的有效期。这就是为了模拟，传统的

qq那样，只有当用户退出之后，才能触发这个操作。在web端，用户怎么退出呢？那就是关闭浏览器。关闭 浏览器之后，客户端就会失去与workerman的链接，此时，就会触发 onClose 事件。此时，我们就可以根据 实际的业务需求，设置退出状态，以及发送好友下线通知。其他在线的好友，接到好友下线的通知后，做出 相应的动作，main.js中的 onmessage：

// 下线

case 'offline': layim.setFriendStatus(data.id, 'offline'); break;

**其他注意事项**

在写作本教程的时候，下载的worker又有更新了，其中的mysql方法已经做了修改，不赞成我之前的那种写法

了，但是我发现，貌似还能用，不过你可以改成官方的用法：

<?php

use \GatewayWorker\Lib\Gateway;

/\*\*

\* 数据库示例，假设有个your\_db\_name库，里面有个user表

\*/

class Events

{

/\*\*

\* 新建一个类的静态成员，用来保存数据库实例

\*/

public static $db = null;

/\*\*

\* 进程启动后初始化数据库连接

\*/

public static function onWorkerStart($worker)

{

self::$db = new Workerman\MySQL\Connection('host', 'port', 'user', 'password', 'db\_name');

}

/\*\*

\* 有消息时触发该方法，根据发来的命令打印2个用户信息

* @param int $client\_id 发消息的client\_id
* @param mixed $message 消息
* @return void

\*/

public static function onMessage($client\_id, $message)

{

// 发来的消息

$commend = trim($message); if($commend !== 'get\_user\_list')

{

Gateway::sendToClient($client\_id, "unknown commend\n"); return;

}

ry();

}

}

// 使用数据库实例

self::$db->select('\*')->from('users')->where('uid>3')->offset(5)->limit(2)->que

// 打印结果

return Gateway::sendToClient($client\_id, var\_export($ret, true));

# 修改个人资料的实现

#### 修个个人资料这一功能，并没与牵扯到与服务器通信的问题，你可以从main.js中看到这部分功能的端倪。

**js功能部分的设计**

我们在layim的ready事件中，触发这个方法

//layim建立就绪

layim.on('ready', function(res){

//点击头像操作

$(".layui-layim-user").click(function(){ layui.use(['layer'], function(){

var layer = layui.layer; layer.open({

type: 2,

title: '修改个人资料',

maxmin: false, //开启最大化最小化按钮 area: ['800px', '600px'], content: chat\_user\_url

});

});

});

});

我们通过 F12 可以看到，用户名那部分的 div的class=“layui-layim-user”，因此我们给这几个div绑定一个点击

事件。用layer打开一个弹出层，来修改用户的个人资料。用的接口地址是 chat\_user\_url，在 index/index.html中可见如下配置：

var chat\_user\_url = "{:url('chatuser/index')}";

**控制器的设计**

接口已经订好了，那么我们在 application\index\controller 新建一个 Chatuser.php 来处理这部分逻辑即可：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Chatuser extends Base

{

//展示当前用户的基本资料 public function index()

{

$user = db('chatuser')->where('id', session('f\_user\_id'))->find();

$this->assign([ 'user' => $user

]);

return $this->fetch();

}

//修改个人资料

public function doChange()

{

if (request()->isAjax()) {

$param = input('post.');

//所有验证应该在后台重新做，此处暂时不做 if (!empty($param['oldpwd'])) {

>find();

$pwd = db('chatuser')->field('pwd')->where('id', session('f\_user\_id'))-

if (md5($param['oldpwd']) != $pwd['pwd']) {

return json(['code' => -1, 'data' => '', 'msg' => '旧密码不正确！']);

}

！']);

if($param['pwd'] != $param['repwd']){

return json(['code' => -2, 'data' => '', 'msg' => '两次密码输入不一致

}

$upData['pwd'] = md5($param['pwd']);

}

//查询获得区域描述

$where = 'id =' . $param['pid']; if(!empty($param['cid'])){

$where .= ' or id=' . $param['cid'];

}else{

$param['city'] = 0;

}

if(!empty($param['aid'])){

$where .= ' or id=' . $param['aid'];

}else{

$param['area'] = 0;

select();

}

$area = db('area')->field('area\_name')->where($where)->order('level asc')->

$areaStr = ''; if(!empty($area)){

foreach($area as $key=>$vo){

$areaStr .= $vo['area\_name'] . '-';

}

$areaStr = rtrim($areaStr, '-');

}else{

$areaStr = '北京-北京市-东城区';

}

unset($area);

$upData['user\_name'] = trim($param['user\_name']);

!empty($param['avatar']) && $upData['avatar'] = $param['avatar'];

$upData['sex'] = $param['sex'];

$upData['age'] = $param['age'];

$upData['pid'] = $param['pid'];

$upData['cid'] = $param['cid'];

$upData['aid'] = $param['aid'];

$upData['area'] = $areaStr; unset($param);

$flag = db('chatuser')->where('id', session('f\_user\_id'))->update($upData); if(false === $flag){

return json(['code' => -3, 'data' => '', 'msg' => '系统错误']);

}

//重置头像

!empty($upData['avatar']) && session('f\_user\_avatar', $upData['avatar']);

return json(['code' => 1, 'data' => '', 'msg' => '修改成功']);

}

$this->error('非法访问');

}

//上传个人头像

public function upAvatar()

{

// 获取表单上传文件

$file = request()->file('avatar');

// 移动到框架应用根目录/public/uploads/ 目录下

if(!is\_null($file)){

$fileInfo = $file->getInfo();

$imgExt = config('common\_img');

$ext = explode('.', $fileInfo['name']);

$ext = array\_pop($ext);

if(!in\_array($ext, $imgExt)){

return json(['code' => -2, 'data' => '', 'msg' => '请上传' . implode(','

, $imgExt) . '的图片']);

}

unset($ext);

$size = config('common\_size'); if($fileInfo['size'] > $size){

return json(['code' => -3, 'data' => '', 'msg' => '上传的图片超过' . $siz

e/1024 . 'kb']);

}

unset($fileInfo);

$info = $file->move(ROOT\_PATH . 'public' . DS . 'uploads'); if($info){

// 成功上传后 获取上传信息

$avatar = '/uploads' . '/' . date('Ymd') . '/' . $info->getFilename(); return json(['code' => 1, 'url' => $avatar, 'msg' => 'success']);

}else{

// 上传失败获取错误信息

return json(['code' => -4, 'url' => '', 'msg' => $file->getError()]);

}

}

return json(['code' => -1, 'data' => '', 'msg' => '修改头像失败']);

}

}

**视图渲染部分**

#### 我们在 application\index\view 先新建一个 chatuser文件夹 并在其下 新建 index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<title>laychat-v3.0 - 修改个人资料</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

</head>

<body>

<div class="main layui-clear" style="padding: 15px">

<div class="layui-form layui-form-pane">

<div id="user\_info">

<div class="layui-form-item">

<label for="user\_name" class="layui-form-label">用户名</label>

<div class="layui-input-inline">

<input type="text" id="user\_name" name="user\_name" autocomplete="of

f"

</div>

class="layui-input" value="{$user.user\_name}">

="110px">

="file">

<input type="hidden" id="user\_avatar" name="user\_avatar"/>

<div class="site-demo-upload" style="float: right">

<img id="LAY\_demo\_upload" src="{$user.avatar}" width="110px" height

<div class="site-demo-upbar">

<input name="avatar" class="layui-upload-file" id="avatar" type

</div>

</div>

</div>

<blockquote class="layui-elem-quote">

修改密码

</blockquote>

<div class="layui-form-item">

<label for="pwd" class="layui-form-label">旧密码</label>

<div class="layui-input-inline">

<input type="password" id="oldpwd" name="oldpwd" autocomplete="off" class="layui-input">

</div>

</div>

<div class="layui-form-item">

<label for="pwd" class="layui-form-label">新密码</label>

<div class="layui-input-inline">

<input type="password" id="pwd" name="pwd" autocomplete="off" class

="layui-input">

</div>

<div class="layui-form-mid layui-word-aux">6到16个字符</div>

</div>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">重复新密码</label>

<div class="layui-input-inline">

<input type="password" id="repwd" name="repwd" autocomplete="off" c

lass="layui-input">

</div>

<div class="layui-form-mid layui-word-aux">6到16个字符</div>

</div>

<div class="layui-form-item fly-form-app">

注：密码改后下次登录用

</div>

<blockquote class="layui-elem-quote">

个人信息

</blockquote>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">性别</label>

<div class="layui-input-block">

<input name="sex" value="1" title="男" {if condition="$user['sex'] eq 1"}checked=""{/if} type="radio">

<input name="sex" value="-1" title="女" {if condition="$user['sex'] eq -1"}checked=""{/if} type="radio">

</div>

</div>

<div class="layui-form-item">

<label for="repwd" class="layui-form-label">年龄</label>

<div class="layui-input-inline">

<input name="age" lay-verify="number" autocomplete="off" class="lay ui-input" type="number"

value="{$user.age}" id="age">

</div>

<div class="layui-form-mid layui-word-aux">只能是数字</div>

</div>

<div class="layui-form-item">

<label class="layui-form-label">居住地区</label>

<div class="layui-input-inline">

<select name="province" lay-filter="province">

<option value="">请选择省</option>

</select>

</div>

<div class="layui-input-inline">

<select name="city" lay-filter="city">

<option value="">请选择市</option>

</select>

</div>

<div class="layui-input-inline">

<select name="area" lay-filter="area">

<option value="">请选择县/区</option>

</select>

</div>

</div>

<div class="layui-form-item">

<button class="layui-btn" id="btn">确认提交</button>

</div>

</div>

</div>

</div>

<script src=" COMMON /layui/layui.js"></script>

<script src=" COMMON /jquery.min.js"></script>

<script src=" COMMON /layui/lay/modules/cityselect.js"></script>

<script type="text/javascript"> var p\_code = '{$user.pid}'; var c\_code = '{$user.cid}'; var a\_code = '{$user.aid}';

var do\_change\_url = "{:url('chatuser/doChange')}"; var up\_avatar\_url = "{:url('chatuser/upavatar')}";

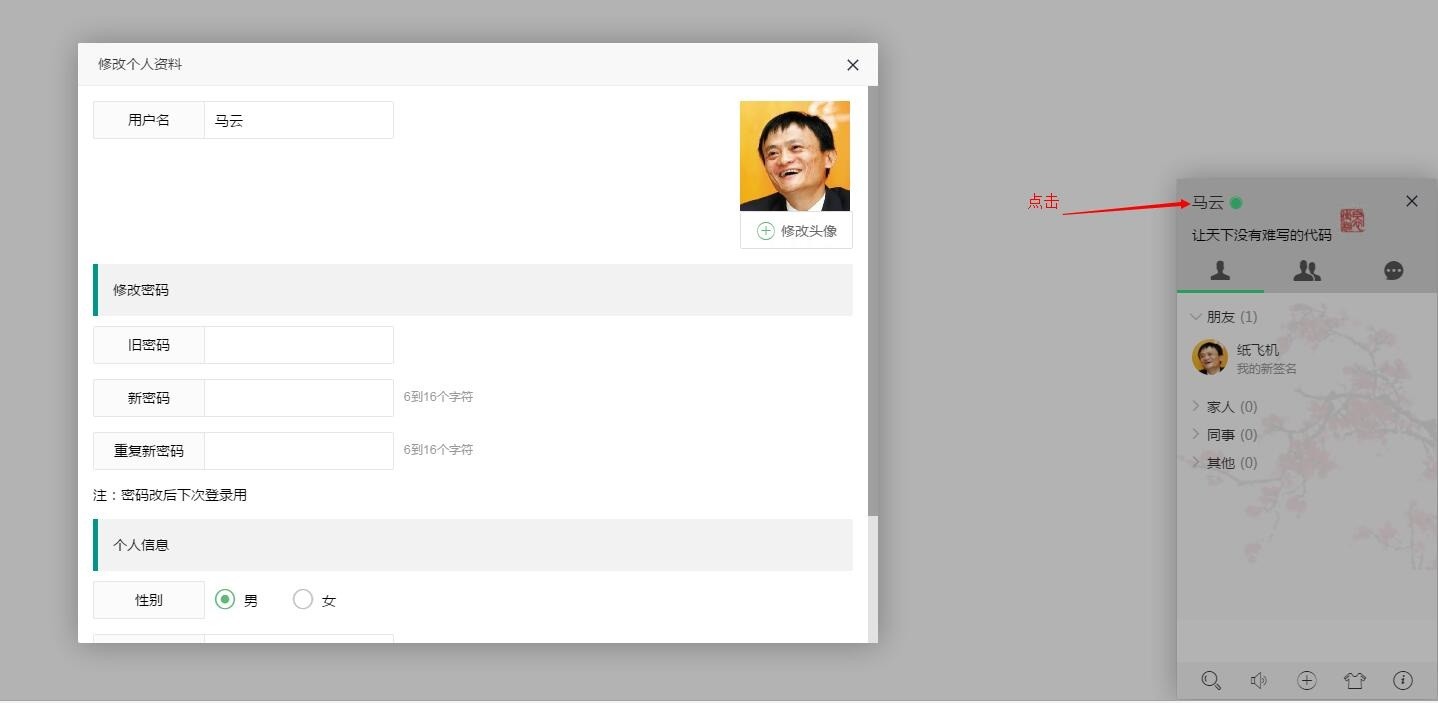
</script>

<script type="text/javascript" src=" COMMON /js/user.js"></script>

</body>

</html>

#### 此时，我们点击用户面包上的，用户名：



即可出现用户资料的编辑页面了。那就可以在这里面实现 用户资料的编辑以及头像的更改。

# 修改个性签名的实现

#### 如何让你的webIM系统像qq那样，可以修改自己的个性签名呢？不用怕，layIM拥有这部分功能的接口：



鼠标点击个性签名的位置，就会出现一个编辑状态。那么，我们怎么实现他的确切功能呢？

**js部分的设计**

layIM提供了一个监听修改个性签名的事件，请看 main.js 中：

//监听签名修改

layim.on('sign', function(value){

$.post(change\_sign\_url, {'sign' : value}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:1500});

}else{

layer.msg(res.msg, {time:1500});

}

}, 'json');

});

用户修改了个性签名之后，会执行一个会掉，地址可以在 index.html中得知：

var change\_sign\_url = "{:url('chatuser/changeSign')}";

**控制器的设计**

从接口地址可以看出，我依旧是在 chatuser这个控制器中：

//修改个性签名

public function changeSign()

{

if(request()->isAjax()){

$sign = input('post.sign');

$flag = db('chatuser')->where('id', session('f\_user\_id'))->setField('sign',

$sign);

if(false === $flag){

return json(['code' => -1, 'data' => '', 'msg' => '系统错误']);

}

//重置签名 session('f\_user\_sign', $sign);

return json(['code' => 1, 'data' => '', 'msg' => '修改成功']);

}

$this->error('非法访问');

}

异步的实现签名的修改，然后通知用户修改成功或者失败即可。

本系统 修改个人资料 和 个性签名 这两部分功能，都不是实时的，也就是说，修改了之后，你的好友看到 数据不会更新。这部分留给以后解决，或者你有兴趣的话可以自己去解决。思路很简单，就是修改的时 候，将修改后的消息重新注入 session以及通知好友即可。

# 切换在线状态的实现

#### 在使用qq的时候，我们都能看到一个功能，我们可以随时的切换自己的在线状态，例如：我切换成隐身，这 样别人就不知道我是在线的。那么，我们如何在webIM中实现这个功能呢？

**js部分设计**

我们能在用户面板中看到用户名旁边有一个绿色的按钮，鼠标点击这个按钮：

可以看到状态切换的面板，那么我们如何监听这个事件呢？没错，layIM同样的做到了对这个事件的监听， main.js 可见：

//在线状态切换

layim.on('online', function(status){

var change\_data = '{"type":"online", "status":"' + status + '", "uid":"' + uid + '"

}';

socket.send(change\_data);

});

当用户点击 在线和隐身的时候，会触发这个时间，并且得到不同的status，我们将这个 status 发送给服务

器，让其通知当前用户的好友，去更新 当前用户的状态。这么说可能很拗口，以现在的用户 马云 来举例吧， 马云切换成隐身状态，系统将通知马云的好友，将马云的头像变暗，表示马云不在线。 main.js中的 socket.onmessage 方法 还可见这段代码：

// 在线

case 'online': layim.setFriendStatus(data.id, 'online'); break;

// 下线

case 'offline': layim.setFriendStatus(data.id, 'offline'); break;

大家都知道，这部分代码是客户端接收到服务器消息之后，做出的动作。那么workerman端，我们该怎么去

写呢？

**服务端的实现**

我们可以在 Events.php 的 onMessage 方法，添加如下的代码：

case 'online':

$db1 = Db::instance('db1'); //数据库链接

// 在线切换状态

$friends = $db1->query("select `user\_id` from `v3\_friends` where `friend\_id` = "

. $message['uid']);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [

'message\_type' => ('online' == $message['status']) ? 'online'

: 'offline',

'id' => $message['uid'],

];

Gateway::sendToClient($user\_client\_id['0'], json\_encode($online\_me

ssage));

}

}

}

return; break;

#### workerman接收到 在线信息切换的状态的消息之后，查询这用户的好友，并逐一通知这个用户的好友，改变

这个用户的状态。即触发了 socket.onmessage 中的 online 和 offline 分之。

**如何验证**

我们该怎么验证这个功能是否正确呢？因为我们的用户登录状态是通过 session 标识的，因此，我们要用两个

浏览器去模拟这个过程。比如 马云 和 纸飞机 是好友关系，我们分别在 chrome 中登录马云，在 firefox 中登 录纸飞机。

注意：这一切的前提是，你的workerman链接正确。

马云切换状态，从纸飞机 那边看，马云的头像亮和暗是否改变了。

# 添加好友的实现

#### 用户自己的功能都能实现了，那么我们该怎么添加好友，然后和好友聊天呢？本章先介绍，如何实现添加好 友的功能

**js部分的设计**

我们在用户主面板的下面，能看到一个 + 号

这里就是加好友和加群组的的地方，我们这里先介绍如何通过这里查找好友，点击这个加号，同样的我们有 对应的js去监听这个事件，在main.js中可见：

//基础配置 layim.config({

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

//查看群员接口

,members: {

url: member\_list\_url

,data: {}

}

,uploadImage: {

url: upload\_img\_url

}

,uploadFile: {

url: upload\_file\_url

}

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

}

]

,initSkin: '3.jpg' //1-5 设置初始背景

,msgbox: msg\_box\_url//消息盒子页面地址，若不开启，剔除该项即可

,find: find\_url //发现页面地址，若不开启，剔除该项即可

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

});

其中，我们可以看到 find 这个配置项 它的地址是 find\_url，那么我们看看 index.html中定义的 find\_url地址是

什么？

var find\_url = "{:url('findgroup/index')}";

**控制器方面的设计**

我们在 application\index\controller 中新建一个 Findgroup.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Findgroup extends Base

{

//显示查询 / 添加 分组的页面

public function index()

{

//查询最新4名非好友，建议用户添加

$uid = session('f\_user\_id');

$friends = db('friends')->field('friend\_id')->where('user\_id', $uid)->select();

$where = '1=1'; if(!empty($friends)){

foreach($friends as $vo){

$fArr[] = $vo['friend\_id'];

}

unset($friends);

$where .= ' and id not in(' . implode(',', $fArr) . ')';

}

here)

$userList = db('chatuser')->field('id,user\_name,area,avatar,sex,age')->where($w

->order('id desc')->limit(4)->select();

//初始化省份

$province = db('area')->field('id,area\_name')->where('parent\_id', 0)->select();

//推荐新建的群组，并且是审核通过的，供用户添加

$groupArr = db('chatgroup')->where('status = 1')->order('id desc')->limit(4)->s

elect();

$this->assign([

'group' => $groupArr, 'age' => config('age'), 'province' => $province, 'user' => $userList

]);

return $this->fetch();

}

}

#### 系统会推荐4名用户供用户添加，由于这个页面是添加群组和添加好友合一的页面，因此，我在这里，也查询

了推荐群组数据。此时，我们可以不必关注这方面的数据。

**视图部分**

我们在 application\index\view 下新建 findgroup 文件夹，然后在其下面新建 index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>查找/添加群组</title>

<link rel="shortcut icon" href="favicon.ico">

<link href=" COMMON /css/bootstrap.min.css?v=3.3.6" rel="stylesheet">

<link href=" COMMON /css/font-awesome.min.css?v=4.4.0" rel="stylesheet">

<link href=" COMMON /css/animate.min.css" rel="stylesheet">

<link href=" COMMON /css/style.min.css?v=4.1.0" rel="stylesheet">

<link href=" COMMON /layui/css/layui.css" rel="stylesheet" media="all">

</head>

<body class="gray-bg">

<div class="wrapper wrapper-content animated fadeInRight">

<div class="col-sm-12">

<div class="tabs-container">

<ul class="nav nav-tabs">

<li class="active"><a data-toggle="tab" href="#tab-1" aria-expanded="tr

ue"> 找人</a>

找群</a>

</li>

<li class=""><a data-toggle="tab" href="#tab-2" aria-expanded="false">

</li>

hod="post">

性别-->

年龄--> 省份id--> 城市id--> 区id-->

</ul>

<div class="tab-content">

<div id="tab-1" class="tab-pane active">

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-content">

<form class="form-horizontal m-t" id="userForm" met

<input type="hidden" name="sex" id="sex"/><!--

<input type="hidden" name="age" id="age"/><!--

<input type="hidden" name="pid" id="pid"/><!--

<input type="hidden" name="cid" id="cid"/><!--

<input type="hidden" name="aid" id="aid"/><!--

<div class="form-group">

<div class="col-sm-4">

<div class="input-group col-sm-12">

<input type="text" class="form-cont

rol" name="user\_name" placeholder="请输入昵称" id="user\_name" >

</div>

</div>

<div class="col-sm-2">

<button data-toggle="dropdown" class="b tn btn-white dropdown-toggle col-sm-12" type="button" aria-expanded="false">

<span id="csex">选择性别</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="javascript:chsex(0);">

不限</a>

</li>

男</a>

>女</a>

<li>

<a href="javascript:chsex(1);">

</li>

<li>

<a href="javascript:chsex(-1);"

</li>

</ul>

</div>

<div class="col-sm-2">

<button data-toggle="dropdown" class="b

tn btn-white dropdown-toggle col-sm-12" type="button" aria-expanded="false">

<span id="cage">选择年龄</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="javascript:chage('0-0'

);">不限</a>

key"}

y}');">{$vo}</a>

-12" type="button" id="charea">

imary" id="search">搜索</button>

</li>

{if !empty($age)}

{foreach name="age" item="vo" key="

<li>

<a href="javascript:chage('{$ke

</li>

{/foreach}

{/if}

</ul>

</div>

<div class="col-sm-3">

<div class="input-group-btn">

<button class="btn btn-white col-sm

<span id="ssq">选择地区</span>

<span class="caret"></span>

</button>

</div>

</div>

<span class="input-group-btn">

<button type="button" class="btn btn-pr

</span>

</div>

</form>

</div>

</div>

</div>

</div>

<!-- 推荐好友 -->

<div class="row">

<div class="col-sm-3"><label id="s\_u\_title">好友推荐</label></di

v>

</div>

<div class="row" id="s\_u\_data">

{if !empty($user)}

{foreach name="user" item="vo"}

<div class="col-sm-3">

<div class="ibox float-e-margins">

<div class="ibox-title">

<h5>{$vo.user\_name}</h5>

<span style="margin-left: 10px"><i class="layui-ico n" {if condition="$vo['sex'] eq 1"}style="color:#7CA3D2 "{else/}style="color:#FDA357"{/ if}>&#xe612;</i></span>

px"/>

<span style="margin-left: 10px">{$vo.age}岁</span>

</div>

<div class="ibox-content">

<div style="margin: 0 auto">

<img src="{$vo.avatar}" width="50px" height="50

<span style="font-size: 10px;width:104px;overfl

ow: hidden;display: inline-block">{$vo.area}</span>

</div>

<div style="margin:10px 50px"><button class="btn bt n-primary" type="button" data-uid="{$vo.id}" onclick="addFriend(this)">加好友</button></ div>

