包含视频（短链生成器）

[Vue补充笔记 1](#_Toc13817)

[秘钥生成 1](#_Toc24947)

[View::make VS view() 2](#_Toc9283)

[View 传递参数至页面的两种写法 2](#_Toc26499)

[全局视图参数 3](#_Toc10341)

[路由规则：限定参数仅能是整数 3](#_Toc31950)

[路由规则：可选路由参数 4](#_Toc21916)

[路由抛出404异常处理 4](#_Toc14214)

[blade模板 子模板覆盖与继承 5](#_Toc16759)

[父界面section在子界面继承 6](#_Toc30341)

[Laravel使用原生的数据库查询 7](#_Toc30952)

[数据库事务与锁 9](#_Toc32392)

[数据库请求构建器 10](#_Toc13058)

[Where子查询与数据集查询 12](#_Toc17013)

[乐观锁与悲观锁 13](#_Toc22186)

[Eloquent默认表名 13](#_Toc14032)

[Eloquent可以任意的使用查询生成器的方法 13](#_Toc21181)

[Eloquent插入数据的两种办法 14](#_Toc20931)

[Eloquent模型Update的两种方法 14](#_Toc1149)

[Eloquent模型的$guarded , $fillable , $table都是每一个模型必须的 15](#_Toc9311)

[Eloquent模型关联表where查询（是关联表不是主表） 15](#_Toc1596)

[使用数据库迁移来修改表 15](#_Toc7944)

[资源路由限制 16](#_Toc20648)

[表单校验的传统写法（非控制器自带方法） 16](#_Toc25134)

[短链URL生成 17](#_Toc11039)

[项目展示 17](#_Toc17842)

[数据库迁移定义 17](#_Toc3059)

[控制器视图内容 18](#_Toc21686)

[路由设置 19](#_Toc20362)

[具体生成短链URL逻辑 20](#_Toc10407)

[让短链URL可以访问 20](#_Toc9657)

[利用队列发送邮件，降低瞬时处理压力 20](#_Toc9653)

[Laravel邮件发送 24](#_Toc20181)

# Vue补充笔记

## 秘钥生成

如果不是从composer生成下来的项目是不会自己创建key，没有key的话，csrf是不能够运行，项目会报错，所以我们需要使用php artisan key generate来进行生成.



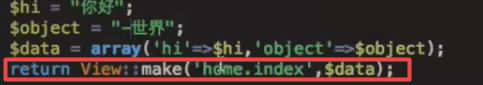
## View::make VS view()



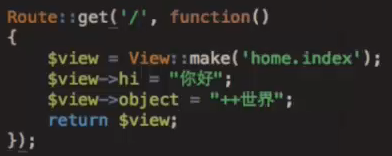


### View 传递参数至页面的两种写法

其实我一直觉得使用view(‘home.admin’,compact(‘a’,’b’,’c’))都是比较好用的.

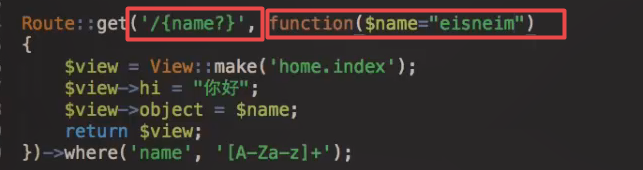


第四种写法

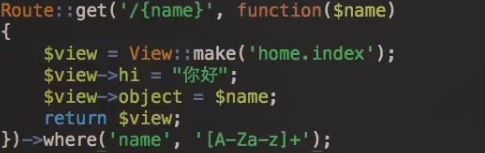




### 全局视图参数



### 路由规则：限定参数仅能是整数



### 路由规则：可选路由参数



### 路由抛出404异常处理



### blade模板 子模板覆盖与继承

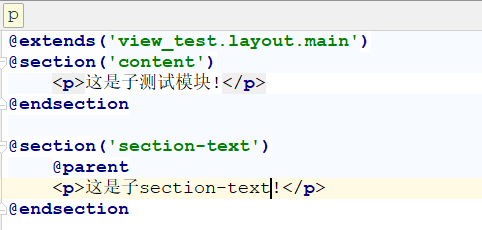
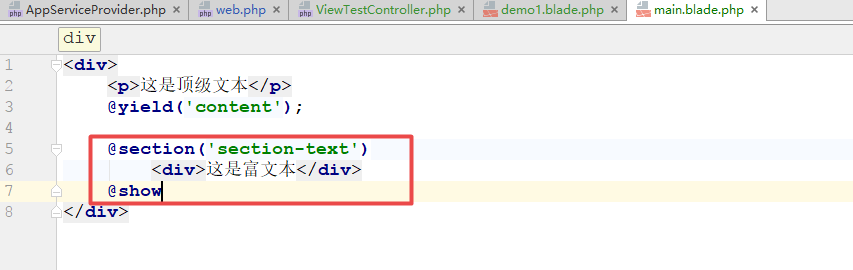




