## 1.laravel 中config的使用

config('app.name', 'Laravel')

在laravel的config中添加配置文件（比如：alipay.php）文件内容为return数组的形式

在方法中使用config()函数获取数据

$config = config('alipay.alyconf');

其中alipay为config下的配置文件名，alyconf为配置文件alipay.php所return 数组的key值

return [

'alyconf' => array (

//应用ID,您的APPID。

'app\_id' => "20180531603",

)

];

使用：

$config = config('alipay.alyconf');

$config即为return的key为alyconf的数组

$alyPay\_appid = $config['app\_id'];

上面是alipay.php的示例

如此，$config就获取了return的数组。

## 2. asset()

asset 函数使用当前请求的 scheme（HTTP 或 HTTPS）为前端资源生成一个 URL：

$url = asset('img/photo.jpg');

laravel自带了laravel-mix，用于对js、css、图片等静态资源进行打包。

生成的文件的命名会是：app.asjduiik2l1323879dasfydua23.js, 即js原文件名+hash+.js后缀，

因为中间的那个hash是随时会变化的，所以在页面引入js文件的时候，就不能写死文件的路径，

而是使用mix('app.js')，此时laravel会自动去匹配当前的app.js对应哪个app+hash+.js的文件

（项目public目录下会有一个mix-manifest.json, 这里面保存了两者的对应关系，每次打包静态资源的时候都会更新该文件）。

而有些时候我们并不希望静态资源的名称中被加上hash值（大部分情况是独自引入的非nodejs模块的第三方库），

这个时候就可以直接使用asset方法，它就是直接简单粗暴地找你给它名称的文件咯。

## 3. 认证指令

@auth 和 @guest 指令可用于快速判断当前用户是否登录：

@auth

// 用户已登录...

@endauth

@guest

// 用户未登录...

@endguest

## 4.route()函数

**为命名路由生成 URL**

正如上面代码所展示的，为给定路由分配名称之后，就可以通过辅助函数 route 为该命名路由生成 URL 或者通过 redirect 函数进行重定向：

// 生成URL

$url = route('profile');

// 生成重定向

return redirect()->route('profile');

## 5.中间件在controller中的使用

路由保护

[路由中间件](http://laravelacademy.org/post/7812.html)可用于只允许通过认证的用户访问给定路由。Laravel 通过定义在 Illuminate\Auth\Middleware\Authenticate 中的 auth 中间件来实现这一功能。由于该中间件已经在 HTTP kernel 中注册，你所要做的仅仅是将该中间件加到相应的路由定义中：

Route::get('profile', function() {

// 只有认证用户可以进入...

})->middleware('auth');

当然，如果你也可以在[控制器](http://laravelacademy.org/post/7836.html" \t "_blank)的构造方法中调用 middleware 方法而不是在路由器中直接定义实现同样的功能：

public function \_\_construct(){

$this->middleware('auth');

}

## 6.中间件使用：

A:创建中间件：php artisan make:middleware CheckToken

B：定义中间件逻辑

C: 中间件注册到指定路由中

方法：

Route::get('profile', function() {

// 只有认证用户可以进入...

})->middleware('auth');

当然，如果你也可以在[控制器](http://laravelacademy.org/post/7836.html)的构造方法中调用 middleware 方法而不是在路由器中直接定义实现同样的功能：

public function \_\_construct(){

$this->middleware('auth');

}

理解中间件的最好方式就是将中间件看做 HTTP 请求到达目标动作之前必须经过的“层”，每一层都会检查请求并且可以完全拒绝它。

## 7.自定义中间件组

当然我们可以自己设置自己的中间件组，以实现更灵活的中间件分配策略：

/\*\*

\* 应用的中间件组.

