# Laravel框架\_02

# 今日目标5

* 能够使用php artisan make:model 命令创建模型

php artisan make:model 目录名/模型名 –m 生成迁移文件

编写迁移文件

php artisan migrate

* 能够使用laravel模型进行数据添加

save

insert

create 必须要在模型中定义允许或拒绝的属性 $fillable $guarded

update

delete

first

get

php artisan make:observer xxx

creating created

* 能够使用模型类的paginate方法实现分页

模型：：paginate(分页数)

模板

{{ 变量->appends([])->links()}}

* 能够掌握Laravel中session数据的存储

session([])

session(‘’)

session()->has()

session()->flush()

session()->flash(key,value)

redirect()->route()->with(key,value)

* 能够使用命令创建中间件

php artisan make:middleware 中间件名称

全局

路由

绑定

指定的路由

指定的路由分组

指定的控制器

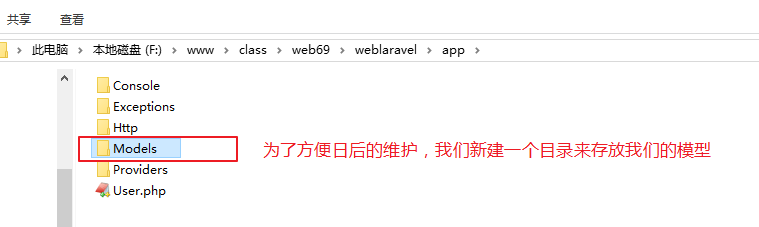
# 一、模型操作(Eloquent ORM)

Laravel 自带的 Eloquent ORM 提供了一个美观、简单的与数据库打交道的方案，每张数据表都对应一个与该表进行交互的“Model模型”，模型允许你在表中进行数据查询，以及插入、更新、删除等操作。

## 1.1、模型文件的位置

默认情况laravel模型在app目录的根目录下面。但这种情况不利于我们日后项目维护，所以我们一般建议手动创建一个目录，这样更加的方便去管理和维护。





## 1.2、创建模型

php artisan **make:model** [目录名/]模型文件名

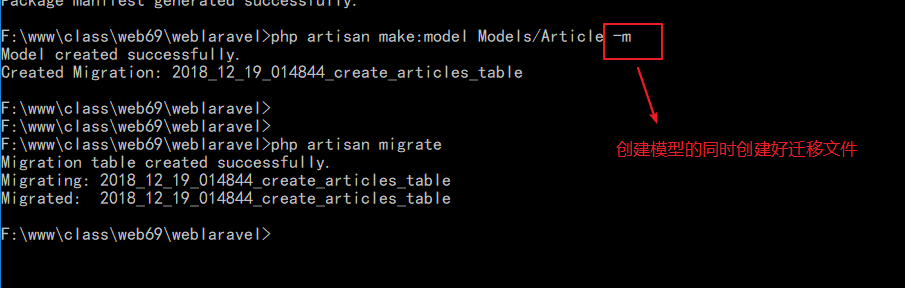
# 分目录的

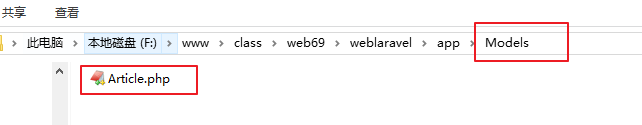
php artisan make:model Models/Member

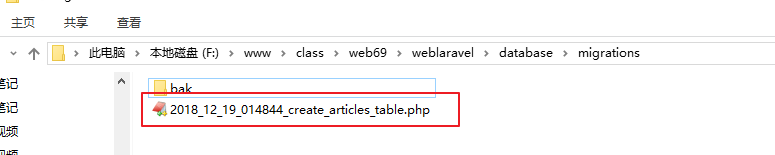
# 创建模板并生成迁移文件 实际工作中，就是创建模型同时就创建好了迁移文件

php artisan make:model Models/Article **–m**

-m 生成迁移文件







## 1.3、模型限制

### 1.3.1、定义表名($table)

模型所对应的默认表名是模型类名复数形式，如果表名不统一则需要指定一下。

// 指定表名

protected $table = 'member';

### 1.3.2、指定主键名($primaryKey)

laravel中默认的主键ID名为id，如果你创建的表字段中主键ID的名称不为id，则需要通过 $primaryKey 来指定一下。

// 指定主键的名称

protected $primaryKey = 'mid';

### 1.3.3、指定时间戳($timestamps)

默认情况下，模型操作会认为在你的数据库表有 created\_at 和 updated\_at 字段。如果你不希望让模型来自动维护这两个字段，可在模型内将 $timestamps 属性设置为 false。

// 时间戳 这里一定要注意它是用的public

**public** $timestamps = false;

### 1.3.4、批量赋值($fillable 与 $guarded)

当通过**create**方法来保存数据的时候，你需要先在你的模型上定义$fillable或 $guarded属性。

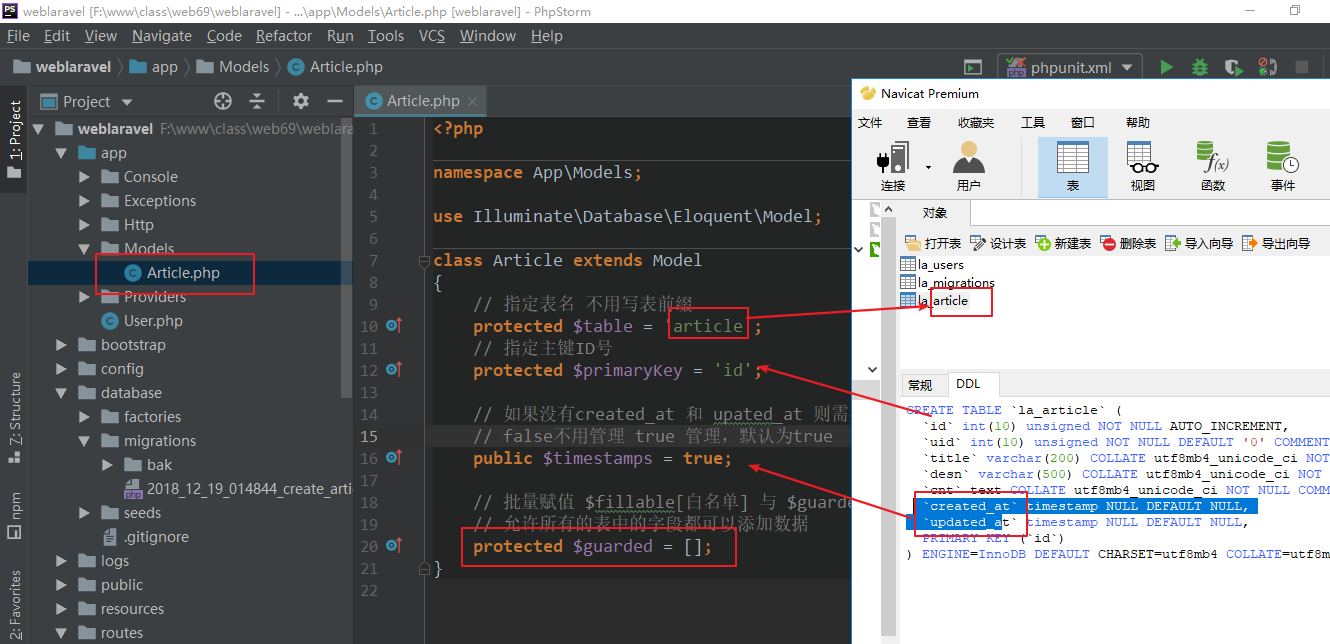
// 批量赋值

# 白名单

protected $fillable = ['允许添加的字段名'];

# 黑名单

protected $guarded = ['拒绝添加的字段名'];



