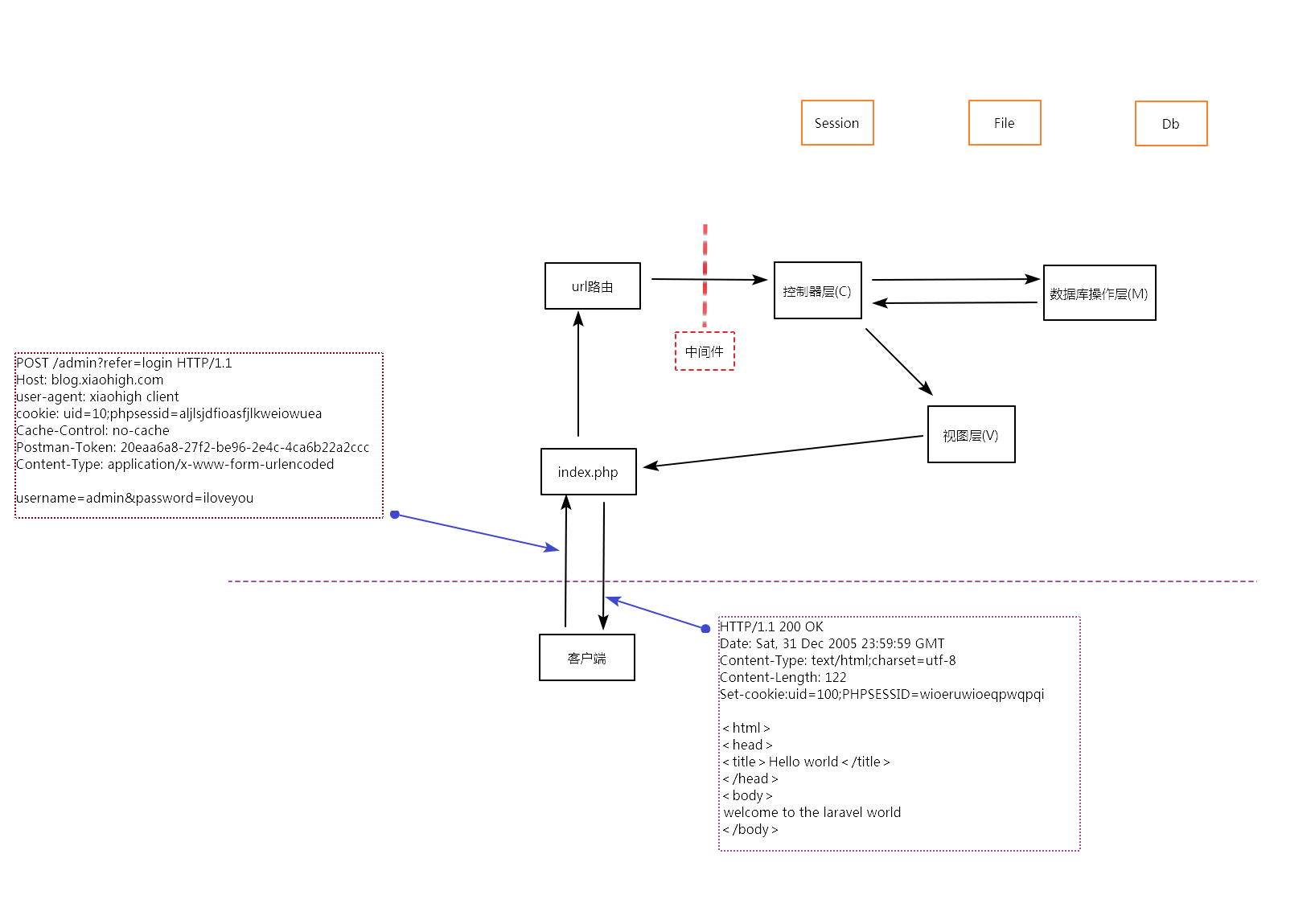
# 第1章：Laravel简介

## 1.1 Laravel运行环境

* Php版本在 7.0.0以上
* Php PDO扩展开启。框架中用PDO链接数据库
* Php OpenSSL扩展开启。框架用于对传输的数据进行加密
* Php Mbstring扩展开启。Mbstring提供了对多字符串的函数，帮助处理php中的多字节编码。
* Php Tokenizer扩展 php代码解析
* Php XML扩展

