目录

[三、 项目初始化 1](#_Toc26896)

[1、基本配置操作 1](#_Toc8128)

[2、 配置虚拟主机 6](#_Toc22954)

[四、 后台基础功能开发 7](#_Toc4463)

[1、 创建管理员数据表 7](#_Toc31553)

[2、 模拟管理员表的数据 9](#_Toc1367)

[3、 后台登录页面的展示 11](#_Toc24487)

[4、 实现验证码的替换 12](#_Toc14705)

[5、 管理员登陆的自动验证 14](#_Toc14996)

[6、 管理登录身份验证【难点】 15](#_Toc29080)

[7、 后台首页 19](#_Toc11123)

[8、 路由保护 21](#_Toc30367)

[五、 管理员管理 23](#_Toc10903)

[1、实现管理员的列表展示 23](#_Toc3884)

[2、使用datatables插件实现无刷新分页 24](#_Toc23843)

[a. 禁用掉第一列（复选框）排序 25](#_Toc18568)

[b. 更改默认的初始化排序操作 25](#_Toc27948)

[六、 RBAC 权限管理 25](#_Toc22201)

[1、 介绍 26](#_Toc12246)

[2、 数据表设计 26](#_Toc20131)

[3、 理清三表之间的关系【重点】 28](#_Toc29058)

[4、 权限管理操作 29](#_Toc20998)

[4.1 展示相关视图 29](#_Toc16453)

# 项目初始化

## 1、基本配置操作

一 时区、本地化、数据库配置、禁用严格模式、删除自带的测用户基础文件 安装debugbar

①创建laravel 项目

1. 直接去找第三方的数据包(不推荐)
2. composer 安装laravel 项目

#composer create-project laravel/laravel=5.4.\* --prefer-dist ./

解析:

laravel/laravel=5.4.\* 指定laravel的版本

--prefer-dist:使用压缩版

./:指的是目录

#composer create-project --prefer-dist laravel/laravel blog “5.6.\*”./

解析:

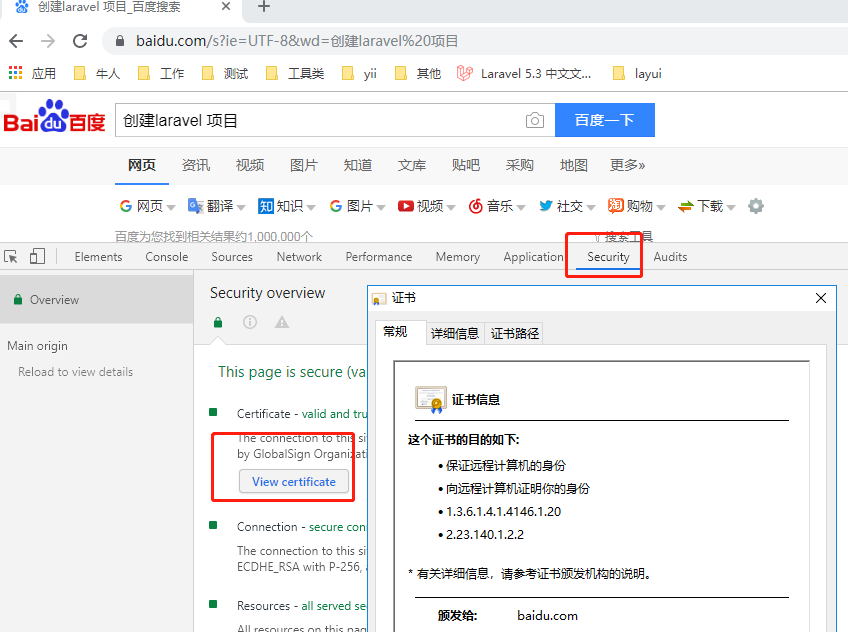
laravel/laravel:包名, 表示创建一个laravel项目

blog:项目目录

--prefer-dist:使用压缩版

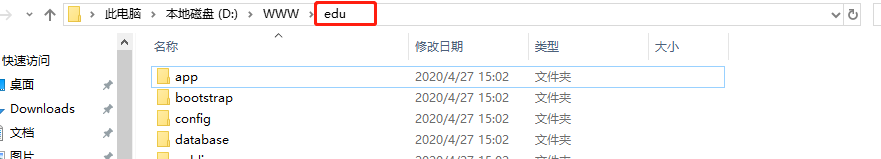
5.6.\*: laravel 对应版本

**插播 如何查看 https证书 :**



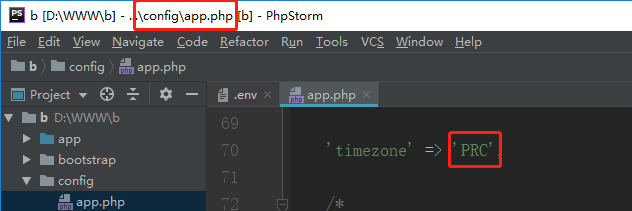


项目目录：



②设置时区 修改 config/app.php

修改 timezone=PRC;