</div>

</div>

</div>

{/foreach}

{/if}

</div>

</div>

<div id="tab-2" class="tab-pane">

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-content">

<form class="form-horizontal m-t" id="commentForm"

method="post" action="">

<div class="form-group">

<div class="col-sm-10">

<div class="input-group">

<input type="text" class="form-cont

rol" name="search\_txt" placeholder="输入群组名称" id="search\_txt">

<span class="input-group-btn">

<button type="button" class="bt

n btn-primary" id="find">搜索</button>

</span>

&nbsp;&nbsp;&nbsp;

<span class="input-group-btn">

<button type="button" class="bt

n btn-w-m btn-warning" id="addGroup">创建群组</button>

</span>

&nbsp;&nbsp;&nbsp;

n btn-w-m btn-primary" id="myGroup">我的群组</button>

<span class="input-group-btn">

<button type="button" class="bt

</span>

abel></div>

</div>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

<!-- 最近新建的分组 -->

<div class="row">

<div class="col-sm-3"><label id="search\_title">最近新添加的群组</l

</div>

<div class="row" id="search\_data">

{if !empty($group)}

{foreach name="group" item="vo"}

<div class="col-sm-3">

<div class="ibox float-e-margins">

<div class="ibox-title">

<h5>{$vo.group\_name}</h5>

</div>

<div class="ibox-content">

<div style="margin: 0 auto">

<img src="{$vo.avatar}" width="50px" height="50

px" style="margin-left:50px"/>

</div>

<div style="margin:10px 50px"><button class="btn bt

n-primary" type="button" data-id="{$vo.id}" data-owner="{$vo.owner\_id}" onclick="joinGr oup(this)">加入</button></div>

</div>

</div>

</div>

{/foreach}

{/if}

</div>

</div>

</div>

</div>

</div>

</div>

<!-- 选择省市区 -->

<div class="wrapper wrapper-content animated" id="show" style="display: none">

<div class="row">

<div class="col-sm-12">

<select class="form-control m-b" id="p">

<option value="0" data-id="0">不限区域</option>

{foreach name="province" item="vo"}

<option value="{$vo.area\_name}" data-id="{$vo.id}">{$vo.area\_name}</opt

ion>

{/foreach}

</select>

</div>

</div>

<div class="row">

<div class="col-sm-12">

<select class="form-control m-b" id="c" disabled>

<option value="0" data-id="0">请选择城市</option>

</select>

</div>

</div>

<div class="row">

<div class="col-sm-12">

<select class="form-control m-b" id="a" disabled>

<option value="0" data-id="0">请选择区</option>

</select>

</div>

</div>

<div class="col-sm-4 col-sm-offset-4" style="margin-bottom: 10px">

<button class="btn btn-primary" type="submit" id="makesure">确认</button>

</div>

</div>

<input type="hidden" id="pname"/>

<input type="hidden" id="cname"/>

<input type="hidden" id="aname"/>

<script src=" COMMON /jquery.min.js"></script>

<script src=" COMMON /bootstrap.min.js?v=3.3.6"></script>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var area\_url = "{:url('findgroup/getArea')}"; var search\_user\_url = "{:url('finduser/index')}";

var search\_group\_url = "{:url('findgroup/search')}"; var add\_group\_url = "{:url('findgroup/addgroup')}"; var my\_group\_url = "{:url('findgroup/myGroup')}"; var join\_group\_url = "{:url('findgroup/joinGroup')}";

var check\_group\_url = "{:url('findgroup/checkGroupAuth')}"; var apply\_friend\_url = "{:url('msgbox/applyFriend')}";

var apply\_group\_url = "{:url('findgroup/applyGroup')}";

</script>

<script src=" COMMON /js/find.js"></script>

</body>

</html>

#### 由于是 加还有和加群合一的页面，因此，代码比较长。再次 点击 + 号

可见这样的添加页面，由于之前我们在 用户注册时候，就已经要求用户填写了一些个人信息，因此这里面的 筛选很容易实现。

我们看到的一堆的接口地址，具体的实现方法在 find.js 中，你可以去查看 find.js的源码，这里就不重复放 出源码了。

**完善控制器中的搜索和添加功能**

用户点击搜索，会向

var search\_user\_url = "{:url('finduser/index')}";

这个接口请求数据，因此，我们在控制器中做了条件的筛选。当用户点击 "加好友" 按钮时：

用户填写完对话框中的信息之后，选择发送申请。即触发了：

var apply\_friend\_url = "{:url('msgbox/applyFriend')}";

申请好友接口。而这些信息，会通过 "消息盒子" 的形式通知被通知人。我们在 application\index\controller

中新建一个 Msgbox.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Msgbox extends Base

{

//申请好友

public function applyFriend()

{

if(request()->isAjax()){

$param = input('post.');

//检测是否被要添加的用户加入了黑名单，若加入，则无法申请添加该好友

$inBlack = db('blacktab')->where('put\_uid=' . session('f\_user\_id') . ' and user\_id=' . $param['uid'])

->find(); if(!empty($inBlack)){

return json(['code' => -2, 'data' => '', 'msg' => '对方已将你加入黑名单'])

;

}

//入库系统消息

$msg = [

'content' => '申请添加你为好友', 'uid' => $param['uid'],

'from' => session('f\_user\_id'), //发起好友申请的uid 'remark' => $param['remark'],

'from\_group' => $param['from\_group'], 'type' => 1,

'read' => 1,

'time' => time()

];

$flag = db('message')->insert($msg); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '系统错误']);

}

return json(['code' => 0, 'data' => '', 'msg' => 'success']);

}

$this->error('非法访问');

}

}

#### 具体的业务逻辑，我注释的很清晰了，你可以阅读一下。到此添加好友的功能算是告一段落，接下来，系统

会通知被添加人对消息做出回应。

**补充：**

添加你好友，用到了 三级联动，这里的接口如下：

//省市区,三级联动

public function getArea()

{

if(request()->isAjax()){

$code = input('post.code');

$result = db('area')->field('id,area\_name')->where('parent\_id = ' . $code)-

>select();

return json(['code' => 1, 'data' => $result, 'msg' => 'success']);

}

$this->error('非法访问');

}

# 消息盒子

#### 在添加好友的时候，提到了 "消息盒子" ，消息盒子的作用就是通知用户一些好友添加消息，以及用户是同意 还是拒接的一些通知，方便用户查看。

**js部分的设计**

//基础配置 layim.config({

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

//查看群员接口

,members: {

url: member\_list\_url

,data: {}

}

,uploadImage: {

url: upload\_img\_url

}

,uploadFile: {

url: upload\_file\_url

}

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

}

]

,initSkin: '3.jpg' //1-5 设置初始背景

,msgbox: msg\_box\_url//消息盒子页面地址，若不开启，剔除该项即可

,find: find\_url //发现页面地址，若不开启，剔除该项即可

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

});

可见 msgbox 这个配置项，我填写的地址是 msg\_box\_url，在index.html可以看到：

var msg\_box\_url = "{:url('Msgbox/index')}";

由此可见，消息盒子打开的页面，是Msgbox的index方法来展示的。我们在main.js中还可以看到这样的代

码：

//获取未读的消息 120s读取一次 setInterval(function(){

$.getJSON(get\_noread\_url, function(res){

if(res.data > 0){ layim.msgbox(res.data);

}

});

}, parseInt(120) \* 1000);

这个功能的用意，是获取用户未读的消息。



消息盒子我没有设计成直接推送的模式，因为个人觉得，用户添加好友之后，直接通过服务器推送给被添 加人，可能会出现消息条数"计数"不准的问题。

我们可以在面板的最下方，看到消息盒子的展示未读消息的条数。我这里是通过轮询的方式去后去用户未读

的消息，因此，在获取用户添加消息的时候，会有一定的延迟。get\_noread\_url 的地址是：

var get\_noread\_url = "{:url('msgbox/getNoRead')}";

**控制器的设计**

Msgbox.php中，我们添加两个方法：

//信息框展示页面

public function index()

{

$this->assign([

'uid' => session('f\_user\_id'), 'username' => session('f\_user\_name'), 'avatar' => session('f\_user\_avatar'), 'sign' => session('f\_user\_sign')

]);

return $this->fetch();

}

//获取当前用户有多少个未读通知 public function getNoRead()

{

if(request()->isAjax()){

$tips = db('message')->where('`uid`=' . session('f\_user\_id') . ' and `read`

=1')->count();

return json(['code' => 1, 'data' => $tips, 'msg' => 'success']);

}

$this->error('非法访问');

}

#### getNoRead用来提醒用户，还有多少条消息没有读取。index方法则用来展示消息盒子的页面

**视图部分**

我们在 application\index\view 新建一个 msgbox 文件夹，然后在其下 新建一个 index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<title>消息盒子</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

<style>

.layim-msgbox{margin: 15px;}

.layim-msgbox li{position: relative; margin-bottom: 10px; padding: 0 130px 10px

60px; padding-bottom: 10px; line-height: 22px; border-bottom: 1px dotted #e2e2e2;}

.layim-msgbox .layim-msgbox-tips{margin: 0; padding: 10px 0; border: none; text

-align: center; color: #999;}

.layim-msgbox .layim-msgbox-system{padding: 0 10px 10px 10px;}

.layim-msgbox li p span{padding-left: 5px; color: #999;}

.layim-msgbox li p em{font-style: normal; color: #FF5722;}

50px;}

.layim-msgbox-avatar{position: absolute; left: 0; top: 0; width: 50px; height:

.layim-msgbox-user{padding-top: 5px;}

.layim-msgbox-content{margin-top: 3px;}

.layim-msgbox .layui-btn-small{padding: 0 15px; margin-left: 5px;}

.layim-msgbox-btn{position: absolute; right: 0; top: 12px; color: #999;}

</style>

</head>

<body>

<ul class="layim-msgbox" id="LAY\_view"></ul>

<textarea title="消息模版" id="LAY\_tpl" style="display:none;">

{{# layui.each(d.data, function(index, item){ if(item.from){ }}

<li data-uid="{{ item.from }}" data-fromGroup="{{ item.from\_group }}" data-id="{{ i tem.id }}">

<a href="javascript:;">

<img src="{{ item.user.avatar }}" class="layui-circle layim-msgbox-avatar">

</a>

<p class="layim-msgbox-user">

<a href="javascript:;">{{ item.user.username||'' }}</a>

<span>{{ item.time }}</span>

</p>

<p class="layim-msgbox-content">

{{ item.content }}

<span>{{ item.remark ? '附言: '+item.remark : '' }}</span>

</p>

{{# if(item.agree == 0){ }}

<p class="layim-msgbox-btn">

<button class="layui-btn layui-btn-small" data-type="agree">同意</button>

<button class="layui-btn layui-btn-small layui-btn-primary" data-type="refu se">拒绝</button>

</p>

{{# } else if(item.agree == 1){ }}

<p class="layim-msgbox-btn">

已同意

</p>

{{# } else { }}

<p class="layim-msgbox-btn">

<em>已拒绝</em>

</p>

{{# } }}

</li>

{{# } else { }}

<li class="layim-msgbox-system">

<p><em>系统：</em>{{ item.content }}<span>{{ item.time }}</span></p>

</li>

{{# }

}); }}

</textarea>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var socket = new WebSocket('ws://127.0.0.1:8282');

layui.use(['layim', 'flow'], function(){ var layim = layui.layim

,layer = layui.layer

,laytpl = layui.laytpl

,$ = layui.jquery

,flow = layui.flow;

var cache = {}; //用于临时记录请求到的数据

//请求消息

var renderMsg = function(page, callback){

//获取消息盒子

$.post('getmsg', { page: page || 1

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

//记录来源用户信息

layui.each(res.data, function(index, item){ cache[item.from] = item.user;

});

callback && callback(res.data, res.pages);

});

};

//消息信息流 flow.load({

elem: '#LAY\_view' //流加载容器

,isAuto: false

,end: '<li class="layim-msgbox-tips">暂无更多新消息</li>'

,done: function(page, next){ //加载下一页 renderMsg(page, function(data, pages){

var html = laytpl(LAY\_tpl.value).render({ data: data

,page: page

}

});

});

});

next(html, page < pages);

//打开页面即把消息标记为已读

$.post('read', {read: 2}, function(){});

//操作

var active = {

//同意

agree: function(othis){

var li = othis.parents('li')

,uid = li.data('uid')

,from\_group = li.data('fromgroup')

,user = cache[uid]

,id = li.data('id');

//选择分组 parent.layui.layim.setFriendGroup({

type: 'friend'

,username: user.username

,avatar: user.avatar

,group: parent.layui.layim.cache().friend //获取好友分组数据

,submit: function(group, index){

//同意接口

$.post('agreeFriend', { uid: uid //对方用户ID

, from\_group: from\_group //对方设定的好友分组

, group: group //我设定的好友分组

, id : id

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

//将好友追加到主面板 parent.layui.layim.addList({

type: 'friend'

, avatar: user.avatar //好友头像

, username: user.username //好友昵称

, groupid: group //所在的分组id

, id: uid //好友ID

, sign: user.sign //好友签名

});

//通知对方将我加入好友列表

var data = '{"type":"addFriend", "toid":"' + uid + '", "id":"{$uid}", "username":"{$username}", "avatar":"{$avatar}", "sign":"{$sign}", "group id":"' + from\_group + '"}';

socket.send(data);

});

}

parent.layer.close(index); othis.parent().html('已同意');

});

}

//拒绝

,refuse: function(othis){

var li = othis.parents('li')

, uid = li.data('uid')

, id = li.data('id');

layer.confirm('确定拒绝吗？', function(index){

$.post('refuseFriend', { uid : uid, //对方用户ID

id : id

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

});

}

layer.close(index); othis.parent().html('<em>已拒绝</em>');

});

}

};

$('body').on('click', '.layui-btn', function(){ var othis = $(this), type = othis.data('type');

active[type] ? active[type].call(this, othis) : '';

});

});

</script>

</body>

</html>

#### 当用户点击消息盒子的时候，会展示消息盒子页面

那么这些数据是怎么来的呢？我们能看到js中有这么一个接口：

//请求消息

var renderMsg = function(page, callback){

//获取消息盒子

$.post('getmsg', {

page: page || 1

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

//记录来源用户信息

layui.each(res.data, function(index, item){ cache[item.from] = item.user;

});

callback && callback(res.data, res.pages);

});

};

#### 因此，我们在Msgbox.php中应该提供一个，获取消息的接口：

//获取通知消息

public function getMsg()

{

if(request()->isAjax()){

$msg = db('message')->where('uid', session('f\_user\_id'))->order('time desc'

)->paginate(5);

//拼装发送人信息 if(empty($msg)){

return json(['code' => 0, 'pages' => 0, 'data' => '', 'msg' => '']);

}

$pages = $msg->lastPage();

$msg = objToArr($msg); foreach($msg as $key=>$vo){

$msg[$key]['time'] = date('Y-m-d H:i'); if(1 == $vo['type']){

$user = db('chatuser')->field('avatar,user\_name,sign')->where('id',

$vo['from'])->find();

$msg[$key]['user'] = [

'id' => $vo['from'], 'avatar' => $user['avatar'],

'username' => $user['user\_name'], 'sign' => $user['sign']

];

}else{

$msg[$key]['user']['id'] = null;

}

}

return json(['code' => 0, 'pages' => $pages, 'data' => $msg, 'msg' => '']);

}

$this->error('非法访问');

}

#### 同时会触发一个标记消息阅读状态的方法

//打开页面即把消息标记为已读

$.post('read', {read: 2}, function(){});

我们在 Msgbox中添加一个read方法，来实现这个功能，标记所有未读的消息为已读，防止重复提醒。

//标记当前推送的消息为已读状态

public function read()

{

if(request()->isAjax()){

$read = input('post.read');

db('message')->where('uid=' . session('f\_user\_id'))->setField('read', $read

);

return true;

}

$this->error('非法访问');

}

# 用意/拒绝好友(1)

#### 我们在消息盒子的界面，已经能看到，当我们查看"消息盒子"中未读的消息时，比如查看好友添加申请时，能 看到有"同意" 和 "拒绝"按钮，我们该如何实现这两个功能呢？

**同意添加好友的实现**

**js部分代码**

我们在 view层的msgbox下的index.html中可见：

//操作

var active = {

//同意

agree: function(othis){

var li = othis.parents('li')

,uid = li.data('uid')

,from\_group = li.data('fromgroup')

,user = cache[uid]

,id = li.data('id');

//选择分组 parent.layui.layim.setFriendGroup({

type: 'friend'

,username: user.username

,avatar: user.avatar

,group: parent.layui.layim.cache().friend //获取好友分组数据

,submit: function(group, index){

//同意接口

$.post('agreeFriend', { uid: uid //对方用户ID

, from\_group: from\_group //对方设定的好友分组

, group: group //我设定的好友分组

, id : id

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

//将好友追加到主面板 parent.layui.layim.addList({

type: 'friend'

, avatar: user.avatar //好友头像

, username: user.username //好友昵称

, groupid: group //所在的分组id

, id: uid //好友ID

, sign: user.sign //好友签名

});

//通知对方将我加入好友列表

var data = '{"type":"addFriend", "toid":"' + uid + '", "id":"{$uid}", "username":"{$username}", "avatar":"{$avatar}", "sign":"{$sign}", "group id":"' + from\_group + '"}';

socket.send(data);

parent.layer.close(index); othis.parent().html('已同意');

});

}

});

}

//拒绝

,refuse: function(othis){

var li = othis.parents('li')

, uid = li.data('uid')

, id = li.data('id');

layer.confirm('确定拒绝吗？', function(index){

$.post('refuseFriend', {

uid : uid, //对方用户ID id : id

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

layer.close(index); othis.parent().html('<em>已拒绝</em>');

});

});

}

};

**控制器的设计**

#### 同意和拒绝走的是两个方法，本处我们讨论"同意"的操作。根据我的注释，你能看到 同意是请求的

agreeFriend 方法，也就是在Msgbox.php 中添加，同意的处理方法 agreeFriend：

//同意好友申请

public function agreeFriend()

{

if(request()->isAjax()){

$param = input('post.');

//建立好友关系

//1、将我与请求人建立关系

$myFriend = [

'user\_id' => session('f\_user\_id'), 'friend\_id' => $param['uid'], 'group\_id' => $param['group']

];

$flag = db('friends')->insert($myFriend); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '系统错误']);

}

unset($myFriend);

//2、将请求人与我建立关系

$yourFriend = [

'user\_id' => $param['uid'], 'friend\_id' => session('f\_user\_id'), 'group\_id' => $param['from\_group']

];

$flag = db('friends')->insert($yourFriend); if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '系统错误']);

}

unset($yourFriend);

//入库系统消息

$msg = [

'content' => session('f\_user\_name') . ' 已经同意你的好友申请', 'uid' => $param['uid'],

'from\_group' => $param['from\_group'], 'type' => 2,

'read' => 1, 'time' => time()

];

$flag = db('message')->insert($msg); if(empty($flag)){

return json(['code' => -3, 'data' => '', 'msg' => '系统错误']);

}

//将此消息标记为已经同意

$flag = db('message')->where('id', $param['id'])->setField('agree', 1); if(empty($flag)){

return json(['code' => -4, 'data' => '', 'msg' => '系统错误']);

}

return json(['code' => 0, 'data' => '', 'msg' => 'success']);

}

$this->error('非法访问');

}

#### 感谢我当初的明智，代码注释的相当清楚了。按照顺序我们来整理一下思路：

当我选择同意的时候

1. 将我和添加人的好友关系 写入 friends 表中

2. 将我同意该好友的申请，写入消息库 message用来提醒添加人，我对该申请的反馈。

3. 将该条申请消息，标记为已经同意状态。 此时，系统会通知 js 执行回调的部分操作：

if(res.code != 0){

return layer.msg(res.msg);

}

//将好友追加到主面板 parent.layui.layim.addList({

type: 'friend'

, avatar: user.avatar //好友头像

, username: user.username //好友昵称

, groupid: group //所在的分组id

, id: uid //好友ID

, sign: user.sign //好友签名

});

//通知对方将我加入好友列表

var data = '{"type":"addFriend", "toid":"' + uid + '", "id":"{$uid}", "username":"{$u

sername}", "avatar":"{$avatar}", "sign":"{$sign}", "groupid":"' + from\_group + '"}'; socket.send(data);

parent.layer.close(index); othis.parent().html('已同意');

因为该申请添加我的好友信息，已经全部传递过来了，同意的一方，只需要 直接调用 addList将该好友追加到

好友面板。

注意：此处的添加到好友面板，对方的面板中并没有"我"的存在，因此必须通过服务器向添加者通知这个 消息。

然后可以看到，我们向workerman发送了一个 addFriend 消息

**workerman的逻辑**

在 Events.php 的 onMessage 方法，添加接收处理这个消息的功能：

case 'addFriend':

$client\_id = Gateway::getClientIdByUid($message['toid']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [

'message\_type' => 'addFriend', 'data' => [

'username' => $message['username'], 'avatar' => $message['avatar'], 'id' => $message['id'],

'type' => 'friend',

'sign' => $message['sign'], 'groupid' => $message['groupid'],

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

直接通知添加方，准备添加这个好友到面板。

为什么代码是选择对方在线才通知呢？ 因为用户好友列表，在用户登录的时候，通过ajax从数据库拉取的，在对方同意我方好友申请的时候，已 经在firends表中将还有关系绑定了，因此用户只要登录，就能拉取到最新的好友列表。另外，对方不在 线，workerman也无法通知到该客户端，不判断的话，可能会出现推送错误。

**main.js中的layIM处理逻辑**

我们在 main.js 的 socket.onmessage 方法中可见：

// 添加好友

case 'addFriend':

//console.log(data.data); layim.addList(data.data); break;

这就是我在服务端 将所需的数据拼装完毕的好处，此时在客户端，只需要直接调用方法渲染即可，不必再次

的进行数据拼装，造成开发难度的增加。

# 同意/拒绝好友(2)

#### 拒绝添加好友的逻辑，比起同意好友申请要简单，因为我们不需要通知对方添加我到面板以及我也不需要将 对方添加到好友面板，只需要将拒绝的消息写入消息盒子(message表)中即可，这样，由于消息盒子的轮询机 制，这样就可以通知到申请方查看好友申请的最新动态。

**拒绝好友的实现**

**js部分的设计**

我们在 view层的 msgbox的index.html,可见拒绝部分的逻辑代码如下：

refuse: function(othis){

var li = othis.parents('li')

, uid = li.data('uid')

, id = li.data('id');

layer.confirm('确定拒绝吗？', function(index){

$.post('refuseFriend', {

uid : uid, //对方用户ID id : id

}, function(res){

if(res.code != 0){

return layer.msg(res.msg);

}

layer.close(index); othis.parent().html('<em>已拒绝</em>');

});

});

}

决绝是一个谨慎的操作，因此，采用的是layer的 confirm 来让用户确认是否要这样做。我们可见，用户拒绝

时，请求的是 refuseFriend方法。 回调执行的代码也很简单，就是将消息状态改为"已拒绝"。

**控制器的设计**

我们在 Msgbox.php 中添加 refuseFriend 方法：

//拒绝好友申请

public function refuseFriend()

{

if(request()->isAjax()){

$param = input('post.');

//将此消息标记为拒绝

$flag = db('message')->where('id', $param['id'])->setField('agree', 2); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '系统错误']);

}

//入库系统消息

$msg = [

'content' => session('f\_user\_name') . ' 拒绝了你的好友申请', 'uid' => $param['uid'],

'type' => 2,

'read' => 1, 'time' => time()

];

$flag = db('message')->insert($msg); if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '系统错误']);

}

return json(['code' => 0, 'data' => '', 'msg' => 'success']);

}

$this->error('非法访问');

}

拒绝的逻辑如下：

1. 将该条申请消息 标记为"已经拒绝" 2. 在messge表中加入一条 拒绝 消息

加入到消息系统中后的消息，只要等待 msgbox 的轮询获取。即可通知对方，我拒绝了他的申请。

# 好友聊天的实现

#### 前面添加好友的功能已经实现了，那么我们现在来添加一个好友，实现好友之间的聊天功能吧。 当点击一个好友的头像时，会弹出聊天窗口



我们只需要在聊天窗口中输入想要发送的信息，点击发送即可。

如果好友不在线，那我们的消息还能发送给对方吗？这个问题，在看了后面的代码，你就会得到答案。

我们来看一下，我们输入的信息，是如何发送给对方的。

**js部分代码**

//发送消息

layim.on('sendMessage', function(res){

//console.log(res); if(To.type === 'friend'){

layim.setChatStatus('<span style="color:#FF5722;">对方正在输入。。。</span>');