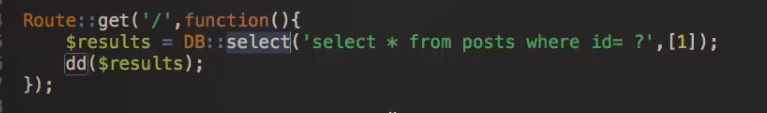
### 父界面section在子界面继承

一定要使用@show

如果写@endsection,会出现内容隐藏的现象



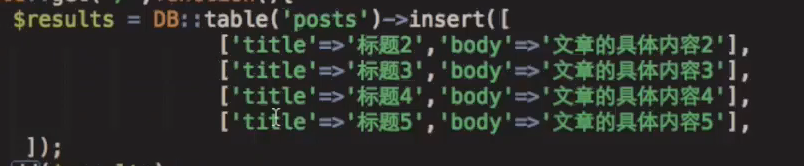
### Laravel使用原生的数据库查询

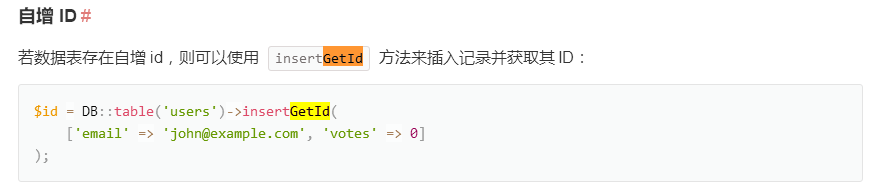


### 数据库事务与锁

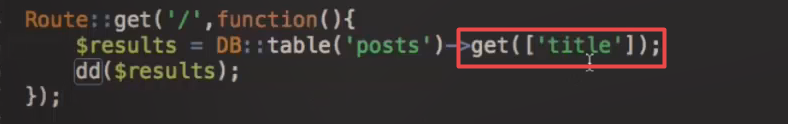


### 数据库请求构建器





Get方法指定显示查询的内容



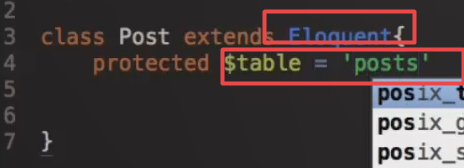
### Where子查询与数据集查询



### 乐观锁与悲观锁



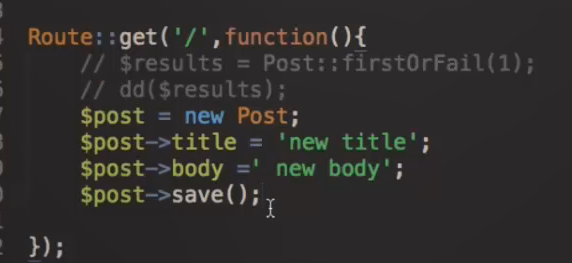
### Eloquent默认表名



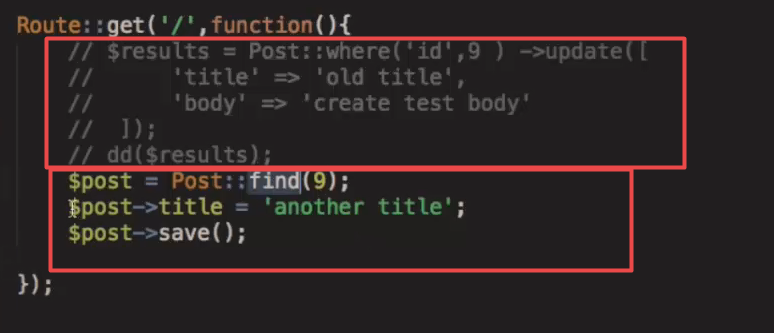
### Eloquent可以任意的使用查询生成器的方法



### Eloquent插入数据的两种办法



### Eloquent模型Update的两种方法



### Eloquent模型的$guarded , $fillable , $table都是每一个模型必须的

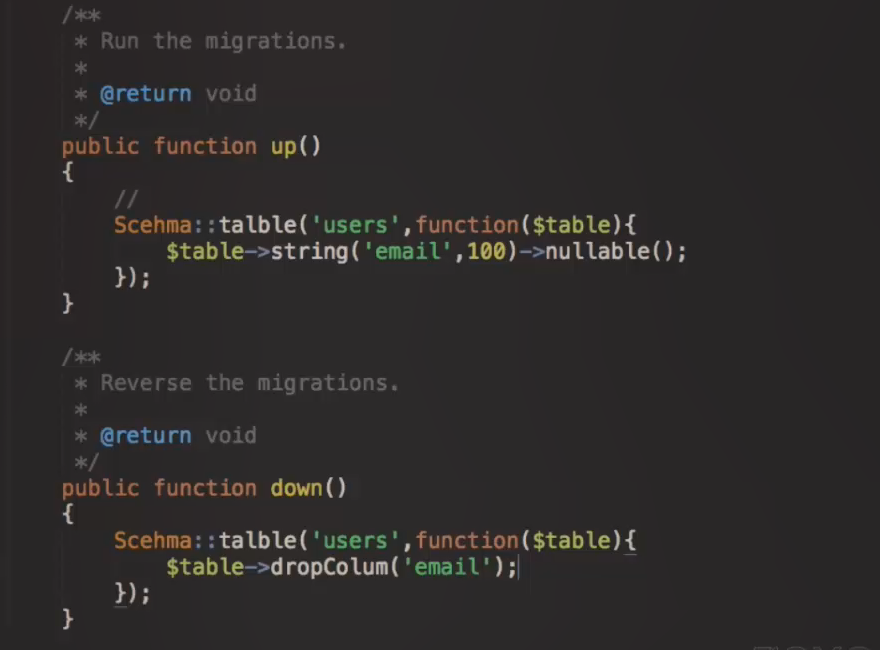
### Eloquent模型关联表where查询（是关联表不是主表）



### 使用数据库迁移来修改表

首先必须添加一个新的数据库迁移的新文件

必须将原来的表的列先删除掉，不然容易报错,



### 资源路由限制



## 表单校验的传统写法（非控制器自带方法）

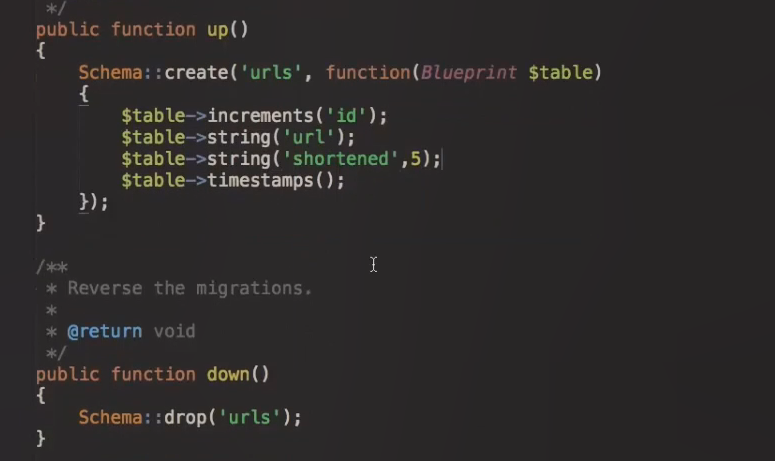
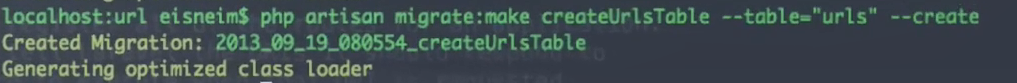