\*

\* @var array

\*/

protected $middlewareGroups = [

'web' => [

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

// \Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

'api' => [

'throttle:60,1',

'auth:api',

],

'blog' => [

'token',

]

];

我们修改 routes/web.php 下面的中间件分配方式：

Route::group(['middleware'=>['blog']],function(){

Route::get('/', function () {

return view('welcome', ['website' => 'Laravel']);

});

Route::view('/view', 'welcome', ['website' => 'Laravel学院']);

});

这样我们访问 http://blog.test 和 http://blog.test/view 的时候都要带上 token=laravelacademy.org 参数，否则就会跳转到 Laravel 学院网站。

## 8. 中间件参数

### 1.中间件不能return json\_encode()

2.中间件还可以接收额外的自定义参数，例如，如果应用需要在执行给定动作之前验证认证用户是否拥有指定的角色，可以创建一个 CheckRole 来接收角色名作为额外参数。

额外的中间件参数会在 $next 参数之后传入中间件：

<?php

namespace App\Http\Middleware;

use Closure;

class CheckRole

{

/\*\*

\* 处理输入请求

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @param string $role

\* @return mixed

\* translator http://laravelacademy.org

\*/

public function handle($request, Closure $next, $role)

{

if (! $request->user()->hasRole($role)) {

// Redirect...

}

return $next($request);

}

}

中间件参数可以在定义路由时通过 : 分隔中间件名和参数名来指定，多个中间件参数可以通过逗号分隔：

Route::put('post/{id}', function ($id) {

//

})->middleware('role:editor');

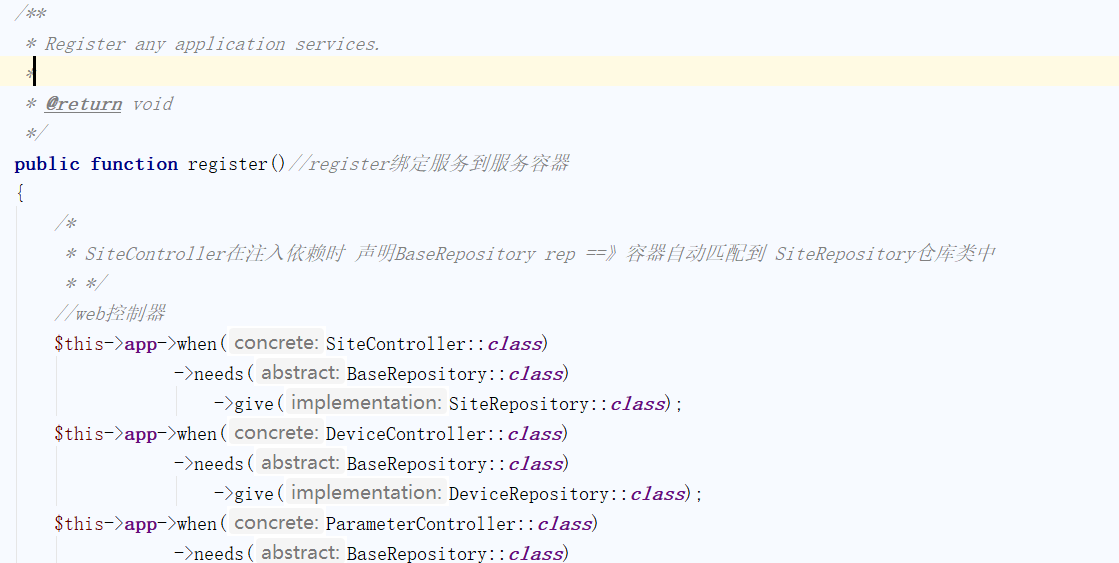
## 9. Repository 模式

## 10. [::before和::after伪元素的用法](https://www.cnblogs.com/starof/p/4459991.html)

## 11. 服务容器

Laravel 服务容器是一个用于管理类依赖和执行依赖注入的强大工具

Laravel 提供了一个功能强大的 IoC 容器。这个 IoC 容器在 Laravel 中被称作服务容器，是整个 Laravel 框架最核心的部分，在它的调度下，框架各个组件可以很好的组合在一起工作。

  
php artisan数据库迁移

<https://www.jellythink.com/archives/229>

<https://www.cnblogs.com/huangbx/p/Laravel_4.html>

<https://www.cnblogs.com/huangbx/p/Laravel_3.html>

php artisan migrate:install

php artisan migrate

php artisan db:seed

基本指令：<https://blog.csdn.net/lw545034502/article/details/88076016>

p()执行迁移文件操作，down()是完全相反。就像word文档中的撤销命令——撤销所做的修改。例如，我们想创建一个“authors”表，在up()中创建，在down()中撤销

