# 博客第一天

注册登陆

使用thinkphp的模型进行注册，创建公共文件夹common建立Model方便使用，可以在模型里写方法进行数据添加，然后再控制器执行改方法，目前使用静态方法。前端使用AJAX进行交互

要使用自动添加数据，一定要加入protected $auto=['regtime','salt','token','token\_exptime'];有对应的字段 才能在使用protected function setRegtimeAttr()添加数据。

添加数据注册数据方法 userreg ：：public static function userreg($data)

{ $res = self::create($data,true);//添加数据}

控制器方法调用User:: userreg ($data);

如果在控制器进行数据添加可以使用

$data=input(‘post.’) //获取传来的信息

$user=new UserModel($data);

$user=$ user ->allowField(true)->save();

如果保存成功模型方法返回['code'=>1,'msg'=>'注册成功']；控制器调用方法后接收到信息后，要将返回的信息转为json $res= User::login($data);return json($res);，传给前端，前端接受控制器的信息 如果res.code == "1"=登陆成功，跳转。

检验规则定义 validate 注册时需要对数据进行检验

$userValidate=new UserValidate();

if(!$userValidate->scene('index')->check($data))

protected $rule=[ 'account' => 'require|unique:user',]定义规则

protected $message=[ 'account.require' => '账号不能为空',]定义提示信息

protected $scene = 'reg' => ['account','passwd''],定义使用场景，例如你只需要检查账号密码，那你在这里写账号密码，然后调用方法是

$userValidate->scene('reg')->check($data),在这里加上就行。

遇上的问题:验证码验证错误->缓存引起的错误，MD5加密存进数据库登陆取出来是检验失败，2个值不一样🡪原因不明，然后采用直接存进数据库，拿出来再加密。

流程 前端（用户填写信息注册）->控制器接收参数后:调用模型里面定义的方法（userreg）->模型（方法）把运行后的结果再返回给控制器->控制器将接收结果改为json再发放给前端->前端对接收到的信息进行分类：code=1 成功跳转 code=0 失败。

# 博客第二天

邮件发送

首先要登陆163邮箱，开启SMTP服务，开通授权码

主要是调用PHPMailer.在用户注册完成后系统会发送激活邮件$result=Base::sendTextmail($res->email,$res->account,url('index/Login/active','','',true)."?token=".$res->token); 发送邮件后用户进行激活，将isactive改为1.激活原理：控制器根据链接传过来的token进行查找数据库，查找到数据库相应的用户 需要查找的字段【token,token\_exptime,account,passwd,email】，如果没有这个用户，返回'code'=>0 如果存在该用户继续往下执行 首先判断当前时间是否超过激活时间token\_exptime 如果不是 继续往下执行

$res->isactive=1;$result=$res->save();

if($result===0)代表没有执行，证明isactive已经激活1，

if($result===1)代表执行，证明isactive成功激活1，

if($result===flase)代表错误，证明Token已经被更新。

链接失效后，重新发送邮件

重新发送邮件需要更新token和token\_exptime db('user')->where('token',$token)->update(['token'=>$newstoken,'token\_exptime'=>$time]); 因为这步将$token，所以$result===flase证明Token已经被更新。