## 短链URL生成

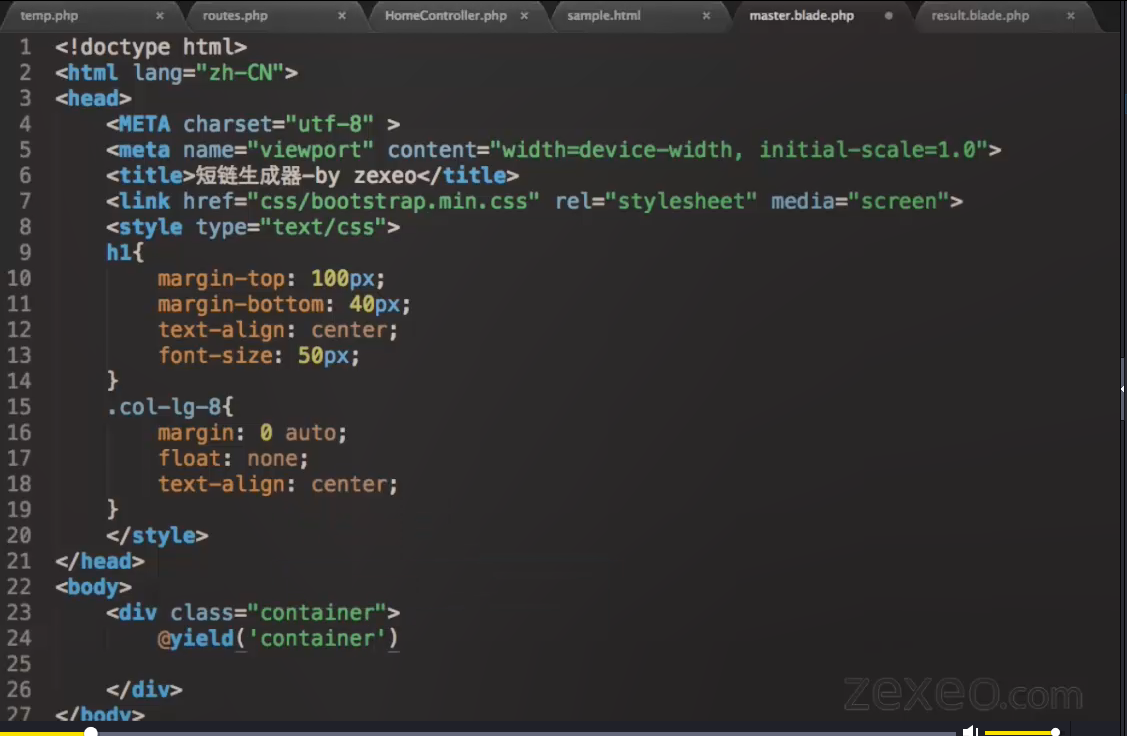
### 项目展示

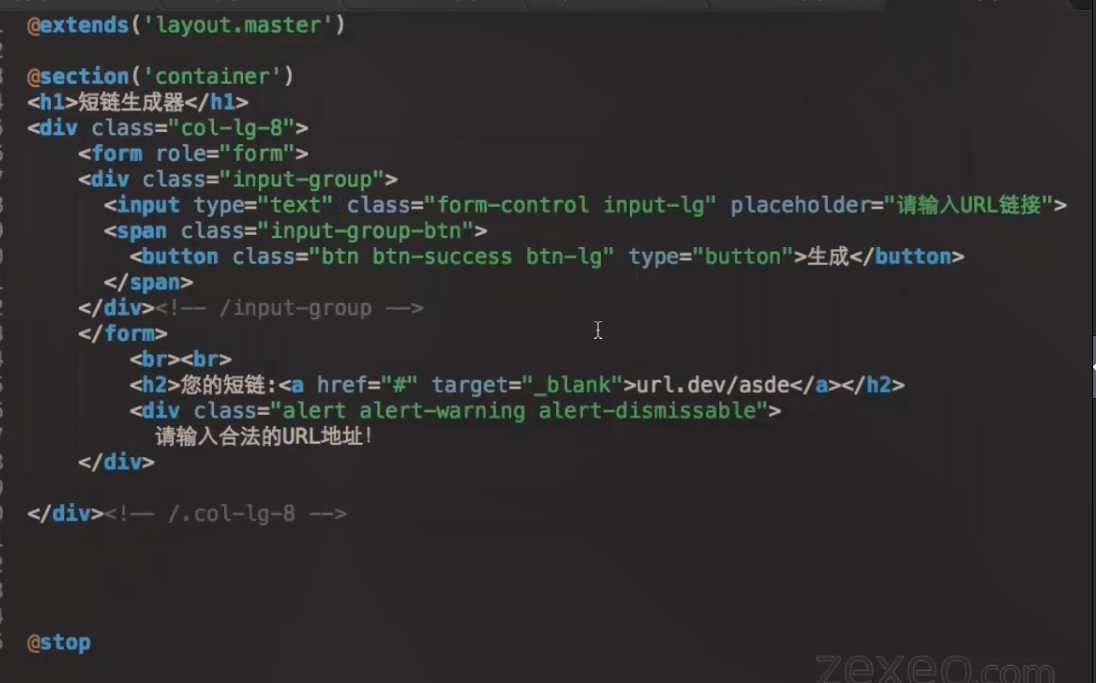


### 数据库迁移定义