## 1.4、CURD基本操作

### 1.4.1、添加数据

# 方法一

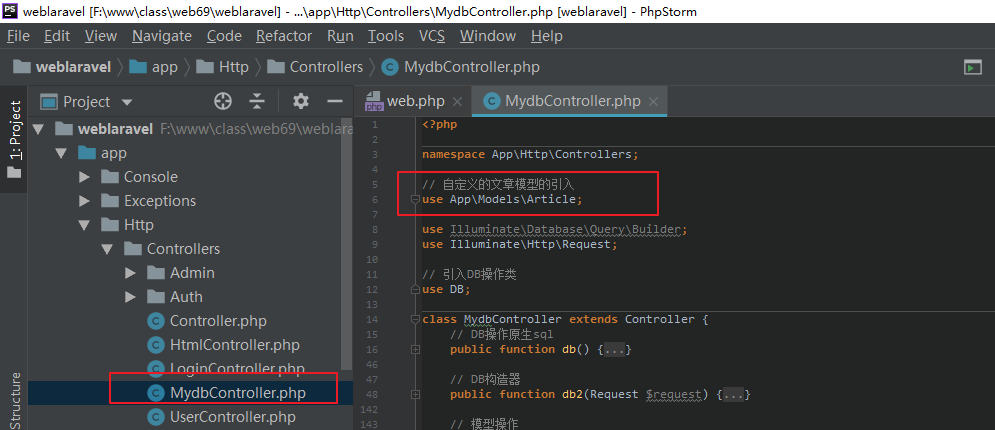
save() # 对象属性方式 返回模型对象

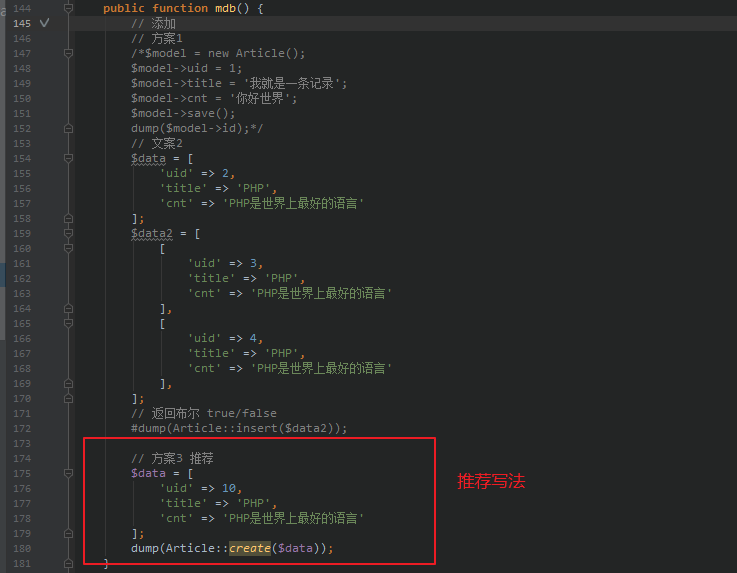
# 方法二

insert() #数组方式 可以添加一条或多条，返回true/false

# 方法三

create() # 数组方式 需要在模型中设置好fillable或guarded属性 返回是模型对象





### 1.4.2、查询数据

# 查询单条

first() find($id) 要想让find不报错，一定字段为id或者模型中定义了主键名称

# 查询多条

get()

# 条件

where()

# 排序

orderBy()

# 取指定的值

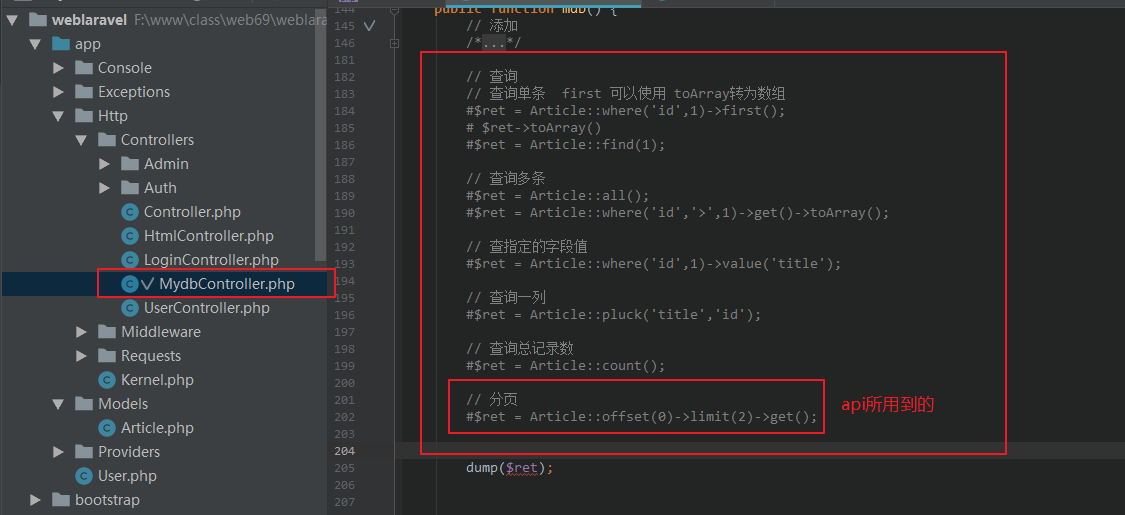
value()

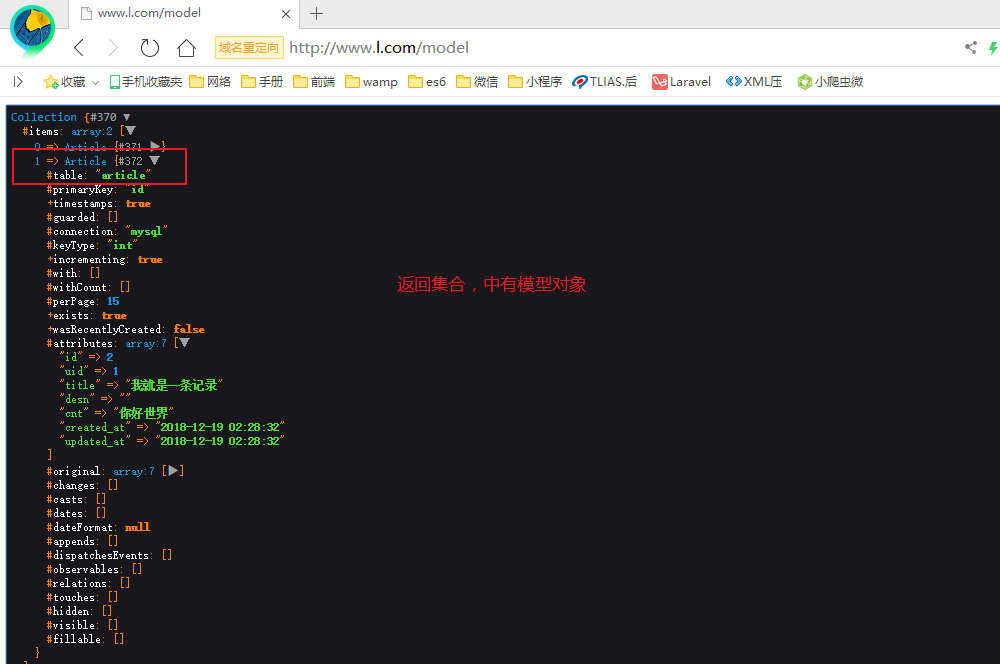
# 取指定的列

pluck()

# 查询总记录数

count()