}

// 发送消息

var mine = JSON.stringify(res.mine); var to = JSON.stringify(res.to);

var login\_data = '{"type":"chatMessage","data":{"mine":'+mine+', "to":'+to+'}}';

socket.send( login\_data );

});

layim会监听 sendMessage 事件，当用户输入，发送消息的时候，会触发这个方法。可以看到，我们向

workerman发送了一个chatMessage类型的消息。

**workerman部分代码**

当用户将发送的消息推送给 服务器 时，服务器该怎么去处理呢？我们来看一下 Events.php 的 onMessage 方

法：

case 'chatMessage':

$db1 = Db::instance('db1'); //数据库链接

// 聊天消息

$type = $message['data']['to']['type'];

$to\_id = $message['data']['to']['id'];

$uid = $\_SESSION['id'];

']),

$chat\_message = [

'message\_type' => 'chatMessage', 'data' => [

'username' => $\_SESSION['username'], 'avatar' => $\_SESSION['avatar'],

'id' => $type === 'friend' ? $uid : $to\_id, 'type' => $type,

'content' => htmlspecialchars($message['data']['mine']['content

'timestamp' => time() \* 1000,

]

];

// 加入聊天log表

$param = [

'from\_id' => $uid, 'to\_id' => $to\_id,

'from\_name' => $\_SESSION['username'], 'from\_avatar' => $\_SESSION['avatar'],

'content' => htmlspecialchars($message['data']['mine']['content']), 'timeline' => time(),

'need\_send' => 0

];

switch ($type) {

// 私聊

case 'friend':

// 插入

$param['type'] = 'friend';

if (empty(Gateway::getClientIdByUid($to\_id))) {

$param['need\_send'] = 1; //用户不在线,标记此消息推送

$client\_id);

}

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToUid($to\_id, json\_encode($chat\_message));

// 群聊

case 'group':

$param['type'] = 'group';

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToGroup($to\_id, json\_encode($chat\_message),

}

return;

break;

#### 因为每一条聊天记录都需要入表记录，所以这里，首先我们将这些数据整理好，加入到 v3\_chatlog 表中，方

便用户后期的查看。此外在群中发送消息和对好友发送消息，都通过该方法，只是消息的格式不同，我们这 里讨论，好友与好友之间的交流。

实际的应用中，聊天消息可能会很多，而且并发量可能也会很大。通过传统的mysql去记录，会有一定的 瓶颈，这里可以考虑扩展 mongodb去存储这些大量的而又不重要的数据。

// 插入

$param['type'] = 'friend';

if (empty(Gateway::getClientIdByUid($to\_id))) {

$param['need\_send'] = 1; //用户不在线,标记此消息推送

}

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToUid($to\_id, json\_encode($chat\_message));

此时我会检测用户是否在线。如果用户不在线，设置需要推送格式 为 1，标识此消息需要推送。那你可能会

问，我记录在此处，那什么时候这些离线消息会推送给该好友呢？

不知你还记得，用户登录时触发的 Events.php 的 init消息类型吧，我们在 收到用户登录这个 init消息之后， 做了这么一个动作：

//查询最近1周有无需要推送的离线信息

$time = time() - 7 \* 3600 \* 24;

$resMsg = $db1->select('id,from\_id,from\_name,from\_avatar,timeline,content')->from('v3\_c hatlog')

->where("to\_id= {$uid} and timeline > {$time} and type = 'friend' and need\_send = 1" )

->query();

//var\_export($resMsg); if (!empty($resMsg)) {

foreach ($resMsg as $key => $vo) {

$log\_message = [

'message\_type' => 'logMessage', 'data' => [

'username' => $vo['from\_name'], 'avatar' => $vo['from\_avatar'], 'id' => $vo['from\_id'],

'type' => 'friend',

'content' => htmlspecialchars($vo['content']), 'timestamp' => $vo['timeline'] \* 1000,

]

];

Gateway::sendToUid($uid, json\_encode($log\_message));

//设置推送状态为已经推送

$db1->query("UPDATE `v3\_chatlog` SET `need\_send` = '0' WHERE id=" . $vo['id']);

}

}

#### 在用户登录时候，系统会从日志表中读取这些尚未推送的消息，推送给该好友，做到消息的不遗漏。这部分

的消息，会用过 main.js中的代码体现：

// 离线消息推送

case 'logMessage': setTimeout(function(){layim.getMessage(data.data)}, 1000); break;

**如何验证是否可以互相聊天**

依旧是选择两个互为好友的账号，分两个浏览器登录，互发消息试试。

# 创建群组的实现

#### 创建群组的入口和添加好友是一样的，点击面板最下面的 + 号 会看到如下的页面：

这部分的页面在哪里呢？如果你认真看了 《添加好友的实现》 这一章节的话，你会记得，查找群组和添加好 友在一个页面。都是通过 main.js 中的：

//基础配置

layim.config({

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

//查看群员接口

,members: {

url: member\_list\_url

,data: {}

}

,uploadImage: {

url: upload\_img\_url

}

,uploadFile: {

url: upload\_file\_url

}

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

},

{

alias: 'map'

,title: '地理位置'

,icon: '&#xe634;'

},

]

,initSkin: '3.jpg' //1-5 设置初始背景

,msgbox: msg\_box\_url//消息盒子页面地址，若不开启，剔除该项即可

,find: find\_url //发现页面地址，若不开启，剔除该项即可

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

});

#### 其中的 find 接口， find\_url

var find\_url = "{:url('findgroup/index')}";

**控制器部分**

我们来重温这部分的代码， findgroup.php 的 index方法：

//显示查询 / 添加 分组的页面

public function index()

{

//查询最新4名非好友，建议用户添加

$uid = session('f\_user\_id');

$friends = db('friends')->field('friend\_id')->where('user\_id', $uid)->select();

$where = '1=1'; if(!empty($friends)){

foreach($friends as $vo){

$fArr[] = $vo['friend\_id'];

}

unset($friends);

$where .= ' and id not in(' . implode(',', $fArr) . ')';

}

here)

$userList = db('chatuser')->field('id,user\_name,area,avatar,sex,age')->where($w

->order('id desc')->limit(4)->select();

//初始化省份

$province = db('area')->field('id,area\_name')->where('parent\_id', 0)->select();

//推荐新建的群组，并且是审核通过的，供用户添加

$groupArr = db('chatgroup')->where('status = 1')->order('id desc')->limit(4)->s

elect();

$this->assign([

'group' => $groupArr, 'age' => config('age'), 'province' => $province, 'user' => $userList

]);

return $this->fetch();

}

#### 其实这里，主要就是

$groupArr = db('chatgroup')->where('status = 1')->order('id desc')->limit(4)->select();

这部分，与我们的群组管理关系密切。选择几个推荐的群组，供用户快速添加。

**视图部分**

视图部分和《添加好友的实现》 这一章节的完全相同，我在这里就主要截取本章密切相关的页面做讲解。

<div id="tab-2" class="tab-pane">

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-content">

<form class="form-horizontal m-t" id="commentForm"

method="post" action="">

<div class="form-group">

<div class="col-sm-10">

<div class="input-group">

<input type="text" class="form-cont

rol" name="search\_txt" placeholder="输入群组名称" id="search\_txt">

<span class="input-group-btn">

<button type="button" class="bt

n btn-primary" id="find">搜索</button>

</span>

&nbsp;&nbsp;&nbsp;

<span class="input-group-btn">

<button type="button" class="bt

n btn-w-m btn-warning" id="addGroup">创建群组</button>

</span>

&nbsp;&nbsp;&nbsp;

n btn-w-m btn-primary" id="myGroup">我的群组</button>

<span class="input-group-btn">

<button type="button" class="bt

</span>

abel></div>

</div>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

<!-- 最近新建的分组 -->

<div class="row">

<div class="col-sm-3"><label id="search\_title">最近新添加的群组</l

</div>

<div class="row" id="search\_data">

{if !empty($group)}

{foreach name="group" item="vo"}

<div class="col-sm-3">

<div class="ibox float-e-margins">

<div class="ibox-title">

<h5>{$vo.group\_name}</h5>

</div>

<div class="ibox-content">

<div style="margin: 0 auto">

<img src="{$vo.avatar}" width="50px" height="50

px" style="margin-left:50px"/>

</div>

<div style="margin:10px 50px"><button class="btn bt

n-primary" type="button" data-id="{$vo.id}" data-owner="{$vo.owner\_id}" onclick="joinGr oup(this)">加入</button></div>

</div>

</div>

</div>

{/foreach}

{/if}

</div>

</div>

#### 从 find.js 中可以看出：

//创建群组

$("#addGroup").click(function(){ layui.use(['layer'], function(){

var layer = layui.layer; layer.ready(function(){

$.getJSON(check\_group\_url, function(res){ if(1 == res.code){

//iframe层

layer.open({

type: 2,

title: '创建群组',

shade: [0.2, '#393D49'],

maxmin: false, //开启最大化最小化按钮 area: ['700px', '270px'], content: add\_group\_url

});

}else{

layer.alert(res.msg, {closeBtn:0,icon:2,title:'友情提示'});

});

});

}

}, 'json');

//close\_layer(); //关闭当前的layer层

});

#### 添加群组，先后触发了两个接口：

var check\_group\_url = "{:url('findgroup/checkGroupAuth')}"; var add\_group\_url = "{:url('findgroup/addgroup')}";

这里面的设计思路是什么呢？

为了更好的控制前台用户的创建群组的行为，我们在管理员后台做了一个 群组基础配置 功能，对创建群 组是否需要审核，是否允许用户创建群组，最多能创建多少个群组以及最多能加入多少个群组，做了细致 的控制。因此，用户创建群组的时候，应该判断，他是否可以创建群组。所以，此处触发了两个接口。

**权限检测接口的设计与实现**

我们来看看，checkGroupAuth 该如何去实现：

//检测群组创建权限

public function checkGroupAuth()

{

//用户点击创建群组的时候，检测用户是否还可以创建群粗

$config = readConfig();

//Array ( [make] => 1 [pass] => -1 [maxgroup] => 10 [maxjoin] => 15 ) if(-1 == $config['make']){

return json(['code' => -1, 'data' => '', 'msg' => '系统不允许创建群组']);

}

//检测用户当前创建的群组数是否到达了最大要求

$cNum = db('chatgroup')->where('owner\_id', session('f\_user\_id'))->count(); if($cNum >= $config['maxgroup']){

return json(['code' => -2, 'data' => '', 'msg' => '您已经创建了' . $cNum . '

个群组,不可再创建!']);

}

return json(['code' => 1, 'data' => '', 'msg' => 'ok']);

}

读取系统的 群组基础配置 ，如果系统此时不允许创建群组，直接返回错误，否则检测当前用户是否已经创建

了到系统规定的最大可创建群组数。

chatgroup 表是专门记录系统创建的群组信息的表

**添加群组的控制器设计**

下面来看看，通过权限验证之后，如何添加群组 addgroup

//添加群组

public function addGroup()

{

if (request()->isAjax()) {

//检测创建群组权限

$auth = $this->checkGroupAuth()->getData(); if(1 != $auth['code']){

return json(['code' => -5, 'data' => '', 'msg' => $auth['msg']]);

}

$param = input('post.');

if (empty($param['group\_name'])) {

return json(['code' => -1, 'data' => '', 'msg' => '群组名不能为空']);

}

$param['owner\_name'] = session('f\_user\_name');

$param['owner\_id'] = session('f\_user\_id');

$param['owner\_avatar'] = session('f\_user\_avatar');

$param['owner\_sign'] = session('f\_user\_sign');

$param['addtime'] = time();

//如果开启了群组审核功能，那么新创建的群组必须通过审核才可以使用

//默认不需审核

$status = 1;

$config = readConfig(); if(1 == $config['pass']){

$status = -1; //等待审核

}

$param['status'] = $status;

$groupId = db('chatgroup')->insertGetId($param); if (empty($groupId)) {

return json(['code' => -2, 'data' => '', 'msg' => '添加群组失败']);

}

unset($param);

//将自己加入群组

$groupDetail = [

'user\_id' => session('f\_user\_id'), 'user\_name' => session('f\_user\_name'), 'user\_avatar' => session('f\_user\_avatar'), 'user\_sign' => session('f\_user\_sign'), 'group\_id' => $groupId,

];

$flag = db('groupdetail')->insert($groupDetail); if (empty($flag)) {

return json(['code' => -3, 'data' => '', 'msg' => '添加群组失败']);

}

$return = [

'join\_id' => session('f\_user\_id'), 'group\_id' => $groupId

];

return json(['code' => 1, 'data' => $return, 'msg' => '创建群组 成功']);

}

return $this->fetch();

}

**添加群组的视图**

#### application\index\view\findgroup 下新建 addgroup.html

<!DOCTYPE

<html>

<head>

<meta

<meta

">

html>

charset="utf-8">

name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

<title>laychat-v3.0 - 创建群组</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

</head>

<body>

<div class="main layui-clear" style="margin-top:15px">

<div class="layui-form layui-form-pane" style="width: 70%;margin: 0 auto">

<div class="layui-form-item">

<label for="group\_name" class="layui-form-label">群名称</label>

<div class="layui-input-block">

<input type="text" id="group\_name" name="group\_name" class="layui-input

" >

</div>

</div>

<input type="hidden" id="group\_avatar"/>

<div class="layui-form-item">

<label for="group\_name" class="layui-form-label">群头像</label>

<div class="layui-input-block">

<input name="avatar" class="layui-upload-file" type="file" id="avatar"/

>

<span style="margin-left: 5px" id="upimg">

</span>

</div>

</div>

<div class="layui-form-item" style="float:right">

<button class="layui-btn" id="sub">提交</button>

</div>

<div class="layui-form-item fly-form-app">

注：上传的图片只能是png、jpg，且大小为100kb以内<br/>

</div>

</div>

</div>

<script src=" COMMON /layui/layui.js"></script>

<script src=" COMMON /jquery.min.js"></script>

<script type="text/javascript">

var add\_group\_url = "{:url('findgroup/addgroup')}"; var up\_img\_url = "{:url('Findgroup/upGroupAvatar')}";

</script>

<script src=" COMMON /js/group.js"></script>

</body>

</html>

#### 展示如下：

输入群组名称和上传合适的 群头像即可。上传群头像用到的接口：

var up\_img\_url = "{:url('Findgroup/upGroupAvatar')}";

这里采用的是 异步上传图片的方式，也是用的layui的一个插件。

/\*\*

\* 上传图片方法

\* @param $param

\*/

public function upGroupAvatar()

{

$auth = $this->checkGroupAuth()->getData(); if(1 != $auth['code']){

return json(['code' => -5, 'data' => '', 'msg' => $auth['msg']]);

}

// 获取表单上传文件

$file = request()->file('avatar');

// 移动到框架应用根目录/public/uploads/ 目录下

if(!is\_null($file)){

$fileInfo = $file->getInfo();

$imgExt = config('common\_img');

$ext = explode('.', $fileInfo['name']);

$ext = array\_pop($ext);

if(!in\_array($ext, $imgExt)){

return json(['code' => -2, 'data' => '', 'msg' => '请上传' . implode(','

, $imgExt) . '的图片']);

}

unset($ext);

$size = config('common\_size'); if($fileInfo['size'] > $size){

return json(['code' => -3, 'data' => '', 'msg' => '上传的图片超过' . $siz

e/1024 . 'kb']);

}

unset($fileInfo);

$info = $file->move(ROOT\_PATH . 'public' . DS . 'uploads'); if($info){

// 成功上传后 获取上传信息

$avatar = '/uploads' . '/' . date('Ymd') . '/' . $info->getFilename(); return json(['code' => 1, 'url' => $avatar, 'msg' => 'success']);

}else{

// 上传失败获取错误信息

return json(['code' => -4, 'url' => '', 'msg' => $file->getError()]);

}

}

return json(['code' => -1, 'data' => '', 'msg' => '上传群头像失败']);

}

#### 由于本系统 用两张表 来记录群组的先关信息

v3\_chatgroup 这张表记录的是群组的创建的基本信息 v3\_groupdetail 这张表记录的群组中的成员信息。

因此，当用户成功创建群组之后，我们需要将用户创建群组的 基本信息 加入到 v3\_chatgroup，以及将用户 加入到 v3\_groupdetail，因为管理员(创建者)也是属于这个群组的。我们在 addGroup 这个方法中，已经看到 了这部分的操作逻辑。接下来我们看看，用户创建了群组，在哪里展现呢？

**如何看到我创建的群组**

在 group.js 中有这么一段操作

$.post(add\_group\_url, {'group\_name' : group\_name, 'avatar' : avatar}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:2000});

//推送给socket服务器，将管理员加入群组

var group\_data = '{"type":"addGroup","join\_id":"' + res.data.join\_id + '", ' + '"avatar":"' + avatar + '", "id":"' + res.data.group\_id + '", "groupname":"'

+ group\_name + '"}'; socket.send(group\_data);

setTimeout(function(){

var index = parent.layer.getFrameIndex(window.name); //先得到当前iframe层的索引 parent.layer.close(index); //再执行关闭

}, 1500);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json');

这部分的前部分 向 add\_group\_url即 addGroup 方法发送添加群组的方法，当收到成功的回调之后，我们看

到，其向workerman发送一个添加群组的"消息"，这是为什么呢？

在前面的 addGroup 方法，用户其实已经真实的创建了群组，并且自己也加入了自己的群组。(这里说的真 实，就是这些信息已经入库了)但是此时我们在用户面板上：

此处，依旧无法看到自己的群组，按理说我创建的群组，我也加入这个群组了，我是可以在这个群组聊天的 啊。为什么没有呢？降到这里，我首先得回头给大家补充一点，也就是渲染首页时，我没讲的一个接口。

**首页面板的群组获取接口**

一个用户可能加入了很多个群组，在用户登录的时候，我们应该能看到我加入过的群组，这样，我可以在里

面聊天。这部分该如何实现呢？其实 layIM在这方面已经设置了接口：

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

我们在 layim的初始化接口，能看到这么一个接口，当然这里面有获取我的好友列表，同时也兼有获取用户群

组的列表，我们来看看 user\_list\_url 对应的 Index.php 下的 getList 方法：

//查询当前用户的所处的群组

$groupArr = db('groupdetail')->alias('j')->field('c.group\_name groupname,c.id,c

.avatar')

->join('v3\_chatgroup c', 'j.group\_id = c.id')->where('j.user\_id', $uid)

->group('j.group\_id')->select();

这里我做了用户所属群组的查取，并组织成相应的格式。这样就可以在用户群组面板中看到我加入的群组

了，就像这样：

那有了群组，我怎么看群组中的成员呢？群组中的成员，也是有接口可以调用的：

//查看群员接口 members: {

url: member\_list\_url

,data: {}

}

没错就是这个接口，我们在配置项中能看到 member\_list\_url 对应的是：

var member\_list\_url = "{:url('index/getMembers')}";

那么我们看看layim的文档，需要什么格式的数据，getMembers 按照这样的格式，将该群组中的成员展示出

来就行了。

//获取组员信息

public function getMembers()

{

$id = input('param.id');

//群主信息

$owner = db('chatgroup')->field('owner\_name,owner\_id,owner\_avatar,owner\_sign')-

>where('id = ' . $id)->find();

//群成员信息

$list = db('groupdetail')->field('user\_id id,user\_name username,user\_avatar ava tar,user\_sign sign')

->where('group\_id = ' . $id)->select();

$return = [

'code' => 0, 'msg' => '', 'data' => [

'owner' => [

'username' => $owner['owner\_name'], 'id' => $owner['owner\_id'], 'owner\_id' => $owner['owner\_avatar'], 'sign' => $owner['owner\_sign']

],

'list' => $list

]

];

return json( $return );

}

#### 此时，用户点击，群组聊天面板的 下箭头 就可以看到群组中的成员了

**再讲如何看到我创建的群组**

这个是用户登录之后，从数据库拉取的群组信息，用户创建完群组之后，也不可能从群组重新拉取一遍，除

非刷新页面，这个是TCP链接，刷新页面，等于断开了一次了。如果不和用户的UID关联的话，其实就算是和 以前的客户端属于不同的用户了。那我们该如何在不刷新的情况下，在创建完群组之后，就在群组面板看到 呢？ 别怕，layIM提供了添加群组到面版的方法。我们先来具体看看，用户创建群组成功之后，推送的消息：

//推送给socket服务器，将管理员加入群组

var group\_data = '{"type":"addGroup","join\_id":"' + res.data.join\_id + '", ' +

'"avatar":"' + avatar + '", "id":"' + res.data.group\_id + '", "groupname":"' + grou p\_name + '"}';

socket.send(group\_data);

#### workerman怎么处理的吧：

case 'addGroup':

// 创建群组

$client\_id = Gateway::getClientIdByUid($message['join\_id']); Gateway::joinGroup($client\_id['0'], $message['id']); // 将该用户加入群组

$add\_message = [ 'message\_type' => 'addGroup', 'data' => [

'type' =>

'avatar' 'id'

'group',

=>

=>

'groupname'

$message['avatar'],

$message['id'],

=> $message['groupname']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message)); return;

break;

我们可以看到，其向需要添加群组的客户端推送一个 addGroup 类型的消息，接下来，我们看看 main.js 如何

处理：

// 添加 分组信息

case 'addGroup':

//console.log(data.data); layim.addList(data.data); break;

调用了 layim.addList 将该群组加入到 群面板。

# 按条件查找群组

#### 按条件查找群组的页面，前面我已经展示了多次了，《创建群组的实现》 这一节已经展示过了，界面如下：

当系统中检测有需要推荐的群组时，系统会推荐群组，供用户添加。我们通过 find.js 看到 ：

//搜索群组

$("#find").click(function(){

layui.use(['layer'], function(){ var layer = layui.layer;

var search\_txt = $("#search\_txt").val(); if( '' == search\_txt ){

layer.ready(function(){ layer.msg('群组名称不能为空', {'time' : 2000});

});

return;

}

$.getJSON(search\_group\_url, {'search\_txt':search\_txt}, function(res){ var \_html = "";

$("#search\_title").text('搜索结果：');

if( 1 == res.code ){

$.each(res.data, function(k, v){

\_html += '<div class="col-sm-3"><div class="ibox float-e-margin s"><div class="ibox-title">';

\_html += '<h5>' + v.group\_name + '</h5></div><div class="ibox-c ontent"><div style="margin: 0 auto">';

\_html += '<img src="' + v.avatar + '" width="50px" height="50px " style="margin-left:50px"/>';

\_html += '</div><div style="margin:10px 50px"><button class="bt

n btn-primary" type="button" data-id="' + v.id + '" onclick="joinGroup(this)">加入</butt on></div>';

\_html += '</div></div></div>';

});

$("#search\_data").html(\_html);

}else{

\_html += '<div class="col-sm-12"><div class="ibox float-e-margins">

<div class="ibox-content">';

\_html += '<div class="form-group"><div class="col-sm-10"><label sty le="color: red">' + res.msg + '</label>';

\_html += '</div></div></div></div></div>';

$("#search\_data").html(\_html);

}

});

});

});

#### 搜索群组我已经绑定了 search 接口

var search\_group\_url = "{:url('findgroup/search')}";

其实查询的接口的逻辑也很简单，查询符合用户查询条件的 群组信息返回即可 search 方法：

//搜索查询群组

public function search()

{

$groupName = input('param.search\_txt');

$find = db('chatgroup')->where("group\_name like '%" . $groupName . "%'")->where('sta tus')->select();

if( empty($find) ){

return json( ['code' => -1, 'data' => '', 'msg' => '您搜的群不存在' ] );