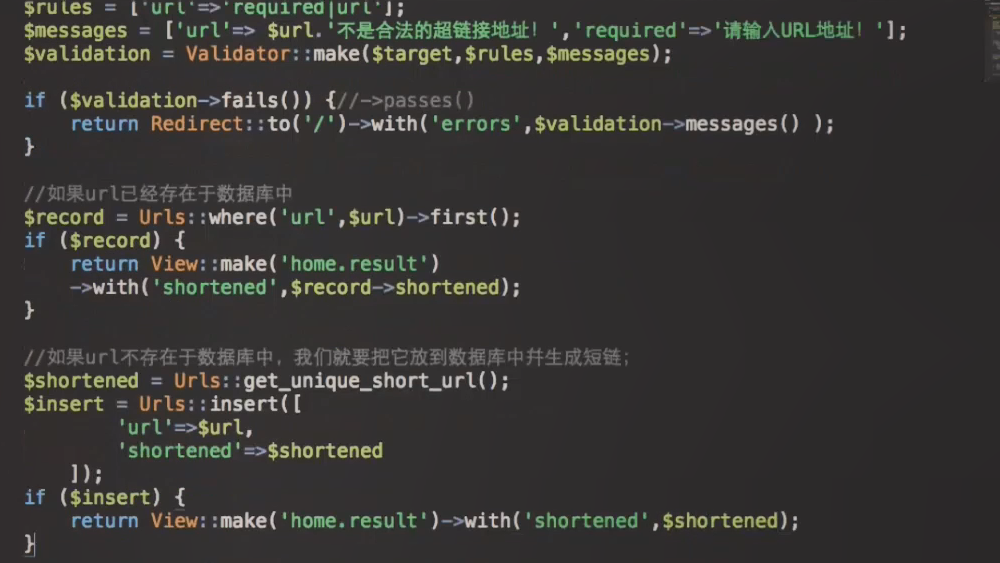
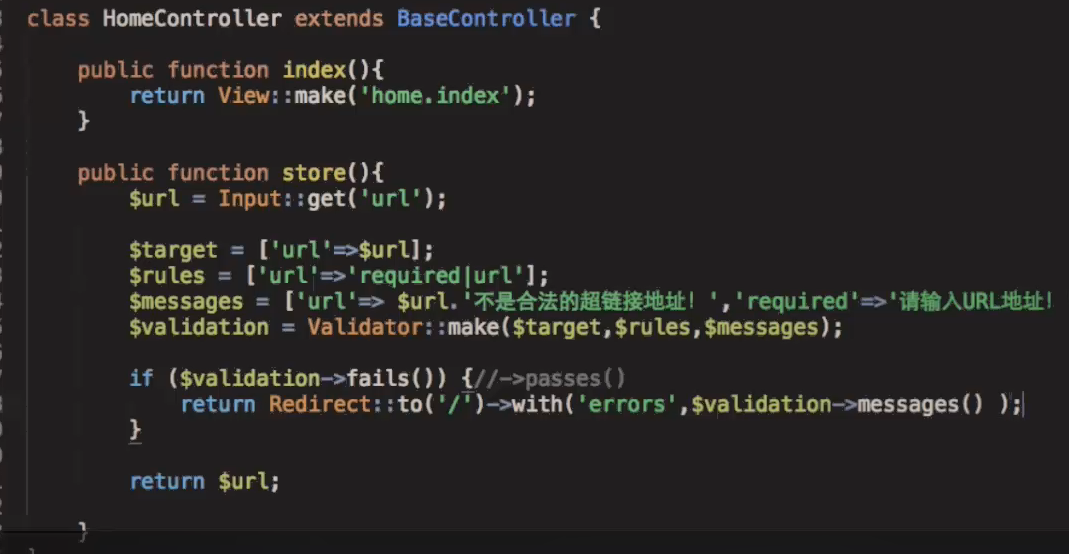
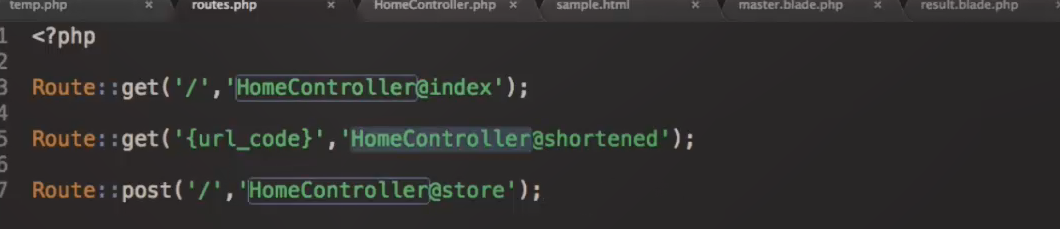
### 控制器视图内容