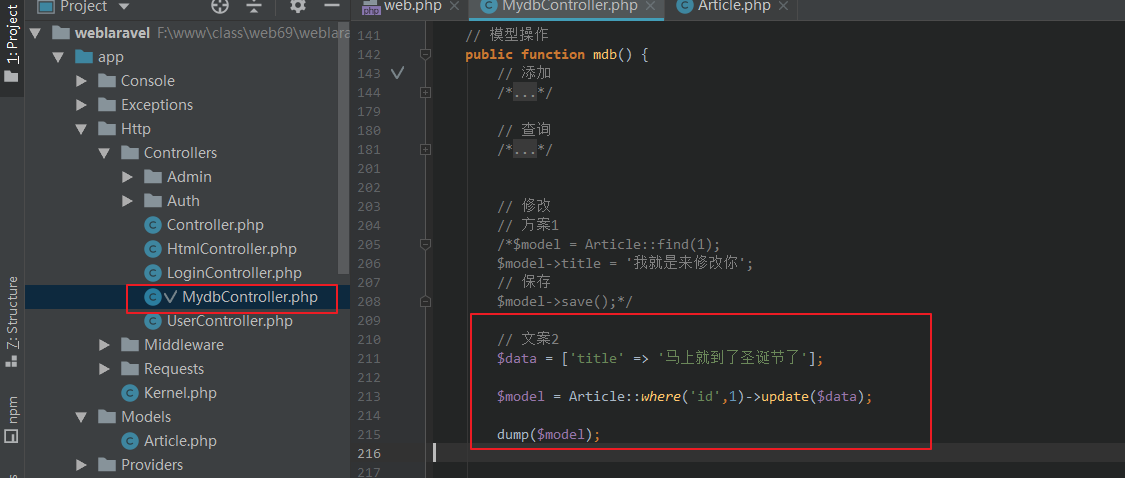
### 1.4.3、修改数据

# 方法一

save() # 对象属性方式 返回的是模型对象

# 方法二

update() # 数组 返回受影响行数



### 1.4.4、删除数据

# 删除 一定要指定条件删除

delete() # 对象删除

destroy($id) 静态方法删除 默认表字段主键名为id,如果不是则需要模型中定义一下

# 软删除

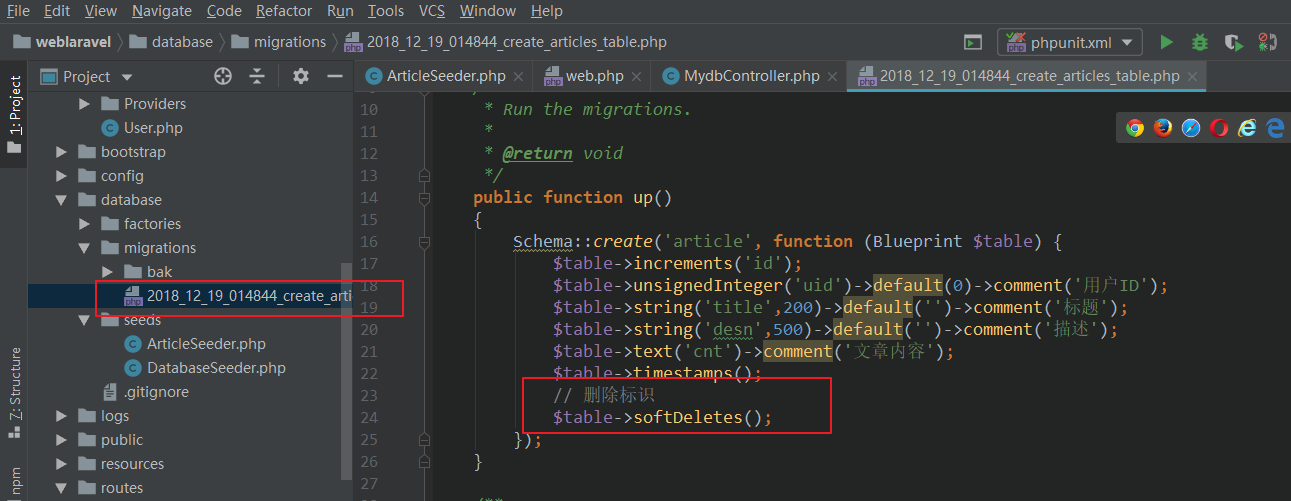
1、在表字段中要有deleted\_at字段 可以在迁移文件中直接添加一个方法 $table->softDeletes()

2、在模型中引入 trait use SoftDelele;

protected $dates = [删除标识字段名 deleted\_at]

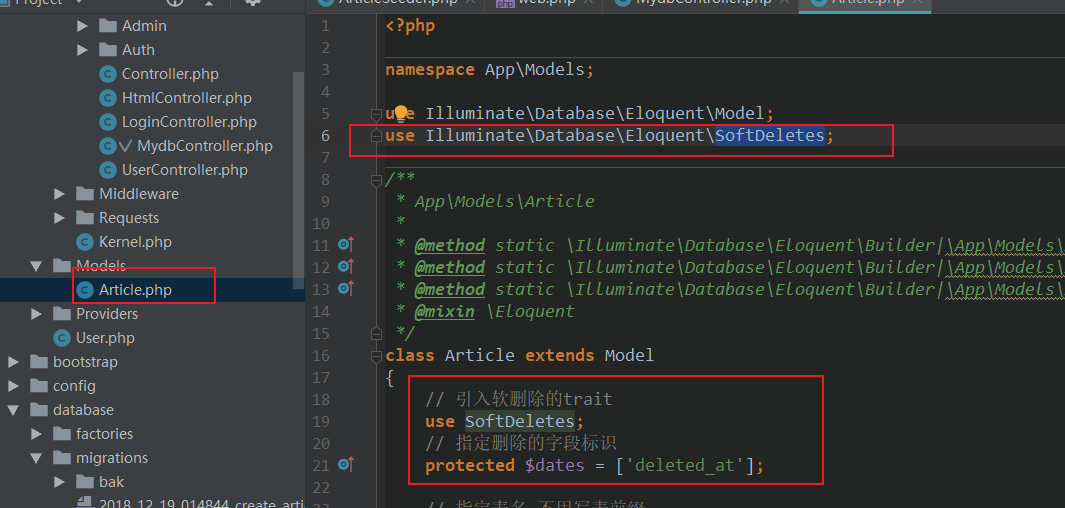
软删除

修改迁移文件

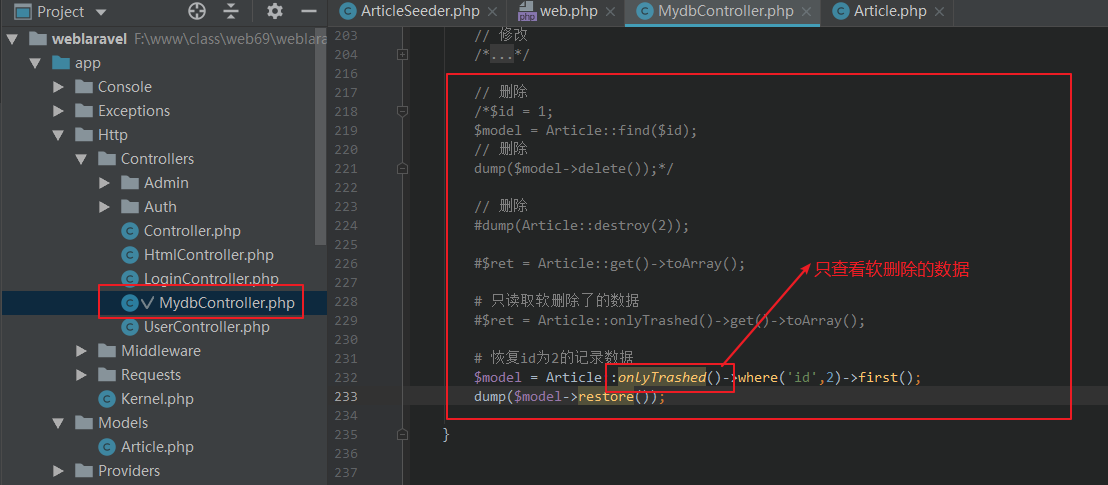


php artisan migrate:refresh --seed

修改模型文件



控制器



软删除说白就是给标识字段中添加一些数据，让此字段不为null

## 1.5、faker来进行数据填充

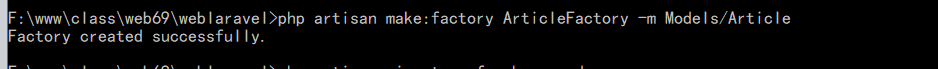
<https://packagist.org/packages/fzaninotto/faker>