重新发送邮件

**做这个功能时发生的问题：一.安装PHPMailer出现问题，多次安装失败，然后换了一个thinkphp文件包来写项目，前一个文件包安装不了问题没有解决。**

**二．邮件发送不成功，有2个至没有设置好，$mail->SMTPSecure要使用ssl，对应的端口号$mail->Port为994或者465.**

**还有使用了try{}catch{}导致发生了错误没有发送成功，后面改为判读if()就成功了，在后面用回try{}catch{}也没有发生错误。可能是一些神奇因素影响，原因不明。**

**流程：注册成功->发送邮件->用户进行激活->当用户运行这个链接，执行active（）这个方法将isactive改为1，->链接如果超时，重新发送邮件激活。**

# 博客第三天

判断是否登陆

在公共控制器中创建函数 isLogin()，判断是否存在用户名和code->{用于安全，后面解释}如果不存在，返回false。存在返回true。

基于安全问题进行改进：

用户行为：用户可能会直接在浏览器创建cookie然后绕过登陆

应对：登录的成功的时候产生一个cookie->code，这是一个固定值，该值由（用户账号+SALT ）进行MD5加密，其可以看作是一个固定值，如果有用户伪造cookie->name 来产生一个cookie->code如果产生的cookie->code与后台加密产生的一样就可以登陆成功，但是不大可能，判断登陆时：对浏览器中cookie->name先用decrypt（）进行解密，解密后再与SALT进行MD5加密，与浏览器中已有的cookie->code进行对比，相同即登陆，不同返回登陆。所以如果要绕过登陆，用户需要知道 SALT值 and 我对code的加密方式 AND 我对cookie->name的加密解密方式，因为出现在浏览器中的cookie->name已经在后台进行了encrypt（）加密。所以用户先要进行解密。

用户登录失败3次，要验证码验证。

1. 服务器记录用户错误次数，$logincount，要根据不同的用户来显示是否要出现验证码，所以要将用户名存进缓存cache($data['account'].'retry')，用户名存进缓存后就创建Cookie: cookie('codecount',$count+1,3600); 因为$coun包含了用户信息，所以不同用户Cookie不同，count也不同，后台验证3次后返回信息加上count同时code设置为3，来告诉前端已经超过3次，出现验证码
2. 后台控制器将logincount存进前段页面

前段根据code返回值来确定执行哪一步

密码错误返回code=2，然后执行count++ 次数加以，因为后台已经记录了你的次数，所以不加也行 登陆后清除缓存cache($data['account'].'retry', NULL);因为cookie可以用浏览器清除，所以只需要清除缓存即可。

Blog主页需要登陆，所以要定义行为，行为在thinkphp有封装，

Tags.Php,创建在需要限制的根目录下 这文件是统一定义行为

return [

'action\_begin'=> [

'app\\blog\\behavior\\Login',

],

]

然后创建行为文件夹在文件夹里进行 行为的定义的PHP文件，例如Login PHP

public *function* actionBegin(&$params)

{

$res=Base::isLogin();

if(!$res)

{

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

}

}

actionBegin 行为标签 用作与Tags.php绑定行为。

博客后台管理

上传图片

Thinkphh封装好了上传方法public static function upload

$info = $file->move(ROOT\_PATH . 'public' . DS . $dir);

如果用户已经传过logo但是想更改的话，就需要把之前已经上传过的图片进行删除，执行upload方法是传入3个参数，一个是文件$file，一个是上一次存进数据库的logo数据$oldfile，最后是你想存放的文件夹$dir->有默认值 默认upload upload方法里。如果$oldfile不为空，

if($oldfile!==null){

//项目根目录/public/$oldfile

$path=ROOT\_PATH.'public/'.$oldfile;

if(is\_file($path) && file\_exists($path)){

@unlink($path);

}

}

# 个人资料管理

先将用户原有的数据显示出来，通过$result = $result->toArray();将数据变为数据，让后用assign穿给前台

如果有登记先个人资料，首先对前台传来的数据进行检验，然后要进行个人资料的查询，查询方法：

$res=db('user')->where('account',session('username'))->find();如果查到先获取ID，将ID传进新表，将传来的数据经行更新，如果查不到，直接返回。

可以利用model进行更新$cateMode=model("Blogcate");

# 个人密码修改

获取前段页面数据，检查旧密码是否与之前相同，如果不相同，不通过，如果相同，允许修改。

# 分类列表

添加分类

添加数据进blogcate表，里面带有文件上传，使用LayUI进行 依旧使用thinkphp封装好的函数进行

删除分类

首先要判断按钮是否带有参数id。然后根据获取按钮带的数据 用AJAX进行传递参数，从而删除数据

编辑分类

根据按钮传来的cid，获取到blogcate表中相应的数据，用户点击提交，先判断用户是否修改了数据，如果修改了返回code=》1 $result=$cateMode->isUpdate()->save($data); 修改成功。如果cid==null 参数错误。使用到弹出层，Yayui里面的弹出层，