### trait:

<https://www.jb51.net/article/105579.htm>

## 12.API认证：

目的：在前后端进行交互的时候，为了数据安全性而进行的安全认证

前端向后端发起一个请求时，如果别人知道我们请求的url，通过浏览器浏览，那么我们的数据就会被获取。因此为了限制不是所有人都可以请求到我们的数据，我们就需要进行API认证，只让符合我们规则的请求获取数据。

## 13.路由：

### 命名路由，路由分组：

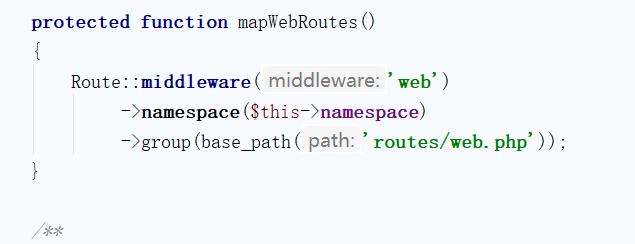
Routes/web.php定义了Web界面的路由，这些路由被分配到Web中间件中。

定义在 routes/web.php 中的路由可以通过在浏览器地址栏输入相应的 URL 进行访问

routes/api.php 中的路由是无状态的，这是因为被分配到了 api 中间件组

在定义路由的时候，我们像这样定义Route::*any*(**'/test'**,**'test@index'**)->name(**'test'**);

路由中的第二个参数并没有给控制器完整的命名空间。这是因为在RouteServiceProvider 在加载路由文件的时候给他指定了命名空间，因此定义路由的时候只需要App\Http\Controllers后面的命名空间即可。



在api.php中我们一般像这样

因为默认的命名空间是App\Http\Controllers,因此我们要给给api\..的命名空间。同时我们可以在路由组中给路由一个api命名空间，这样就省去在每个第二个参数前加上api了。

## 14.控制器：

后端服务器—服务器资源

用户—用户资源

后端登录(前端)—请求者 第三方应用

接口测试的时候带上令牌？？

## 15. 自定义 api 接口的错误消息

https://blog.csdn.net/hjh15827475896/article/details/84801163

## 16. debug\_backtrace()

获取调用过的函数---从内到外获取

<?php  
function one($str1, $str2)  
{  
two("Glenn", "Quagmire");  
}  
function two($str1, $str2)  
{  
three("Cleveland", "Brown");  
}  
function three($str1, $str2)  
{  
print\_r(debug\_backtrace());  
}  
  
one("Peter", "Griffin");  
?>

上面代码的输出如下所示：

Array  
(  
[0] => Array  
(  
[file] => C:webfoldertest.php  
[line] => 7  
[function] => three  
[args] => Array  
(  
[0] => Cleveland  
[1] => Brown  
)  
)  
[1] => Array  
(  
[file] => C:webfoldertest.php  
[line] => 3  
[function] => two  
[args] => Array  
(  
[0] => Glenn  
[1] => Quagmire  
)  
)  
[2] => Array  
(  
[file] => C:webfoldertest.php  
[line] => 14  
[function] => one  
[args] => Array  
(  
[0] => Peter  
[1] => Griffin  
)  
)  
)

## 17．Laravel本地项目打开网页速度慢

原因：框架中引用了谷歌的字体

需要删除对谷歌字体的引用

### public/css/app.css

@import url(https://fonts.googleapis.com/css?family=Raleway:300,400,600);@charset "UTF-8"; ...