}

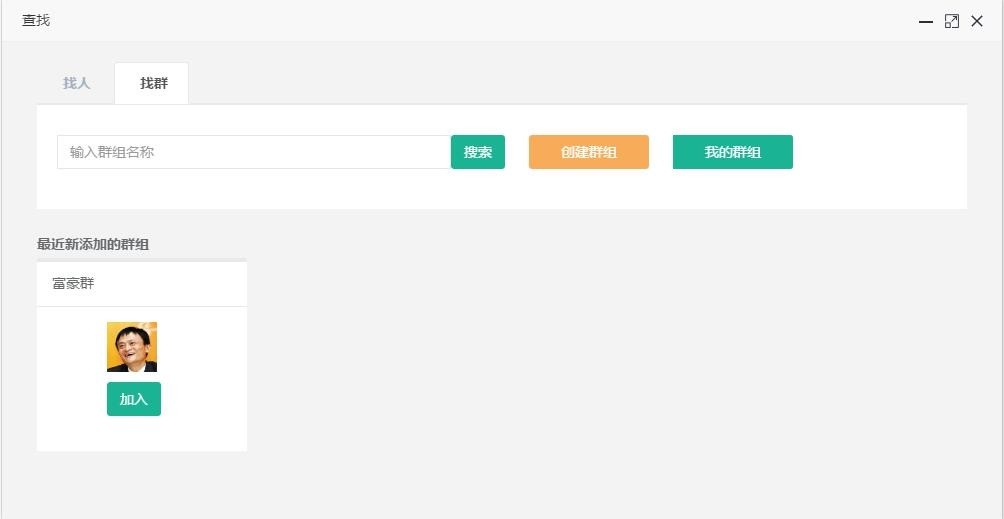
return json( ['code' => 1, 'data' => $find, 'msg' => 'success' ] );

}

这个功能其中的一些小细节，你可以去完善一下，例如：推荐群组的显示问题，还有就是与后台控制更加 密切的问题。

# 申请加入群组

#### 我们主面板的 + 号，即可展现申请加群组的页面。这个前面，我们已经做过了。当别的用户创建了群组之 后，在查找页面可见：



如何设计加入群组的操作呢？这部分的逻辑，我们放到了 find.js 中

//加入群组

function joinGroup(obj){

layui.use(['layim', 'jquery'], function() { var layim = layui.layim,

$ = layui.jquery;

layim.add({

type: 'group'

, username: $(obj).parent().parent().siblings().find('h5').text()

, avatar: $(obj).parent().siblings().find('img').attr('src')

, submit: function(group, remark, index){

//通知对方

$.post(apply\_group\_url, {

owner : $(obj).attr('data-owner')

, group : $(obj).attr('data-id')

, remark: remark

}, function(res){ if(1 == res.code){

var group\_name = $(obj).parent().parent().siblings().find('h5')

.text();

'src');

r('data-owner') + '"' +

var group\_avatar = $(obj).parent().siblings().find('img').attr( var group\_id = $(obj).attr('data-id');

var apply\_data = '{"type":"applyGroup", "to\_id":"' + $(obj).att

', "join\_id": "' + res.data.uid + '", "groupname" : "' + gr

oup\_name + '"' +

', "groupid" : "' + group\_id + '", "join\_name" : "' + res.d

ata.uname + '"' +

',

+ '", "join\_sign": "'

"remark" : "' + remark + '", "groupavatar"

: "' + group\_

avatar

+ res.data.sign + '"' + ', "join\_avatar"

:

"'

+

res.data.av

atar + '"}';

socket.send(apply\_data);

layer.msg('申请已发送，请等待管理员确认', { icon: 1

,shade: 0.5

}, function(){ layer.close(index);

});

}else{

layer.msg(res.msg, { icon: 1

,shade: 0.5

}, function(){ layer.close(index);

});

}

});

}

});

});

}

#### 当用户点击加入群组，需要添加者输入验证信息，等待管理员确认。我们可以看到， 此处调用了

apply\_group\_url 接口

var apply\_group\_url = "{:url('findgroup/applyGroup')}";

接收到回调之后，会向服务器(workerman)发送一个 applayGroup消息

var apply\_data = '{"type":"applyGroup", "to\_id":"' + $(obj).attr('data-owner') + '"' +

', "join\_id": "' + res.data.uid + '", "groupname" : "' + gr

oup\_name + '"' +

', "groupid" : "' + group\_id + '", "join\_name" : "' + res.d

ata.uname + '"' +

', "remark" : "' + remark + '", "groupavatar" : "' + group\_ avatar + '", "join\_sign": "'

+ res.data.sign + '"' + ', "join\_avatar" : "' + res.data.av

atar + '"}'; socket.send(apply\_data);

通知管理员，有人申请加入群组。下面我们先来看看，向thinkphp5发送的请求如何处理，然后回头处理，回

调的部分。

**控制器的实现**

applyGroup 方法：

//申请加组

public function applyGroup()

{

if(request()->isAjax()){

$param = input('post.');

//检测该用户是否已经加入了该群组

$isJoin = db('groupdetail')->where('user\_id=' . session('f\_user\_id') . ' an d group\_id=' . $param['group'])->find();

if(!empty($isJoin)){

return json(['code' => -1, 'data' => '', 'msg' => '您已经加入了该群组']);

}

unset($isJoin);

$return = [

'uid' => session('f\_user\_id'), 'uname' => session('f\_user\_name'), 'avatar' => session('f\_user\_avatar'), 'sign' => session('f\_user\_sign')

];

return json(['code' => 1, 'data' => $return, 'msg' => 'success']);

}

$this->error('非法访问');

}

#### 当用户申请加入群组的时候，我们首先判断用户是否已经是该群组的成员了，重复的添加是没有意义的。如

果用户没有加入该群组，则返回用户可添加群组的各种有效的身份信息。前台 js 收到有效的 返回信息之后， 之后后面的回调，向workerman推送入群申请。下面我们看一下，workerman如何处理这个入群申请消息。

**workerman代码的实现**

依旧是在 Events.php 中的 onMessage 方法中加入一个 消息处理分支：

case 'applyGroup':

// 申请加入群组

// 通知群组管理员，进行入群申请

$client\_id = Gateway::getClientIdByUid($message['to\_id']); if(!empty($client\_id)){

$apply\_message = [

'message\_type' => 'applyGroup', 'data' => [

'uid' => $message['to\_id'], //群组的管理员

'groupname' => $message['groupname'], 'groupid' => $message['groupid'], 'groupavatar' => $message['groupavatar'],

'joinid' => $message['join\_id'], //申请加入群组的id

'joinname' => $message['join\_name'], //申请加入群组的用户名 'joinsign' => $message['join\_sign'], //申请加组的的用户签名 'joinavatar' => $message['join\_avatar'], // 申请人头像 'remark' => $message['remark'] //附加信息

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($apply\_message))

;

}

return; break;

#### 这里的逻辑很简单，就是向管理员 推送申请者的 申请信息。而对应的 客户端 $client\_id['0'] 会接收到这部分

的消息推送。在 main.js 中会有相应的处理。

**js部分的设计**

从服务器(workerman) 推送过来的数据，我们都会在 main.js 中做接收处理。socket.onmessage 方法中，加

入一个 消息处理分支：

// 申请加入群组

case 'applyGroup':

//console.log(data.data);

//询问框

var index = layer.confirm(

data.data.joinname + ' 申请加入 ' + data.data.groupname + "<br/> 附加 信息： " + data.data.remark , {

btn: ['同意', '拒绝'], //按钮 title: '加群申请',

closeBtn: 0,

icon: 3

}, function(){

$.post(join\_group\_url,

{

'user\_id': data.data.joinid, 'user\_name': data.data.joinname, 'user\_avatar': data.data.joinavatar, 'user\_sign': data.data.joinsign, 'group\_id': data.data.groupid

},

function(res){

if(1 == res.code){

+ data.data.joinid + '"' +

var join\_data = '{"type":"joinGroup", "join\_id":"'

', "group\_id": "' + data.data.groupid + '", "gr

oup\_avatar": "' + data.data.groupavatar + '"' +

', "group\_name": "' + data.data.groupname + '"}

';

socket.send(join\_data);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json'); layer.close(index);

}, function(){

});

break;

#### 这里能看到，此处的 消息处理相比其他的要复杂一些。这个原因很简单，因为用户申请加群之后，管理员 有"同意"还是"拒绝"的权利，"拒绝"部分我们可以不做任何的处理。(layIM并没有给加群添加类似于加好友那样 的消息盒子机制，因此此处拒绝不做处理。)主要我们来看看同意部分的逻辑，用户管理员点击同意之后，会 向join\_group\_url发送一个请求，我们从前面的配置，很容易看出join\_group\_url是请求哪个php文件。

var join\_group\_url = "{:url('findgroup/joinDetail')}";

我们来看看，joinDetail如何处理管理员同意用户加入群组的逻辑：

//同意加入群组

public function joinDetail()

{

if(request()->isAjax()){

$param = input('post.');

$flag = db('groupdetail')->insert($param); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => "系统错误"]);

}

return json(['code' => 1, 'data' => '', 'msg' => 'success']);

}

$this->error('非法访问');

}

v3\_groupdetail 表主要记录用户所属群组的详细信息，用户和群组 是多对多的关系，即一个用户可以加入 多个群组同时一个群组可以拥有多个用户。

前端的 js 已经将需要加入群组的信息，完全推送过来，此处只要入库即可完成 用户申请-->管理员同意 这个

过程的操作，剩下的就是聊天面板的操作了。实际上此时用户已经添加了群组。我们接下来看看，剩下的面 板操作。

从上面的 js 我们可以看出，当用户信息入库之后，依旧有一个回调向 workerman 推送数据：

var join\_data = '{"type":"joinGroup", "join\_id":"' + data.data.joinid + '"' +

', "group\_id": "' + data.data.groupid + '", "group\_avatar": "' + data.data.group avatar + '"' +

', "group\_name": "' + data.data.groupname + '"}'; socket.send(join\_data);

这一部分推送的主要目的是为了，让申请者 的 聊天面板中 出现该群组的入口。



即上图是以的群聊入口。向workerman推送了这样的消息，我们又明确了这部分的目的，处理起来就很简单

了。workerman服务器只需要向 添加者 客户端 推送添加群组入口到面板的消息即可。Events.php 中加入如 下的分支：

推送 户加入群组

case 'joinGroup':

//将申请人加入讨论组

$client\_id = Gateway::getClientIdByUid($message['join\_id']); //若在线实时

if( !empty($client\_id) ){

Gateway::joinGroup($client\_id['0'], $message['group\_id']); //将该用

$add\_message = [

'message\_type' => 'addGroup', 'data' => [

'type' => 'group',

'avatar' => $message['group\_avatar'], 'id' => $message['group\_id'], 'groupname' => $message['group\_name']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

#### 这里同样是判断用户在线再推送，是因为，用户在登录的时候，需要拉取群组信息。这部分是从数据库读取 的和workerman服务器无关。逻辑很简单，就是推送一条 addGroup 类型的数据，以及相关的需求信息。这 时，我们只要在 main.js 中接收并处理即可

// 添加 分组信息

case 'addGroup':

//console.log(data.data);

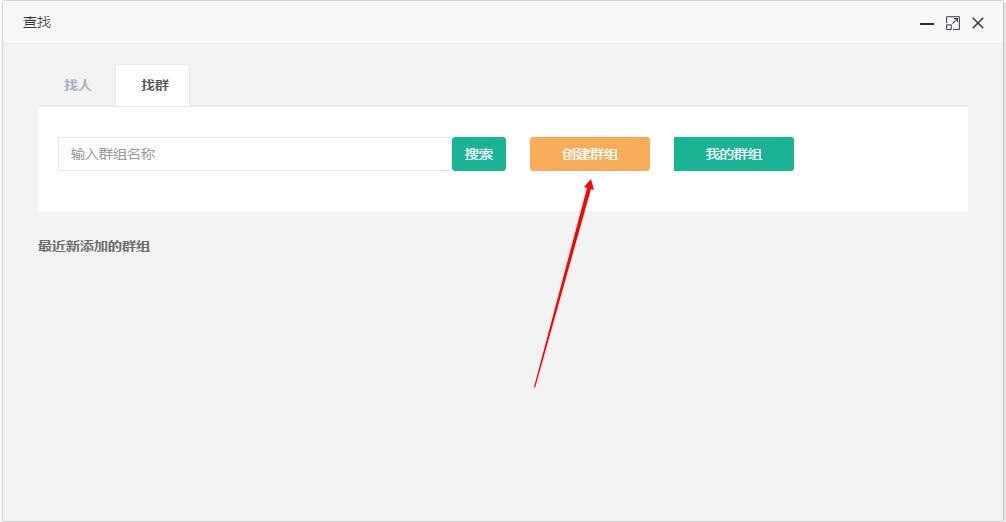
layim.addList(data.data); break;

layIM 提供了添加 群组到面板的方法，很简单便可以实现这个步骤。

本系统的添加群组申请没有重复推送的步骤，也就是说，如果管理员不在线，申请入群的操作将无法被管 理员接收。关于这方面的优化，感兴趣的可以 仿照 离线消息的推送方式来完善，或者你有其他更好的方 式也行。

# 管理我的群组

#### 用户创建的群组，用户应该拥有完全的支配权。那么，我们怎么实现这部分的功能呢？同样的，在创建群组 的旁边，我们能看到一个我的群组：



借用的旧图，不要纠结这些细节。哈哈

**功能需求**

我的群组用户应该具有以下的功能：

1. 查看我创建的所有群组

2. 管理我的群组中的所有成员

3. 解散我的群组

**具体的实现**

所有用户创建的群组信息，都会存入 v3\_chatgroup 中，因此很容易调取用户的 创建的群组信息。每个群组

的成员信息，又在v3\_groupdetail 中有详细的信息。因此成员信息也很容易调取。解散就更好理解了，就是 删除呗。好了，思路有了，那就开始吧。 由于前面我们知道，这个找群的页面，拥有大量的信息，由于是两个大功能合并的页面 (找人和群功能)。我们 来看一下，我的群组触发的是哪个接口吧？查看 find.js

//管理的群组

$("#myGroup").click(function(){ window.location.href = my\_group\_url;

});

管理我的群组，是用的简单暴力的 跳转，看一下 my\_group\_url

var my\_group\_url = "{:url('findgroup/myGroup')}";

调取的是 myGroup 方法，其实通过前面的分析，我们也很好理解，这个功能该怎么去实现，无非就是读取该

用户的创建的群组信息，因此，很容易实现这个方法 myGroup

//管理我的群组

public function myGroup()

{

if (request()->isAjax()) {

$groupId = input('param.id');

$users = db('groupdetail')->field('user\_name,user\_id,user\_avatar,group\_id')

->where('group\_id', $groupId)

->paginate(4);

return json(['code' => 1, 'data' => $users, 'msg' => 'success']);

}

$group = db('chatgroup')->field('id,group\_name name')->where('owner\_id', sessio n('f\_user\_id'))->select();

$this->assign([

'group' => json\_encode($group)

]);

return $this->fetch();

}

这个方法实现了两个功能，一个是调取我创建的所有的群组，一个是用户点击群组的时候，展示用户详情。

同时渲染了一个页面，给大家展示一下，我设计的页面，这样也能很快的知道我们页面的设计思路：

**我的群组的实现**

左边是一个群组树，右边是一个群组成员列表，上面有解散群组的按钮，右上角给用户展示当前操作的群组

信息。因为我设计的是用户点击群组树上的节点展示数据。因此，要给用户一个友好的提示。好了，来看一 下，具体的实现吧，在application\index\view\findgroup 下新建一个 mygroup.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>群组列表</title>

<meta name="renderer" content="webkit">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="format-detection" content="telephone=no">

<link rel="stylesheet" type="text/css" href=" COMMON /layui/css/layui.css" media=

"all">

<link rel="stylesheet" type="text/css" href=" COMMON /bootstrap/css/bootstrap.css " media="all">

<link rel="stylesheet" type="text/css" href=" COMMON /global.css" media="all">

<link rel="stylesheet" type="text/css" href=" CSS /personal.css" media="all">

</head>

<body>

<section class="layui-larry-box">

<div class="larry-personal">

<div class="layui-tab">

<blockquote class="layui-elem-quote mylog-info-tit">

<button class="layui-btn layui-btn-small" onclick="history.back();">

<i class="layui-icon" title="后退"></i>

</button>

<button class="layui-btn layui-btn-danger layui-btn-small" onclick="bre ak\_up();">解散群组</button>

<div style="float: right">

<span id="g\_name" style="font-size: 14px;color: red"></span>

<span id="m\_num" style="font-size: 14px;color: red"></span>

</div>

</blockquote>

<div class="larry-separate"></div>

<div class="layui-tab-content larry-personal-body clearfix mylog-info-box">

<!-- 操作日志 -->

<div class="layui-tab-item layui-field-box layui-show col-sm-4">

<ul id="mygroup"></ul>

</div>

<div class="layui-tab-item layui-field-box layui-show col-sm-8">

<table class="layui-table table-hover" lay-even="" lay-skin="nob">

<thead>

<tr>

<th></th>

<th>群成员ID</th>

<th>群成员头像</th>

<th>群成员名</th>

<th>操作</th>

</tr>

</thead>

<tbody id="view">

</tbody>

</table>

<div id="page">

</div>

</div>

</div>

</div>

</div>

</section>

<input type="hidden" id="group\_id"/>

<input type="hidden" id="now\_page"/>

<script type="text/javascript" src=" COMMON /layui/layui.js"></script>

<!-- 展示用户信息模板 -->

<script id="userList" type="text/html">

{{# for(var i = 0, len = d.data.length; i < len; i++){ }}

<tr>

<td></td>

<td>{{ d.data[i].user\_id }}</td>

<td><img src="{{ d.data[i].user\_avatar }}" width="25px" height="25px"/></td>

<td>{{ d.data[i].user\_name }}</td>

<td><button class="layui-btn layui-btn-danger layui-btn-mini" onclick="letOut({

{ d.data[i].user\_id }})">移除出群</button></td>

</tr>

{{# } }}

</script>

<script type="text/javascript">

var detail\_url = "{:url('findgroup/mygroup')}";

var remove\_member\_url = "{:url('findgroup/removeMember')}"; var remove\_group\_url = "{:url('findgroup/removeGroup')}"; var group\_node = {$group};

</script>

<script type="text/javascript" src=" COMMON /js/mygroup.js"></script>

</body>

</html>

#### 这里用到了 laytpl和laypage以及laytree等一些插件，不了解的，可以自行查看手册。都在layer的官网。核心 的操作，我都放到了mygroup.js 中，其中展示群组树的代码：

layui.tree({

elem: '#mygroup' //指定元素

, click: function (item) { //点击节点回调 if(0 != item.id){

layer.ready(function(){

var loading = layer.load(0, {shade: false});

$("#g\_name").text(item.name);

$("#group\_id").val(item.id);

getMembers($, laytpl, laypage, item.id, 1, loading);

});

}

}

, nodes: [ //节点

{

name: '我的群组'

, id: 0

, spread : true

, children: group\_node

}

]

});

$(".layui-tree-leaf").html('&#xe613;');

没什么好讲的，查看手册就知道具体的用法。

**管理我的群组成员的是实现**

这里的管理成员，只要分为查看成员，以及将成员移踢出。查看很简单，就是从v3\_groupdetail中拉取群成员

信息，移除同样是讲该用户的信息从 v3\_groupdetail 删除即可。我们关注两个接口：

var detail\_url = "{:url('findgroup/mygroup')}";

var remove\_member\_url = "{:url('findgroup/removeMember')}";

第一个前面我们也有讲到，查看群组用户详情，这不部分的js代码详情，在mygroup.js中可以看到：

//异步获取数据

function getMembers($, laytpl, laypage, id, curr, loading){

$("#now\_page").val(curr); //标注当前页面

$.getJSON(detail\_url, { page: curr || 1, id : id

}, function(res){

layer.close(loading); if(1 == res.code){

$("#m\_num").html('(' + res.data.total + '人)');

var gettpl = document.getElementById('userList').innerHTML; laytpl(gettpl).render(res.data, function(html){

document.getElementById('view').innerHTML = html;

});

//分页 laypage({

cont: 'page'

,pages: Math.ceil(res.data.total/res.data.per\_page)

,first: false

,last: false

,curr: curr || 1 //当前页

,jump: function(obj, first){

if(!first){ //点击跳页触发函数自身，并传递当前页：obj.curr getMembers($, laytpl, laypage, id, obj.curr, loading);

}

});

}

}

}

});

#### 这部分是 在点击树节点时触发的，这里用到了的插件，请详细阅读手册，了解用法。

请认真阅读 laytree 和 laytpl 以及 laypage 的用法手册。

接下来，我看移除群成员的设计 remove\_member\_url，调用的是 removeMember 方法，其实就是删除该成

员即可。 js部分：

//移除群成员

function letOut(uid) {

layui.use(['layer', 'laytpl', 'laypage'], function() { var layer = layui.layer,

$ = layui.jquery,

laytpl = layui.laytpl, laypage = layui.laypage;

layer.ready(function(){

//询问框 layer.confirm('确定移除该成员？', {

btn: ['确定','取消'], title: '友情提示', closeBtn: 0,

icon: 3

}, function(){

var group\_id = $('#group\_id').val(); if('' == group\_id){

layer.msg('数据异常', {time:2000});

return ;

}

var page = $('#now\_page').val() || 1; //需要刷新的页面

$.post(remove\_member\_url, {uid: uid, gid : group\_id}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:1000});

// 通知被移除的用户，从其面板中将群组信息移除

var remove\_data = '{"type":"removeMember", "remove\_id": "' + ui d + '", "group\_id" : "' + group\_id + '"}';

socket.send(remove\_data);

setTimeout(function(){

getMembers($, laytpl, laypage, group\_id, page);

}, 1000);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json');

}, function(){

});

}

});

});

#### 从这段js，我们可以看到，貌似还有不少的逻辑可做。为什么我说只要从 v3\_groupdetail 删除该用户即可呢？

在实际的数据中删除该用户，该用户重新登录之后，就无法进入该群组了。但是，如果我们此次不将用户面 板中的 群组入口移除，以及不从 socket通信组中把用户移除的话，用户依旧是可以在该群组中说话和接收信 息的。因此，我们在这变看到了：

// 通知被移除的用户，从其面板中将群组信息移除

var remove\_data = '{"type":"removeMember", "remove\_id": "' + uid + '", "group\_id" : "'

+ group\_id + '"}'; socket.send(remove\_data);

向workermnan发送了一段 removeMember 的消息。当然，这个是回调做的事情，我们在此之前，也就是处

理 remove\_member\_url 这个接口的时候，应该讲用户从数据库中移除：

//移出成员出组

public function removeMember()

{

if(request()->isAjax()){

$uid = input('param.uid');

$groupId = input('param.gid');

();

$canNot = db('chatgroup')->field('owner\_id')->where('id=' . $groupId)->find

if(empty($canNot)){

return json(['code' => -1, 'data' => '', 'msg' => '异常操作']);

}

if($uid == $canNot['owner\_id']){

return json(['code' => -2, 'data' => '', 'msg' => '不可移除群主']);

}

//该群的管理员不是当前的操作人，则为非法操作 if($canNot['owner\_id'] != session('f\_user\_id')){

return json(['code' => -3, 'data' => '', 'msg' => '非法操作']);

}

$flag = db('groupdetail')->where('user\_id = ' . $uid . ' and group\_id = ' .