## 1.2 Laravel 工作原理



# 第2章：安装

## 2.1 composer安装

### 2.1.1 composer 简介

1.composer 是PHP用来管理依赖关系的工具。

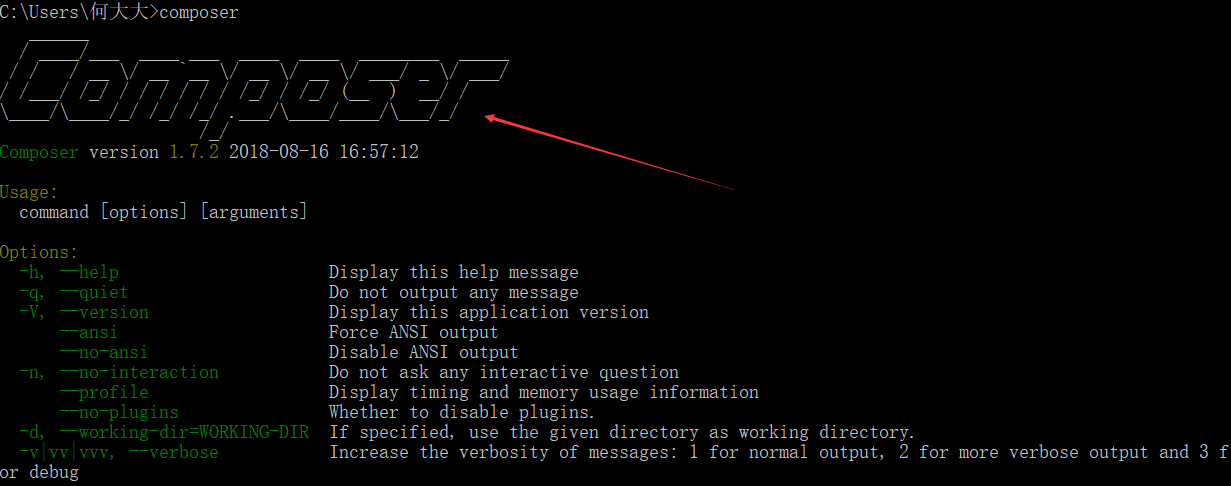
2.工具包地址 <http://packagist.org/>

### 2.1.2 Window环境

在windows下安装采用傻瓜式下一步安装

下载地址：http://getcomposer.org/download/

测试:cmd命令下输入composer，显示已下表示安装成功。



### 2.1.3 Linux 环境

1.下载composer.phar

2.放到linux服务器下，同时切换同级目录下执行 php composer.phar

3.全局执行 mv composer.phar /usr/local/bin/composer

### 2.1.4 composer中国全量镜像

composer config -g repo.packagist composer https://packagist.phpcomposer.com

### 2.1.5 composer的基本使用

* 【单文件下载】

单文件下载需要，知道下载的名称(可先在https://packagist.org/ 中搜索)

如：

1. 验证码：composer require gregwar/captcha
2. 二维码：composer require endroid/qrcode
3. 数据库包、日志包等等

* 【多文件下载】

1.多文件下载需要在指定文件夹下创建composer.json,填写相关配置

{

"require": {

"monolog/monolog": "1.20.0",

}

}

//都是双引号

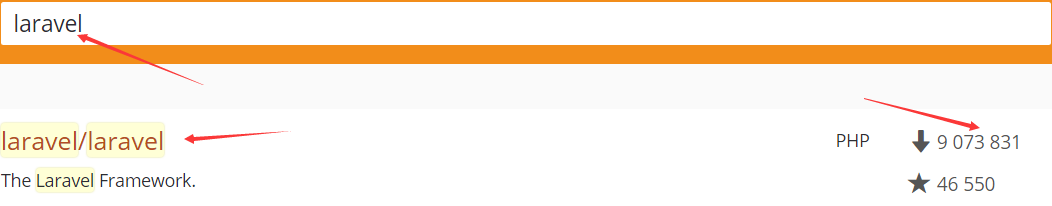
1. 在cmd命令行下执行 composer install
2. 如果是更新完composer.json文件 输入composer update来更新