### resources/assets/sass/app.scss

// Fonts

@import url("https://fonts.googleapis.com/css?family=Raleway:300,400,600");

## 18. Eloquent数据库操作

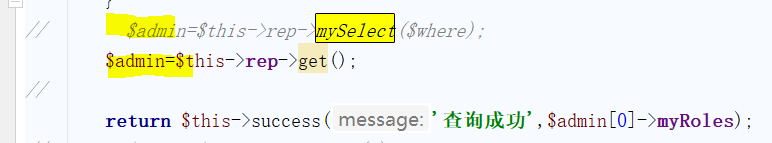
Fisrt()查询 查询到结果返回，查询不到返回null

Get()查询 查询到返回结果数组集，查询不到返回空数组

### 关联关系

参考：<https://laravelacademy.org/post/22018#toc-5>

注意：查询多条数据并获取他的关联表的数据时，多条数据必须是个object



像这样 *$admin=$this->rep->mySelect($where)返回的是数组* $admin[0]->**myRoles** 会报错

### 定义关联关系

## 19. CSS3自定义滚动条样式

/\*定义滚动条高宽及背景 高宽分别对应横竖滚动条的尺寸\*/

::-webkit-scrollbar

{

width: 16px;

height: 16px;

background-color: #F5F5F5;

}

/\*定义滚动条轨道 内阴影+圆角\*/

::-webkit-scrollbar-track

{

-webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);

border-radius: 10px;

background-color: #F5F5F5;

}

/\*定义滑块 内阴影+圆角\*/

::-webkit-scrollbar-thumb

{

border-radius: 10px;

-webkit-box-shadow: inset 0 0 6px rgba(0,0,0,.3);

background-color: #555;

}

参考：<https://blog.csdn.net/hanshileiai/article/details/40398177>

## 20.视频播放插件

### 20.1 H5原生video标签

//h5原生video标签  
<**video width="300" height="240" controls**>  
 <**source src="**{{asset(**'static/video/SWE.mp4'**)}}**" type="video/mp4"**>  
 <**source src="movie.ogg" type="advideo/ogg"**>  
</**video**>

### 20.2 video.js插件

<**video id="example\_video\_1"  
 class="video-js vjs-default-skin vjs-big-play-centered vjs-16-9"  
 controls  
 preload="auto"  
 data-setup="{}"  
 poster="http://vjs.zencdn.net/v/oceans.png"**>  
 <**source src="**{{asset(**'static/video/SWE.mp4'**)}}**" type="video/mp4"**>  
</**video**>

参考资料：

<https://segmentfault.com/a/1190000018914486?utm_source=tag-newest#articleHeader2>

<https://blog.csdn.net/a0405221/article/details/80923090>

## 21.formdata

### 概述:

FormData类型其实是在XMLHttpRequest 2级定义的，它是为序列化表以及创建与表单格式相同的数据（当然是用于XHR传输）提供便利。

### 个人理解：

不通过表单构造数据，而是使用FormData来构造与表单格式相同的数据进行提交。

### 获取变单input的text,passworde类型元素

Demo:coludupdate

假如现在页面已经有一个表单

<form id="myForm" action="" method="post">

<input type="text" name="name">名字

<input type="password" name="psw">密码

<input type="submit" value="提交">

<input id="text" type="text" name="text">

</form>

// 获取页面已有的一个form表单

var form = document.getElementById("myForm");

// 用表单来初始化

var formData = new FormData(form);

// 我们可以根据name来访问表单中的字段

var name = formData.get("name"); // 获取名字

var psw = formData.get("psw"); // 获取密码

// 当然也可以在此基础上，添加其他数据

formData.append("token","kshdfiwi3rh");

对于jquery不通过form获取input值，直接通过input获取

*// var formData2 = new FormData();  
// formData2.append('text',$('#text').val())  
// console.log(formData2.get('text'))*

### 获取变单input的file类型元素

#### 单个file获取

formData.append(**'file'**, *$*(**'#file'**)[0].files[0]); *// 固定格式*

#### 多个file获取

**for**(**var** i=0;i<*$*(**'#file'**)[0].files.length;i++){  
 formData.append(**'file[]'**, *$*(**'#file'**)[0].files[i]); *// 固定格式*}