$groupId)->delete();

if(empty($flag)){

return json(['code' => -4, 'data' => '', 'msg' => '操作失败']);

}

return json( ['code' => 1, 'data' => '', 'msg' => '移除成功'] );

}

$this->error('非法访问');

}

#### 接下来，我们重点看看，这个socket消息，workerman该如何处理？接到这个消息之后，我们应该将该用户从

当前的 讨论组中移除，用到了 workerman的方法

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

这个很重要，然后通知被移除的成员客户端，删除该成员面板中的 群组入口。看看 Events.php 的具体实现：

case 'removeMember':

//将移除群组的成员的群信息移除，并从讨论组移除

$client\_id = Gateway::getClientIdByUid( $message['remove\_id'] ); if( !empty($client\_id) ){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

$del\_message = [

'message\_type' => 'delGroup', 'data' => [

'type' => 'group',

'id' => $message['group\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($del\_message));

}

return; break;

通知的消息 发出了 delGroup，同样的，我们 客户端的 main.js 需要对这个消息做出相应的处理，调用layim

提供的移除群信息的功能即可

// 删除面板的群组

case 'delGroup':

//console.log(data.data); layim.removeList({

type: 'group'

,id: data.data.id //群组ID

});

break;

**解散我的群组的实现**

解散我的群组的，整体的设计思路就是，将该群组的信息从数据库中移除，并通知所有的群成员，移除主面

板的群组入口。

//解散群组

function break\_up(){ layui.use(['layer'], function(){

var layer = layui.layer,

$ = layui.jquery;

var group\_id = $("#group\_id").val(); if('' == group\_id){

layer.alert('请先点击选择要解散的群组', {icon:2, title:'友情提示', closeBtn:0});

return ;

}

layer.ready(function(){

//询问框

var group\_name = $("#g\_name").text(); layer.confirm(group\_name, {

btn: ['确定','取消'],

title: '确定解散', closeBtn: 0,

icon: 3

}, function(){

var group\_id = $('#group\_id').val(); if('' == group\_id){

layer.msg('数据异常', {time:2000});

return ;

}

$.post(remove\_group\_url, {gid : group\_id}, function(res){ if(1 == res.code){

// 通知断开链接

var break\_data = '{"type":"breakUp", "uids":"' + res.data + '", "group\_id":"' + group\_id + '"}';

socket.send(break\_data);

layer.msg(res.msg, {time:1000});

$("#group\_id").val('');

$("#g\_name").text(); setTimeout(function(){

window.location.reload();

}, 1000);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json');

}, function(){

});

});

});

}

#### 这里我们解散群组的接口是：

var remove\_group\_url = "{:url('findgroup/removeGroup')}";

思路我们已经知道了，看一下 removeGroup 的具体实现：

//解散群组

public function removeGroup()

{

if(request()->isAjax()){

$groupId = input('param.gid');

//检测解散群组的人是否是该群组的管理员

$can = db('chatgroup')->field('id')->where('id = ' . $groupId . ' and owner

\_id = ' . session('f\_user\_id'))

->find();

if(empty($can)){

return json(['code' => -1, 'data' => '', 'msg' => '非法操作']);

}

//删除群组

$flag = db('chatgroup')->where('id', $groupId)->delete(); if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '解散群组失败']);

}

->select();

// 查出该群组的所有的用户id

$userIds = db('groupdetail')->field('user\_id')->where('group\_id', $groupId)

$uids = ''; foreach($userIds as $vo){

$uids .= $vo['user\_id'] . ',';

}

$uids = rtrim($uids, ','); unset($userIds);

//删除群成员

$flag = db('groupdetail')->where('group\_id', $groupId)->delete(); if(empty($flag)){

return json(['code' => -3, 'data' => '', 'msg' => '解散群成员失败']);

}

return json(['code' => 1, 'data' => $uids, 'msg' => '成功解散该群']);

}

$this->error('非法访问');

}

#### thinkphp5删除这一些列信息之后，返回了 $uids ，此后，js开始发送 消息给 workerman

// 通知断开链接

var break\_data = '{"type":"breakUp", "uids":"' + res.data + '", "group\_id":"' + group\_i d + '"}';

socket.send(break\_data);

向workerkman发送了需要通知的端口uid 数据为：uids，因此，workerman只需要逐一通知即可。

Events.php

case "breakUp":

// 解散群组

$uids = explode(',', $message['uids']); foreach($uids as $key=>$vo){

$client\_id = Gateway::getClientIdByUid($vo); if(!empty($client\_id)){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

$del\_message = [

'message\_type' => 'delGroup', 'data' => [

'type' => 'group',

'id' => $message['group\_id']

]

];

));

Gateway::sendToClient($client\_id['0'], json\_encode($del\_message

}

}

return; break;

#### 通知客户端的依旧是 delGroup 格式的数据，main.js 的具体处理逻辑 跟移除单个成员一样，就不做重复的解 释了。

# 群组聊天

#### 在群组中聊天的功能跟和好友聊天的功能很相似。layim通过同一个 js 方法监听，只是得到的数据格式不同。

**js部分**

//发送消息

layim.on('sendMessage', function(res){

//console.log(res); if(To.type === 'friend'){

layim.setChatStatus('<span style="color:#FF5722;">对方正在输入。。。</span

>');

}

// 发送消息

var mine = JSON.stringify(res.mine); var to = JSON.stringify(res.to);

var login\_data = '{"type":"chatMessage","data":{"mine":'+mine+', "to":'+to+

'}}';

socket.send( login\_data );

});

可见依旧是好友聊天的那部分 js，只不过区分在于 var mine = JSON.stringify(res.mine); var to =

JSON.stringify(res.to);这两部分的代码。

**workman的实现**

case 'chatMessage':

$db1 = Db::instance('db1'); //数据库链接

// 聊天消息

$type = $message['data']['to']['type'];

$to\_id = $message['data']['to']['id'];

$uid = $\_SESSION['id'];

']),

$chat\_message = [

'message\_type' => 'chatMessage', 'data' => [

'username' => $\_SESSION['username'], 'avatar' => $\_SESSION['avatar'],

'id' => $type === 'friend' ? $uid : $to\_id, 'type' => $type,

'content' => htmlspecialchars($message['data']['mine']['content

'timestamp' => time() \* 1000,

]

];

// 加入聊天log表

$param = [

'from\_id' => $uid, 'to\_id' => $to\_id,

'from\_name' => $\_SESSION['username'], 'from\_avatar' => $\_SESSION['avatar'],

'content' => htmlspecialchars($message['data']['mine']['content']), 'timeline' => time(),

'need\_send' => 0

];

switch ($type) {

// 私聊

case 'friend':

// 插入

$param['type'] = 'friend';

if (empty(Gateway::getClientIdByUid($to\_id))) {

$param['need\_send'] = 1; //用户不在线,标记此消息推送

}

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToUid($to\_id, json\_encode($chat\_message));

// 群聊

case 'group':

$param['type'] = 'group';

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToGroup($to\_id, json\_encode($chat\_message),

$client\_id);

}

return; break;

#### 你会发现和 前面的 好友聊天依旧一样的，只是当时，我并没有说

$type = $message['data']['to']['type'];

区别就在于此，根据这 $type 的不同，处理是群聊 还是 和好友聊天。

# 右键退出群组

#### layim只在主面板的会话历史中提供了右键功能，因此。想要在群组实现右键功能，需要自己去扩展源码：



其实仿照他之前写的代码，很容易扩展出这个功能。找到 layim.js 的源码，此段在可以搜索标题看到：

// 自定义主面板 群组右键功能 --- NickBai layim-list-group layimMain.find('.layim-list-group').on('contextmenu', 'li', function (e) {

var othis = $(this);

var html = '<ul data-id="' + othis[0].id + '">' + '<li layim-event="leaveOut">退出该群</li>';

if (othis.hasClass('layim-null')) return; layer.tips(html, this, {

tips: 1

, time: 0

, anim: 5

, fixed: true

, skin: 'layui-box layui-layim-contextmenu'

, success: function (layero) { var stopmp = function (e) {

stope(e);

}

});

};

layero.off('mousedown', stopmp).on('mousedown', stopmp);

$(document).off('mousedown', hide).on('mousedown', hide);

$(window).off('resize', hide).on('resize', hide);

});

#### 我给这个功能，埋了一个

layim-event="leaveOut"

事件，这样，我在外部，就可以调用这个事件去触发退群的操作。那么怎么判断我要退出的是哪个群呢？

data-id="' + othis[0].id + '"

这部分就会有效的显示群组的标识信息，我们只需要通过 dom 查找到该数据，便可以得知要操作哪个群组。

如何在外部触发这个 leaveOut 事件呢？这时，我们又该修改源码了。

//对外API

var LAYIM = function () { this.v = v;

$('body').on('click', '\*[layim-event]', function (e) {

var othis = $(this), methid = othis.attr('layim-event');

events = $.extend(events, my\_events || {}); //合并外部用户自定义的events events[methid] ? events[methid].call(this, othis, e) : '';

});

};

在对外的 api 这个中，我来合并一下我的事件

events = $.extend(events, my\_events || {}); //合并外部用户自定义的events

这样，我只需要在外部定义一个 my\_events 对象，即可调用layim的内部方法了。我的好友面板的右键功能，

都是在这个 事件中扩展出的。否则我们无法用到 layim的内部功能。

//外部自定义我的事件 my\_events = {

//退出群组

, leaveOut: function(othis, e){

var group\_id = othis.parent().attr('data-id'); var index = layer.confirm('确定退出该群组？', {

btn: ['确定', '取消'], title: '友情提示', closeBtn: 0,

icon: 3

}, function(){

$.post('/index/Tools/leaveGroup', {'group\_id' : group\_id}, function(res

){

if(1 == res.code){ layer.msg(res.msg, {time: 1500}); layim.removeList({

type: 'group'

, id: res.data.group\_id

});

// 退出讨论组

var leave\_data = '{"type": "leaveGroup", "leave\_id":"' + res.da ta.uid + '", "group\_id":"' + res.data.group\_id + '"}';

socket.send(leave\_data);

}else{

layer.msg(res.msg, {time: 1500});

}

layer.close(index);

}, 'json');

}, function(){

});

}

}

#### 从接口我们可以看到，用户点击 退出群组的时候，会向 /index/Tools/leaveGroup 发送请求，其实这理的处理

逻辑很简单，只需要 将该用户从 v3\_groupdetail 表中删除即可。我们在 application\index\controller 下新建 一个 Tools.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Tools extends Base

{

//离开群组

public function leaveGroup()

{

if(request()->isAjax()){

$delStr = 'layim-group';

$groupId = input('group\_id');

$groupId = substr($groupId, strlen($delStr), strlen($groupId));

$me = session('f\_user\_id');

//检测是不是管理员要退出该群组

$group = db('chatgroup')->field('owner\_id')->where('id', $groupId)->find(); if($me == $group['owner\_id']){

return json(['code' => -2, 'data' => '', 'msg' => '管理员不可以直接退出'])

;

}

$flag = db('groupdetail')->where('user\_id=' . $me . ' and group\_id=' . $gro upId)->delete();

if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '系统错误']);

}

$return = [

'uid' => $me, 'group\_id' => $groupId

];

return json(['code' => 1, 'data' => $return, 'msg' => '退出成功']);

}

$this->error('非法访问');

}

}

#### 完成了 用户从 数据库中物理删除之后，向 workerman发送一个消息。

// 退出讨论组

var leave\_data = '{"type": "leaveGroup", "leave\_id":"' + res.data.uid + '", "group\_id": "' + res.data.group\_id + '"}';

socket.send(leave\_data);

其实这个消息的目的，就是通知workerman的主要目的就是，让workerman讲我从这个群组中移除。在此之

前，js已经调用了接口移除了群聊入口面板

layim.removeList({ type: 'group'

, id: res.data.group\_id

});

Events.php

case 'leaveGroup':

// 退出群组

$client\_id = Gateway::getClientIdByUid($message['leave\_id']); if( !empty($client\_id) ){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

}

return; break;

# 发送语音

#### web端的语音解决方案不是很理想，本系统加入的发送语音，仅可以当尝鲜用吧。在第一次启动录音的时候， 会出现一点问题，后面则比较稳定。录音，采用的是 RecordRTC 这个插件，感兴趣的可以先去了解一下。其 实主要看我封装好的 audio.js 也可以，因为这个插件官方的例子，以及可以找到的好多例子，都是录音和视 频同时开启的，如下的配置，是我通过琢磨，最后总结出的配置：

const startBtn = document.getElementById('say'); const stopBtn = document.getElementById('over—say');

let recordRTC;

// 开始录制

const startRecording = () => { startBtn.disabled = true; navigator.getUserMedia({

audio: true, video : false

}, (stream) => {

recordRTC = RecordRTC(stream, { bufferSize: 16384, type : 'audio' }); recordRTC.startRecording();

stopBtn.disabled = false;

}, (error) => {

alert(JSON.stringify(error)); console.error(error);

});

};

// 结束录制

const stopRecording = () => { startBtn.disabled = false; stopBtn.disabled = true; recordRTC.stopRecording(() => {

const file = new File([recordRTC.getBlob()], `${Date.now()}.ogg`);

//console.log(file);

uploadToServer(recordRTC, function(progress, fileURL){ if(progress === 'ended') {

return;

}

});

};

});

//a File object can be easily uploaded to any server

//https://developer.mozilla.org/en-US/docs/Using\_files\_from\_web\_applications

startBtn.onclick = startRecording; stopBtn.onclick = stopRecording;

//上传到服务器

function uploadToServer(recordRTC, callback) {

var blob = recordRTC instanceof Blob ? recordRTC : recordRTC.blob; var fileType = blob.type.split('/')[0] || 'audio';

var fileName = (Math.random() \* 1000).toString().replace('.', '');

if (fileType === 'audio') {

fileName += '.' + (!!navigator.mozGetUserMedia ? 'ogg' : 'wav');

} else {

fileName += '.webm';

}

// create FormData

var formData = new FormData(); formData.append(fileType + '-filename', fileName); formData.append(fileType + '-blob', blob);

callback('Uploading ' + fileType + ' recording to server.'); makeXMLHttpRequest(save\_audio\_url, formData, function(progress) {

if (progress !== 'upload-ended') {

callback(progress); return;

}

});

}

//异步上传

function makeXMLHttpRequest(url, data, callback) { var request = new XMLHttpRequest(); request.onreadystatechange = function() {

if (request.readyState == 4 && request.status == 200) { var reObj = eval('(' + request.response + ')');

$("#audio\_src").val(reObj.data); callback('upload-ended');

}

};

request.upload.onloadstart = function() { callback('Upload started...');

};

");

request.upload.onprogress = function(event) {

callback('Upload Progress ' + Math.round(event.loaded / event.total \* 100) + "%

};

request.upload.onload = function() { callback('progress-about-to-end');

};

request.upload.onload = function() { callback('progress-ended');

};

request.upload.onerror = function(error) { callback('Failed to upload to server'); console.error('XMLHttpRequest failed', error);

};

request.upload.onabort = function(error) { callback('Upload aborted.'); console.error('XMLHttpRequest aborted', error);

};

request.open('POST', url); request.send(data);

}

#### 在页面中放一个隐藏域，来放置录音文件的地址：

<input type="hidden" id="audio\_src"/>

我们看到，录制完成的语音，我在这里让他上传到指定的位置：

var save\_audio\_url = "{:url('tools/saveAudio')}";

看看我的录音代码中怎么处理的吧

makeXMLHttpRequest(save\_audio\_url, formData, function(progress) { if (progress !== 'upload-ended') {

callback(progress); return;

}

});

而saveAudio方法，只需要将这段语音保存起来，并把存储的地址返回回来。这样就可以在聊天面板中调取这

个录音，发给好友。这个地址就是放在

<input type="hidden" id="audio\_src"/>

这也是我设计这个隐藏域的用意所在，方便调用录音地址。

//保存录音

public function saveAudio()

{

$file = request()->file('audio-blob');

$info = $file->getInfo();

$saveName = md5($info['tmp\_name'] . time()) . '.ogg';

$savePath = ROOT\_PATH . 'public' . DS . 'uploads' . DS . date('Ymd') . DS; if(!is\_dir($savePath)){

mkdir($savePath); chmod($savePath, 0666);

}

// 移动到框架应用根目录/public/uploads/ 目录下

if(move\_uploaded\_file($info['tmp\_name'], $savePath . $saveName)){

$src =

return

}else{

return

}

DS . 'uploads' . DS . date('Ymd') . DS . $saveName;

json(['code' => 1, 'data' => $src, 'msg' => '']);

json(['code' => -1, 'data' => '', 'msg' => '失败']);

}

在聊天面板中扩展出一个录音按钮

当时做这个系统的时候，layim版本是 3.0.1，layui也没有提供比较切合的语音图标，目前最新的版本有提 供这个图标，感兴趣的可以替换一下。



看一下，如何在聊天面板中，扩展功能。

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

}

]

我们在layim的基础配置中，能看到这么一个配置项，这里就是扩展功能的按钮。如何实现这部分功能呢？

layim给我们提供了监听这部分功能你的接口：

//监听自定义工具栏点击录音

layim.on('tool(voice)', function(insert, send){

layui.use(['layer'], function(){ var layer = layui.layer;

var box = layer.open({ type: 1,

title: '发送语音',

maxmin: false, //开启最大化最小化按钮 skin: 'layui-layer-molv',

anim: 3,

area: ['200px', '250px'], content: $("#audio\_box"), cancel: function(index){

$("#tips").text('');

$("#a\_pic").attr('src', '/static/common/images/audio.png'); layer.close(index);

return false;

}

});

//点击开始录音

$("#say").bind('click', function(){

$("#tips").text('说话中......');

$("#a\_pic").attr('src', '/static/common/images/audio.gif');

$("#over—say").removeClass('layui-btn-disabled');

$(this).addClass('layui-btn-disabled');

});

var isSend = false;

//结束录音

$("#over—say").bind('click', function(){ if(!isSend){

$("#tips").text('');

$("#a\_pic").attr('src', '/static/common/images/audio.png');

$("#say").removeClass('layui-btn-disabled');

$(this).addClass('layui-btn-disabled'); layer.close(box);

var index = layer.load(1, {

shade: [0.1,'#fff'] //0.1透明度的白色背景

});

setTimeout(function(){

var audioSrc = $("#audio\_src").val(); if('' == audioSrc){

layer.msg('您没有发送语音', {time:1000});

}else{

insert('audio[' + audioSrc + ']'); send(); //自动发送

});

}

});

}

layer.close(index);

}, 1500);

isSend = true;

});

#### 这里做一个 延时处理

setTimeout(function(){

var audioSrc = $("#audio\_src").val(); if('' == audioSrc){

layer.msg('您没有发送语音', {time:1000});

}else{

insert('audio[' + audioSrc + ']'); send(); //自动发送

}

layer.close(index);

}, 1500);

是因为，录音文件上传到tp5，并保存下来，这是一个异步实现的过程，和 直接调用layim发送方法

insert('audio[' + audioSrc + ']'); send(); //自动发送

之间存在矛盾，如果不做延迟处理，总会在点击停止录音之后，立马会从 audioSrc 中取出语音地址，而这个

语音地址并没有被写入，从而导致发送失败

# 发送图片

#### 发送图片的功能，layim给了比较好的接口，我们只要实现服务器逻辑部分即可。js部分的接口：

,uploadImage: {

url: upload\_img\_url

}

而这个 upload\_img\_url 你可以根据自己的需求去定义，我定义的处理地址是：

var upload\_img\_url = "{:url('upload/uploadimg')}";

我们需要在thinkphp5中的这个指定的方法，把这个图片文件保存起来，并把保存地址返回给js即可完成图片

发送的的问题。发送的图片，是以流的形式发送的，只需要用tp5封装好的文件上传函数即可，接收的名字叫 file。

$file = request()->file('file');

在 application\index\controller 下新建 Upload.php

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Upload extends Base

{

//上传图片

public function uploadImg()

{

$config = readCtConfig(); //读取系统配置的上传文件配置

$file = request()->file('file');

$fileInfo = $file->getInfo();

if($fileInfo['size'] > 1024\*1024\*$config['img\_size']){

// 上传失败获取错误信息

return json( ['code' => -2, 'data' => '', 'msg' => '文件超过' . $config['img

\_size'] . 'M'] );

}

//检测图片格式

$ext = explode('.', $fileInfo['name']);

$ext = array\_pop($ext);

$extArr = explode('|', $config['img\_ext']); if(!in\_array($ext, $extArr)){

return json(['code' => -3, 'data' => '', 'msg' => '只能上传' . $config['img\_ ext'] . '的文件']);

}

// 移动到框架应用根目录/public/uploads/ 目录下

$info = $file->move(ROOT\_PATH . 'public' . DS . 'uploads'); if($info){

$src = '/uploads' . '/' . date('Ymd') . '/' . $info->getFilename();

return json(['code' => 0, 'data' => ['src' => $src ], 'msg' => '']);

}else{

// 上传失败获取错误信息

return json(['code' => -1, 'data' => '', 'msg' => $file->getError()]);

}

}

}

#### 这里由于我在后台做了用户上传图片的限制，因此比较多的过程消耗在对图片的校验上。

# 发送文件

#### 发送文件和发送图片的处理方式几乎相同。layim提供了上传文件的接口：

,uploadFile: {

url: upload\_file\_url

}

你只需要给他规定一个 地址：

var upload\_file\_url = "{:url('upload/uploadFile')}";

然后和上传图片一样的 thinkphp5 处理方式即可：

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Upload extends Base

{

//上传文件

public function uploadFile()

{

$config = readCtConfig(); //读取系统配置的上传文件配置

$file = request()->file('file');

$fileInfo = $file->getInfo();

if($fileInfo['size'] > 1024\*1024\*$config['file\_size']){

// 上传失败获取错误信息

return json( ['code' => -2, 'data' => '', 'msg' => '文件超过' . $config['fil e\_size'] . 'M'] );

}

//检测文件格式

$ext = explode('.', $fileInfo['name']);

$ext = array\_pop($ext);

$extArr = explode('|', $config['file\_ext']); if(!in\_array($ext, $extArr)){

return json(['code' => -3, 'data' => '', 'msg' => '只能上传' . $config['file

\_ext'] . '的文件']);

}

// 移动到框架应用根目录/public/uploads/ 目录下

$info = $file->move(ROOT\_PATH . 'public' . DS . 'uploads'); if($info){

$src = '/uploads' . '/' . date('Ymd') . '/' . $info->getFilename();

return json(['code' => 0, 'data' => ['src' => $src ], 'msg' => '']);

}else{

// 上传失败获取错误信息

return json(['code' => -1, 'data' => '', 'msg' => $file->getError()]);

}

}

}

# 用户空间功能

#### 由于我实在是找不到合适的空间入口，就把他之前的版权出信息，修改一下：



看一下源码的位置

//关于

, about: function () {

/\*layer.alert('版本： '+ v + '<br>版权所有：<a href="[http://layim.layui.com](http://layim.layui.com/)" t arget="\_blank">layim.layui.com</a>', {

title: '关于 LayIM'

,shade: false

});\*/

layer.alert('<a href="/index/userzone/index" target="\_blank">点击，进入我的空

间</a>', {

title: false

, icon: 6

, btn: ''

});

}

由于直接通过 js 打开新的页面会被浏览器阻拦，因此我在这边，设计了一个用户自己点击 进入我的空间 的操

作。进入空间的接口也一目了然，/index/userzone/index，因此，我们需要在 application\index\controller 新建一个 Userzone.php 。

**空间需求**

在这里设计一个简化版的空间，本空间实现以下的功能：

1. 发表说说

2. 查看我以及我的好友发表的说说。

3. 对这些说说进行评论

发表说说的入口，放在空间内，用户点击发表按钮，弹出发表框。我们这里需要设计一个 存放说说的表， v3\_mblog

DROP TABLE IF EXISTS `v3\_mblog`; CREATE TABLE `v3\_mblog` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '说说的id',

`post\_user` varchar(155) NOT NULL COMMENT '发表人名',

`post\_uid` int(11) NOT NULL,

`post\_avatar` varchar(255) NOT NULL COMMENT '发表人的头像',

`content` text NOT NULL COMMENT '发表内容',

`post\_time` int(10) NOT NULL COMMENT '发表时间', PRIMARY KEY (`id`),

KEY `post\_uid` (`post\_uid`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

以及一个用户评论表，v3\_comment

DROP TABLE IF EXISTS `v3\_comment`; CREATE TABLE `v3\_comment` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '评论的id',

`blog\_id` int(11) NOT NULL COMMENT '关联的说说id',

`com\_user` varchar(155) NOT NULL COMMENT '评论人名',

`com\_uid` int(11) NOT NULL COMMENT '评论人用户id',

`com\_avatar` varchar(255) NOT NULL COMMENT '评论人头像',

`content` text NOT NULL COMMENT '评论内容',

`com\_time` int(10) NOT NULL COMMENT '评论时间', PRIMARY KEY (`id`),

KEY `blog\_id` (`blog\_id`) USING BTREE, KEY `com\_uid` (`com\_uid`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

好了，准备完毕了，我们开始吧。

**展示说说的实现**

首先用户点击进入我的空间，即访问 userzone/index 的index方法，此时我们只需要从 v3\_friends 表中将用

户的好友找出，并从v3\_mblog中将这些好友的说说查询出来渲染给用户看即可。考虑到数据量可能会很大， 我们这里采用分页。评论部分，我是在页面渲染的时候，再去查询这些评论。通过的是 getComment 方法。 这个方法定义在：application\common.php

//获取评论

function getComment($blogId)