* 更新composer

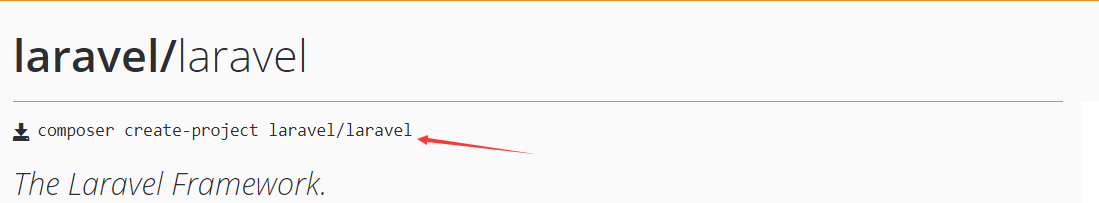
composer self-update

## 2.2 composer安装Laravel

1.在Packagist.org中搜索laravel



1. 安装 默认安装最新版本



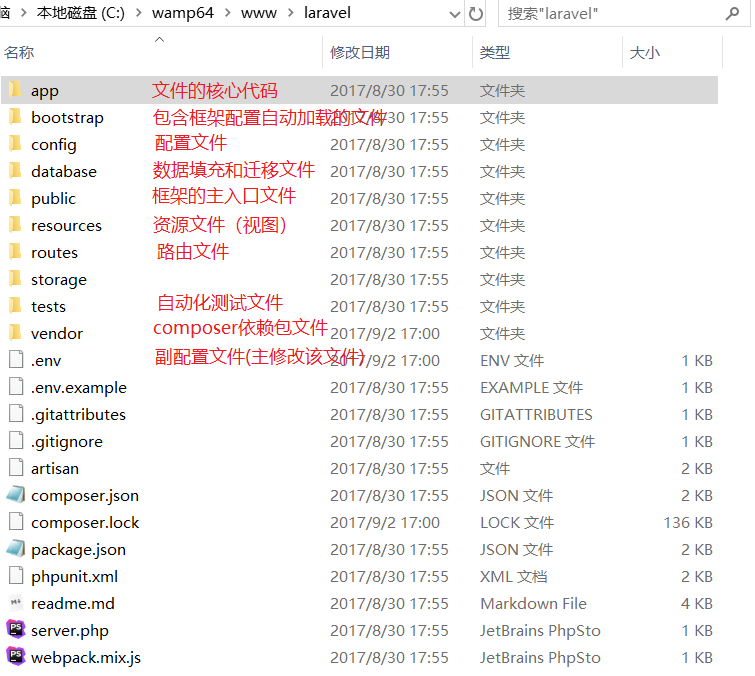
composer create-project --prefer-dist laravel/laravel your-project-name "5.5.\*"

安装5.5版本

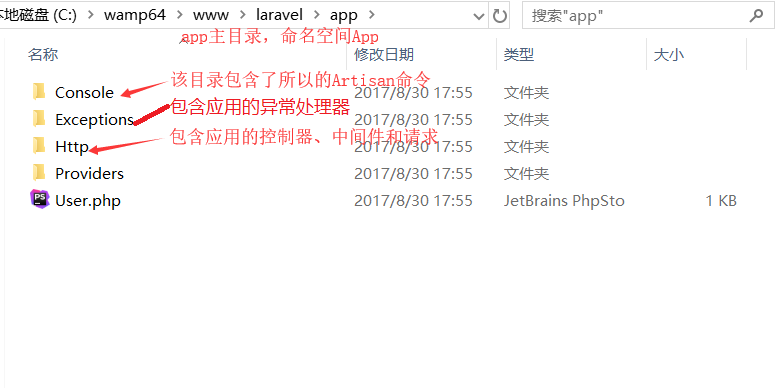
# 第3章 开发

## Laravel目录结构

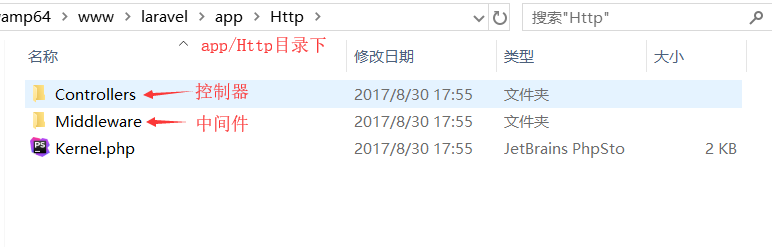
【主目录】



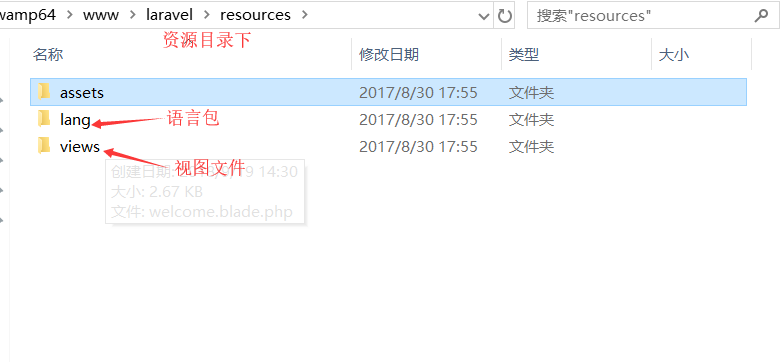
【App目录】



【Http目录】



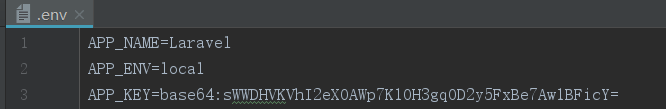
【Resources目录】



## 开发配置

### 3.2.1开发前

1. 配置虚拟主机
2. linux环境下,storage和vendor目录要让服务器有写入的权限(chmod 777 storage)。如果没有写的权利会报错或空白页
3. 程序秘钥，这是默认生成的，如果没有可使用命令创建 。



php artisan key:generate

1. 修改时区 config/app.php

‘timezone’ => ’PRC’,

### 3.2.2 开发中

1.读取和设置配置

读取：Config::get(‘app.timezone’);

设置：Config::set(‘app.locale’,’cn’);

2.获取环境配置

env(‘DB\_HOST’)

3.关闭和启动应用

关闭：php artisan down

启动：php artisan up

## 路由

Laravel框架路由存放在route/web.php文件夹中。

### 3.3.1基本路由

Route::get(‘/my’,function(){return ‘Hyt-get’})

Route::post(‘/my’,function(){return ‘Hyt-post’})

Route::put(....)