larvel从5之后默认就已经安装了此扩展包

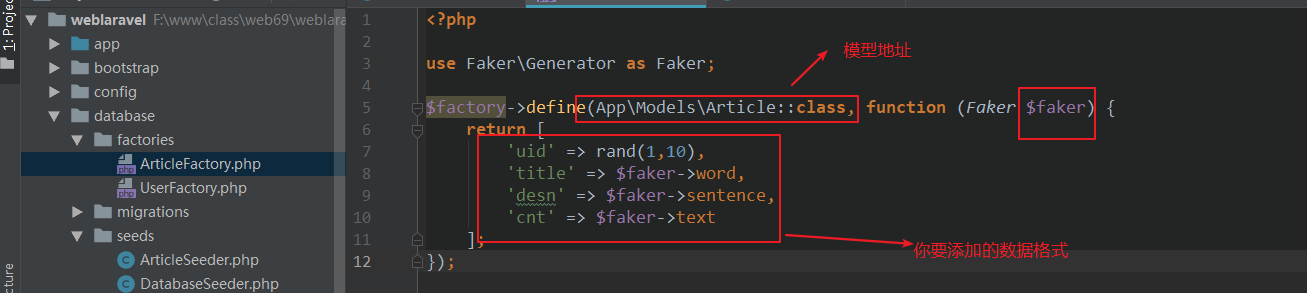


### 第1步：创建数据工厂

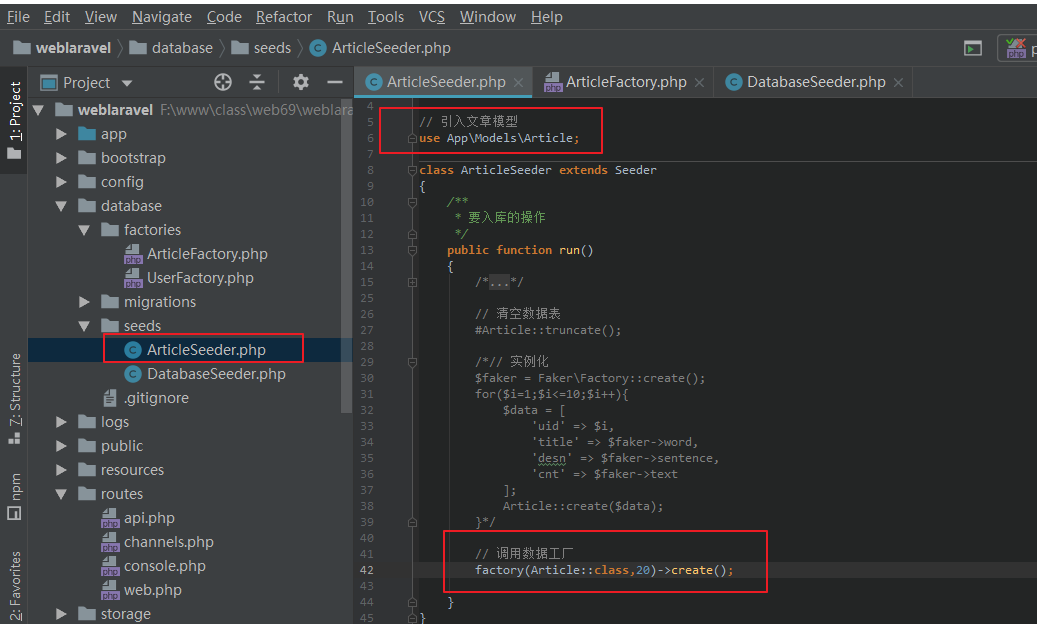
php artisan make:factory 工厂的文件名 -m 模型名(目录 Models/Article)