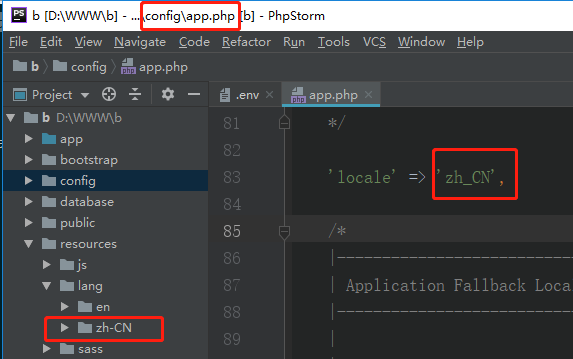
③项目本地化

下载语言包

命令: composer require caouecs/laravel-lang:~6.0

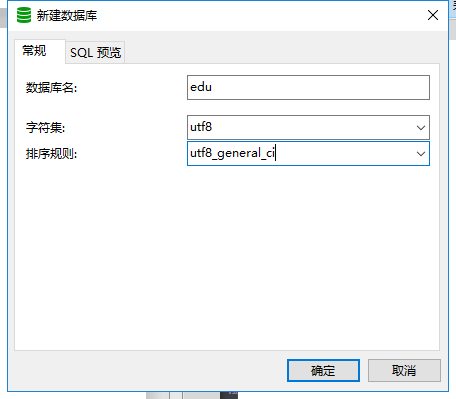
将语言包文件从vendor/caouecs/laravel-lang/src 中复制到 resources/lang 目录下

修改config/app.php 中的关于本地化的配置项 locale =>“zh-CN”

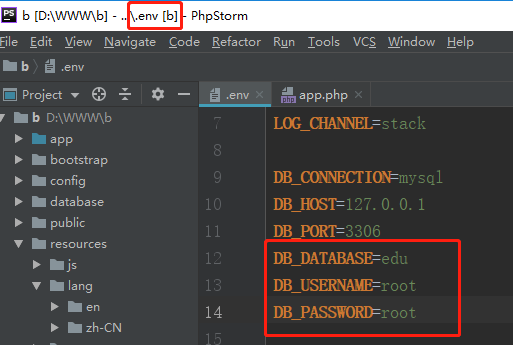


④配置数据库

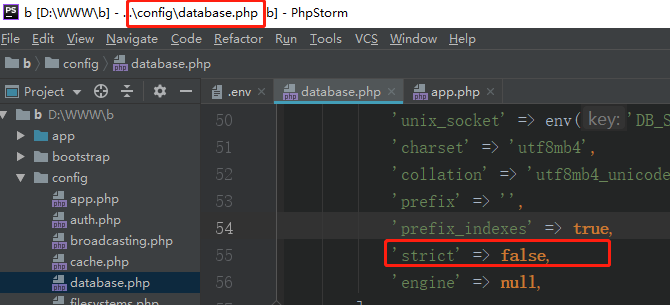
创建数据库 edu\_qz04



修改项目数据库

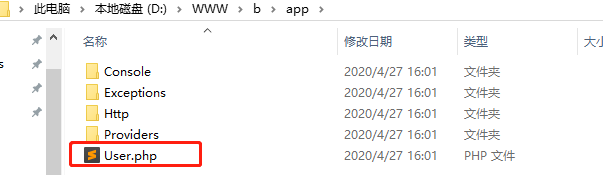


推荐禁用mysql的严格模式 需要修改文件 config/database.php 的strict=false;

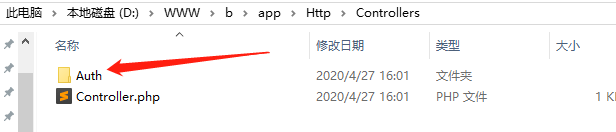


⑤删除系统自带的不需要的文件

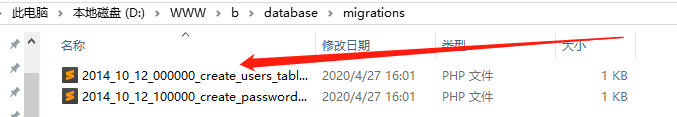
删除 app/user.php模型文件

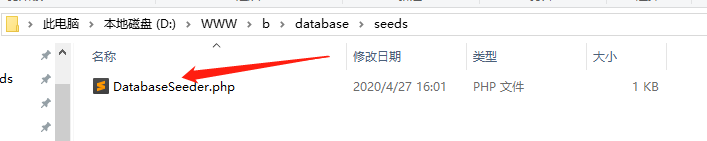


删除控制器里的app/http/controllers/Auth 目录

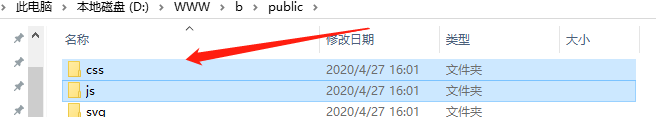


删除database/migrations/目录和database/seeds目录下的全部文件





删除public 下的 css 与js



删除resourse/views下的welcome.blade.php



⑥ （可选）安装debugbar 工具条, 要求PHP版本大于等于7.0

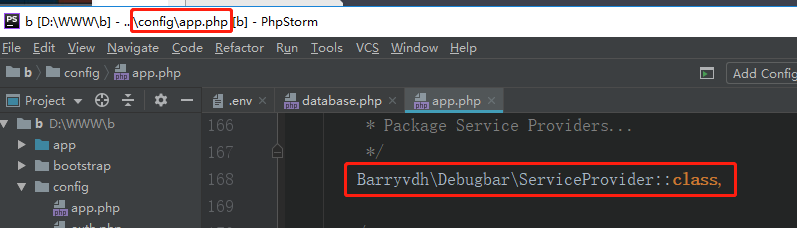
地址：<https://packagist.org/packages/barryvdh/laravel-debugbar>