{

$list = db('comment')->where('blog\_id', $blogId)->select(); if(empty($list)){

echo "";

}else{

$html = '';

foreach($list as $key=>$vo){

$html .= '<a href="javascript:;" class="pull-left"><img alt="image" src="'

. $vo['com\_avatar'] . '"></a>';

$html .= '<div class="media-body"><a href="javascript:;" style="color:#337A B7">' . $vo['com\_user'];

$html .= '&nbsp;&nbsp;&nbsp;&nbsp;</a>' . $vo['content'] . '<br/>';

$html .= '<small class="text-muted">' . date('Y-m-d H:i', $vo['com\_time'])

. '</small></div>';

}

echo $html;

}

}

#### 我们来主要看一下，渲染说说的主方法。

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Userzone extends Base

{

//指定用户的空间首页

//进入我的空间，则显示所有的好友的更新的状态,以及相关的评论信息

//进入某人的空间，则集中显示这个人的最近更新的内容以及相关评论 public function index()

{

//查询自己，以及好友发布的说说

$uid = session('f\_user\_id');

$friends = db('friends')->field('friend\_id')->where('user\_id', $uid)->select();

$uids = ''; if(!empty($friends)){

foreach($friends as $key=>$vo){

$uids .= $vo['friend\_id'] . ',';

}

}

c')

$uids .= $uid;

$blogs = db('mblog')->where('post\_uid in(' . $uids . ')')->order('post\_time des

->paginate(config('blog\_page'));

$this->assign([

'user' => session('f\_user\_name'), 'blogs' => $blogs

]);

return $this->fetch();

}

}

#### 我们再在 application\index\view 下面新建一个 userzone 文件夹并在其下新建 index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>laychat-v3.0 - 空间</title>

<link rel="shortcut icon" href="favicon.ico">

<link href=" COMMON /css/bootstrap.min.css?v=3.3.6" rel="stylesheet">

<link href=" COMMON /css/font-awesome.min.css?v=4.4.0" rel="stylesheet">

<link href=" COMMON /css/animate.min.css" rel="stylesheet">

<link href=" COMMON /css/style.min.css?v=4.1.0" rel="stylesheet">

<link href=" COMMON /css/tool.css" rel="stylesheet">

<link href=" COMMON /layui/css/layui.css" rel="stylesheet">

<style>

.social-body{padding-left:50px;padding-right:50px;line-height:160%;}

.media-body{padding-left:20px;padding-right:20px;line-height:160%;}

.social-footer .media-body img{width: 22px;height: 22px}

</style>

</head>

<body class="gray-bg" style="background:#F7F7F7 url(' COMMON /images/zone.jpg') repea t-x;">

<div class="wrapper wrapper-content animated fadeInRight">

<div class="row">

<div class="col-sm-3"></div>

<div class="col-sm-6">

<div class="ibox">

<div class="ibox-content text-center">

<h3 class="m-b-xxs">{$user}的空间</h3>

</div>

</div>

<div id="blog\_flow">

<!-- 动态模板 -->

{if !empty('blogs')}

{foreach name="blogs" item="vo"}

<div class="social-feed-separated">

<div class="social-avatar">

<a href="">

<img alt="image" src="{$vo.post\_avatar}">

</a>

</div>

<div class="social-feed-box">

<div class="social-avatar">

<a href="javascript:;" style="color:#337AB7">

{if condition="$vo['post\_user'] eq $user"}我{else/}{$vo

.post\_user}{/if}

###}</small>

</a>

<small class="text-muted">{$vo.post\_time|date="Y-m-d H:i",

</div>

<div class="social-body">

<p>{$vo.content}</p>

<div class="btn-group">

<button class="btn btn-white btn-xs" data-id="{$vo.id}"

onclick="comment(this)">

<i class="fa fa-comments"></i> 评论

</button>

</div>

</div>

<div class="social-footer">

<div class="social-comment">

{$vo.id|getComment=###}

</div>

</div>

</div>

</div>

{/foreach}

{/if}

</div>

<!-- 动态模板 -->

<div id="page" style="float: right">

{$blogs->render()}

</div>

</div>

</div>

</div>

<!-- 漂浮工具条 -->

<div class="floating\_ck">

<dl>

<dt></dt>

<dd class="consult">

<span>发表说说</span>

</dd>

<dd class="return">

<span onClick="gotoTop();return false;">返回顶部</span>

</dd>

</dl>

</div>

<!-- 发表说说的box -->

<div id="blog\_box" style="display: none;padding:15px">

<textarea class="layui-textarea" id="LAY\_demo1" style="display: none">

</textarea>

<div class="site-demo-button" style="margin-top: 20px;float:right">

<button class="layui-btn site-demo-layedit" data-type="content">发表</button>

</div>

</div>

<!-- 评论box -->

<div id="comment\_box" style="display: none;padding:15px">

<textarea class="layui-textarea" id="LAY\_demo2" style="display: none">

</textarea>

<div class="site-demo-button" style="margin-top: 20px;float:right">

<button class="layui-btn site-demo-layedit" data-type="comment" id="c\_btn">发表<

/button>

</div>

</div>

<script src=" COMMON /jquery.min.js?v=2.1.4"></script>

<script src=" COMMON /bootstrap.min.js?v=3.3.6"></script>

<script src=" COMMON /js/top.js"></script>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var img\_url = "{:url('userzone/upImg')}";

var post\_tips\_url = "{:url('userzone/postTips')}";

var post\_comment\_url = "{:url('userzone/postComment')}";

</script>

<script type="text/javascript" src=" COMMON /js/zones.js"></script>

</body>

</html>

#### 最终效果如下：

展示这些说话只是查询的事情，很容易就可以做到。

**发表说说的实现**

发表说说的实现逻辑就是，用户点击右边的 发表说说按钮，然后在弹出的对话框中输入要发表的说说内容即

可。用户发表的说说内容加入v3\_mblog中，就是一个插入的数据的操作。没什么好说的，直接看处理代码 吧：

//发表说说

public function postTips()

{

if(request()->isAjax()){

$content = input('post.content'); if(empty($content)){

return json(['code' => -1, 'data' => '', 'msg' => '发表说说不能为空']);

}

$param = [

'post\_user' => session('f\_user\_name'), 'post\_uid' => session('f\_user\_id'), 'post\_avatar' => session('f\_user\_avatar'), 'content' => $content,

'post\_time' => time()

];

$flag = db('mblog')->insert($param); if(empty($flag)){

return json(['code' => -3, 'data' => '', 'msg' => '系统错误']);

}

return json(['code' => 1, 'data' => '', 'msg' => '发表成功']);

}

$this->error('非法访问');

}

//上传说说图片

public function upImg()

{

// 获取表单上传文件

$file = request()->file('file');

// 移动到框架应用根目录/public/uploads/ 目录下

if(!is\_null($file)){

$fileInfo = $file->getInfo();

$imgExt = config('common\_img');

$ext = explode('.', $fileInfo['name']);

$ext = array\_pop($ext);

if(!in\_array($ext, $imgExt)){

return json(['code' => -2, 'data' => '', 'msg' => '请上传' . implode(','

, $imgExt) . '的图片']);

}

unset($ext);

$size = config('common\_size'); if($fileInfo['size'] > $size){

return json(['code' => -3, 'data' => '', 'msg' => '上传的图片超过' . $siz

e/1024 . 'kb']);

}

unset($fileInfo);

$info = $file->move(ROOT\_PATH . 'public' . DS . 'uploads'); if($info){

// 成功上传后 获取上传信息

$avatar = '/uploads' .

return json(['code' =>

}else{

// 上传失败获取错误信息

return json(['code' =>

}

'/' . date('Ymd') . '/' . $info->getFilename();

0, 'data'

=> ['src' => $avatar], 'msg' => '']);

-4, 'url'

=> '', 'msg' => $file->getError()]);

}

return json(['code' => -1, 'data' => '', 'msg' => '上传图片失败']);

}

**发表评论的实现**

#### 用户点击说说下面的评论按钮，即弹出评论弹出层，输入评论内容即可，入 v3\_comment 库。

//发表评论

public function postComment()

{

if(request()->isAjax()){

$blogId = input('post.blog\_id');

$content = input('post.content'); if(empty($content)){

return json(['code' => -1, 'data' => '', 'msg' => '发表评论不能为空']);

}

$param = [

'blog\_id' => $blogId,

'com\_user' => session('f\_user\_name'), 'com\_uid' => session('f\_user\_id'), 'com\_avatar' => session('f\_user\_avatar'), 'content' => $content,

'com\_time' => time()

];

$flag = db('comment')->insert($param); if(empty($flag)){

return json(['code' => -3, 'data' => '', 'msg' => '系统错误']);

}

return json(['code' => 1, 'data' => '', 'msg' => '评论成功']);

}

$this->error('非法访问');

}

#### 没多大的难度，此处不做过多的解释了。

# 查看聊天记录

#### 查看聊天记录这个接口，layim也提供了很好的接口。这就是layim的强大之处。看一下这个接口怎么配置：

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

看一下我门 chatlog\_url 指向哪里：

var chatlog\_url = "{:url('Chatlog/index')}";

我们在 application\index\controller 下新建一个 Chatlog.php 来处理这部分的逻辑。页面，layim给了我们一

个demo页，我们将其移植到 application\index\view\chatlog\ index.html 也就是我们看到的 聊天记录主页 面。

**控制器的实现**

<?php

// +----------------------------------------------------------------------

// | laychat-v3.0

// +----------------------------------------------------------------------

// | Author: NickBai <[1902822973@qq.com](mailto:1902822973@qq.com)>

// +----------------------------------------------------------------------

namespace app\index\controller;

class Chatlog extends Base

{

//聊天记录

public function index()

{

$this->assign([

'perPage' => config('log\_page')

]);

return $this->fetch();

}

}

这种配置项，config('log\_page')，我就不一 一列举了，我还在源码中给出。

**完善渲染模板**

我们将layim给我们的chatlog模板拿过来，进行完善。application\index\view\chatlog\ index.html

<!DOCTYPE

<html>

<head>

<meta

<meta

">

html>

charset="utf-8">

name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

<title>聊天记录</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

<style>

body .layim-chat-main{height: auto;}

</style>

</head>

<body>

<div class="layim-chat-main">

<ul id="LAY\_view"></ul>

</div>

<div id="LAY\_page" style="margin: 0 10px;"></div>

<textarea title="消息模版" id="LAY\_tpl" style="display:none;">

{{# layui.each(d.data.data, function(index, item){ if(item.id == parent.layui.layim.cache().mine.id){ }}

<li class="layim-chat-mine">

<div class="layim-chat-user">

<img src="{{ item.avatar }}">

<cite>

<i>{{ layui.data.date(item.timestamp\*1000) }}</i>

{{ item.username }}

</cite>

</div>

<div class="layim-chat-text">{{ layui.layim.content(item.content) }}</div>

</li>

{{# } else { }}

<li>

<div class="layim-chat-user">

<img src="{{ item.avatar }}">

<cite>

{{ item.username }}

<i>{{ layui.data.date(item.timestamp\*1000) }}</i>

</cite>

</div>

<div class="layim-chat-text">{{ layui.layim.content(item.content) }}</div>

</li>

{{# }

}); }}

</textarea>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var chat\_log\_url = "{:url('chatlog/detail')}"; var per\_page = "{$perPage}";

</script>

<script type="text/javascript" src=" COMMON /js/chatlog.js"></script>

</body>

</html>

#### 通过一步去获取聊天记录，我们只要讲这个聊天记录按照要求返回即可。

var chat\_log\_url = "{:url('chatlog/detail')}";

请求的是详情接口，我们在聊天详情中查询出这些数据。

**聊天记录详情**

//聊天记录详情

public function detail()

{

if(request()->isAjax()) {

$perPage = config('log\_page');

$id = input('id');

$type = input('type');

$flag = input('flag'); //此处为标识是否获取总数

$uid = session('f\_user\_id');

$field = 'from\_name username,from\_id id,from\_avatar avatar,timeline timesta mp,content';

if('friend' == $type) {

$where = "((from\_id={$uid} and to\_id={$id}) or (from\_id={$id} and to\_id

={$uid})) and type='friend'";

if(!empty($flag)){

$result = db('chatlog')->field('id')->where($where)->count();

}else{

$result = db('chatlog')->field($field)

->where($where)->order('timeline desc')->paginate($perPage);

}

if(empty($result)) {

return json(['code' => -1, 'data' => '', 'msg' => '没有记录']);

}

return json(['code' => 1, 'data' => $result, 'msg' => 'success']);

} else if('group' == $type) { if(!empty($flag)){

$result = db('chatlog')->field('id')->where("to\_id={$id} and type='

group'")->count();

}else{

$result = db('chatlog')->field($field)->where("to\_id={$id} and type

='group'")->order('timeline desc')

->paginate($perPage);

}

if(empty($result)) {

return json(['code' => -1, 'data' => '', 'msg' => '没有记录']);

}

return json(['code' => 1, 'data' => $result, 'msg' => 'success']);

}

}

$this->error('非法访问');

}

#### 只需要将我们的 chatlog 表中符合需求的数据查出，即可完成。



# 进入好友空间

#### 进入好友空间，是通过在好友的头像上通过右键弹出页面进入。

这里我通过扩展 原有的功能。简化了这部分的操作逻辑，添加了好几项功能。看一下我怎么扩展的，大体展 示一下，我扩展的可能不是很好，js不是很懂。

//自定主面板联系人右键功能 --- NickBai,修改了主模板渲染 添加了 layui-friend 样式名

layimMain.find('.layui-friend').on('contextmenu', 'li', function (e) { var othis = $(this);

var html = '<ul data-id="' + othis[0].id + '" data-index="' + othis.data('i ndex') + '">' +

入黑名单</li>' +

'<li layim-event="enterZone">进入空间</li><li layim-event="joinBlack">加

'<li layim-event="changeGroup">移动分组</li><li layim-event="removeFrien

d">删除该好友</li>' +

'<li layim-event="reportFriend">举报</li></ul>';

if (othis.hasClass('layim-null')) return; layer.tips(html, this, {

tips: 1

, time: 0

, anim: 5

, fixed: true

, skin: 'layui-box layui-layim-contextmenu'

, success: function (layero) { var stopmp = function (e) {

stope(e);

}

});

};

layero.off('mousedown', stopmp).on('mousedown', stopmp);

$(document).off('mousedown', hide).on('mousedown', hide);

$(window).off('resize', hide).on('resize', hide);

});

#### 这里进入qq空间，我定义的是 enterZone 事件，在main.js 中，监听这个事件

//外部自定义我的事件 my\_events = {

//进入好友的空间

enterZone: function (othis, e) {

var friend\_id = othis.parent().attr('data-id');

layer.alert('<a href="/index/Userzone/uZone/fid/' + friend\_id + '" target= "\_blank">点击，进入他的空间</a>', {

title: false

, icon: 6

, btn: ''

});

}

}

只是跟 进入我的空间，多传入了一个 fid 即好友的id，此时我们在 Userzone 的 uZone 方法，查询出这个好

友的全部说说，展示即可。

//进入指定好友的空间 public function uZone()

{

$delStr = 'layim-friend';

$friendId = input('fid');

$friendId = substr($friendId, strlen($delStr), strlen($friendId));

//查询属于该用户的说说信息

$blogs = db('mblog')->where('post\_uid', $friendId)->order('post\_time desc')->pa ginate(config('blog\_page'));

$this->assign([

'user' => db('chatuser')->field('user\_name')->where('id', $friendId)->find(

),

'blogs' => $blogs

]);

return $this->fetch();

}

在 application\index\view\userzone 下新建 uzone.html 展示出这些信息：

<!DOCTYPE

<html>

<head>

<meta

<meta

html>

charset="utf-8">

name="viewport" content="width=device-width, initial-scale=1.0">

<title>laychat-v3.0 - 空间</title>

<link rel="shortcut icon" href="favicon.ico">

<link href=" COMMON /css/bootstrap.min.css?v=3.3.6" rel="stylesheet">

<link href=" COMMON /css/font-awesome.min.css?v=4.4.0" rel="stylesheet">

<link href=" COMMON /css/animate.min.css" rel="stylesheet">

<link href=" COMMON /css/style.min.css?v=4.1.0" rel="stylesheet">

<link href=" COMMON /css/tool.css" rel="stylesheet">

<link href=" COMMON /layui/css/layui.css" rel="stylesheet">

<style>

.social-body{padding-left:50px;padding-right:50px;line-height:160%;}

.media-body{padding-left:20px;padding-right:20px;line-height:160%;}

.social-footer .media-body img{width: 22px;height: 22px}

</style>

</head>

<body class="gray-bg" style="background:#F7F7F7 url(' COMMON /images/zone.jpg') repea t-x;">

<div class="wrapper wrapper-content animated fadeInRight">

<div class="row">

<div class="col-sm-3"></div>

<div class="col-sm-6">

<div class="ibox">

<div class="ibox-content text-center">

<h3 class="m-b-xxs">{$user.user\_name}的空间</h3>

</div>

</div>

<div id="blog\_flow">

<!-- 动态模板 -->

{if !empty('blogs')}

{foreach name="blogs" item="vo"}

<div class="social-feed-separated">

<div class="social-avatar">

<a href="">

<img alt="image" src="{$vo.post\_avatar}">

</a>

</div>

<div class="social-feed-box">

<div class="social-avatar">

<a href="javascript:;" style="color:#337AB7">

{if condition="$vo['post\_user'] eq $user"}我{else/}{$vo

.post\_user}{/if}

###}</small>

</a>

<small class="text-muted">{$vo.post\_time|date="Y-m-d H:i",

</div>

<div class="social-body">

<p>{$vo.content}</p>

<div class="btn-group">

<button class="btn btn-white btn-xs" data-id="{$vo.id}"

onclick="comment(this)">

<i class="fa fa-comments"></i> 评论

</button>

</div>

</div>

<div class="social-footer">

<div class="social-comment">

{$vo.id|getComment=###}

</div>

</div>

</div>

</div>

{/foreach}

{/if}

</div>

<!-- 动态模板 -->

<div id="page" style="float: right">

{$blogs->render()}

</div>

</div>

</div>

</div>

<!-- 漂浮工具条 -->

<div class="floating\_ck">

<dl>

<dt></dt>

<dd class="return">

<span onClick="gotoTop();return false;">返回顶部</span>

</dd>

</dl>

</div>

<!-- 发表说说的box -->

<div id="blog\_box" style="display: none;padding:15px">

<textarea class="layui-textarea" id="LAY\_demo1" style="display: none">

</textarea>

<div class="site-demo-button" style="margin-top: 20px;float:right">

<button class="layui-btn site-demo-layedit" data-type="content">发表</button>

</div>

</div>

<!-- 评论box -->

<div id="comment\_box" style="display: none;padding:15px">

<textarea class="layui-textarea" id="LAY\_demo2" style="display: none">

</textarea>

<div class="site-demo-button" style="margin-top: 20px;float:right">

<button class="layui-btn site-demo-layedit" data-type="comment" id="c\_btn">发表<

/button>

</div>

</div>

<script src=" COMMON /jquery.min.js?v=2.1.4"></script>

<script src=" COMMON /bootstrap.min.js?v=3.3.6"></script>

<script src=" COMMON /js/top.js"></script>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var img\_url = "{:url('userzone/upImg')}";

var post\_tips\_url = "{:url('userzone/postTips')}";

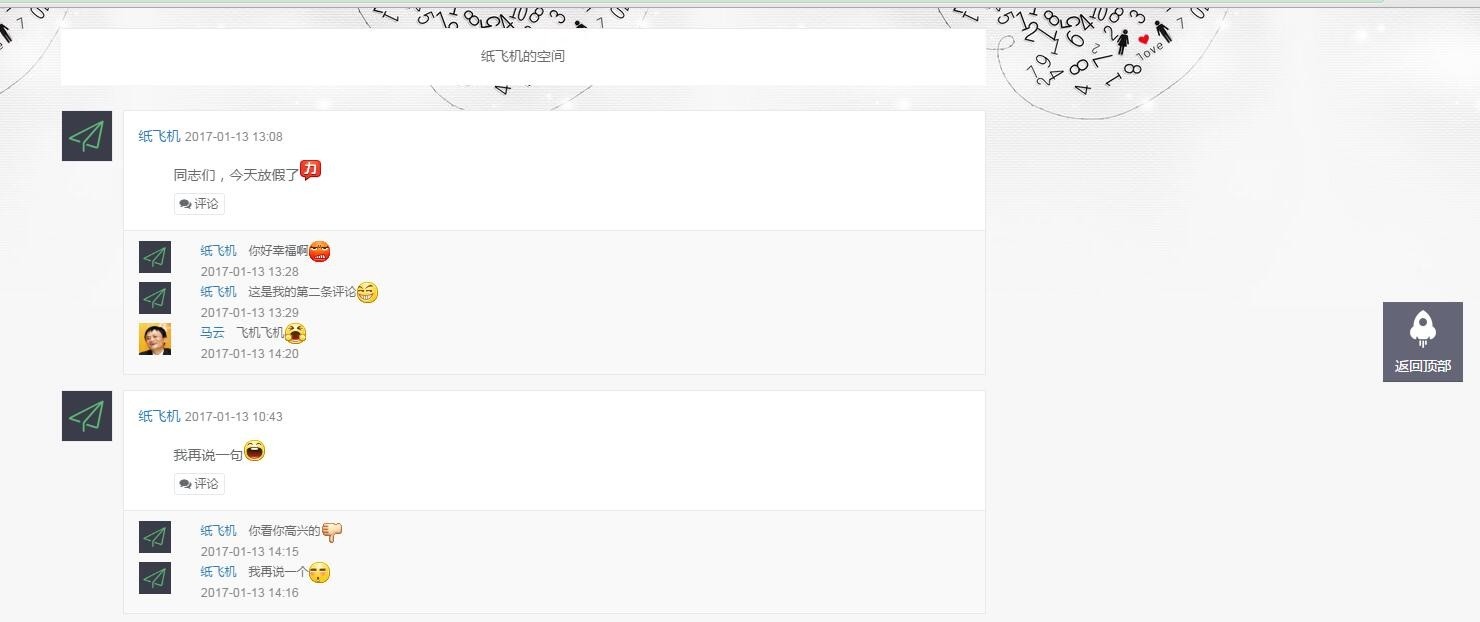
var post\_comment\_url = "{:url('userzone/postComment')}";

</script>

<script type="text/javascript" src=" COMMON /js/zones.js"></script>

</body>

</html>



# 将好友加入黑名单

#### 将好友加入黑名单之后，该好友便不能与我聊天，同时该好友也不能添加我为好友了。所以，我们需要设计 一个表来存放这些黑名单信息，v3\_blacktab

DROP TABLE IF EXISTS `v3\_blacktab`; CREATE TABLE `v3\_blacktab` (

`user\_id` int(11) NOT NULL COMMENT '操作人的id',

`put\_uid` int(11) NOT NULL COMMENT '被加入黑名单的id',

`addtime` int(10) NOT NULL COMMENT '执行时间', KEY `user\_id` (`user\_id`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

依旧是自己扩展的右键其中的一个功能，js 监听这个这个事件：

//将用户加入黑名单

, joinBlack: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

//询问框 layer.confirm('确定要将他加入黑名单？', {

btn: ['是的', '算了'], title: '友情提示', icon: 3,

closeBtn: 0

}, function(){

$.getJSON("/index/Tools/joinBlack/fid/" + friend\_id, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:1500}); layim.removeList({

type: 'friend'

,id: res.data.to\_id

});

//通知被加入黑名单的用户，删除我

var black\_data = '{"type":"black","to\_id":"' + res.data.to\_id + '", "del\_id":"' + res.data.del\_id + '"}';

socket.send(black\_data);

}else{

layer.msg(res.msg, {time:1500});

}

});

}, function(){

});

}

#### 首先用户确定加入好友到黑名单之后，会向 /index/Tools/joinBlack 方法发送一个请求。其实这个方法处理的

逻辑也很简单，只需要将该好友从好友表中删除，并把该好友加入到 黑名单表中，就可以完成好友的拉黑 了。

//加入黑名单

public function joinBlack()

{

if(request()->isAjax()){

$delStr = 'layim-friend';

$friendId = input('fid');

$friendId = substr($friendId, strlen($delStr), strlen($friendId));