Route::delate(...)

Route::any(...)

注意：CSRF保护

laravel中使用POST、PUT、DELETE路由的任何HTML表单都应该包含CSRF令牌字段，否则，这个请求将会被拒绝

<form method="POST" action="/profile">

{{ csrf\_field() }}

...

</form>

### 3.3.2 带参数的路由

Route::get(‘/goods/{id}’,function($id){})

### 3.3.3 路由别名

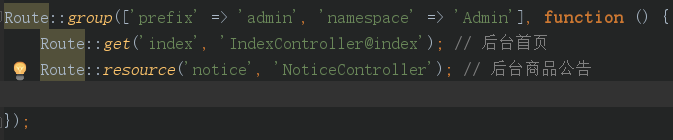
Route::get(‘/Home/Goods/show/{id}’,[‘as’=>’goods’],function($id){});

### 3.3.3 路由组

Route::group([],function(){

Route::get(‘/my’,’HYT-group’);

});



### 3.3.4 404页面设置

1.errors目录需要自己创建(resources/views/errors)

2.在该目录下设置404模板 404.blade.php(laravel中自己会找)

### 3.3.5 CSRF保护

在3.3.1中已经简单说了一下CSRF保护，这是laravel特有的。同时还要注意,AJAX的post也要验证。

$.ajaxSetup({

headers: {

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

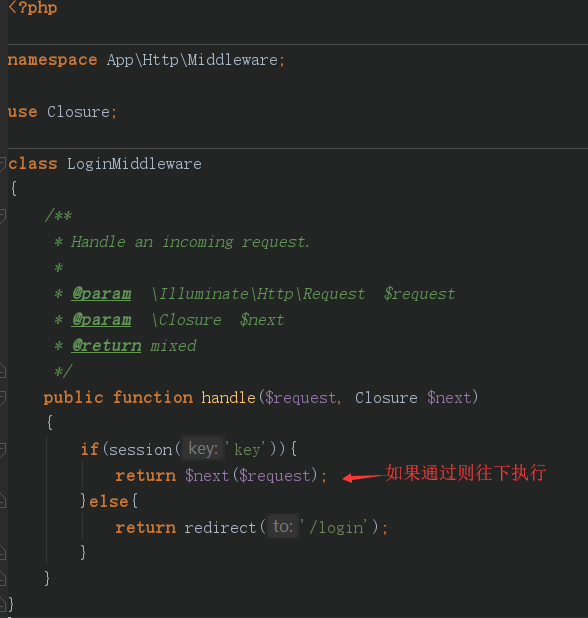
}});

## 中间键

### 创建

cmd 命令下：

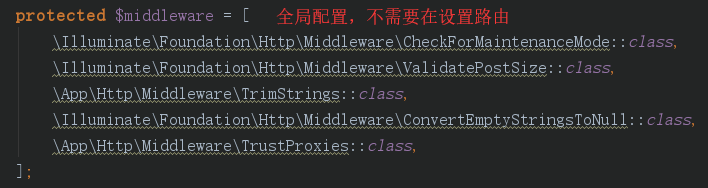
php artisan make:middleware LoginMiddleware



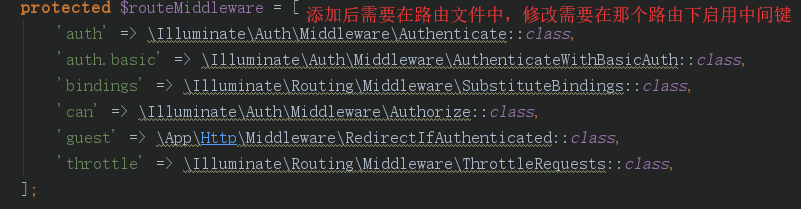
### 配置

中间键的配置在Http目录下Kernel.php文件

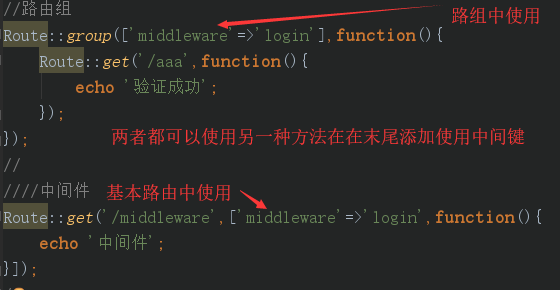
* 【全局配置】 namepsae\中间键名::class,



* 【路由配置】



使用路由配置需要：



Route::get(‘/my’,function(){

echo ‘111’;

})->middleware(‘login’);

## 控制器

### 普通控制器

* 【创建】

php artisan make:controller UserController

* 【访问】

1. 普通访问：Route::get(‘/User/add’,’Admin\UserController@add’)
2. 带参数访问：Route::get(‘/User/edit/{id}’,’Admin\UserController@edit’);
3. 别名方法