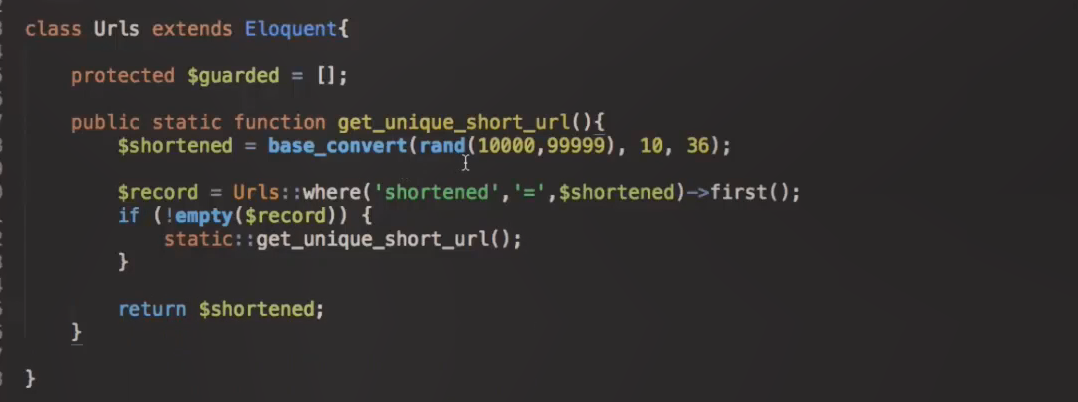


### 路由设置

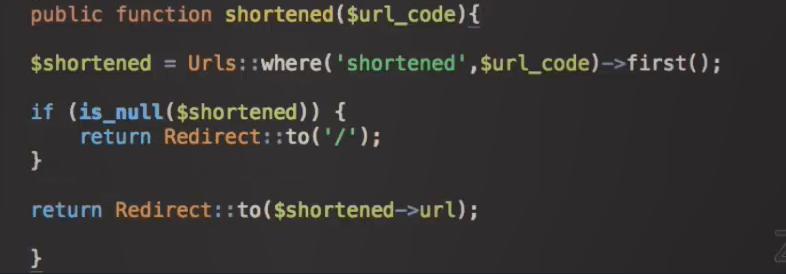
当前是Laravel4的写法，后面补充Laravel5写法



### 具体生成短链URL逻辑



### 让短链URL可以访问



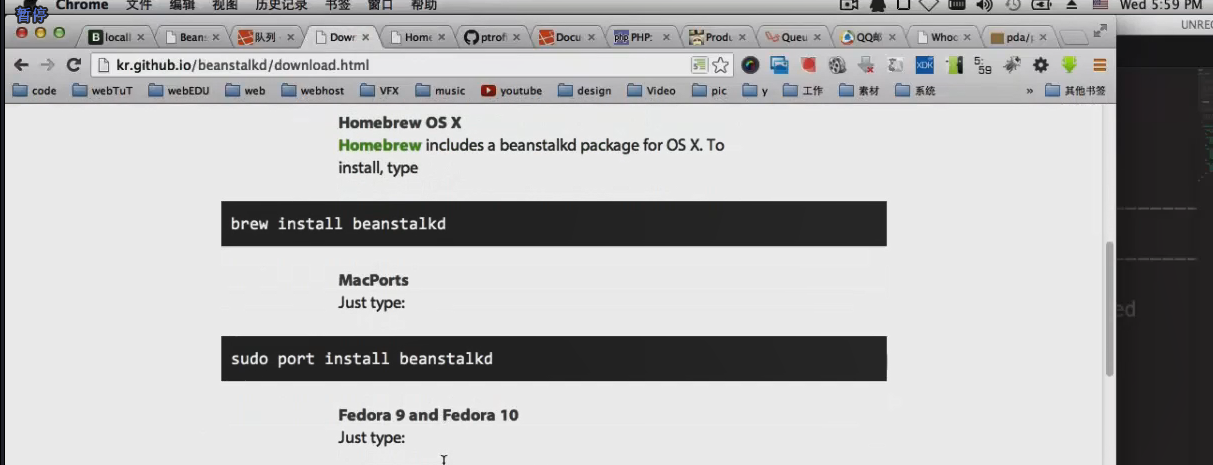