，然后将编辑分类改为AJAX提交，还要用到弹出层的关闭

Thinkphp5 自定义跳转

默认跳转页面对应的模板文件 // 'dispatch\_success\_tmpl' => THINK\_PATH . 'tpl' . DS . 'dispatch\_jump.tpl', // 'dispatch\_error\_tmpl' => THINK\_PATH . 'tpl' . DS . 'dispatch\_jump.tpl', //自定义模板文件 'dispatch\_success\_tmpl' => APP\_PATH . 'tpl' . DS . 'success.tpl', 'dispatch\_error\_tmpl' => APP\_PATH . 'tpl' . DS . 'error.tpl'

具体模板文件的放置位置呢可以根据自己的喜好进行设置，在这里我将其设置到应用目录下的tpl目录中。在这里两个模板文件可以共用一个，也可以分开指定，这个根据自己的实际情况进行指定。

关于可以变量可以参考自带的模板，常用的如下：

$code //状态码

$msg //提示信息

$wait //等待时间

$url //跳转地址

$data //自定义数据,可以是数组，也可以是字符串

[danger]变量在模板中的使用和模板变量的使用是一样的。

2. 页面跳转

页面倒计时跳转的实现，使用的是js的方式，js部分的代码如下：

<script type="text/javascript">

(function(){

var wait = document.getElementById('wait'),

href = document.getElementById('href').href;

var interval = setInterval(function(){//创建一个定时任务

var time = --wait.innerHTML;

if(time <= 0) {

location.href = href;//实现跳转

clearInterval(interval);

};

}, 1000);

})();

</script>

3. 资源文件的引入

资源文件的引入和页面模板中的引入方式是一样的。 例：

<script src="\_\_COMMON\_\_/js/jquery.min.js"></script>

<script src="\_\_COMMON\_\_/layer/layer.js"></script>

Thinkphp5 自定义分页

我们可以在根目录下建立存放自定义分页的文件夹，访问自定义分页文件，但需要注册命名空间

我们可以在公共文件中加入\think\Loader::addNamespaceAlias('mywork','app\my\\');注册命名空间。调用时我们

$res=$PicsModel->order("addtime Desc")

->paginate($limit,false);

相册列表中，根据相册的cid，页面中返回对应的分类，方法是在模型中先在模型中根据cid查找相册分类表获取分类名称，再返回，

public function getCidAttr($value)

{

$cate=Picscate::where('cid',$value)->value("name");

return $cate;

}

### 刷新

location.reload();

thinkphp 命令

php think build --module test 创建模块

php think make:controller admin/index 创建控制器

php think clear 清除缓存

php think make:model admin/index 创建模型

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

$query->order('id', 'Asc');

插件安装

实现该功能的首要条件要在原插件包下添加if(!is\_file(ADDON\_PATH.$name.DS."config.php")){

continue;

}如果没有这个判断，插件是否安装都会显示，是因为是自动识别，这个是要是判断是否存在config文件

要使用插件开发，先要安装think-addons插件

插件放在Addons文件夹下，创建插件文件夹还有里面的文件要和文件夹名称相同，要使用插件需要在appcation\extra下登记。

插件发开后，用户可以选择自己喜欢的插件进行安装，后台管理中，插件安装列表实现：我们需要在控制器中首先获取插件的信息：$info->在插件开发时已经登记，在think-addons安装包下的common文件下封装了get\_addon\_list 方法，可以直接调用该发放来过去插件信息。该方法实现原理：首先找到ADDons文件夹，让后遍历文件夹，如果文件夹目录等于 “.”&&”..”就跳出，如果遍历到的是文件也直接跳出来，如果是addons\test。要判断test是否是一个文件目录,因为test可以文件，图片，目录，所以上一步只是排除了文件，未排除其他可能，所有我们还要判断他是否为文件目录。如果不是跳出，如果是继续判断是否存在与目录名字相同的PHP文件。如果是继续判断PHP是否为插件文件，拿类名进行判断。是就继续 创建一个插件对象$addon = new $class();判断是否是有效插件$addon->checkInfo()；然后$info=$addon->info;获取插件信息，再判断是否存在config.php文件，如果存在代表插件已经安装，状态为1

if (!is\_file($info\_file)){

$info['status']=0;

}else{

$info['status']=1;