### formData里面存储的数据形式

一对key/value组成一条数据，key是唯一的，一个key可能对应多个value

### 添加数据

我们可以通过append(key, value)来添加数据，如果指定的key不存在则会新增一条数据，如果key存在，则添加到数据的末尾

formData.append("k1", "v1");

formData.append("k1", "v2");

formData.append("k1", "v1");

formData.get("k1"); // "v1"

formData.getAll("k1"); // ["v1","v2","v1"]

### 设置修改数据

我们可以通过set(key, value)来设置修改数据，如果指定的key不存在则会新增一条，如果存在，则会修改对应的value值。

formData.append("k1", "v1");

formData.set("k1", "1");

formData.getAll("k1"); // ["1"]

### 判断是否该数据

我们可以通过has(key)来判断是否对应的key值

formData.append("k1", "v1");

formData.append("k2",null);

formData.has("k1"); // true

formData.has("k2"); // true

formData.has("k3"); // false

### 删除数据

通过delete(key)，来删除数据

formData.append("k1", "v1");

formData.append("k1", "v2");

formData.append("k1", "v1");

formData.delete("k1");

formData.getAll("k1"); // []

### 遍历

我们可以通过entries()来获取一个迭代器，然后遍历所有的数据，

formData.append("k1", "v1");

formData.append("k1", "v2");

formData.append("k2", "v1");

var i = formData.entries();

i.next(); // {done:false, value:["k1", "v1"]}

i.next(); // {done:fase, value:["k1", "v2"]}

i.next(); // {done:fase, value:["k2", "v1"]}

i.next(); // {done:true, value:undefined}

## 22.FileReader

## 23.A标签之定义样式

参考：<https://blog.csdn.net/tian_fang/article/details/4802552>

Css为一些特殊效果定义的伪类：

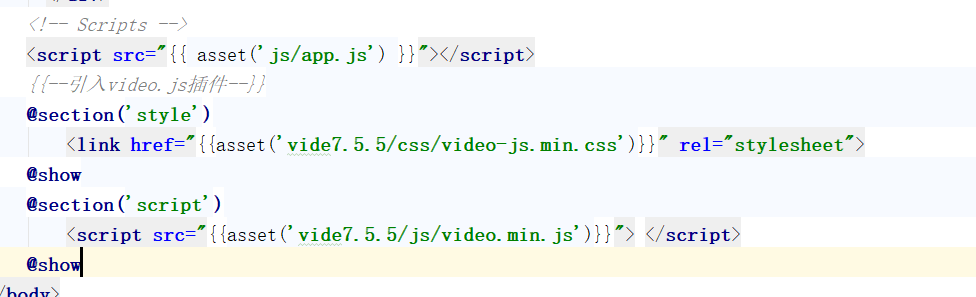
link,hover,visited,active

a:link，定义正常链接的样式；  
    a:visited，定义已访问过链接的样式；  
    a:hover，定义鼠标悬浮在链接上时的样式；  
    a:active，定义鼠标点击链接时的样式。

text-decoration:none;清除下划线

## 24. laraevl在不同的blade模板中使用不同的css和js

父模板：



子模板



## 25.laravel的文件上传

### 1.磁盘定义

config/filesystems.php默认配置了local(本地驱动),public(公共磁盘),

s3(**S3 驱动配置**)三个磁盘。可以自定义磁盘

如：**'fileinput'**=>[  
 **'driver'**=>**'local'**,  
 **'root'** => public\_path(**'uploads/video'**),  
],

### 2. 公共磁盘public

用于存储可以被公开访问的文件，默认存储位置在storage/app/public目录下。

使用php artisan storage:link创建软链

### 3.本地驱动

自定义本地驱动

**'fileinput'**=>[  
 **'driver'**=>**'local'**,  
 **'root'** => public\_path(**'uploads/video'**),  
],

使用

Storage::disk('fileinput')->put('file.txt', 'Contents');

将文件存储到fileinput本地驱动定义的root目录下

### 4.s3

略

### 5. 获取硬盘实例

//Storage门面没有调用disk方法，调用的方法会自动被传递到默认磁盘

Storage::put('avatars/1', $fileContents);