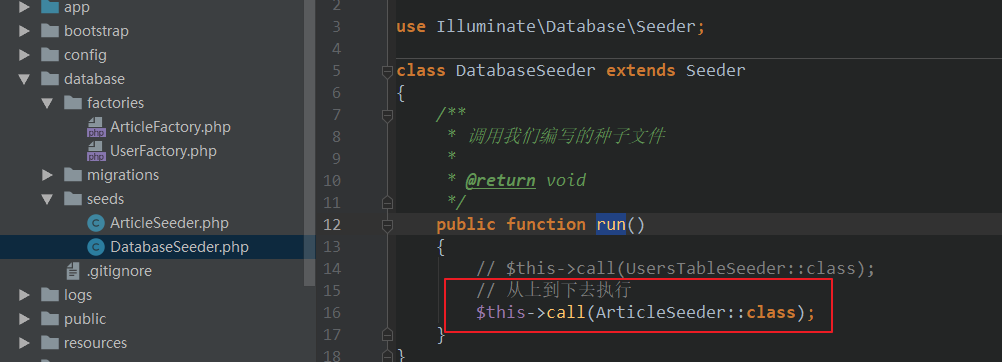




### 第2步：在种子文件中调用

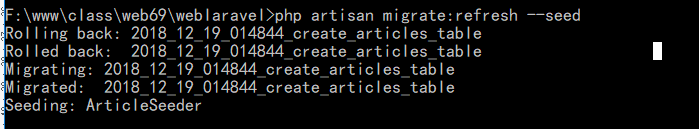


### 第3步：DatabaseSeeder文件中调用种子



### 第4步：执行种子

php artisan migrate:refresh --seed

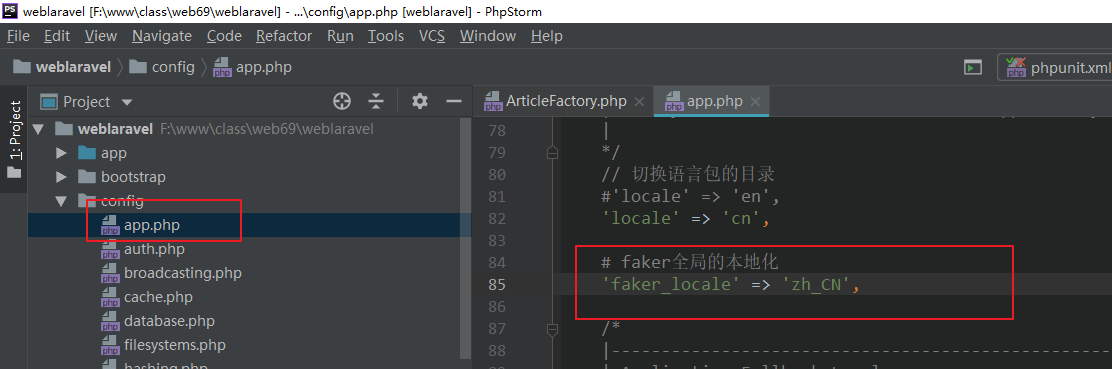


### 第5步：本地化

方案1

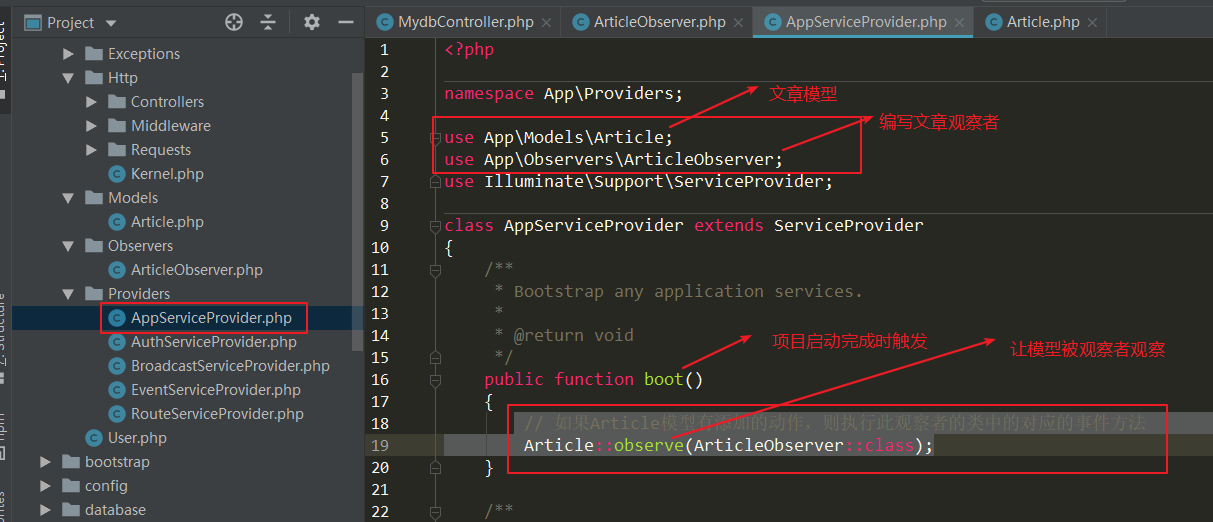


方案2 推荐

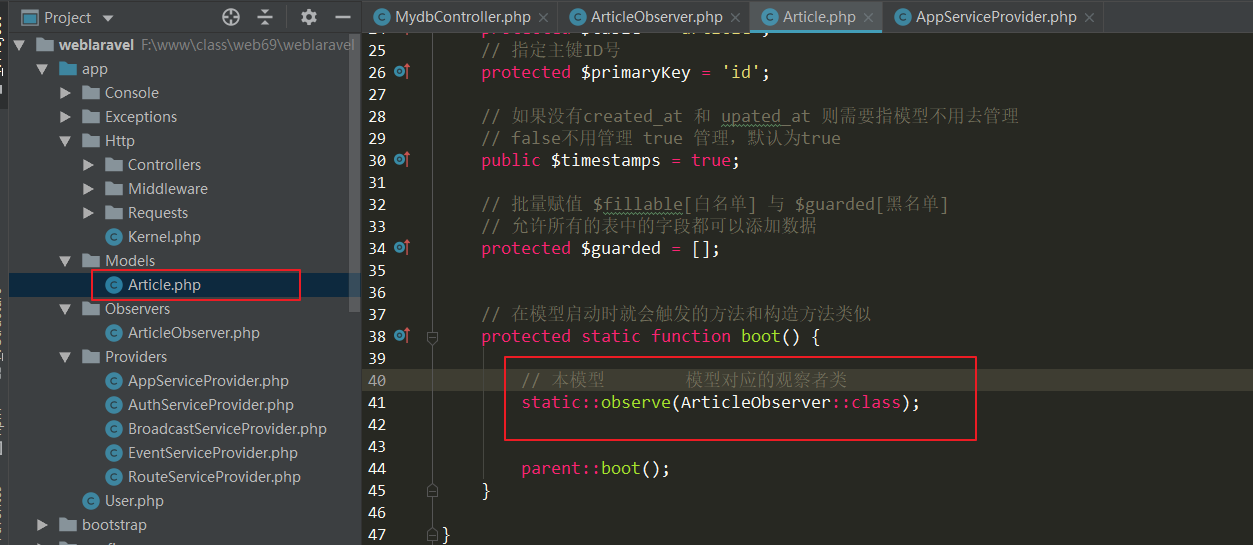


## 1.6、模型事件

方案1：在AppServiceProvider中的boot方法中注册模型对应的观察者。



方案2：在本模型中去注册观察 推荐



# 二、数据分页

在laravel里面要完成分页是很简单的，laravel是直接调用模型的分页方法，返回对应的数据和分页的字符串。

# 模型

# 标准分页

Model::paginate(每页显示的记录数)

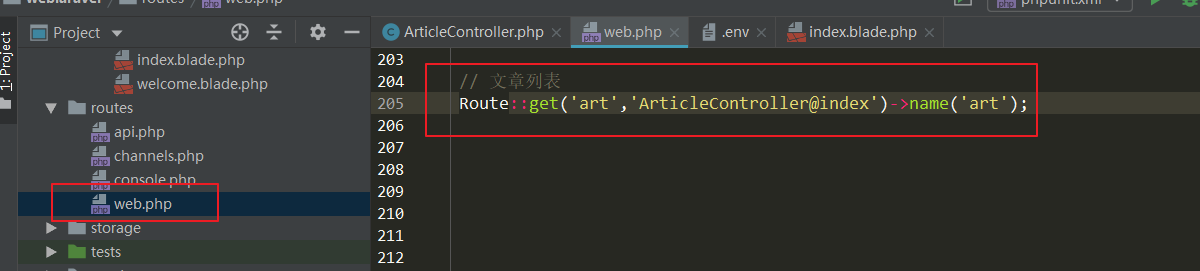
# 简单分页

Model::simplePaginate(每页显示的记录数)