}

安装插件

生成对应的文件进行标识-> config文件。

首先要在插件的安装方法install下,编写安装程序，获取插件信息，创建config文件$result=create\_config($content);插件控制器Addons->install方法，控制插件安装，先创建插件对象$class=get\_addon\_class($name);$result=(new $class)->install();返回结果判断失败错误。

# 卸载插件

该功能是以有没有config文件为条件，所以卸载插件就是删除config文件。

$name=$this->getName();

$file=ADDON\_PATH.$name.DS.config("addons.fileflag");

if(!is\_file($file))

{

return true;

}

$result=@unlink($file);

if(!$result)

{

return false;

}

return true;

## 安装插件->带静态资源，数据库

如果有静态资源，将插件中的静态资源转移到public/static下，需要

//判断是否有静态资源文件

if(is\_dir(ADDON\_PATH.$name.DS."static"))

{

//将静态资源文件移动到public/static/addons

@rename(ADDON\_PATH.$name.DS."static".DS.$name, ROOT\_PATH."public/static/addons".DS.$name);

}

如果需要数据库

安装的同时执行数据库语句，将sql文件放在插件下

卸载反之而行。

## 离线安装插件

原理🡪前端接收安装包->后台解压文件->将压缩包下的文件及目录转移

解压文件

打开zip文件包，获取压缩包文件数量 $docnum = $zip->numFiles; 遍历压缩包文件 for($i = 0; $i < $docnum; $i++) {} 判断压缩包里面的目录文件if($statInfo['crc'] == 0) 默认$statInfo['crc']==0为目录。找到该目录后在addon根文件下创建该目录。mkdir($toDir.'/'.substr($statInfo['name'], 0,-1));如果不是目录文件就将文件转移到刚创建的文件夹下copy('zip://'.$zipfile.'#'.$statInfo['name'], $toDir.'/'.$statInfo['name']);原理将缓存文件的文件转移到刚新建的文件夹下。判断是否是目录文件需要用到2个循环，因为循环时，率先循环的是文件，然后再到目录，如果放在一起的话会发生要转移到的文件夹不存在，因为还没有创建。

解压代码：protected function uzip($filename){

//解压缩

$zip = new \ZipArchive;

//要解压的文件

$zipfile = $filename;

$res = $zip->open($zipfile);

if($res!==true){

return false;

}

//要解压到的目录

$toDir = ADDON\_PATH;

if(!file\_exists($toDir)) {

mkdir($toDir,755);

}

//获取压缩包中的文件数（含目录）

$docnum = $zip->numFiles;

$addonname="";

//遍历压缩包中的文件

for($i = 0; $i < $docnum; $i++) {

$statInfo = $zip->statIndex($i);

//判断是否为目录，是为0

if($statInfo['crc'] == 0) {

// dump($toDir.'/'.substr($statInfo['name'], 0,-1));die;

mkdir($toDir.'/'.substr($statInfo['name'], 0,-1));

$addonname=substr($statInfo['name'], 0,-1);

}

}

for($i = 0; $i < $docnum; $i++) {

$statInfo = $zip->statIndex($i);

if($statInfo['crc'] != 0) {

copy('zip://'.$zipfile.'#'.$statInfo['name'], $toDir.'/'.$statInfo['name']);

}

}

return $addonname;

}

权限添加及排序

添加权限，要显示出等级分类。层次感

采取方法

用pid来表示自己的所属分类，如顶级id是2，那属于这个分类的第一层插件管理pid即为2 他自身id为5，如果继续分类，pid而且该分类属于插件管理，那么pid即为5。Pid所属的一直是上一层id。

问题：系统管理是第一层，基本设置是第二层，插件安装时第三层，二三层都属于第一层。如何让插件安装在系统管理的范围内，因为插件安装的pid是属于基本设置。

因为基本设置在系统管理里面，所以我们用递归就可以将同一类的分归一组。

方法：

public static *function* getRuleall($pid=0,$level=0){

$res=self::where('pid',$pid)->select();

static $arr=[];

foreach ($res as $v){

$v['level']=$level;

$arr[]=$v;

self::getRuleall($v['id'],$level+1);

}

return $arr;