* 【注意】

1.Larevel中的路由控制器的第二个参数一定要是反斜线\

2.控制器到方法一定是@

3.与TP5有差别

### 资源控制器

* 【创建】

php artisan make:controller UserController --resource

* 【使用】



* 【注意】

PUT/PATCH、DELETE请求时需要伪造表单方法。

因为HTML表单不能生成PUT、PATCH或者DELETE请求，因此我们需要添加一个隐藏域字段\_method来伪造这些HTTP动作。method\_field辅助函数可以帮你创建这个字段(laravel封装好的)

{{method\_field(‘PUT’)}} //PATCH、DELETE

## 请求

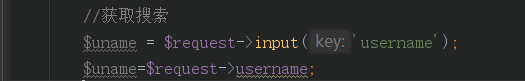
### 3.6.1 基本信息获取

1. 获取请求方法 $request->methoD();
2. 检测方法 $request->isMethod(‘post’);
3. 请求路径 $request->path();
4. 获取完整url $request->url();
5. 获取IP $request->ip();
6. 获取端口 $request->getPort();

### 3.6.2 提取请求参数

1. 基本获取 $name = $request->input(‘name’);

从Request $request 里面获取数据，不管是post/get都可以用$request->input()来获取。同时还有另一种方法，如：$request->input(‘name’)也可以用$request->name来接收。



1. 设置默认值 $request->input(‘name’,’Sally’);

如果name有值，就用name，如果没有就用Sally。类似三元运算符。

1. 提取所有的参数 $input =$request->all();
2. 检测是否存在 $request->has(‘name’);

布尔值返回

1. 提取部分参数 $request->only([‘username’,’password’]);

只提取username、password

1. 选择提取参数 $request->except([‘create\_card’])

除了create\_card其他都获取。一般都出去\_token

1. 获取请求头里的信息 $request->header(‘Connection’);

### 3.6.3上传文件操作

1. 检测是否有文件上传 $request->hasFile(‘img’);

一般有上传文件的话，是但多获取的，比如$request->except(‘img’)后采用123步依次处理上传文件

1. 获取上传文件的后缀 $request->file(‘ime’)->getClienOriginalExtension();

肯定的除了获取后缀以外，还要重新给上传文件整合新名字。

1. 将文件移动到指定位置 $request->file(‘img’)->move(‘路径’,’新名字’)

路径这点一定要注意：目录名前不要加斜线(/)，如果真要加可用点斜线(./)。因为，加上斜线就走磁盘的跟目录而不是public的跟目录。

【实例】



### 3.6.4 Cookie操作

使用Cookie时需要引入cookie类



如果不引入，可在使用Cookie时在前面加上反斜线。(\Cookie::)

* 【设置】

1.第一种方法

\Cookie::queue(‘name’,’Hyt’,10);

10表示有效时间。

2.第二种方法

respoense(‘名称’)->withCookie(‘name’,’Hyt’,10);

* 【获取】

1. 第一种方法

\Cookie::get(‘name’);

1. 第二种方法

$request->cookie(‘name’);

### 3.6.5 闪存信息

* 【设置】

闪存原理就是将信息存储session/cookie中 使用一次就删除。

1. 将所有的请求参数写入闪存中 $request->flash();
2. 除去某些参数之外的参数写入闪存中 $request->flashExcept(‘\_token’);



1. 将部分参数写入闪存中 $request->flashOnly(‘title’,’price’);

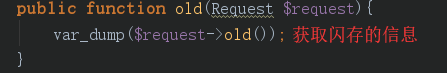
只存入title、price字段的信息。

1. 携带信息返回 return back()->withInput();

携带闪存的信息返回。

* 【获取】

Laravel5.5中获取闪存的信息用old方法来获取



## 3.7响应

### 3.7.1 返回json

return response()->json([‘a’=>100,’b’=>200]);

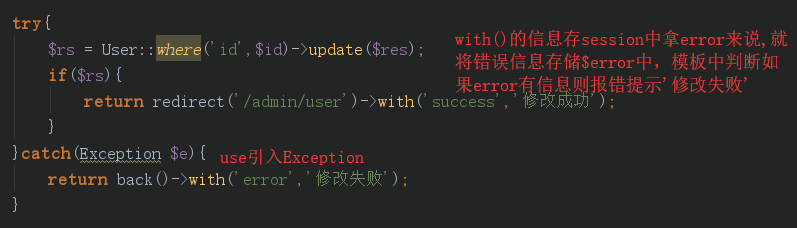
### 3.7.2 下载文件

return response()->download(‘’);

### 3.7.3 页面跳转

return redirect(‘/goods/add’);

一般配置这try{}catch(){}语法来使用,成功让你跳转，失败让你返回



## 3.8模板

### 3.8.1 解析模板

1.解析

return view(‘user.index’);

2.分配数据

return view(‘user.index’,[‘title’=>’Hyt’]);

3.与Tp5的差别

其实解析模板的方法有两种，第一种是.还有一种与TP通用的就是/。而TP5的第一种方法适用:

### 3.8.2 模板引擎blade

Laravel框架中使用balde。如：index.blade.php

* 模板中使用变量

Laravel中使用{{$username}}，两个{{}} 来解析。

* 模板中使用函数

{{$time()}}

* 设置默认值

{{$username or ‘Hyt’}}

类似三元运算符：$username有值就用，无值就用Hyt

* 显示html代码

{!!$name!!}

如：百度编辑器存入数据库的数据有HTML的标签。在修改时就用该方式来显示

* 引入子视图

@include(‘header’)

* 模板的继承

1. 继承整个模板

@extends(‘layout.admin’)

继承layout文件夹下的admin.blade.php模板

1. 重写模板

重写前需要明确重写模板中的那个部分。需要先用占位符明确。

2.1占位符

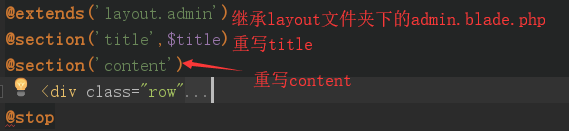
单一占位：@yield(‘title’)

范围占位：@section(‘content’) @show

2.2重写

单一重写：@section(‘title’,’Hyt’)

范围重写：@section(‘content’) @endsection(或用@stop)



* 流程控制

@if() @elseif() @else @endif

* 循环控制

@foreach() @endforeach

## 3.9 数据库操作

### 3.9.1 支持数据库的类型

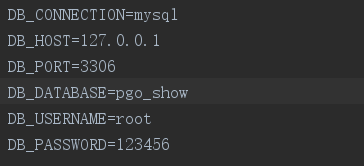
Laravel框架中支持一下几种数据库：

MySQL、Postgres、SQLite、SQL Server

### 3.9.2 数据库的配置

文件位置：config/datebase.php (一般不动该文件)

一般配置都在.env文件中配置修改。



### 3.9.3 数据库基本操作

使用DB是需要先引入



* 【查询】

DB::select();



* 【插入】

DB::insert();



* 【更新】

DB::update();



* 【删除】

DB::delete();



* 【一般语句】

DB::statement(‘’);

* 【事物操作】

DB::beginTransaction() 开启

DB::rollBack() 回滚

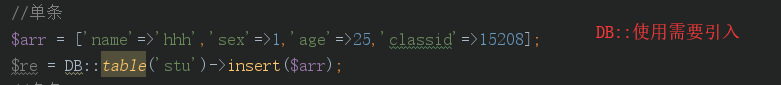
DB::commit() 提交

### 3.9.4 构造器

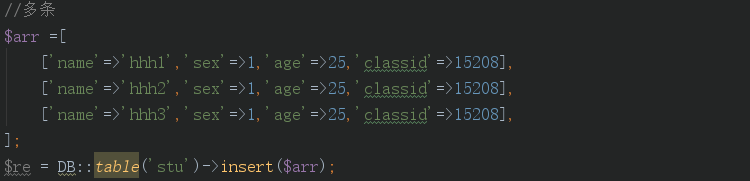
* 【增】

DB::table(‘表名’)->insert(‘数据’);

1. 单条



1. 多条



* 【删】

DB::table(‘表名’)->where(‘条件’)->delete();



* 【改】

DB::table(‘表名’)->where(‘条件’)->update();



* 【查】

1. get()查询全部



1. first()查询单条 //还要一个是find(查询条件)





1. 查询单条结果中的某个字段 value()



1. 获取一列数据 pluck()



* 【连贯操作】
* 设置获取字段



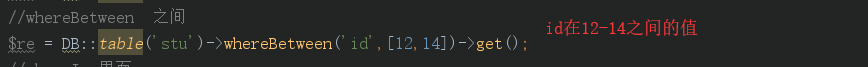
* 条件查询
  1. like模糊查询



* 1. orWhere



* 1. whereBetween



* 1. whereIn



* 排序



* 分页

DB::table(‘users’)->paginate(‘显示条数’);

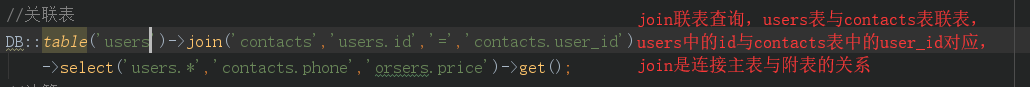
模板中：5.1版本的是 {!!$users->render()!!} 5.5版本是{!!$user->links()!!}

1. 将附加参数到分页链接中:appends([‘sort’ => ‘votes’])->links();

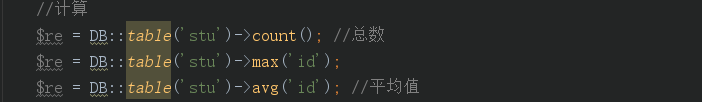


1. 总条数

* 关联表



* 计算



### 3.9.5 Eloquent ORM(Model)

就是mvc中的Model和tp5中的类似，需要cmd命令行创建。

使得每个数据库表都有一个对应的「模型」可用来跟数据表进行交互。

同样控制器在使用是需要引入。

* 【创建】

Cmd命令行下:

php artisan make:mode User

默认创建在app目录下的Model目录(会自动创建)