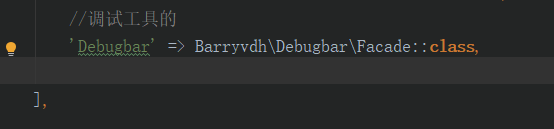
命令：composer require barryvdh/laravel-debugbar --dev

安装好之后添加config/app.php中的providers和aliases 数组的文件

Barryvdh\Debugbar\ServiceProvider::class,

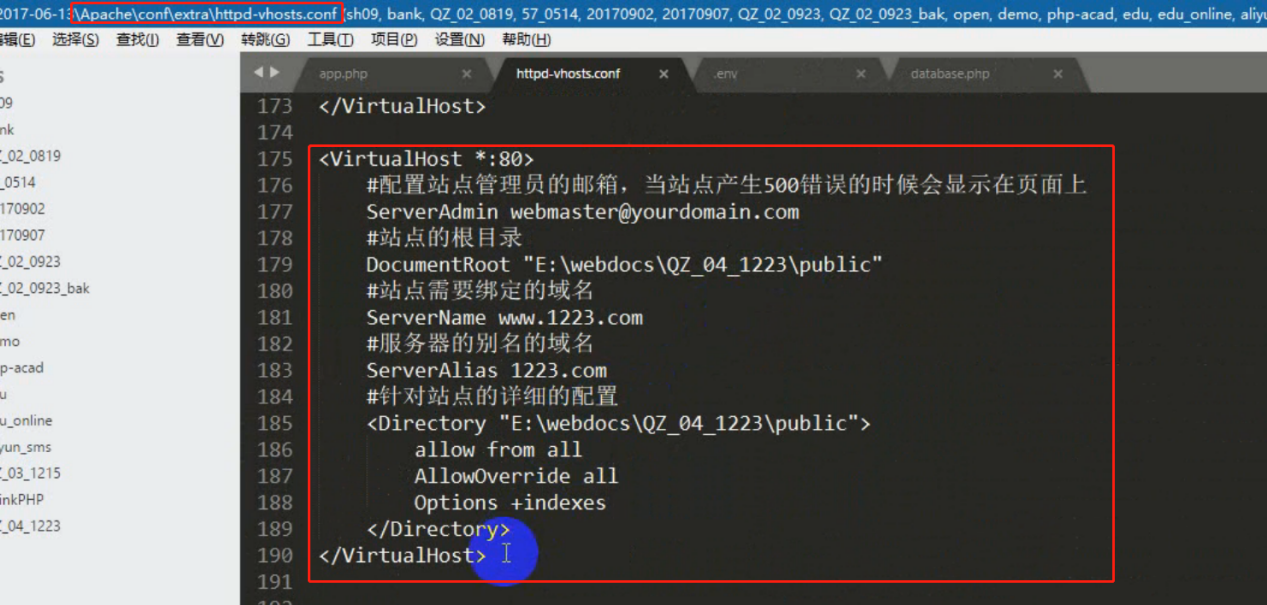
'Debugbar' => Barryvdh\Debugbar\Facade::class,