Storage::disk('fileinput')->put('avatars/1', $fileContents);

### 6. 获取文件

$contents = Storage::get('file.jpg');

$contents = Storage::disk('fileinput')->get('file.jpg');

所有文件路径都是相对于配置文件中指定的磁盘默认根目录，get 方法用于获取给定文件的内容，该方法将会返回该文件的原生字符串内容。

exists 方法用于判断给定文件是否存在于磁盘上：

$exists = Storage::disk('fileinput')->exists('file.jpg');

### 7.下载文件

download 方法可用于生成强制用户浏览器下载给定路径文件的响应。download 方法接收一个文件名作为第二个参数用于决定用户下载时看到的文件名。最后，你可以传递一个 HTTP 请求头数组作为该方法的第三个参数：

return Storage::download('file.jpg');

return Storage::download('file.jpg', $name, $headers);

### 8.文件 URL

使用 local 或 s3 驱动时，可以使用 url 方法获取给定文件的 URI。如果你使用的是 local 驱动，通常会在给定路径前加上 /storage，并返回该文件的相对 URL；如果使用的是 s3 或 rackspace 驱动，则会返回完整的远程 URL：

use Illuminate\Support\Facades\Storage;

$url = Storage::url('file1.jpg');

### 9. 存储文件

put 方法可用于存储原生文件内容到磁盘。此外，还可以传递一个 PHP 资源到 put 方法，该方法将会使用 Flysystem 底层的流支持。在处理大文件的时候推荐使用文件流：

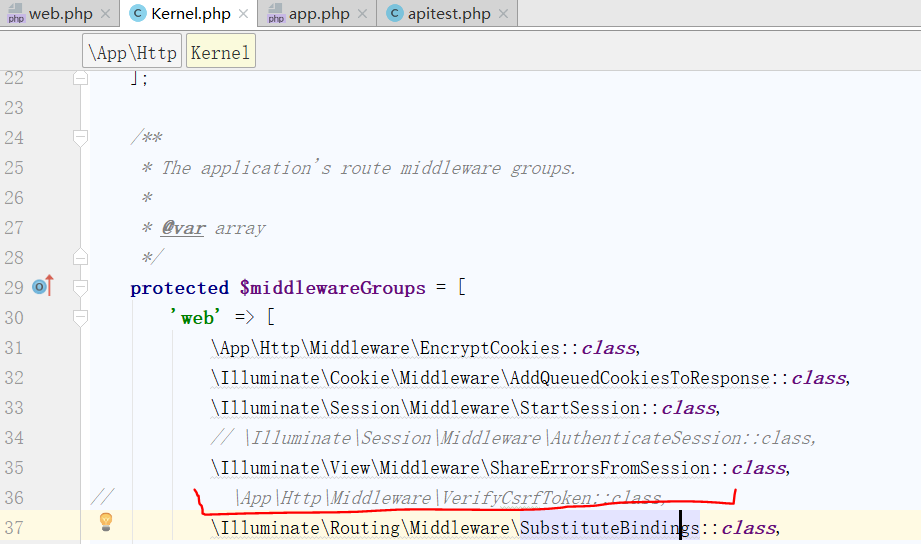
use Illuminate\Support\Facades\Storage;

Storage::put('file.jpg', $contents);

Storage::put('file.jpg', $resource);

## 26.laravel新建项目post请求419

### 1. 关闭CSRF



### 2. 首先在页面加上

<meta name="csrf-token" content="{{ csrf\_token() }}" />

然后请求的在header里面加上

|  |
| --- |
| 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content') |

## 27. OAuth2.0认证原理

OAuth的作用就是让"客户端"安全可控地获取"用户"的授权，与"服务商提供商"进行互动。

## 28.laravel去除google字体防止页面加载过慢

删除app.blade.php对google字体的引用，和下面的部分



## 29.Auth中间件

## 30. GuzzleHttp安装

composer require guzzlehttp/guzzle

文档：

<https://guzzle-cn.readthedocs.io/zh_CN/latest/>

### GuzzleHttp访问https报错

参考<https://www.jianshu.com/p/392f966fc4d1>



访问这个网址 [https://curl.haxx.se/ca/cacert.pem](https://link.jianshu.com?t=https:/curl.haxx.se/ca/cacert.pem) 下载文件  
然后修改 php.ini curl.cainfo = "D:\cacert.pem" cacert.pem文件 随便放在哪，没限制。