## 利用队列发送邮件，降低瞬时处理压力

异步处理，延迟发送

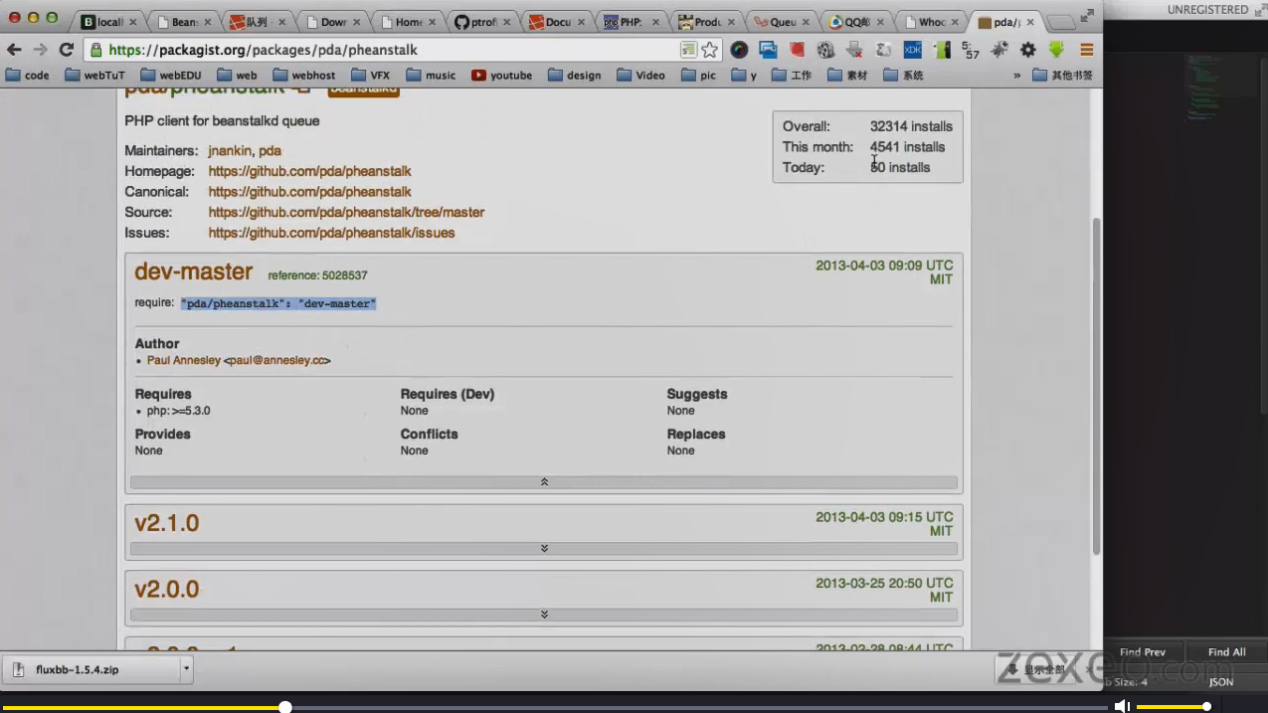
减缓服务器压力

本次队列使用BeanTalked来进行处理

安装地址



Php插件

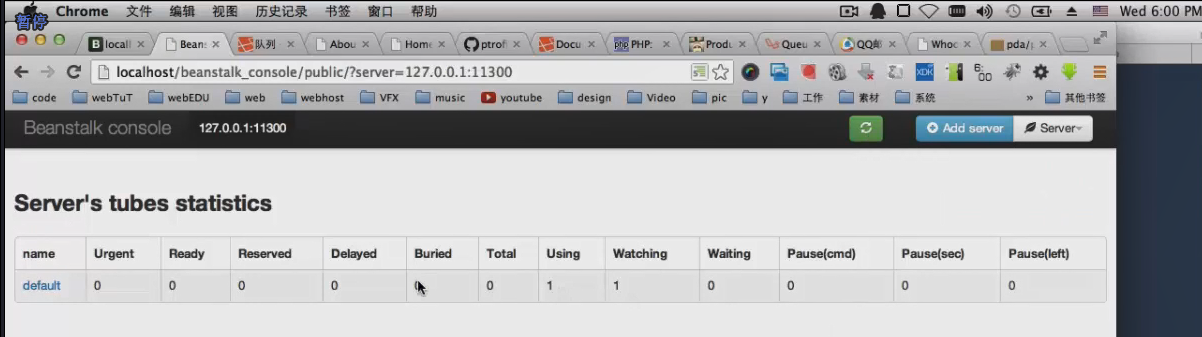


Composer install “pda/pheanstalk”