* 【调试】

创建完成后需要进行调式修改。

1. 添加表名



1. 添加主键名

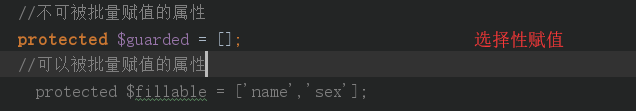


1. 该模型是否被自动维护时间戳



Laracel默认情况下，Eloquent会认为在你的数据库表中有created\_at和updated\_at字段。如果你不需要就将$timestamps = false; 同时还要注意这个设置时public 不是protected。

1. 设置可添加或不可添加的属性



不可赋值：$guarded=[];

可被批量赋值：$fillable = [];

* 【实例】



* 【补充】

Laravel框架或Tp5框架中也好所有的增、删、改、查语句中，不过有多少个条件查询，但执行语句(insert、delete、uodate、select(get、first)等等)一定要放到最后，否则会报错。

### 3.9.6 模型关系

* 【一对一】

在这里我们用user表和userinfo表来举例说明。

假定：user表的ID关联userinfo表中的uid。这里我们用user表来关联userinfo表。

首先需要在User模型中创建以一个userinfo的方法,在userinfo方法内部调用hasOne()方法来关联userinfo表。

public function userinfo(){

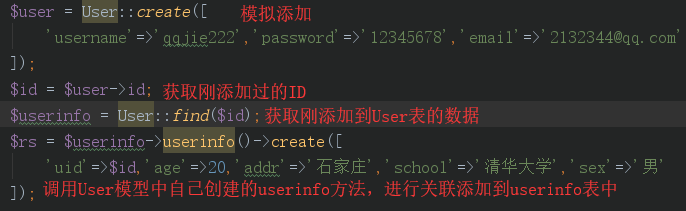
return $this->hasOne(‘App\Http\Model\Admin\Userinfo’,’uid’);

/hasOne参数1：是Userinfo模型的命名空间+名称/

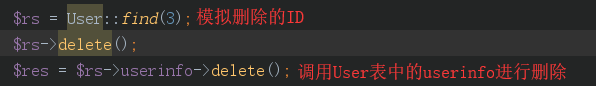
/hasOne参数2：是Userinfo表中的外键/

}

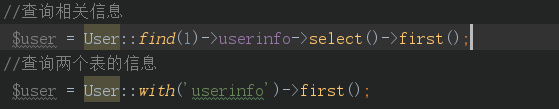
* 添加 添加User表数据，Userinfo表的数据跟着添加。



* 删除



* 查询



* 【一对多】

类似一对一，但需要修改hasOne方法为hasMany()方法

public function hasMany(){

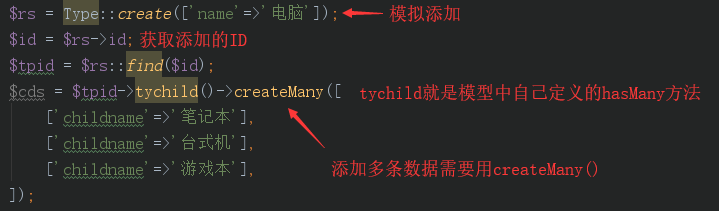
return $this->hasMany(‘App\Http\Model\Admin\Userinfo’,’uid’);

/hasOne参数1：是Userinfo模型的命名空间+名称/

/hasOne参数2：是Userinfo表中的外键/

}

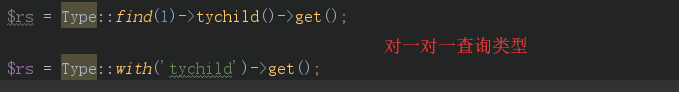
* 添加



* 删除



* 查询



* 【多对多】

后期补充

## 3.10 设置自定义函数/类

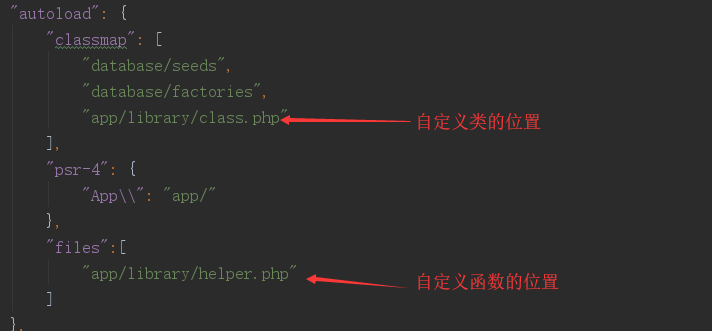
拿自定义函数来举例

1. 创建文件(名字随意)

app/library/helper.php (自定义函数)

app/library/class.php(自定义类)