}

代码解释

该方法要重第一层开始执行，所以要传第一层需要的参数。

$pid=0代表所属分类 $level=0代表所在层数

先找出顶级的分类，创建一个静态数组🡪一定要静态数组。因为使用静态数组可以把上一次递归的值保存下来。第一次使用该函数得到所有的顶级分类。然后第二次递归时使用的是第一次所得到的顶级分类的id来做pid。因为第二次递归属于下一层，所以$level要加一。因为使用的是顶级分类的id来作为pid进行函数调用，所以第二次得到的数组放在某一个顶级id的数组内。第三次同理，刚开始与第一次有点区别而已。

无限权限

Thinkphp模型事件

Init是注册该模型事件 做删除之后出发的事件

protected static *function* init()

{

self::event('after\_delete', *function* ($data) {

$uid=$data->id;

db('auth\_group\_access')->where("uid",$uid)->delete();

});

} 删除之后

# 数据表优化 百度解释

当我们使用mysql进行delete数据，delete完以后，发现空间文件ibd并没有减少，这是因为碎片空间的存在，举个例子，一共公司有10号员工，10个座位，被开除了7个员工，但这些座位还是保留的，碎片整理就像，让剩下的3个员工都靠边坐，然后把剩下的7个作为给砸掉，这样就能释放出空间了

OPTIMIZE TABLE reorganizes the physical storage of table data and associated index data, to reduce storage space and improve I/O efficiency when accessing the table.

好处除了减少表数据与表索引的物理空间，还能降低访问表时的IO，这个比较理解，整理之前，取数据需要跨越很多碎片空间，这时需要时间的，整理后，想要的数据都放在一起了，直接拿就拿到了，效率提高

拿一张大表做碎片整理，整理之前是96G

[root@localhost myshard]# du -ch tbl\_immsg\_bigo\_96.ibd |grep total

3.5G    total

当执行命令时

optimise table tbl\_immsg\_bigo\_96;

会有很多慢查询的告警，在告一个waiting for table metadata lock的状态

ID: 121

USER: db\_myshard\_rw

HOST: 127.0.0.1:56326

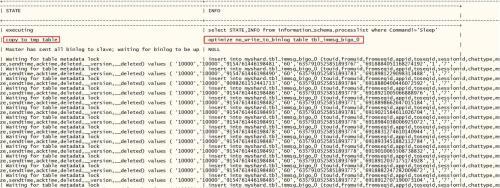
DB: myshard

COMMAND: Execute

TIME: 1214

STATE: Waiting for table metadata lock

INFO: insert into myshard.tbl\_immsg\_bigo\_0 (touid,fromuid,fromseqid,appid



这是因为optimize table的本质，是alter table

mysql 5.5 的改表过程如下

1.创建一张新的临时表 tmp

2.把旧表锁住，禁止插入删除，只允许读写 （这就是为什么上面的insert语句都停留在waiting for table metadata lock）

3.把数据不断的从旧表，拷贝到新的临时表，(这就是上面报copy to tmp table)

4.等表拷贝完后，进行瞬间的rename操作

5.旧表删除掉

所以optimize最大的问题是锁表，锁表会导致insert，delete，update语句堵住，上面等待了1214秒，还在继续，所以第一个结论：在使用optimize table的时候，确保不要有任何dml语句，确保业务切走，否则可能会出事故

为什么要锁表呢？

alter过程里，数据不停从旧表拷贝到新表，如果这个时候旧表被delete了数据了，那旧表与新表的数据就不一致了，到最后rename 新表 to 旧表表名 时候，数据量就多了

如果在拷贝数据的过程中，对旧表数据的delete，同时对新表也做delete,那数据就一致了，对于update和insert也一样，这个功能可以通过 insert触发器，delete触发器，update触发器实现

pt-online-schema-change就利用3个触发器完成在线改表，也能完成在线碎片整理，命令使用

--alter="ENGINE=InnoDB"

相当于optimize table的效果

具体命令如下，最好放在脚本里面实现，因为一次不止整理一个表，可以把整个数据库的表都碎片整理

 pt-online-schema-change

  -h地址

  -P端口号

  -u用户名

  -p密码

  --database=数据库

  t=表名字

  --charset=utf8

  --max-lag=300

  --check-interval=5

  --alter="ENGINE=InnoDB"