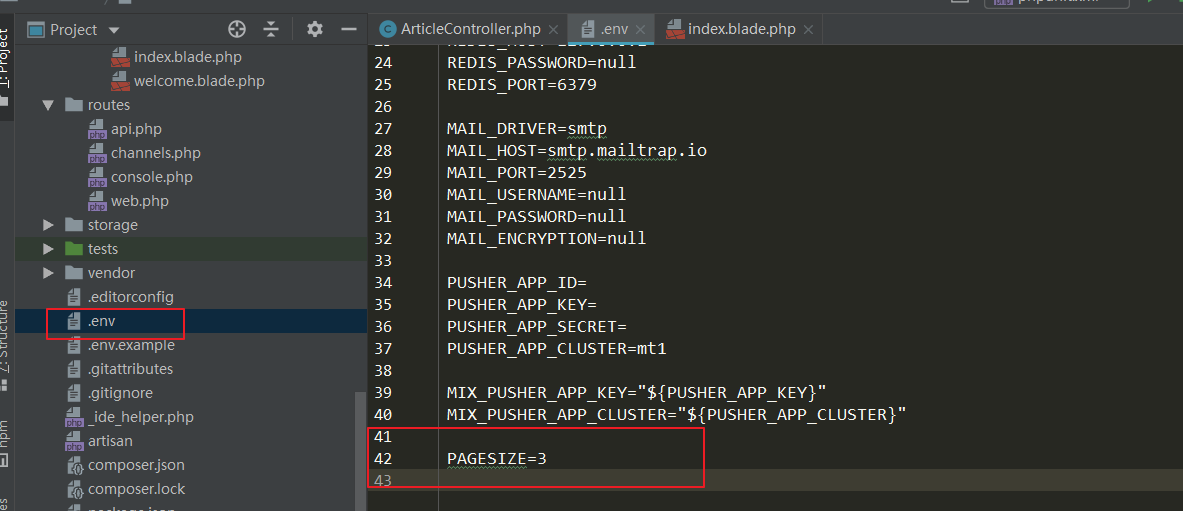
# 模板

{{ $model->links() }}

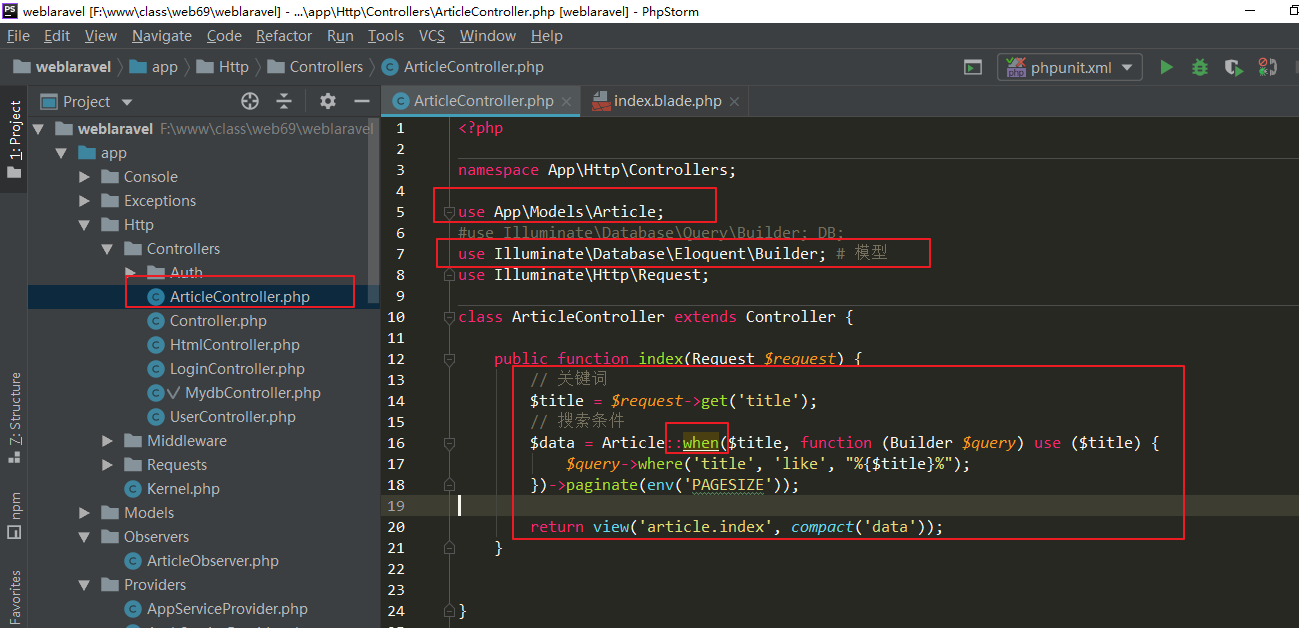
定义路由



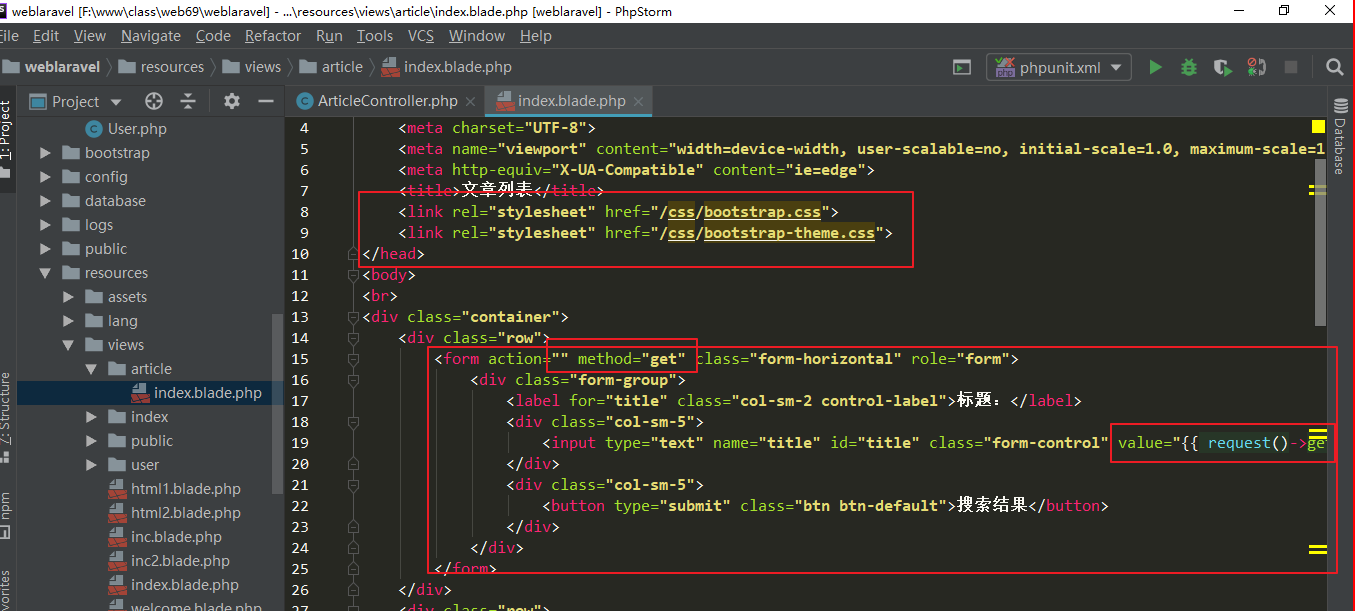
定义分页的数量

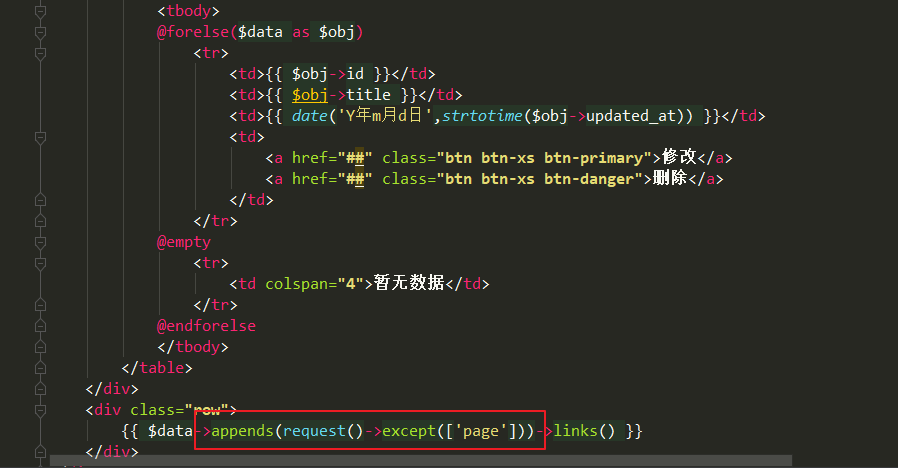


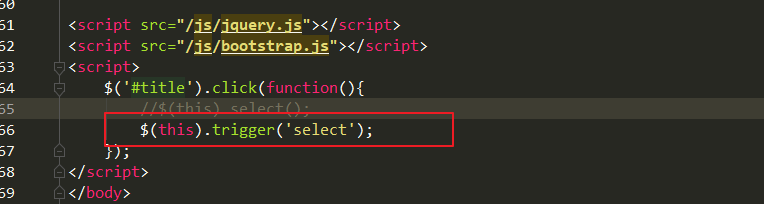
控制器中实现分页搜索功能



视图







效果



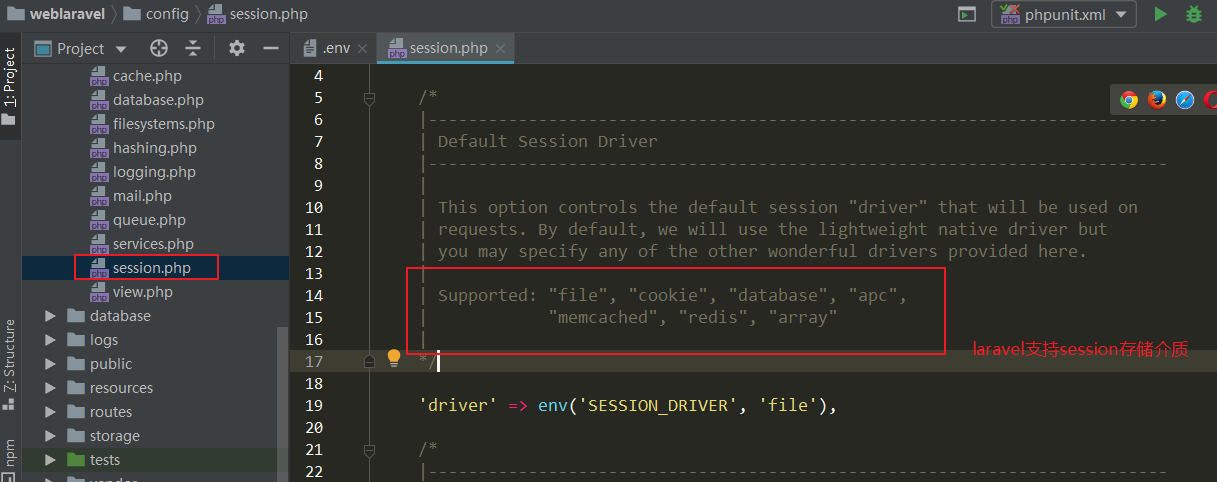
# 三、session处理

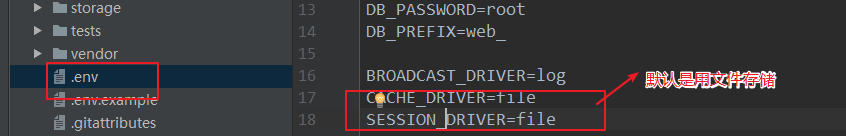
laravel中的session默认存到文件中

session文件的目录：storage\framework\sessions

操作session的方法在laravel中有2种方案，有类来操作也有利用辅助函数来操作。

* Session类来操作
* 辅助函数来操作 session()