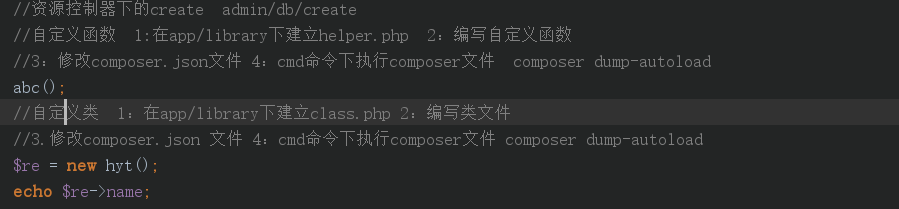
1. 修改composer.json



1. 执行

Cmd命令下:composer dump-autoload

1. 使用



## 3.11 哈希加密

Laravel通过Hash来提供加密保护，如果你在当前laravel应用程序中使用了内置的LoginController和RegisterController类，他们将自动使用Bcrypt进行注册和身份验证。

同时需要引入Hash类。



### 3.11.1 加密

Hash::make(‘加密对象’);



### 3.11.2 验证

Hash验证加密时，不想md5加密的思路一样，加密后与数据库对比。因为Hash加密，即使相同的值也加密后是不一样的。

Hash::check(‘验证对象’,’对比对象’);

if (Hash::check('plain-text', $hashedPassword)) {

// 密码对比...}

## 3.12 加密解密

加密解密是另一种加密解密的方法与Hash是有区别的。

### 3.12.1 加密

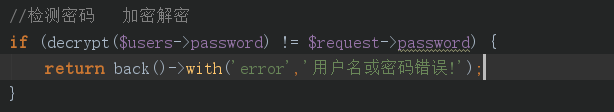
使用encrypt辅助函数来加密。(Hash是类的方法)

encrypt(‘加密对象’)

### 3.12.2 解密

使用decrypt辅助函数来解密一个值，比如你在注册时使用encrypt将密码加密后存储数据库，那么在你登陆的时候使用decrypt将数据库的值解密后去输入的对比。

decrypt(‘解密对象’)



## 3.13 表单验证

表单验证有两种方法，第一种就是在控制器中直接验证(个人起名控制器验证)，第二种方法就是cmd命令行下创建表单请求验证。

### 3.13.1 控制器验证

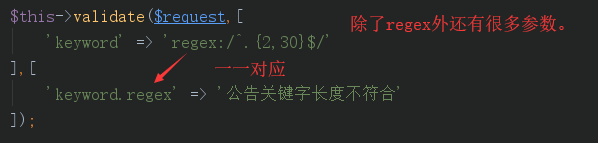
控制器验证直接在需要使用的方法中进行验证。

$this->validate($request, [

'title' => 'required|unique:posts|max:255',

'body' => 'required',

'publish\_at' => 'nullable|date',]);



可选字段标记nullable.

### 3.13.2 表单请求验证

* 【创建】

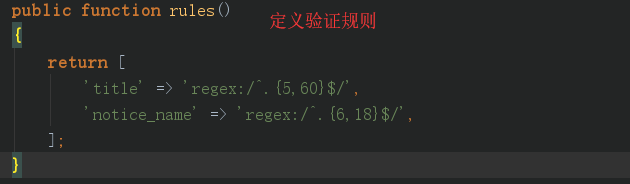
cmd命令行下创建：

php artisan make:request IndexRequest

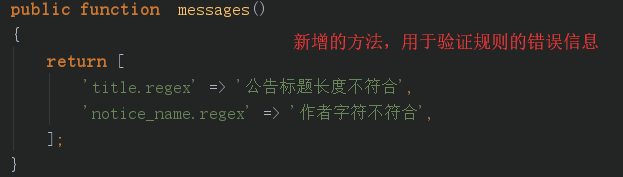
默认会在app/Http/Requests目录下创建生成。

* 【使用】

1. 在rules()方法下定义验证规则



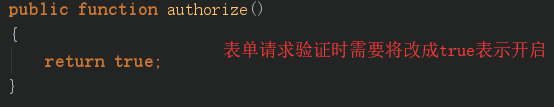
1. 新建messages()方法来获取定义验证规则的错误信息



* 【注意】

使用表单请求验证时需要开启、引用。

开启：



引用：





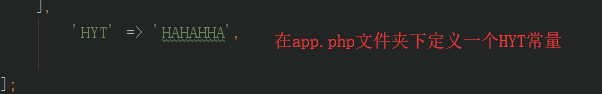
## 3.14 零碎

### 3.14.1 打印

dump()打。dd()打印后die。

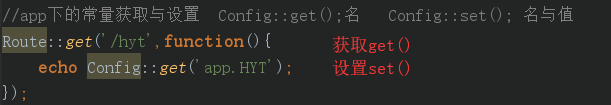
### 3.14.2 常量

app.php文件下定义常量。



使用：

Config::get(‘app.HYT’)



### 3.14.3 public目录下注意

public目录下一般存放静态的img、css、js文件。但文件命名不能时需要注意与路由的名称区分一下，否则会显示报错，说你权限不够。