$param = [

'user\_id' => session('f\_user\_id'), 'put\_uid' => $friendId,

'addtime' => time()

];

$flag = db('blacktab')->insert($param); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '加入黑名单失败']);

}

$mine = session('f\_user\_id');

//将其从好友中删除

$flag = db('friends')->where('user\_id = ' . $mine . ' and friend\_id = ' . $ friendId)->delete();

if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '加入黑名单失败']);

}

//并从被拉黑的好友中，将我删除

$flag = db('friends')->where('user\_id = ' . $friendId . ' and friend\_id = '

. $mine)->delete();

if(empty($flag)){

return json(['code' => -3, 'data' => '', 'msg' => '加入黑名单失败']);

}

$return = [

'to\_id' => $friendId, 'del\_id' => $mine

];

return json(['code' => 1, 'data' => $return, 'msg' => '加入成功']);

}

$this->error('非法请求');

}

#### 接到成功的回调之后，向workerman发送一条信息，让workerman向被拉入黑名单的用户发送一条从主面板

移除好友的消息。这样在我将该好友拉入黑名单，该好友的好友列表中我也会消失。而我的好友面板中，通 过js

layim.removeList({ type: 'friend'

,id: res.data.to\_id

});

已经移除了该好友，之后发送消息给workerman

//通知被加入黑名单的用户，删除我

var black\_data = '{"type":"black","to\_id":"' + res.data.to\_id + '", "del\_id":"' + res.d ata.del\_id + '"}';

socket.send(black\_data);

而workerman只需要通知相应的客户端即可。

case 'black':

$client\_id = Gateway::getClientIdByUid($message['to\_id']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [ 'message\_type' => 'black', 'data' => [

'id' => $message['del\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

客户端接收到这个 black 类型的消息之后，将要移除的人从主面板中删除。

//加入黑名单 case 'black':

//console.log(data.data); layim.removeList({

type: 'friend'

, id: data.data.id //好友或者群组ID

});

break;

# 移动好友分组

#### 移动好友的分组，主要的设计目的是：可以动态的改变好友的分组。

目前主面板的分组有 朋友、家人、同事、其他。这部分是通过配置项实现的。

比如目前的 好友纸飞机在 朋友中，我想改变他的分组，右键选择移动好友分组，可以方便快捷的管理好友

依旧是通过扩展实现的功能，我们来看一下，如何监听这个事件：

//改变用户的群组

, changeGroup: function(othis, e){

//改变群组模板

var elemAddTpl = ['<div class="layim-add-box">'

, '<div class="layim-add-img"><img class="layui-circle" src="{{ d.data. avatar }}"><p>' +

'{{ d.data.name||"" }}</p></div>'

, '<div class="layim-add-remark">'

, '{{# if(d.data.type === "friend" && d.type === "setGroup"){ }}'

, '<p>选择分组</p>'

, '{{# } if(d.data.type === "friend"){ }}'

, '<select class="layui-select" id="LAY\_layimGroup">'

, '{{# layui.each(d.data.group, function(index, item){ }}'

, '<option value="{{ item.id }}">{{ item.groupname }}</option>'

, '{{# }); }}'

, '</select>'

, '{{# } }}'

, '{{# if(d.data.type === "group"){ }}'

, '<p>请输入验证信息</p>'

, '{{# } if(d.type !== "setGroup"){ }}'

, '<textarea id="LAY\_layimRemark" placeholder="验证信息" class="layui-te xtarea"></textarea>'

, '{{# } }}'

, '</div>'

, '</div>'].join('');

var friend\_id = othis.parent().attr('data-id');

$.getJSON('/index/Tools/getNowUser/fid/' + friend\_id, function(res){ if(1 == res.code){

var index = layer.open({ type: 1,

skin: 'layui-layer-rim', //加上边框

area: ['430px', '260px'], //宽高

btn: ['确认', '取消'], title: '移动分组',

content: laytpl(elemAddTpl).render({ data: {

name: res.data.user\_name

, avatar: res.data.avatar

, group: parent.layui.layim.cache().friend

, type: 'friend'

}

, type: 'setGroup'

})

, yes: function (index, layero) {

var groupElem = layero.find('#LAY\_layimGroup'); var group\_id = groupElem.val(); //群组id

$.post('/index/Tools/changeGroup', {'group\_id' : group\_id,

'user\_id' : res.data.id},

function(data) {

if (1 == data.code) { layer.msg(data.msg, {time: 1500});

//先从旧组移除，然后加入新组

layim.removeList({ type: 'friend'

,id: res.data.id

});

//加入新组 layim.addList({

type: 'friend'

,avatar: res.data.avatar

,username: res.data.user\_name

,groupid: group\_id

,id: res.data.id

,sign: res.data.sign

});

layer.close(index);

} else {

layer.msg(data.msg, {time: 2000});

}

});

}else{

}

}, 'json');

}

});

}

layer.msg(res.msg, {time: 2000});

#### 整个页面，是仿照添加好友页面实现的。

**设计思路**

由于选择移动分组的时候，要展示用户的头像和姓名之类的信息，因此打开之时，我们请求了

/index/Tools/getNowUser 获取当前的用户信息。确定一定分组之后，我们请求 /index/Tools/changeGroup 接口，去修改用户所在的分组信息。js 要做的部分，就是将该好友从旧的分组中删除，并在新的分组中加入。

**具体实现方法**

获取当前用户的接口 getNowUser ：

//获取要分组的用户信息

public function getNowUser()

{

if(request()->isAjax()){

$delStr = 'layim-friend';

$friendId = input('fid');

$friendId = substr($friendId, strlen($delStr), strlen($friendId));

//获取该用户的信息

$userInfo = db('chatuser')->field('id,user\_name,avatar,sign')->where('id',

$friendId)->find();

if(empty($userInfo)){

return json(['code' => -1, 'data' => '', 'msg' => '不存在该好友']);

}

return json(['code' => 1, 'data' => $userInfo, 'msg' => 'suceess']);

}

$this->error('非法请求');

}

#### 确切的返回 当前用户账号，下面就是根据用户的选择，改变好友的分组信息。changeGroup

//移动分组

public function changeGroup()

{

if(request()->isAjax()){

//更新用户的分组信息

$groupId = input('post.group\_id');

$userId = input('post.user\_id');

$mine = session('f\_user\_id');

userId)

$flag = db('friends')->where('user\_id = ' . $mine . ' and friend\_id = ' . $

->setField('group\_id', $groupId); if(false === $flag){

return json(['code' => -1, 'data' => '', 'msg' => '移动分组失败']);

}

if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '该用户已经在这个分组了']

);

}

return json(['code' => 1, 'data' => $userId, 'msg' => '移动成功']);

}

$this->error('非法访问');

}

#### 其实就是更新用户的 group\_id 字段。接下来便是 js的操作了。

//先从旧组移除，然后加入新组 layim.removeList({

type: 'friend'

,id: res.data.id

});

//加入新组 layim.addList({

type: 'friend'

,avatar: res.data.avatar

,username: res.data.user\_name

,groupid: group\_id

,id: res.data.id

,sign: res.data.sign

});

# 删除好友

#### 删除好友和拉黑好友的区别就是，不需要将好友加入黑名单表(v3\_blacktab)，也不需要显示该好友重新添加 我。处理的逻辑更加的简单。

**设计思路**

用户右键选择要删除的用户之后，只需发送请求给php，让其从好友表中删除两个人的好友关系。并且返回争

取信息给 js ，js向workerman发送数据，通知被删除的好友客户端，从主面板中移除我的信息。

**具体的实现**

对右键删除好友事件的监听：

es){

// 删除好友

, removeFriend: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

//询问框 layer.confirm('确定删除该好友？', {

btn: ['确定', '取消'], title: '友情提示', closeBtn: 0,

icon: 3

}, function(){

$.post('/index/Tools/removeFriend', {'user\_id' : friend\_id}, function(r

if(1 == res.code){ layer.msg(res.msg, {time: 1500}); layim.removeList({

type: 'friend'

, id: res.data.to\_id

});

//通知被删除的用户，删除我

var black\_data = '{"type":"delFriend","to\_id":"' + res.data.to\_ id + '", "del\_id":"' + res.data.del\_id + '"}';

socket.send(black\_data);

}else{

layer.msg(res.msg, {time: 1500});

}

}, 'json');

}, function(){

});

}

#### 用户选择确认 删除之后，请求 /index/Tools/removeFriend 接口，移除好友关系

//移除好友

public function removeFriend()

{

if(request()->isAjax()){

//删除好友

$userId = input('post.user\_id');

$delStr = 'layim-friend';

$userId = substr($userId, strlen($delStr), strlen($userId));

$mine = session('f\_user\_id');

//先将该用户从操作者的好友中删除

$flag = db('friends')->where('user\_id=' . $mine . ' and friend\_id=' . $user Id)->delete();

if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '删除失败']);

}

//再从该用户中删除操作者

$flag = db('friends')->where('user\_id=' . $userId . ' and friend\_id=' . $mi ne)->delete();

if(empty($flag)){

return json(['code' => -2, 'data' => '', 'msg' => '删除失败']);

}

$return = [

'to\_id' => $userId, 'del\_id' => $mine

];

return json(['code' => 1, 'data' => $return, 'msg' => '删除成功']);

}

$this->error('非法访问');

}

#### 此时js收到成功的回调之后通过js，删除我的好友面板中的该好友：

layim.removeList({ type: 'friend'

, id: res.data.to\_id

});

并向workerman发出删除请求

//通知被删除的用户，删除我

var black\_data = '{"type":"delFriend","to\_id":"' + res.data.to\_id + '", "del\_id":"' + r es.data.del\_id + '"}';

socket.send(black\_data);

workerman收到消息之后，通知被删除的好友，做出相应的动作。Events.php

case 'delFriend':

$client\_id = Gateway::getClientIdByUid($message['to\_id']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [

'message\_type' => 'delFriend', 'data' => [

'id' => $message['del\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

同样的，客户端收到 delFriend 消息之后，做出移除好友的动作。

//删除好友

case 'delFriend':

//console.log(data.data); layim.removeList({

type: 'friend'

, id: data.data.id //好友或者群组ID

});

break;

# 举报好友

#### 当与某个好友的聊天中，发现该好友有哪些不当的言论时，可以动用次功能向管理员举报次用户。举报的页 面相对复杂一些，因此我们需要建立一个页面来渲染。

**设计思路**

监听到用户选择举报好友信息之后，展示给用户举报好友的页面，我们事先有一些预定的举报条目，供用户

选择。当然这些条目可以通过配置来修改。我们新建一个 举报表 v3\_report 来记录这一切的举报。

DROP TABLE IF EXISTS `v3\_report`; CREATE TABLE `v3\_report` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`report\_uid` int(11) NOT NULL COMMENT '举报人id',

`report\_user` varchar(155) NOT NULL COMMENT '举报人名',

`reported\_uid` int(11) NOT NULL COMMENT '被举报人id',

`reported\_user` varchar(155) NOT NULL COMMENT '被举报人名',

`addtime` int(11) NOT NULL COMMENT '举报时间',

`content` text COMMENT '举报证据',

`report\_type` varchar(155) NOT NULL COMMENT '举报类型大类',

`report\_detail` varchar(155) NOT NULL COMMENT '举报类型小类', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

逻辑很简单，只需要将举报信息入库，等待管理员的处理就可以了。

**具体的实现**

监听用户举报好友事件：

//举报好友

, reportFriend: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

layer.open({

type: 2,

title: '举报好友', shadeClose: true,

skin: 'layui-layer-molv', //加上边框 shade: 0.3,

area: ['400px', '400px'],

content: '/index/Tools/reportFriend/user\_id/' + friend\_id

});

}

可以看到请求的是 /index/Tools/reportFriend 方法，该方法会渲染一个页面，并且给出被举报的用户的信

息，友好的提示用户

//举报好友

public function reportFriend()

{

$userId = input('user\_id');

$delStr = 'layim-friend';

$userId = substr($userId, strlen($delStr), strlen($userId));

$user = db('chatuser')->field('id,user\_name')->where('id', $userId)->find();

$reportType = config('report\_type');

$reportDetail = config('report\_detail');

$this->assign([ 'user' => $user,

'retype' => $reportType, 'redetail' => $reportDetail

]);

return $this->fetch();

}

渲染的页面 reportfriend.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1

">

<title>laychat-v3.0 - 创建群组</title>

<link rel="stylesheet" href=" COMMON /layui/css/layui.css">

<style>

.tab\_hover{background:#1E9FFF;color:white}

.layui-form-pane blockquote{cursor: pointer}

</style>

</head>

<body>

<div class="main layui-clear" style="margin-top:15px;padding: 10px">

<blockquote class="layui-elem-quote">

确定举报 {$user.user\_name} ?

</blockquote>

<div class="layui-form layui-form-pane" style="width: 80%;margin: 0 auto" id="retyp

e">

{if !empty($retype)}

{foreach name="retype" item="vo" key="key"}

<blockquote class="layui-elem-quote layui-quote-nm retype" data-id="{$key}" dat

a-data="{$vo}">

{$vo}

<i class="layui-icon" style="float: right">&#xe602;</i>

</blockquote>

{/foreach}

{/if}

</div>

<!-- detail -->

{if !empty($redetail)}

{foreach name="redetail" item="vo" key="key"}

<div class="layui-form layui-form-pane" style="width: 80%;margin: 0 auto;display: n one" data-id="{$key}">

{foreach name="vo" item="v" key="k"}

<blockquote class="layui-elem-quote layui-quote-nm redetail" data-data="{$v}">

{$v}

<i class="layui-icon" style="float: right">&#xe602;</i>

</blockquote>

{/foreach}

<button class="layui-btn layui-btn-mini" style="float: right">上一步</button>

</div>

{/foreach}

{/if}

<!-- final -->

<div class="layui-form layui-form-pane" style="margin: 0 auto;display: none" id="fi nal">

<div class="layui-field-box">

我们还会将您于 {$user.user\_name} 的最近 5条 聊天记录作为证据一并提交

</div>

<button class="layui-btn layui-btn-small" style="float:right;margin-top: 10px">

确定提交</button>

</div>

</div>

<input type="hidden" id="report\_type"/>

<input type="hidden" id="report\_detail"/>

<script src=" COMMON /layui/layui.js"></script>

<script type="text/javascript">

var reported\_uid = "{$user.id}";

var reported\_user = "{$user.user\_name}";

var do\_report\_url = "{:url('tools/doReport')}";

layui.use(['layer', 'jquery'], function(){ var layer = layui.layer,

$ = layui.jquery;

$(function(){

$(".layui-form-pane blockquote").hover(function(){

$(this).addClass('tab\_hover');

}, function(){

$(this).removeClass('tab\_hover');

}) ;

$(".retype").click(function(){

var type\_id = $(this).attr('data-id');

$("div[data-id=" + type\_id + "]").show();

$('#retype').hide();

$("#report\_type").val($(this).attr('data-data'));

$("#report\_detail").val('');

});

$(".redetail").click(function(){

$("#report\_detail").val($(this).attr('data-data'));

$(this).parent().hide();

$("#final").show();

});

$(".layui-btn-mini").click(function(){

$(this).parent().hide();

$('#retype').show();

});

$("#final button").click(function(){

var report\_type = $("#report\_type").val(); var report\_detail = $("#report\_detail").val(); if('' == report\_type || '' == report\_detail){

layer.msg('举报消息有误', {time:1500});

return ;

}

$.post(do\_report\_url,

{

'reported\_uid' : reported\_uid, 'reported\_user' : reported\_user,

'report\_type' : report\_type, 'report\_detail' : report\_detail

}

, function(res){

if(1 == res.code){

layer.msg(res.msg, {time : 1500}); setTimeout(function(){

var index = parent.layer.getFrameIndex(window.name); //先得

到当前iframe层的索引

parent.layer.close(index); //再执行关闭

}, 1500);

}else{

layer.msg(res.msg, {time : 1500});

}

}, 'json');

});

});

});

</script>

</body>

</html>



#### 用户根据导向，举报该用户即可。最后提交给 处理函数 doReport，入库

//处理举报好友

public function doReport()

{

if(request()->isAjax()){

$param = input('post.');

$param['report\_uid'] = session('f\_user\_id');

$param['report\_user'] = session('f\_user\_name');

$param['addtime'] = time();

//查询与这个用户的5条聊天记录作为证据

$log = db('chatlog')->field('content,timeline')

->where('from\_id=' . $param['reported\_uid'] . ' and to\_id=' . $param['r

eport\_uid'])

->limit(5)->order('timeline desc')->select();

$param['content'] = serialize($log);

$flag = db('report')->insert($param); if(empty($flag)){

return json(['code' => -1, 'data' => '', 'msg' => '举报失败']);

}

return json(['code' => 1, 'data' => '', 'msg' => '举报成功，耐心等待处理消息。']

);

}

$this->error('非法访问');

}

# 数据字典

#### 本文档介绍本系统用到的所有表 v3\_admin 后台管理员表

DROP TABLE IF EXISTS `v3\_admin`; CREATE TABLE `v3\_admin` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`login\_name` varchar(155) NOT NULL,

`password` varchar(255) NOT NULL, PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_applygroup 前台用户申请加群表

DROP TABLE IF EXISTS `v3\_applygroup`;

CREATE TABLE `v3\_applygroup` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`apply\_uid` int(11) NOT NULL COMMENT '申请人用户id',

`apply\_uname` varchar(155) NOT NULL COMMENT '申请人用户名',

`group\_id` int(11) NOT NULL COMMENT '申请加入的群组',

`owner\_id` int(11) NOT NULL COMMENT '该群组的管理员id',

`remark` varchar(255)

`addtime` int(10) NOT PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT

NOT NULL COMMENT '加群附加信息',

NULL COMMENT '申请时间',

CHARSET=utf8;

v3\_area 全国地区表

DROP TABLE IF EXISTS `v3\_area`;

CREATE TABLE `v3\_area` (

`id` int(10) NOT NULL COMMENT 'ID',

`area\_name` varchar(50) NOT NULL COMMENT '栏目名',

`parent\_id` int(10) NOT NULL COMMENT '父栏目',

`level` tinyint(1) NOT NULL,

`sort` tinyint(3) unsigned DEFAULT '50' COMMENT '排序', PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

v3\_blacktab 黑名单表

DROP TABLE IF EXISTS `v3\_blacktab`; CREATE TABLE `v3\_blacktab` (

`user\_id` int(11) NOT NULL COMMENT '操作人的id',

`put\_uid` int(11) NOT NULL COMMENT '被加入黑名单的id',

`addtime` int(10) NOT NULL COMMENT '执行时间', KEY `user\_id` (`user\_id`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_chatgroup 聊天群组表

DROP TABLE IF EXISTS `v3\_chatgroup`; CREATE TABLE `v3\_chatgroup` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '群组id',

`group\_name` varchar(155) NOT NULL COMMENT '群组名称',

`avatar` varchar(155) DEFAULT NULL COMMENT '群组头像',

`owner\_name` varchar(155) DEFAULT NULL COMMENT '群主名称',

`owner\_id` int(11) DEFAULT NULL COMMENT '群主id',

`owner\_avatar` varchar(155) DEFAULT NULL COMMENT '群主头像',

`owner\_sign` varchar(155) DEFAULT NULL COMMENT '群主签名',

`addtime` int(10) DEFAULT NULL,

`status` tinyint(1) DEFAULT NULL COMMENT '1审核通过 -1待审核 -2审核不通过', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_chatlog 聊天记录表

DROP TABLE IF EXISTS `v3\_chatlog`;

CREATE TABLE `v3\_chatlog` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`from\_id` int(11) NOT NULL COMMENT '会话来源id',

`from\_name` varchar(155) NOT NULL DEFAULT '' COMMENT '消息来源用户名',

`from\_avatar` varchar(155) NOT NULL DEFAULT '' COMMENT '来源的用户头像',

`to\_id` int(11) NOT NULL COMMENT '会话发送的id',

`content` text NOT

`timeline` int(10)

`type` varchar(55)

NULL COMMENT '发送的内容',

NOT

NOT

`need\_send` tinyint(1) PRIMARY KEY (`id`),

NULL COMMENT '记录时间',

NULL COMMENT '聊天类型',

DEFAULT '0' COMMENT '0 不需要推送 1 需要推送',

KEY `fromid` (`from\_id`) USING BTREE,

KEY `toid` (`to\_id`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_chatuser 前台用户表

DROP TABLE IF EXISTS `v3\_chatuser`;

CREATE TABLE `v3\_chatuser` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(155) DEFAULT NULL,

`pwd` varchar(155) DEFAULT NULL COMMENT '密码',

`sign` varchar(255) DEFAULT NULL,

`avatar` varchar(255) DEFAULT NULL,

`sex`

`age`

`pid`

`cid`

`aid`

tinyint(1) DEFAULT '1' COMMENT '性别 1男 -1女',

int(3) DEFAULT '18' COMMENT '年龄',

int(10)

int(10) int(10)

DEFAULT

DEFAULT DEFAULT

'110000'

'110000'

'110101'

COMMENT '所在省份id',

COMMENT '所在城市id', COMMENT '所在区id',

`area` varchar(255) DEFAULT '北京-北京市-东城区' COMMENT '所在区域描述',

`status` tinyint(1) DEFAULT '0' COMMENT '0下线 1在线', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_comment 空间评论表

DROP TABLE IF EXISTS `v3\_comment`; CREATE TABLE `v3\_comment` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '评论的id',

`blog\_id` int(11) NOT NULL COMMENT '关联的说说id',

`com\_user` varchar(155) NOT NULL COMMENT '评论人名',

`com\_uid` int(11) NOT NULL COMMENT '评论人用户id',

`com\_avatar` varchar(255) NOT NULL COMMENT '评论人头像',

`content` text NOT NULL COMMENT '评论内容',

`com\_time` int(10) NOT NULL COMMENT '评论时间', PRIMARY KEY (`id`),

KEY `blog\_id` (`blog\_id`) USING BTREE, KEY `com\_uid` (`com\_uid`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_friends 好友关系表

DROP TABLE IF EXISTS `v3\_friends`; CREATE TABLE `v3\_friends` (

`user\_id` int(11) NOT NULL COMMENT '用户id',

`friend\_id` int(11) NOT NULL COMMENT '朋友id',

`group\_id` int(11) NOT NULL COMMENT '朋友所属组别id', KEY `user\_id` (`user\_id`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_groupdetail 加入群组详情表

DROP TABLE IF EXISTS `v3\_groupdetail`; CREATE TABLE `v3\_groupdetail` (

`user\_id` int(11) NOT NULL,

`user\_name` varchar(155) NOT NULL,

`user\_avatar` varchar(155) NOT NULL,

`user\_sign` varchar(155) NOT NULL,

`group\_id` int(11) NOT NULL

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_mblog 用户发表的说说表

DROP TABLE IF EXISTS `v3\_mblog`; CREATE TABLE `v3\_mblog` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '说说的id',

`post\_user` varchar(155) NOT NULL COMMENT '发表人名',

`post\_uid` int(11) NOT NULL,

`post\_avatar` varchar(255) NOT NULL COMMENT '发表人的头像',

`content` text NOT NULL COMMENT '发表内容',

`post\_time` int(10) NOT NULL COMMENT '发表时间', PRIMARY KEY (`id`),

KEY `post\_uid` (`post\_uid`) USING BTREE

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

v3\_message 消息盒子表

DROP TABLE IF EXISTS `v3\_message`; CREATE TABLE `v3\_message` (

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '信息的id',

`content` varchar(255) NOT NULL COMMENT '消息内容',

`uid` int(11) NOT NULL COMMENT '接收人用户id',

`from` int(11) NOT NULL COMMENT '发送人用户id',

`from\_group` int(11) NOT NULL DEFAULT '0' COMMENT '默认消息来源群组id',

`type` tinyint(1) NOT NULL COMMENT '消息类型 1用户消息 2系统消息',

`remark` varchar(255) NOT NULL COMMENT '附加消息',

`href` varchar(255) DEFAULT NULL COMMENT '消息跳转',

`read` tinyint(1) DEFAULT '1' COMMENT '消息阅读状态 1未读 2已读',

`time` int(11) NOT NULL COMMENT '消息发送时间',

`agree` tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否同意 0默认 1同意 2拒绝', PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

#### v3\_report 用户举报表

DROP TABLE IF EXISTS `v3\_report`; CREATE TABLE `v3\_report` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`report\_uid` int(11) NOT NULL COMMENT '举报人id',