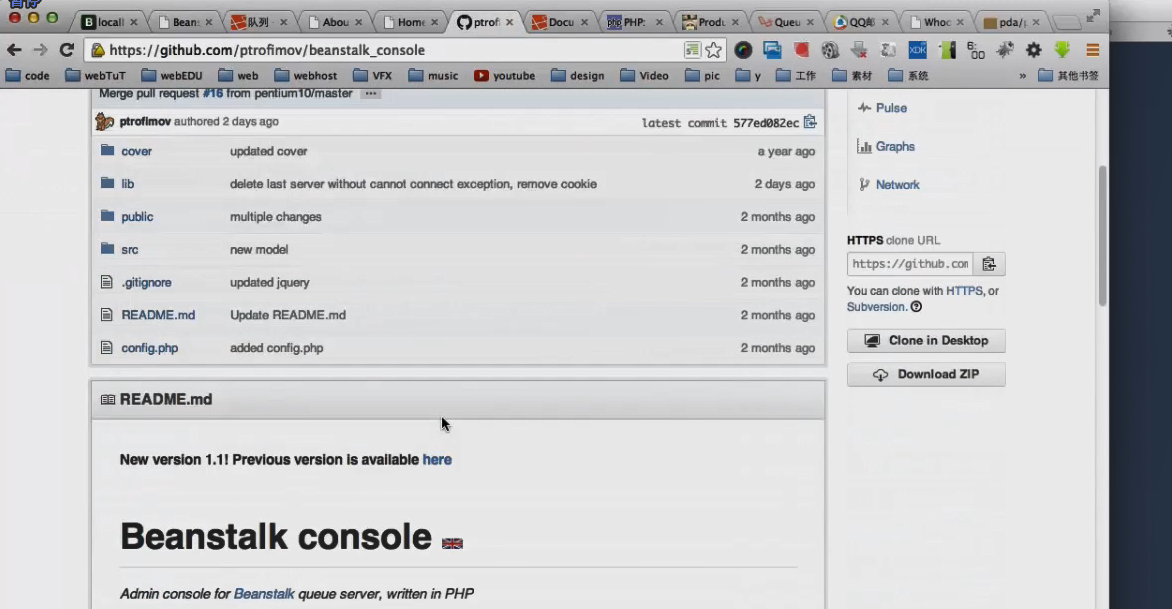
BeansTalk运行指令



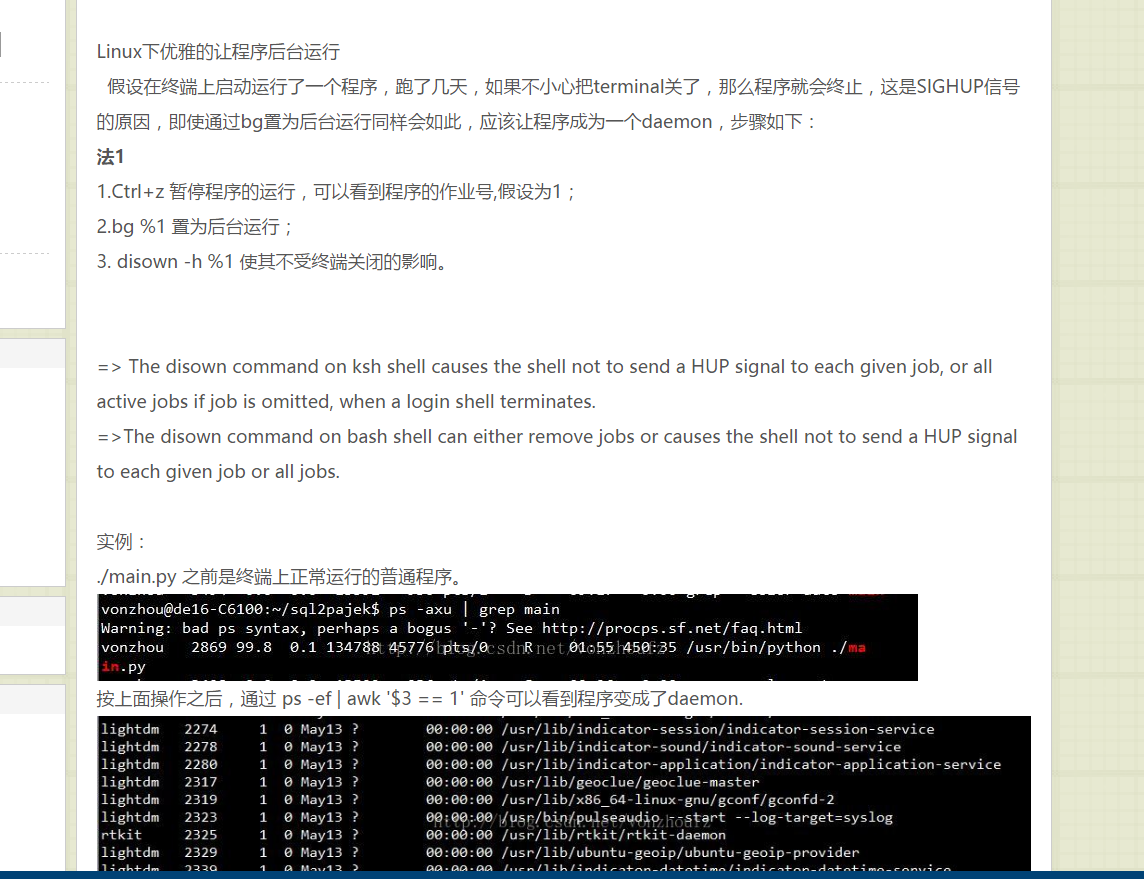
队列管理工具

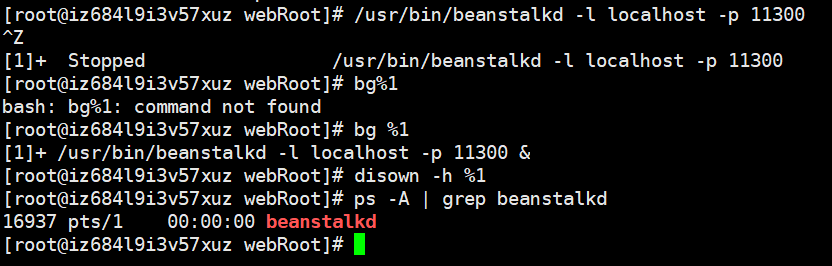


管理工具下载



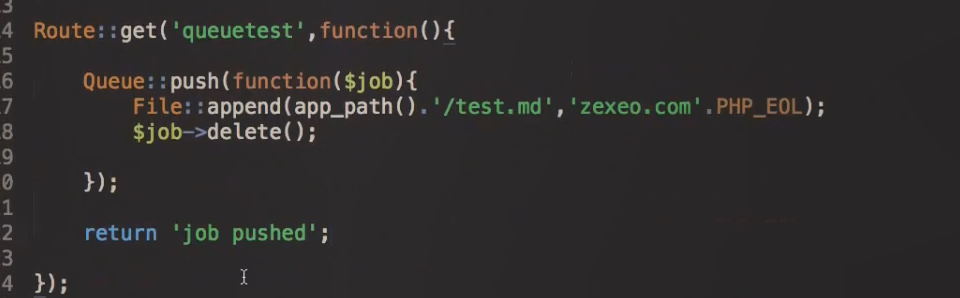
由于Beanstalk不支持后台运行，所以我们需要自己写shell



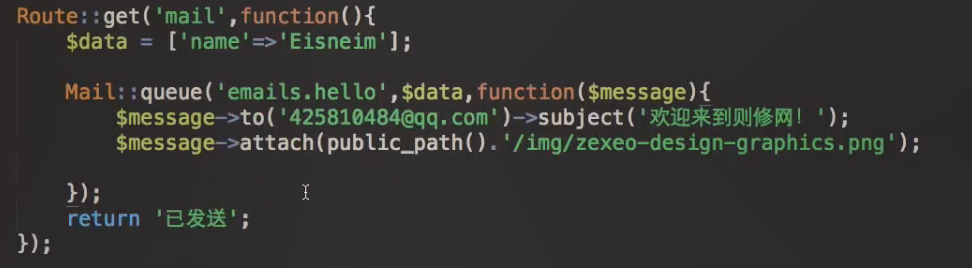


具体实现代码

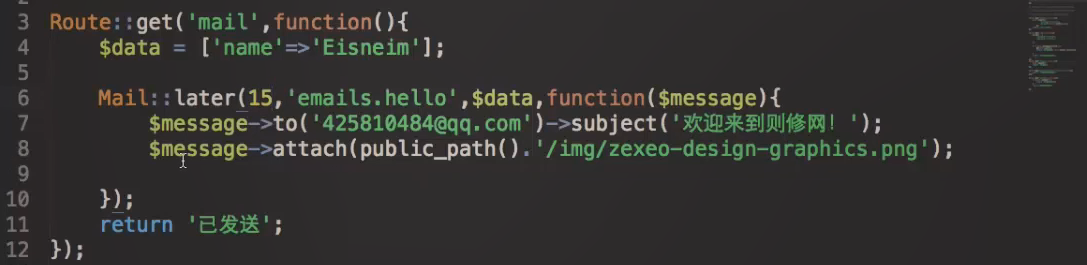
写入文件:



写入邮件:



延迟15s发送



## Laravel邮件发送

<http://www.jianshu.com/p/8ccb2820df23>

注意了:

阿里云主机

主机提供商禁用了第三方的邮件访问通道....坑爹的

## Laravel Redis使用类的引用

laravel redis 扩展冲突的问题分析：Fatal error: Non-static method Redis::keys() cannot be called statically

laravel中，出现 'Fatal error: Non-static method Redis::keys() cannot be called statically' 的问题分析：

1.这里不得不吐槽下。按理说，这个问题，是一个很容易发生的问题！网上各种搜索，很难明白是什么原因。

最多是一个 'redis命名空间冲突'，让我们不要引用 'Redis'，而是

1.将 'config/app.php' 中的 Redis别名，修改为 'Redisxx'，换一个不会冲突的

2.将项目中的 'use Redis;'，改为 'Illuminate\Support\Facades\Redis' 反正就是指定全路径

这确实是正确的，但我之前的环境可以使用，现在的环境不可用，我不能将之前的代码中，所有的都进行替换啊，有点痛苦！

2.找了很多文章，外加昨天正好总结了 "PHP的扩展安装" - http://blog.csdn.net/beyond\_\_devil/article/details/65934701

pecl，pear，phpize，php-config 等，都了解了点(原谅我，现在居然才开始看这个，还干了3年PHP了。。。)

大概了解产生这个问题的原因了，分析如下：

1>我现在默认的集成环境中，系统已经安装了 redis，并且，系统的PHP，也已经安装了 'PhpRedis' 的 PECL扩展，也就是 redis.so // 它已经占据了 'Redis' 命名空间

2>laravel框架，文档中，也指出了，我们使用redis，需要安装 composer require predis/predis // 它也使用的是 'Redis' 命名空间