## 配置虚拟主机

Apache\conf\extra\httpd-vhosts.conf



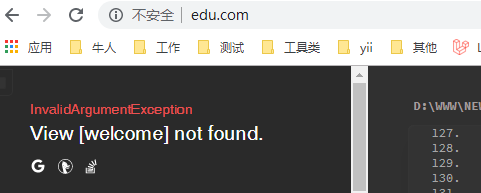
②需要修改apache

③ 修改域名解析操作 ： 修改本地host文件

win +R+ 输入drivers 然后点击etc 文件

④ 测试配置

因为之前已经删掉了welcome.blade.php 所以报错是没有问题的



# 后台基础功能开发

基础功能 ：后台登录、退出、后台首页页面的展示

## 创建管理员数据表

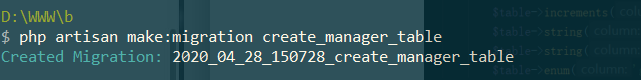
数据字典 通过表格化的形式展示了数据表中应该有的字段信息。

表名: manager

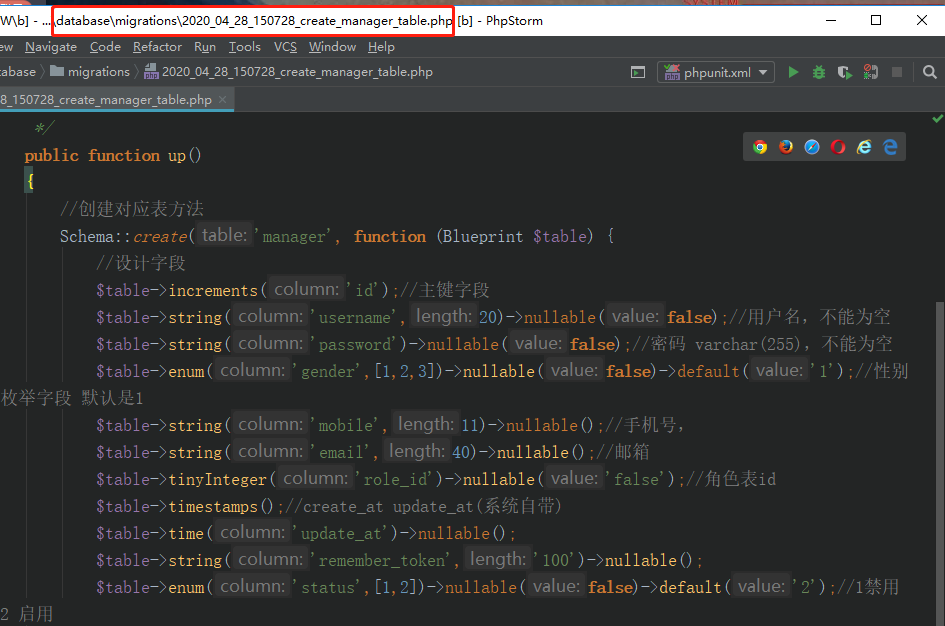
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | manager | | | | | |
| 序号 | 字段 | 类型 | 说明 | 空 | 默认值 | 主键 |
| 1 | id | int | 自增主键 | N | N/A | 是 |
| 2 | username | varchar(20) | 用户名 | N |  |  |
| 3 | password | varchar(255) | 密码 | N |  |  |
| 4 | gender | enum([1,2,3]) | 性别：1男2女3保密 | N | 男 |  |
| 5 | mobile | varchar(11) | 手机号 | Y |  |  |
| 6 | email | varchar(40) | 电子邮箱地址 | Y |  |  |
| 7 | role\_id | tinyint | 角色表中的主键 | Y |  |  |
| 8 | created\_at | datetime | 记录创建的时间 | Y |  |  |
| 9 | updated\_at | datetime | 记录更新的时间 | Y |  |  |
| 10 | remember\_token | varchat(100) | 记录登录功能需要储存的标记 | Y |  |  |
| 11 | status | enum([1,2]) 枚举 | 账号状态 1=禁用2=启用 | N | 启用 |  |

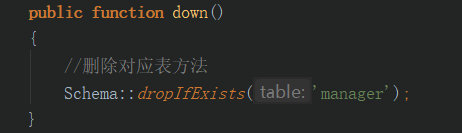
①创建迁移文件

# php artisan make:migration create\_manager\_table



②编写迁移文件的代码

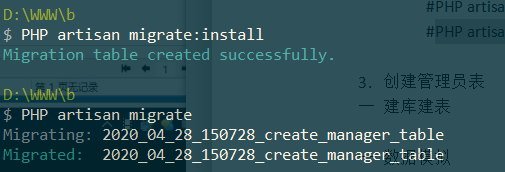




③ 执行迁移文件

#PHP artisan migrate:install //首次使用迁移生成迁移记录表

#PHP artisan migrate

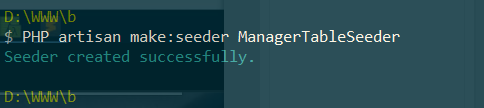




## 模拟管理员表的数据

①创建填充器文件

命令：#PHP artisan make:seeder ManagerTableSeeder



②编写填充器的代码文件

要求：至少模拟出100条数据。[思路：借助循环 + faker代码依赖 ]

注意 ：faker 代码库不需要安装 laravel 创建的时候已经自带了



使用方法参考：<https://packagist.org/packages/barryvdh/laravel-debugbar?query=faker>

使用方法：

a:使用faker\factory::create()创建出生成器；

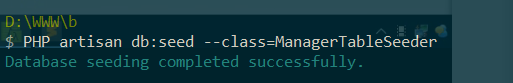
b:通过实例的对象变量去访问属性来生成数据；

提示：**在laravel 项目中 自带一个用于生成密码的函数 bcrypt(密码文明)**



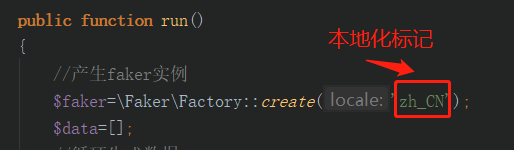
③ 执行填充器文件

#PHP artisan db:seed --class=ManagerTableSeeder



手机号不是大陆的手机号 邮箱不是大陆常用的邮箱 这是因为**生成器没有本地化的操作**

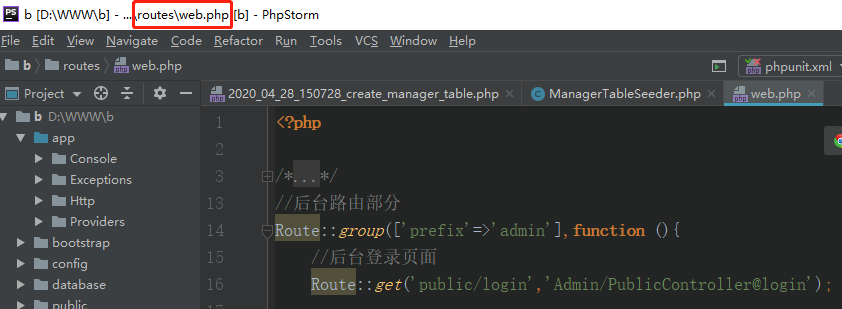
本地化faker 生成器的方法\faker\Factory::create(‘zh\_CN’)：