## 3.1、设置session

// 写入session

// 方法一

Session::put('键名','值');

// 方法二 辅助函数

session(['键名'=>'值']);

注：辅助函数存储session的时候一定要以数组的形式来定义。

## 3.2、读取操作

// 读取session

// 方法一 Session类

dump(Session::get('session名称',[默认值]));

// 方法二 辅助函数

dump(session(' session名称'));

// 读取全部的session

dump(Session::all());

dump(session()->all());

## 3.3、判断某个 Session 值是否存在

// 方法一

dump(Session::has('session名称'));

// 方法二

dump(session()->has(' session名称'));

## 3.4、删除session

// 删除Session

// 方法一

Session::forget('session名称');

if (!Session::has('session名称')){

echo 'name1：已被删除掉<hr>';

}

// 方法二

session()->forget('session名称');

if (!session()->has('session名称')){

echo 'name2：已被删除掉<hr>';

}

-------------------------------------------------

// 删除全部的session数据

// 方法一

Session::flush();

// 方法二

session()->flush();

## 3.5、闪存数据

使用这个方法保存 session，只能将数据保留到下个 HTTP 请求，然后就会被自动删除。

// 方法一

Session::flash('msg','你好世界');

dump(session('msg'));

// 方法二

session()->flash('msg','你好世界');

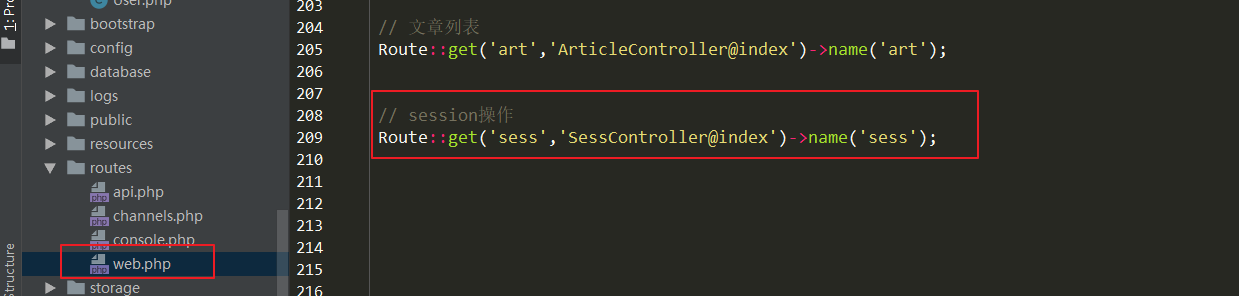
dump(session('msg'));

// with来设置闪存

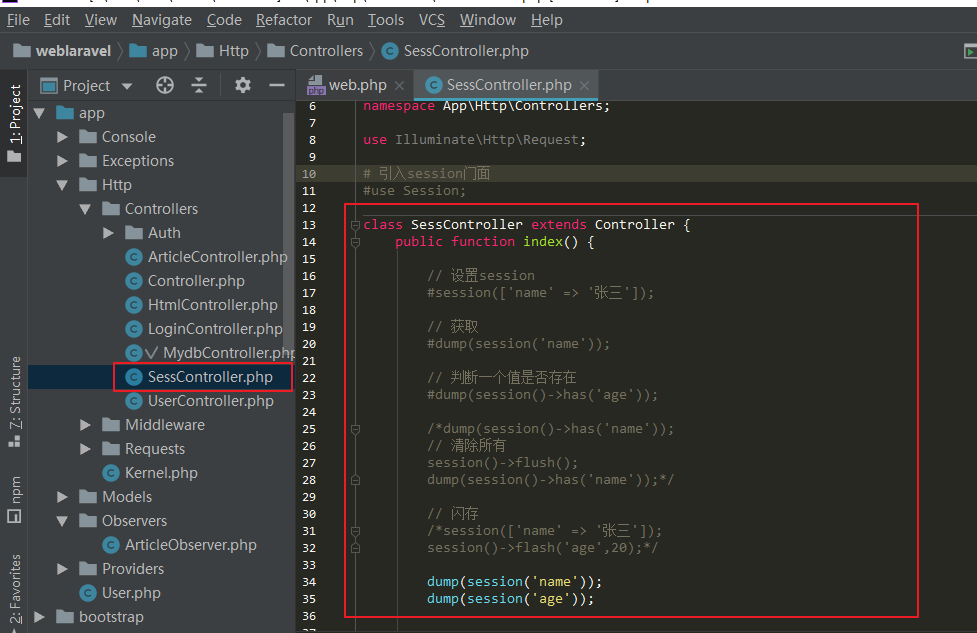
return redirect()->route('session.index')->with('msg','我是成功的');

// 模板中的使用 {{ session(key) }}来读取闪存

路由



控制器



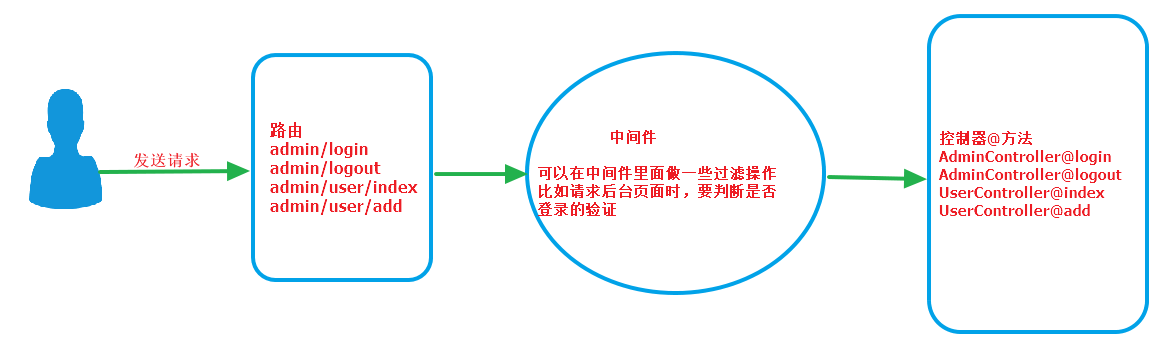
# 四、中间件

中间件作为请求和响应之间的中间人。它是一种过滤机制类型。

在laravel中通过中间件来进行后台用户是否登录的验证操作，防止后台用户翻墙操作。

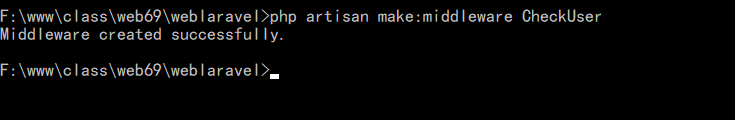
中间件可以通过执行以下命令来创建，所有的中间件都位于app/Http/Middleware目录。