  --max-load="Threads\_running:400"

  --critical-load="Threads\_running:400"

  --nocheck-replication-filters

  --alter-foreign-keys-method=auto

  --execute

使用pt-online-schema-change可以跳过锁表的坑

本来使用碎片整理是因为磁盘使用率96%，但碎片整理时发现磁盘使用率变成99%，差点就爆了

Filesystem      Size  Used Avail Use% Mounted on

/dev/sda2        58G  2.7G   53G   5% /

tmpfs         24G     0   24G   0% /dev/shm

/dev/sda1       485M   32M  428M   7% /boot

/dev/sda5       1.6T  452G  1.1T  31% /data

/dev/sdb1       1.3T  1.2T   25G  99% /data1

这是因为在把旧表拷贝到临时表的时，会把表数据复制一份数据，10G的表，可能复制出来是7G，这个过程磁盘会快速消耗，不小心就会把磁盘撑满造成数据丢失了

为了避免这个坑，应该把整个数据库的表，按照体积从小到大排序,并且把索引文件，表结构去掉，为了方便显示出体积，这里加了一个l参数，实际上是不加的，只获取表名字，然后重定向一个文件里，碎片整理就按照这个顺序

ls -lSr --ignore="\*.frm"

-rw-rw---- 1 mysql mysql   4096 Jul 25 12:33 tables\_priv.MYI

-rw-rw---- 1 mysql mysql   4096 Jul 25 12:33 procs\_priv.MYI

-rw-rw---- 1 mysql mysql   4096 Jul 25 12:33 columns\_priv.MYI

-rw-rw---- 1 mysql mysql   5120 Jul 25 12:33 proxies\_priv.MYI

-rw-rw---- 1 mysql mysql   5120 Jul 25 12:43 db.MYI

-rw-rw---- 1 mysql mysql   8928 Jul 25 12:33 help\_relation.MYD

-rw-rw---- 1 mysql mysql  16384 Jul 25 12:33 help\_keyword.MYI

-rw-rw---- 1 mysql mysql  18432 Jul 25 12:33 help\_relation.MYI

-rw-rw---- 1 mysql mysql  20480 Jul 25 12:33 help\_topic.MYI

-rw-rw---- 1 mysql mysql  22078 Jul 25 12:33 help\_category.MYD

-rw-rw---- 1 mysql mysql  89241 Jul 25 12:33 help\_keyword.MYD

-rw-rw---- 1 mysql mysql 419392 Jul 25 12:33 help\_topic.MYD

可以写一个脚本，统计每个表整理的时间，整理前后的体积比较

## 数据库备份操作

主要运用插件封装的方法进行操作，调用插件方法。

# 实现日志记录的方法

使用thinkphp行为

行为作用，在控制器执行前首先执行该行为，包括控制器中的每一个方法。所以在执行每个方法前 行为方法都将获取控制器和调用方法的信息。获取代码如下

$data['url']=request()->controller()."/".request()->action();

用户信息重session中获取

$uid=session("uid","","admin");

$data['operator']=$uid?$uid:0-🡪该方法是让没用登陆状态下用户id为0。如果没有这个，方法无法执行赋值，引起报错。

要在配置文件夹配置Logs.php。针对节点进行配置，如果没有在extra文件夹下进行配置，也可以自己在控制器下进行配置

config("logs.Login/logout",'退出登录');

问题一定要有

$logs=config("logs");-创建一个logs配置。

全代码

public *function* appEnd(){

//记录日志

//1.获取日志信息

$data['url']=request()->controller()."/".request()->action();

$uid=session("uid","","admin");

$data['operator']=$uid?$uid:0;

$logs=config("logs");

if(isset($logs[$data['url']])){

$description=$logs[$data['url']];

}else{

$description=db("auth\_rule")->where("name",$data['url'])->value("title");

}

$data['description']=$description?$description:"未知";

$data['operate\_time']=time();

$data['operate\_ip']=request()->ip();

LogsModel::create($data);

}

先检查规则表有没有该行为，没有就自己配置