`report\_user` varchar(155) NOT NULL COMMENT '举报人名',

`reported\_uid` int(11) NOT NULL COMMENT '被举报人id',

`reported\_user` varchar(155) NOT NULL COMMENT '被举报人名',

`addtime` int(11) NOT NULL COMMENT '举报时间',

`content` text COMMENT '举报证据',

`report\_type` varchar(155) NOT NULL COMMENT '举报类型大类',

`report\_detail` varchar(155) NOT NULL COMMENT '举报类型小类', PRIMARY KEY (`id`)

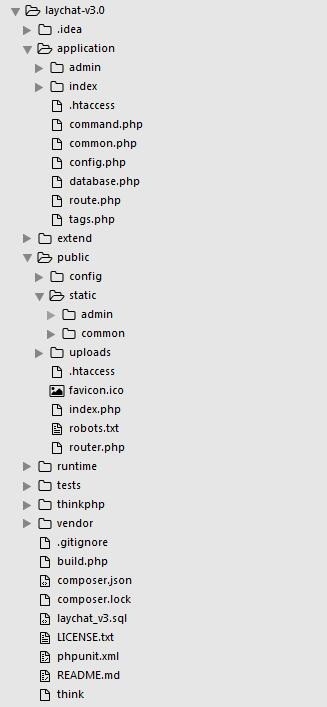
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

# 全国区域数据

#### 本系统用到的全国区域数据：[全国区域数据](http://pan.baidu.com/s/1o7IQzW2)

# 系统整体结构说明

#### 系统整体结构文件如下



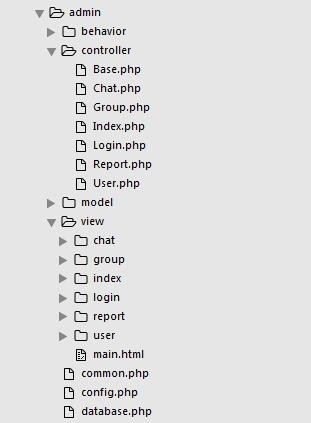
application 程序核心目录

admin 后台模块 index 前台模块

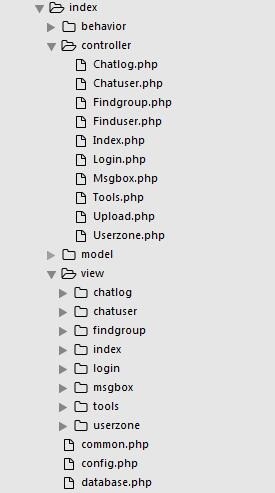
common.php 公用方法 database.php 数据库配置文件

public/config 后台系统配置问价存储目录

public/static 系统静态资源目录 后台模块的所有文件



前台模块的所有文件



# Events.php全部代码

<?php

use \GatewayWorker\Lib\Gateway; use \GatewayWorker\Lib\Db;

/\*\*

\* 主逻辑

* 主要是处理 onConnect onMessage onClose 三个方法
* onConnect 和 onClose 如果不需要可以不用实现并删除

\*/

class Events

{

/\*\*

\* 当客户端发来消息时触发

* @param int $client\_id 连接id
* @param mixed $message 具体消息

\*/

public static function onMessage($client\_id, $data)

{

$message = json\_decode($data, true);

$message\_type = $message['type']; switch ($message\_type) {

case 'init':

// uid

$uid = $message['id'];

// 设置session

$\_SESSION = array(

'username' => $message['username'], 'avatar' => $message['avatar'], 'id' => $uid,

'sign' => $message['sign']

);

// 将当前链接与uid绑定

Gateway::bindUid($client\_id, $uid);

$db1 = Db::instance('db1'); //数据库链接

//查询最近1周有无需要推送的离线信息

$time = time() - 7 \* 3600 \* 24;

$resMsg = $db1->select('id,from\_id,from\_name,from\_avatar,timeline,conte nt')->from('v3\_chatlog')

->where("to\_id= {$uid} and timeline > {$time} and type = 'friend' a

nd need\_send = 1" )

->query();

//var\_export($resMsg); if (!empty($resMsg)) {

foreach ($resMsg as $key => $vo) {

$log\_message = [

'message\_type' => 'logMessage', 'data' => [

'username' => $vo['from\_name'], 'avatar' => $vo['from\_avatar'], 'id' => $vo['from\_id'],

'type' => 'friend',

'content' => htmlspecialchars($vo['content']),

'timestamp' => $vo['timeline'] \* 1000,

]

];

Gateway::sendToUid($uid, json\_encode($log\_message));

=" . $vo['id']);

}

}

//设置推送状态为已经推送

$db1->query("UPDATE `v3\_chatlog` SET `need\_send` = '0' WHERE id

// 通知所有该用户好友，此用户上线，将此用户头像变亮

$friends = $db1->query("select `user\_id` from `v3\_friends` where `frien d\_id` = " . $uid);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [ 'message\_type' => 'online', 'id' => $uid,

line\_message));

}

}

}

];

Gateway::sendToClient($user\_client\_id['0'], json\_encode($on

//查询当前的用户是在哪个分组中,将当前的链接加入该分组

$ret = $db1->query("select `group\_id` from `v3\_groupdetail` where `user

\_id` = {$uid} group by `group\_id`");

if (!empty($ret)) {

foreach ($ret as $key => $vo) {

Gateway::joinGroup($client\_id, $vo['group\_id']); //将登录用户加

入群组

}

}

unset($ret);

//设置用户为登录状态

$db1->query("update `v3\_chatuser` set status=1 where id=" . $uid);

return;

case 'chatMessage':

$db1 = Db::instance('db1'); //数据库链接

// 聊天消息

$type = $message['data']['to']['type'];

$to\_id = $message['data']['to']['id'];

$uid = $\_SESSION['id'];

']),

$chat\_message = [

'message\_type' => 'chatMessage', 'data' => [

'username' => $\_SESSION['username'], 'avatar' => $\_SESSION['avatar'],

'id' => $type === 'friend' ? $uid : $to\_id, 'type' => $type,

'content' => htmlspecialchars($message['data']['mine']['content

'timestamp' => time() \* 1000,

]

];

// 加入聊天log表

$param = [

'from\_id' => $uid, 'to\_id' => $to\_id,

'from\_name' => $\_SESSION['username'], 'from\_avatar' => $\_SESSION['avatar'],

'content' => htmlspecialchars($message['data']['mine']['content']), 'timeline' => time(),

'need\_send' => 0

];

switch ($type) {

// 私聊

case 'friend':

// 插入

$param['type'] = 'friend';

if (empty(Gateway::getClientIdByUid($to\_id))) {

$param['need\_send'] = 1; //用户不在线,标记此消息推送

$client\_id);

}

}

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToUid($to\_id, json\_encode($chat\_message));

// 群聊

case 'group':

$param['type'] = 'group';

$db1->insert('v3\_chatlog')->cols($param)->query();

return Gateway::sendToGroup($to\_id, json\_encode($chat\_message),

return; break;

case 'addFriend':

$client\_id = Gateway::getClientIdByUid($message['toid']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [

'message\_type' => 'addFriend', 'data' => [

'username' => $message['username'], 'avatar' => $message['avatar'], 'id' => $message['id'],

'type' => 'friend',

'sign' => $message['sign'], 'groupid' => $message['groupid'],

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

case 'black':

$client\_id = Gateway::getClientIdByUid($message['to\_id']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [ 'message\_type' => 'black', 'data' => [

'id' => $message['del\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return;

break;

case 'delFriend':

$client\_id = Gateway::getClientIdByUid($message['to\_id']);

//用户在线才通知 if(!empty($client\_id)){

$add\_message = [

'message\_type' => 'delFriend', 'data' => [

'id' => $message['del\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

case 'addGroup':

// 创建群组

$client\_id = Gateway::getClientIdByUid($message['join\_id']); Gateway::joinGroup($client\_id['0'], $message['id']);

$add\_message = [

'message\_type' => 'addGroup', 'data' => [

'type' => 'group',

'avatar' => $message['avatar'], 'id' => $message['id'],

'groupname' => $message['groupname']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message)); return;

break;

case 'applyGroup':

// 申请加入群组

// 通知群组管理员，进行入群申请

$client\_id = Gateway::getClientIdByUid($message['to\_id']); if(!empty($client\_id)){

$apply\_message = [

'message\_type' => 'applyGroup', 'data' => [

'uid' => $message['to\_id'], //群组的管理员

'groupname' => $message['groupname'], 'groupid' => $message['groupid'], 'groupavatar' => $message['groupavatar'],

'joinid' => $message['join\_id'], //申请加入群组的id

'joinname' => $message['join\_name'], //申请加入群组的用户名 'joinsign' => $message['join\_sign'], //申请加组的的用户签名 'joinavatar' => $message['join\_avatar'], // 申请人头像 'remark' => $message['remark'] //附加信息

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($apply\_message))

;

}

return; break;

case 'joinGroup':

//将申请人加入讨论组

$client\_id = Gateway::getClientIdByUid($message['join\_id']); //若在线实时

推送

if( !empty($client\_id) ){

Gateway::joinGroup($client\_id['0'], $message['group\_id']); //将该用

户加入群组

$add\_message = [

'message\_type' => 'addGroup', 'data' => [

'type' => 'group',

'avatar' => $message['group\_avatar'], 'id' => $message['group\_id'], 'groupname' => $message['group\_name']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($add\_message));

}

return; break;

case 'removeMember':

//将移除群组的成员的群信息移除，并从讨论组移除

$client\_id = Gateway::getClientIdByUid( $message['remove\_id'] ); if( !empty($client\_id) ){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

$del\_message = [

'message\_type' => 'delGroup', 'data' => [

'type' => 'group',

'id' => $message['group\_id']

]

];

Gateway::sendToClient($client\_id['0'], json\_encode($del\_message));

}

return; break;

case 'leaveGroup':

// 退出群组

$client\_id = Gateway::getClientIdByUid($message['leave\_id']); if( !empty($client\_id) ){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

}

return; break;

case "breakUp":

// 解散群组

$uids = explode(',', $message['uids']); foreach($uids as $key=>$vo){

$client\_id = Gateway::getClientIdByUid($vo); if(!empty($client\_id)){

Gateway::leaveGroup($client\_id['0'], $message['group\_id']);

$del\_message = [

'message\_type' => 'delGroup', 'data' => [

'type' => 'group',

'id' => $message['group\_id']

]

];

));

Gateway::sendToClient($client\_id['0'], json\_encode($del\_message

}

}

return; break;

case 'online':

$db1 = Db::instance('db1'); //数据库链接

// 在线切换状态

$friends = $db1->query("select `user\_id` from `v3\_friends` where `frien d\_id` = " . $message['uid']);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [

'message\_type' => ('online' == $message['status']) ? 'o

nline' : 'offline',

'id' => $message['uid'],

];

line\_message));

}

}

}

Gateway::sendToClient($user\_client\_id['0'], json\_encode($on

return; break;

case 'ping':

return; default:

echo $data;

}

}

/\*\*

\* 当用户断开连接时触发

\* @param int $client\_id 连接id

\*/

public static function onClose($client\_id)

{

//维持连接的worker退出才触发

if(!empty($\_SESSION['id'])){

$db1 = Db::instance('db1'); //数据库链接

//通知该用户的好友，该用户下线

$friends = $db1->query("select `user\_id` from `v3\_friends` where `friend\_id

` = " . $\_SESSION['id']);

if (!empty($friends)) {

foreach ($friends as $key => $vo) {

$user\_client\_id = Gateway::getClientIdByUid($vo['user\_id']); if (!empty($user\_client\_id)) {

$online\_message = [ 'message\_type' => 'logout', 'id' => $\_SESSION['id'],

\_message));

}

}

];

Gateway::sendToClient($user\_client\_id['0'], json\_encode($online

}

['id']);

}

//设置用户为退出状态

$db1->query("update `v3\_chatuser` set `status` = 0 where id = " . $\_SESSION

}

}

# main.js全部代码

//建立WebSocket通讯 var socket;

layui.use(['layim', 'laytpl'], function(layim){ var laytpl = layui.laytpl;

//基础配置 layim.config({

//初始化接口 init: {

url: user\_list\_url

,data: {}

}

//查看群员接口

,members: {

url: member\_list\_url

,data: {}

}

,uploadImage: {

url: upload\_img\_url

}

,uploadFile: {

url: upload\_file\_url

}

//扩展工具栏

,tool: [

{

alias: 'voice'

,title: '语音'

,icon: '&#xe62c;'

},

{

alias: 'map'

,title: '地理位置'

,icon: '&#xe634;'

},

]

,initSkin: '3.jpg' //1-5 设置初始背景

,msgbox: msg\_box\_url//消息盒子页面地址，若不开启，剔除该项即可

,find: find\_url //发现页面地址，若不开启，剔除该项即可

,chatLog: chatlog\_url //聊天记录页面地址，若不开启，剔除该项即可

});

socket = new WebSocket('ws://127.0.0.1:8282');

//连接成功时触发 socket.onopen = function(){

// 登录

var login\_data = '{"type":"init","id":"' + uid + '", "username":"' + uname + '"

, "avatar":"' + avatar + '", "sign":"' + sign + '"}';

socket.send( login\_data ); console.log("websocket握手成功!");

};

//监听收到的消息

socket.onmessage = function(res){

//console.log(res.data);

var data = eval("("+res.data+")"); switch(data['message\_type']){

// 服务端ping客户端

case 'ping':

//console.log(data); socket.send('{"type":"ping"}'); break;

// 在线

case 'online': layim.setFriendStatus(data.id, 'online'); break;

// 下线

case 'offline': layim.setFriendStatus(data.id, 'offline'); break;

// 检测聊天数据

case 'chatMessage':

//console.log(data.data); layim.getMessage(data.data); break;

// 离线消息推送

case 'logMessage': setTimeout(function(){layim.getMessage(data.data)}, 1000); break;

// 用户退出 更新用户列表

case 'logout':

layim.setFriendStatus(data.id, 'offline'); break;

// 添加好友

case 'addFriend':

//console.log(data.data); layim.addList(data.data); break;

//加入黑名单

case 'black':

//console.log(data.data); layim.removeList({

type: 'friend'

, id: data.data.id //好友或者群组ID

});

break;

//删除好友

case 'delFriend':

//console.log(data.data); layim.removeList({

type: 'friend'

, id: data.data.id //好友或者群组ID

});

break;

// 添加 分组信息

case 'addGroup':

//console.log(data.data); layim.addList(data.data); break;

// 申请加入群组

case 'applyGroup':

//console.log(data.data);

//询问框

var index = layer.confirm(

data.data.joinname + ' 申请加入 ' + data.data.groupname + "<br/> 附加

信息： " + data.data.remark , {

btn: ['同意', '拒绝'], //按钮 title: '加群申请',

closeBtn: 0,

icon: 3

}, function(){

$.post(join\_group\_url,

{

'user\_id': data.data.joinid, 'user\_name': data.data.joinname, 'user\_avatar': data.data.joinavatar, 'user\_sign': data.data.joinsign, 'group\_id': data.data.groupid

},

function(res){

if(1 == res.code){

+ data.data.joinid + '"' +

var join\_data = '{"type":"joinGroup", "join\_id":"'

', "group\_id": "' + data.data.groupid + '", "gr

oup\_avatar": "' + data.data.groupavatar + '"' +

', "group\_name": "' + data.data.groupname + '"}

';

socket.send(join\_data);

}else{

layer.msg(res.msg, {time:2000});

}

}, 'json'); layer.close(index);

}, function(){

});

break;

// 删除面板的群组

case 'delGroup':

//console.log(data.data); layim.removeList({

type: 'group'

,id: data.data.id //群组ID

});

break;

}

};

//外部自定义我的事件 my\_events = {

//进入好友的空间

enterZone: function (othis, e) {

var friend\_id = othis.parent().attr('data-id');

layer.alert('<a href="/index/Userzone/uZone/fid/' + friend\_id + '" target= "\_blank">点击，进入他的空间</a>', {

title: false

, icon: 6

, btn: ''

});

}

//将用户加入黑名单

, joinBlack: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

//询问框 layer.confirm('确定要将他加入黑名单？', {

btn: ['是的', '算了'],

title: '友情提示', icon: 3,

closeBtn: 0

}, function(){

$.getJSON("/index/Tools/joinBlack/fid/" + friend\_id, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:1500}); layim.removeList({

type: 'friend'

,id: res.data.to\_id

});

//通知被加入黑名单的用户，删除我

var black\_data = '{"type":"black","to\_id":"' + res.data.to\_id + '", "del\_id":"' + res.data.del\_id + '"}';

socket.send(black\_data);

}else{

layer.msg(res.msg, {time:1500});

}

});

}, function(){

});

}

//改变用户的群组

, changeGroup: function(othis, e){

//改变群组模板

var elemAddTpl = ['<div class="layim-add-box">'

, '<div class="layim-add-img"><img class="layui-circle" src="{{ d.data. avatar }}"><p>' +

'{{ d.data.name||"" }}</p></div>'

, '<div class="layim-add-remark">'

, '{{# if(d.data.type === "friend" && d.type === "setGroup"){ }}'

, '<p>选择分组</p>'

, '{{# } if(d.data.type === "friend"){ }}'

, '<select class="layui-select" id="LAY\_layimGroup">'

, '{{# layui.each(d.data.group, function(index, item){ }}'

, '<option value="{{ item.id }}">{{ item.groupname }}</option>'

, '{{# }); }}'

, '</select>'

, '{{# } }}'

, '{{# if(d.data.type === "group"){ }}'

, '<p>请输入验证信息</p>'

, '{{# } if(d.type !== "setGroup"){ }}'

, '<textarea id="LAY\_layimRemark" placeholder="验证信息" class="layui-te xtarea"></textarea>'

, '{{# } }}'

, '</div>'

, '</div>'].join('');

var friend\_id = othis.parent().attr('data-id');

$.getJSON('/index/Tools/getNowUser/fid/' + friend\_id, function(res){ if(1 == res.code){

var index = layer.open({ type: 1,

skin: 'layui-layer-rim', //加上边框

area: ['430px', '260px'], //宽高

btn: ['确认', '取消'], title: '移动分组',

content: laytpl(elemAddTpl).render({ data: {

name: res.data.user\_name

, avatar: res.data.avatar

, group: parent.layui.layim.cache().friend

, type: 'friend'

}

, type: 'setGroup'

})

, yes: function (index, layero) {

var groupElem = layero.find('#LAY\_layimGroup'); var group\_id = groupElem.val(); //群组id

$.post('/index/Tools/changeGroup', {'group\_id' : group\_id,

'user\_id' : res.data.id},

function(data) {

if (1 == data.code) { layer.msg(data.msg, {time: 1500});

//先从旧组移除，然后加入新组

layim.removeList({ type: 'friend'

,id: res.data.id

});

//加入新组 layim.addList({

type: 'friend'

,avatar: res.data.avatar

,username: res.data.user\_name

,groupid: group\_id

,id: res.data.id

,sign: res.data.sign

});

layer.close(index);

} else {

layer.msg(data.msg, {time: 2000});

}

});

}else{

}

}, 'json');

}

});

}

layer.msg(res.msg, {time: 2000});

es){

//删除好友

, removeFriend: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

//询问框 layer.confirm('确定删除该好友？', {

btn: ['确定', '取消'], title: '友情提示', closeBtn: 0,

icon: 3

}, function(){

$.post('/index/Tools/removeFriend', {'user\_id' : friend\_id}, function(r

if(1 == res.code){ layer.msg(res.msg, {time: 1500}); layim.removeList({

type: 'friend'

, id: res.data.to\_id

});

//通知被删除的用户，删除我

var black\_data = '{"type":"delFriend","to\_id":"' + res.data.to\_ id + '", "del\_id":"' + res.data.del\_id + '"}';

socket.send(black\_data);

}else{

layer.msg(res.msg, {time: 1500});

}

}, 'json');

}, function(){

});

}

//举报好友

, reportFriend: function(othis, e){

var friend\_id = othis.parent().attr('data-id');

layer.open({

type: 2,

title: '举报好友', shadeClose: true,

skin: 'layui-layer-molv', //加上边框 shade: 0.3,

area: ['400px', '400px'],

content: '/index/Tools/reportFriend/user\_id/' + friend\_id

});

}

//退出群组

, leaveOut: function(othis, e){

var group\_id = othis.parent().attr('data-id'); var index = layer.confirm('确定退出该群组？', {

btn: ['确定', '取消'], title: '友情提示', closeBtn: 0,

icon: 3

}, function(){

$.post('/index/Tools/leaveGroup', {'group\_id' : group\_id}, function(res

){

if(1 == res.code){ layer.msg(res.msg, {time: 1500}); layim.removeList({

type: 'group'

, id: res.data.group\_id

});

// 退出讨论组

var leave\_data = '{"type": "leaveGroup", "leave\_id":"' + res.da ta.uid + '", "group\_id":"' + res.data.group\_id + '"}';

socket.send(leave\_data);

}else{

layer.msg(res.msg, {time: 1500});

}

layer.close(index);

}, 'json');

}, function(){

});

}

}

//监听签名修改

layim.on('sign', function(value){

$.post(change\_sign\_url, {'sign' : value}, function(res){ if(1 == res.code){

layer.msg(res.msg, {time:1500});

}else{

layer.msg(res.msg, {time:1500});

}

}, 'json');

});

//监听自定义工具栏点击 layim.on('tool(map)', function(insert){

layer.msg('暂无', {time:1000});

});

//监听自定义工具栏点击录音

layim.on('tool(voice)', function(insert, send){

layui.use(['layer'], function(){ var layer = layui.layer;

var box = layer.open({ type: 1,

title: '发送语音',

maxmin: false, //开启最大化最小化按钮 skin: 'layui-layer-molv',

anim: 3,

area: ['200px', '250px'], content: $("#audio\_box"), cancel: function(index){

$("#tips").text('');

$("#a\_pic").attr('src', '/static/common/images/audio.png'); layer.close(index);

return false;

}

});

//点击开始录音

$("#say").bind('click', function(){

$("#tips").text('说话中......');

$("#a\_pic").attr('src', '/static/common/images/audio.gif');

$("#over—say").removeClass('layui-btn-disabled');

$(this).addClass('layui-btn-disabled');

});

var isSend = false;

//结束录音

$("#over—say").bind('click', function(){ if(!isSend){

$("#tips").text('');

$("#a\_pic").attr('src', '/static/common/images/audio.png');

$("#say").removeClass('layui-btn-disabled');

$(this).addClass('layui-btn-disabled'); layer.close(box);

var index = layer.load(1, {

shade: [0.1,'#fff'] //0.1透明度的白色背景

});

setTimeout(function(){

var audioSrc = $("#audio\_src").val(); if('' == audioSrc){

layer.msg('您没有发送语音', {time:1000});

}else{

insert('audio[' + audioSrc + ']'); send(); //自动发送

}

layer.close(index);

}, 1500);

isSend = true;

});

}

});

});

//layim建立就绪

layim.on('ready', function(res){

//点击头像操作

$(".layui-layim-user").click(function(){ layui.use(['layer'], function(){

var layer = layui.layer; layer.open({

type: 2,

title: '修改个人资料',

maxmin: false, //开启最大化最小化按钮 area: ['800px', '600px'], content: chat\_user\_url

});

});

});

>');

//发送消息

layim.on('sendMessage', function(res){

//console.log(res); if(To.type === 'friend'){

layim.setChatStatus('<span style="color:#FF5722;">对方正在输入。。。</span

'}}';

});

}

// 发送消息

var mine = JSON.stringify(res.mine); var to = JSON.stringify(res.to);

var login\_data = '{"type":"chatMessage","data":{"mine":'+mine+', "to":'+to+ socket.send( login\_data );

//在线状态切换

layim.on('online', function(status){

var change\_data = '{"type":"online", "status":"' + status + '", "uid":"' +

uid + '"}';

});

});

socket.send(change\_data);

//获取未读的消息 120s读取一次 setInterval(function(){

$.getJSON(get\_noread\_url, function(res){ if(res.data > 0){

layim.msgbox(res.data);

}

});

}, parseInt(120) \* 1000);

});

# laychat系统源码

#### 源码以及数据库文件：[[点击下载]](http://pan.baidu.com/s/1hrNgLRu) 提取密码是：7cag