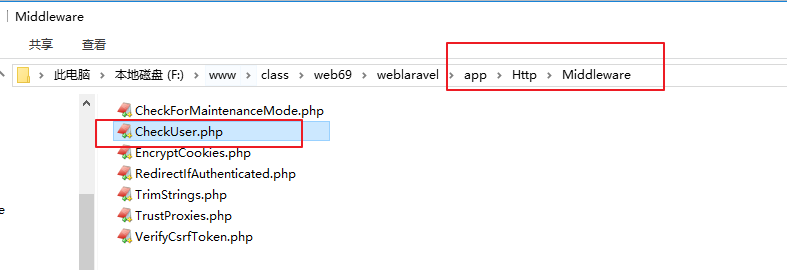
php artisan make:middleware 中间件名

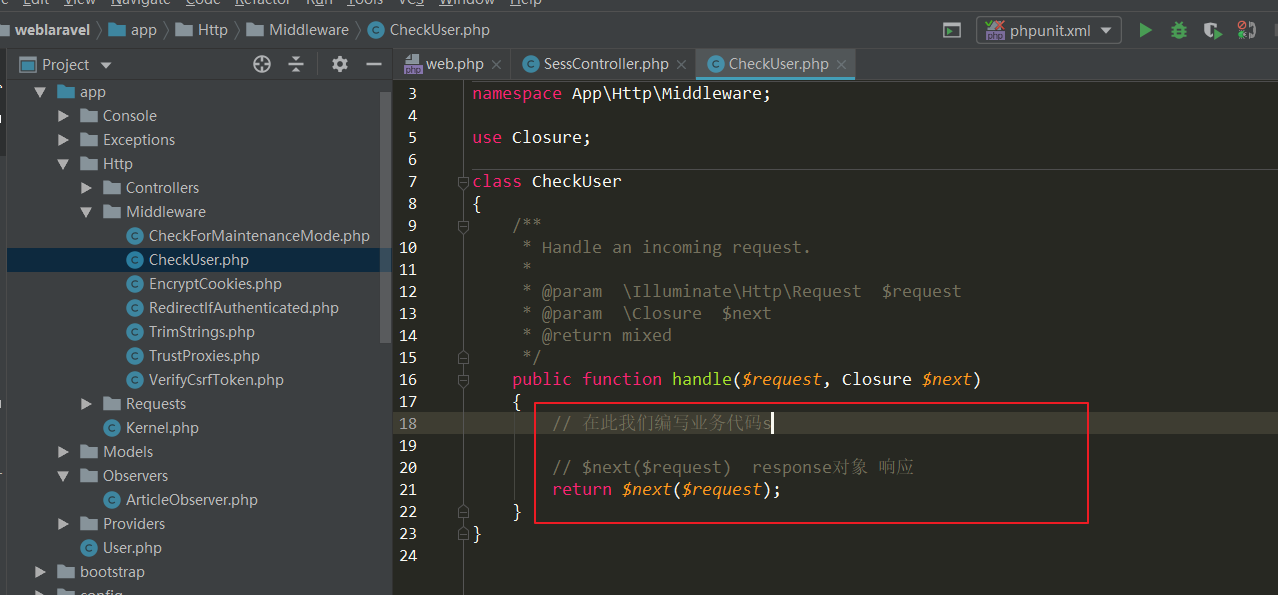


## 4.1、定义中间件

php artisan make:middleware 中间件文件名称







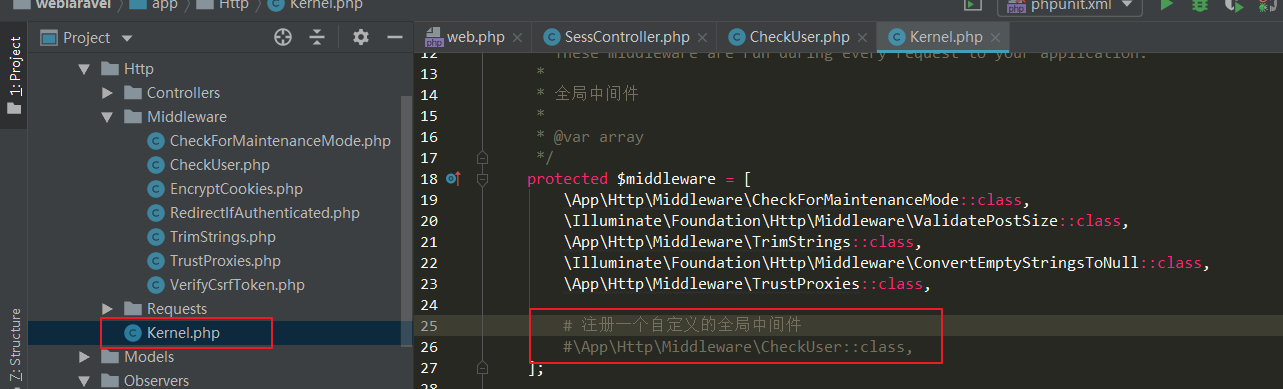
## 4.2、注册中间件

在Laravel有两种类型的中间件。即：【全局中间件】和【路由中间件】

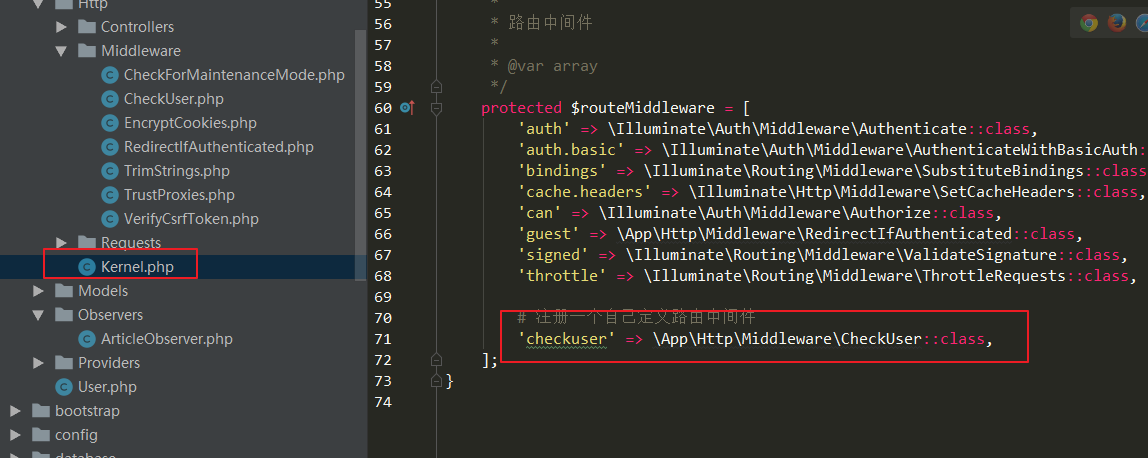
全局中间件将在应用程序的每个HTTP请求运行，而路由中间件将被分配到一个特定的路由。中间件可在**app/Http/Kernel.php**注册

该文件包含两个属性： $middleware 和 $routeMiddleware。$middleware 属性用于注册全局中间件，$routeMiddleware属性用于注册路由指定中间件。

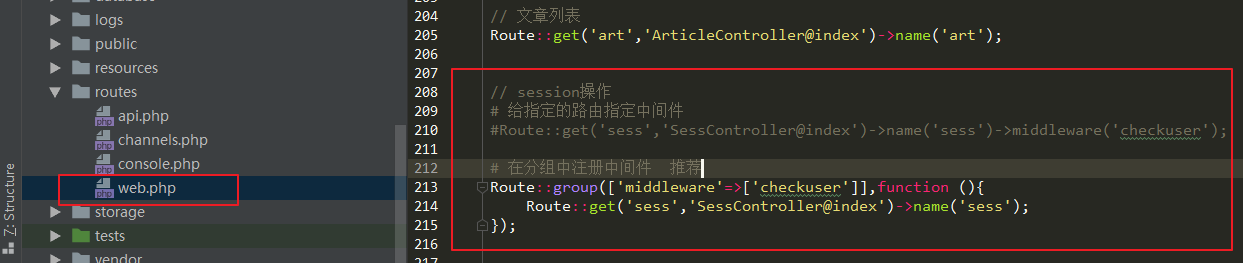
注册一个全局中间件



注册一个路由中间件



如果注册的是路由中间件，则一定需要绑定到指定的路由中



控制器绑定