2者都是Redis扩展，使用了 Redis 命名空间，所以导致冲突！

3.了解了问题所在，我们就可进行修复：

1>就是上面所说的，修改通过 composer 安装的 predis 的命名空间。又分2种：

1)修改 'app/config.php'，将 'Redis'，改为 'RedisDB'。然后，每次引用Redis Facade时，使用 RedisDB 来替换

2)在每个使用 use Redis; 的文件中，我们改为 use Illuminate\Support\Facades\Redis;

2>我的服务器项目，只基于laravel，我在php配置文件中，注释掉 ';extension redis.so'，不使用 pecl 安装的redis扩展，直接使用 composer 安装的redis扩展(也算推荐吧，以后我们得熟悉 composer 来搭建php应用)

参考文章：

https://laravel.com/docs/5.4/redis // laravel "英文" 官方文档(中文翻译的之前的版本，有的信息没有)

1>官方文档中，有这么一句话：

Alternatively, you may install the PhpRedis PHP extension via PECL. The extension is more complex to install but may yield better performance for applications that make heavy use of Redis.

意思是：

你也可以通过PECL安装 PhpRedis 扩展。在大量使用 redis 的PHP应用中，使用 PECL 安装的redis.so扩展，会产生更高的性能(通过我上面的教程，应该了解到 pecl 扩展，是使用 c、c++等开发的)，但是就是安装稍微复杂点

2>接着官方的redis文档，直接红色的警告！！！！

/\*

If you have the PhpRedis PHP extension installed via PECL, you will need to rename the Redis alias in your config/app.php configuration file.

\*/

如果，使用了 pecl 安装的 redis.so，我们在 'config/app.php' 中，就需要重命名 'Redis' 别名。

3>如果，我们就想使用 pecl 安装的 'redis.so' (扩展名为：PhpRedis)，我们需要配置：

config/database.php

'redis' => [

'client' => 'phpredis',

],

除了默认的host ,port, database和password配置，PhpRedis还支持 persistent, prefix, read\_timeout, timeout。可在 'config/database.php' 中配置

总结：

官方文档，就已经解释了很清楚了，我居然不知道。。。

还是我这个问题，太低级了。。。