## 31. json\_encode与json\_decode的区别与用法

Json\_encode将数组或对象编码成json字符串多的函数

json\_decode刚好相反，是将json字符串转成数组或对象

json\_decode（，true）数组

json\_decode（，false）对象默认为false

## 32. Laravel修改passport认证字段

## 33. laravel 429 (Too Many Requests)

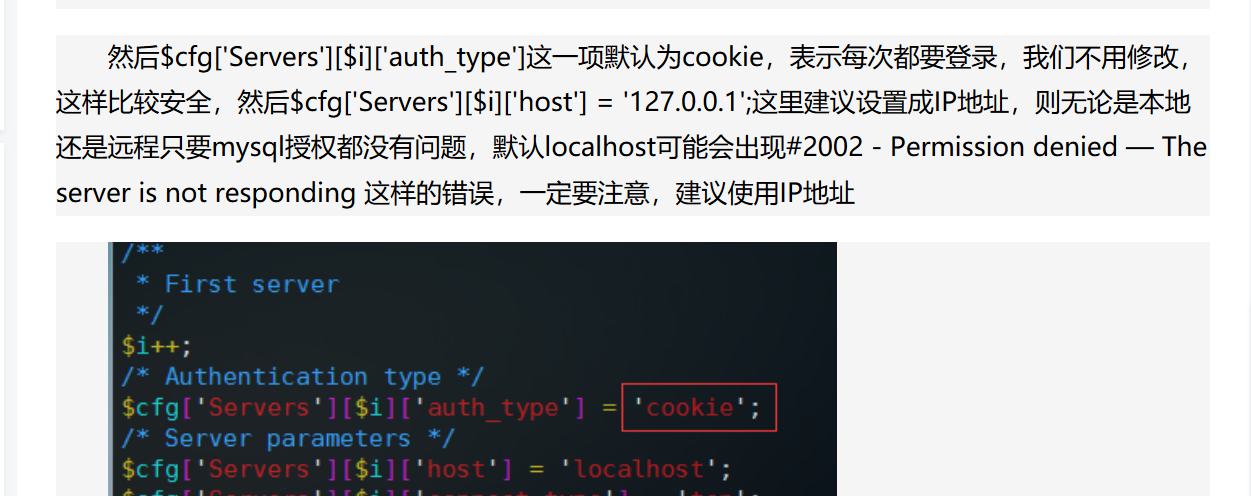
## 34. Eloquent ORM Model 对同一条数据 update 两次导致失败的问题？

<https://learnku.com/laravel/t/9271/does-eloquent-orm-model-cause-two-failures-to-the-same-data-update>

f

## Linux centos安装phpmyadmin

<https://blog.csdn.net/weixin_40325128/article/details/81038715>



Ip地址可以不用换

## 腾讯云http转https

分页查询

Laravel中使用分页查询

接口定义：

*// 分页查询* Route::*get*(**'/user/OrderSelPage'**,**'wechatOrder@OrderSelPage'**);

请求：<http://www.wechatdemo5.com/api/user/OrderSelPage?page=1>

Page后面写几即请求第几页数据

## Laravel swoole

Swoole文档：<https://wiki.swoole.com/>

# 重大bug关于手动创建类laravel显示not exist

我是手动创建的仓库类然后在provider中注册的时候一直报错一开始看到别人说有配置缓存

清除编译缓存

php artisan clear-compiled

清除数据缓存

php artisan cache:clear

# 清除配置缓存

php artisan config:clear

清除路由缓存

php artisan route:clear

然而 并没有用

# 后来使用composer dump-autoload[配置自动加载文件](https://www.cnblogs.com/fogwang/p/11811628.html)

## Postman数组参数测试