## 后台登录页面的展示

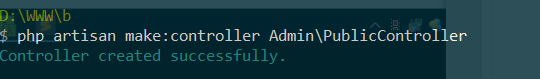
①确定路由

/admin/public/login

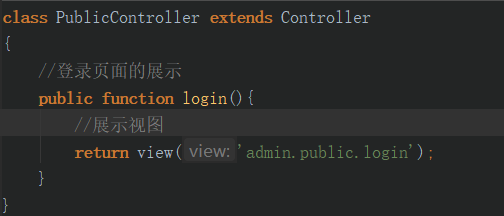


② 创建控制器文件

#php artisan make:controller Admin\PublicController



③ 创建需要的login方法

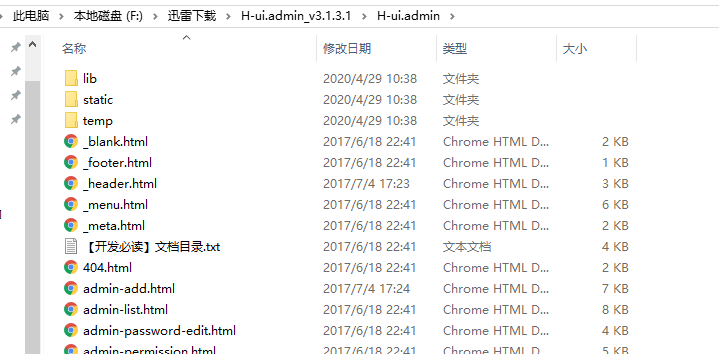


④ 创建需要的视图文件

视图可以选择开源的模板 h-ui.admin

官网：<http://www.h-ui.net/H-ui.admin.shtml>

下载iframe 版本的即可



将h-ui中关于前端的配置文件放入\public\admin中



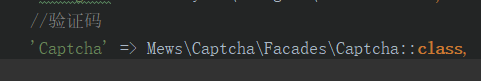
替换js

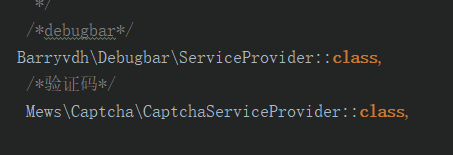
## 实现验证码的替换

验证码的下载地址:<https://packagist.org/packages/mews/captcha>

安装命令： composer require mews/captcha

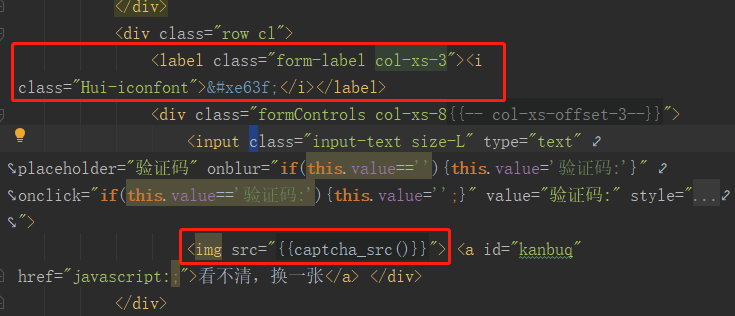
下载安装完成后需要配置config/app.php





需要将验证码写在对应的地方（login.blade.php）

改动:



展示效果:



补充：实现验证码切换操作

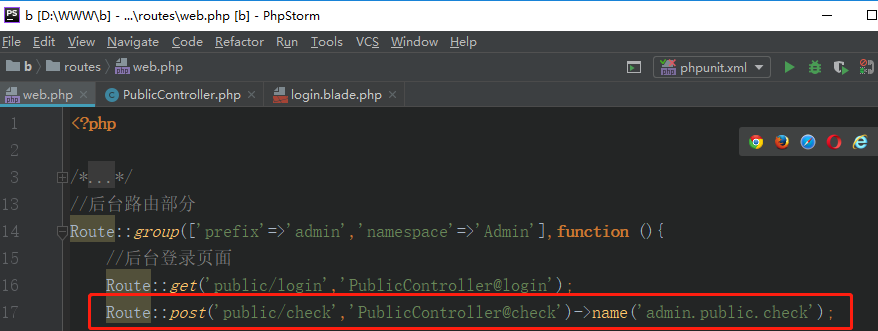
获取验证码链接



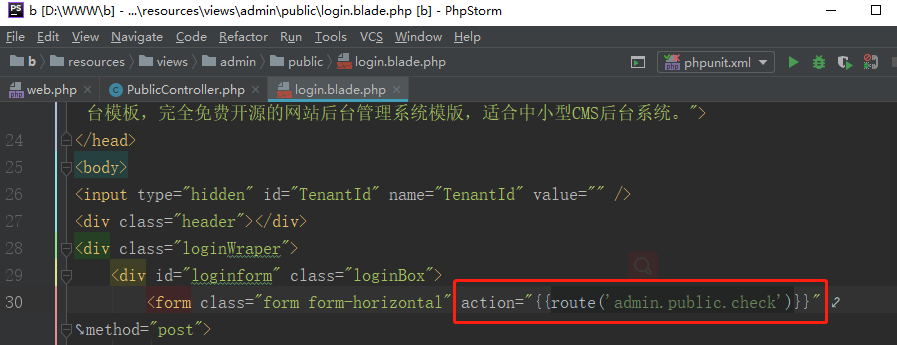
## 管理员登陆的自动验证

1. 提交的路由地址

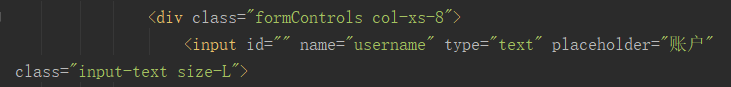
路由：/admin/public/check

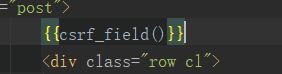


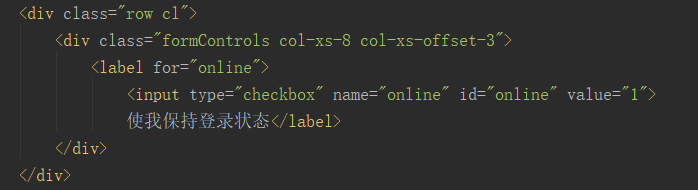
1. 修改登录表单的页面
2. 修改action 的值 换成对应的路由



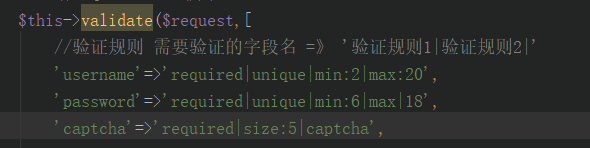
1. 修改表单中的name值



1. 添加csrf验证 carf\_token 代表值 csrf\_field 代表隐藏域 
2. 给“使我保持登录状态” 添加默认值



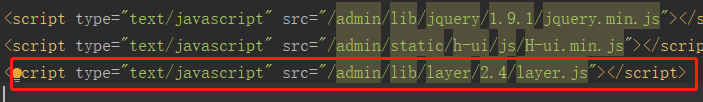
1. 编写public 控制器下的check方法 实现自动验证 （验证数据的有效性）

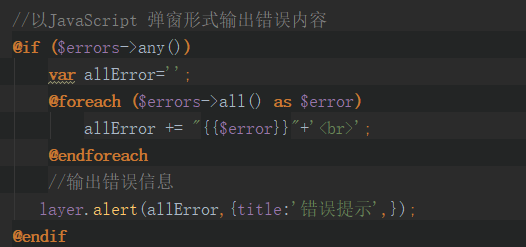


1. 在登录页面上返回错误信息（登录页面失败）

在当前页面如果使用alert会很丑 建议使用JavaScript插件 layer 弹窗插件

在h-ui-admin 中有相关的插件 无需下载 本地引入即可

 语法：layer.alert(‘’,{})



1. **管理登录身份验证【难点】**

在laravel 框架中系统给用户提供了一个比较完备的用户认证机制 使用是通过Auth 门面实现的



Auth 门面提供了记住用户的功能：



**语法：Auth::attempt(‘需要验证的用户信息关联数组’,[‘’bool值])**

访问指定Guard 实例

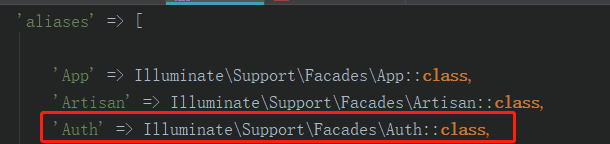


**语法： Auth::guard(指定guard实例)->attempt(需要验证的用户信息关联数组,[bool值]);**

Guard 实例主要是用于区分前后台的。

1. 引入Auth门面

在app.php 文件中已经引入过auth门面



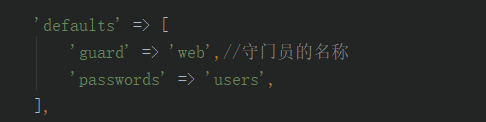
因此 在控制器中使用的时候可以直接use Auth；

1. 使用刚才归纳的语法格式
2. 定义guard(守门员) ，在config/auth.php定义

在该文件中定义存在几个比较重要的键：defaults、guards、providers

其中 各自含义如下:

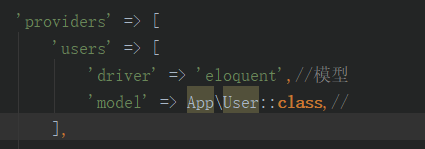
default:用于定义默认的guard实例信息

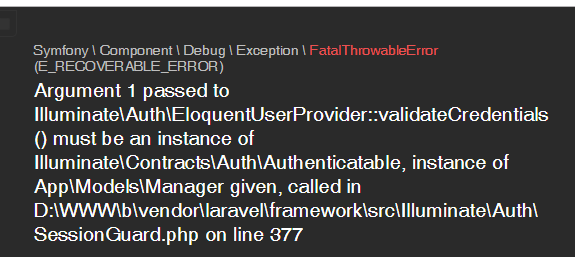


guards： 指定了每个guards 的提供者信息和信息存储的驱动方式



providers：定义每个guard实例的验证数据来源 一般为eloquent 模型 需要去指定模型的 空间位置

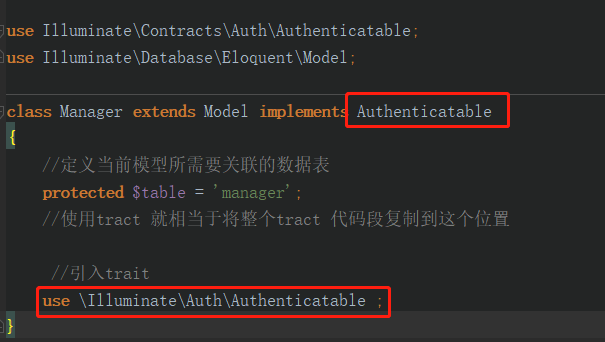


③ 在验证对的情况下会出现报错 

意思：前面的必须是后面的实例

**补充：trait**

**是PHP5.4之后的新特性 主要作用是为了提高代码复用 ，语法与class 类似，区别class 是可以被实例化的 但是trait不可以 Trait 可以被使用 使用方法 use trait 方法名**



根据错误信息判断跳转方向



登录失败的时候可以加上错误信息 传给登录页面

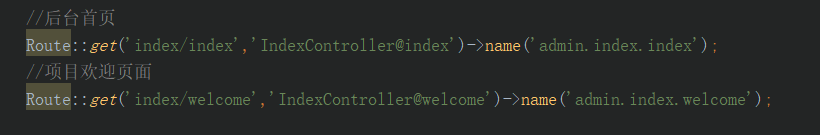
格式：return redirect(‘跳转页面’)->witherrors(‘loginError’=>”错误信息”);

1. **后台首页**

①约定首页路由

/admin/index/index

/admin/index/welcome



1. 创建需要的控制器文件

#PHP artisan make:controller Admin/IndexController ;

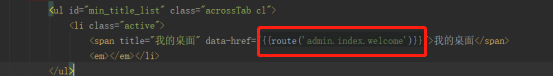


1. 创建需要的两个视图文件 (index.blade.php welcome.blade.php)



记得修改js与css 文件路径

同时需要修改 welcome 页面在index页面的引入笛子 否则之框架页面会引入报错



1. 提升

动态获取用户信息 再页面上展示出来

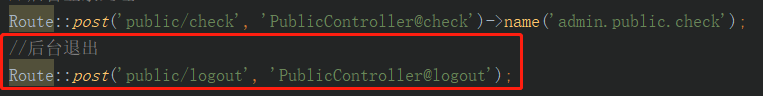
修改视图文件 index.blade.php 页面

获取用户认证

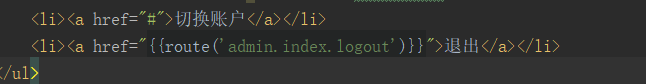


1. 实现用户退出操作

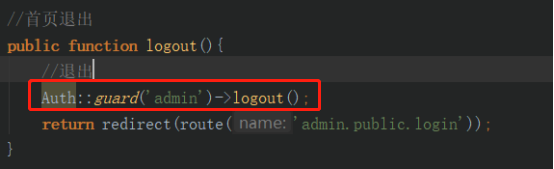
web.php 页面 添加退出路由



index.blade.php 更改路由



PublicController.php 增加logout 方法



1. **路由保护**

含义： 只有通过认证（符合某个条件） 的用户 才能访问指定的路由，该方式称之为路由保护。

路由/地址 -> base控制器（让所有需要判断的控制器继承base控制器）->目标访问控制器

laravel 项目中

路由/地址 -》中间件 ->目标访问控制器

中间件不是控制器文文件 但是也是一个类。中间件在laravel 框架中有2大类:

系统提供的 用户自己定义的 。

使用方法

**路由 -》middleware(中间件名称);**

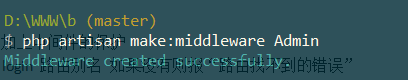
在路由中编写路由保护的代码

注意事项 ：

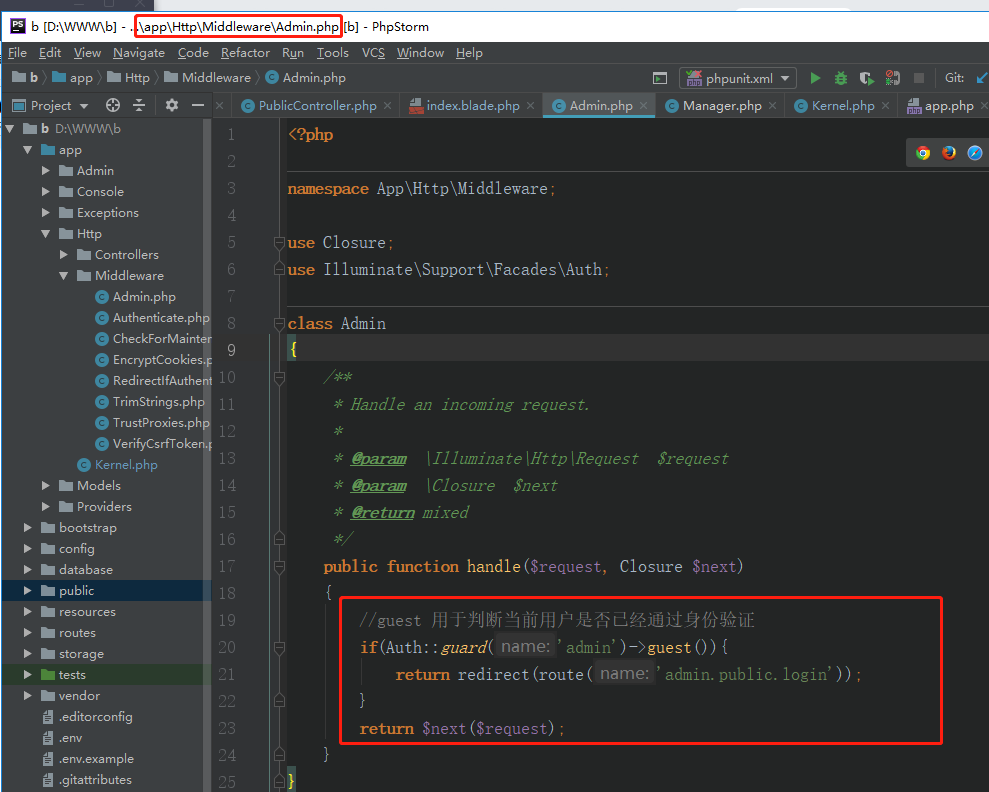
a 需要判断是否登录页面全部都加上中间件的保护

b 在跳转的时候 系统默认使用 login 路由别名 如果没有则报“路由找不到的错误”

命令行添加中间件：

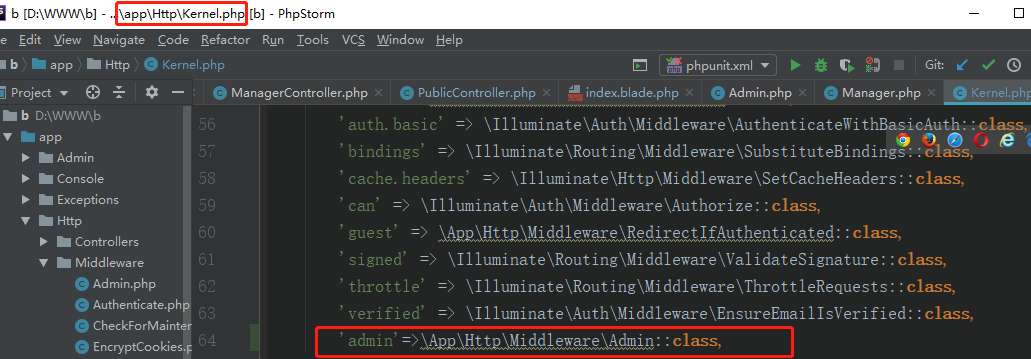


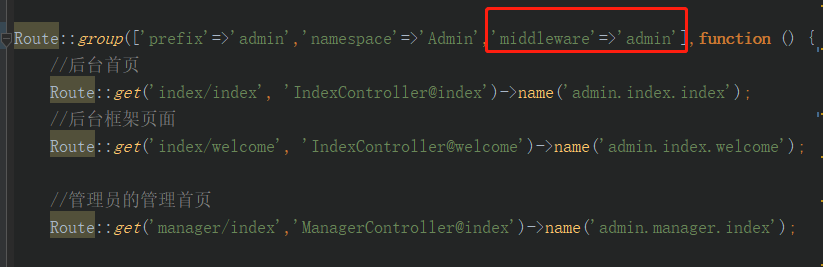
具体代码实现 ：





在app\http\kerdnel注册中间件





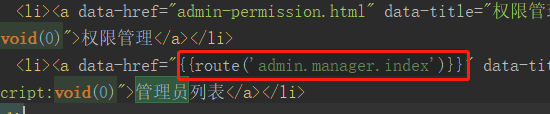
# 管理员管理

1、实现管理员的列表展示

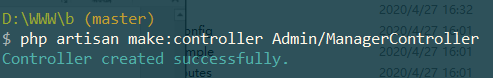
① 定义需要的路由

约定 /admin/manager/index get

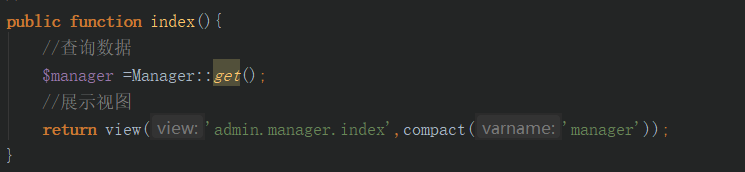
同时修改后台首页的坐车导航菜单的地址



② 生成对应的控制器文件



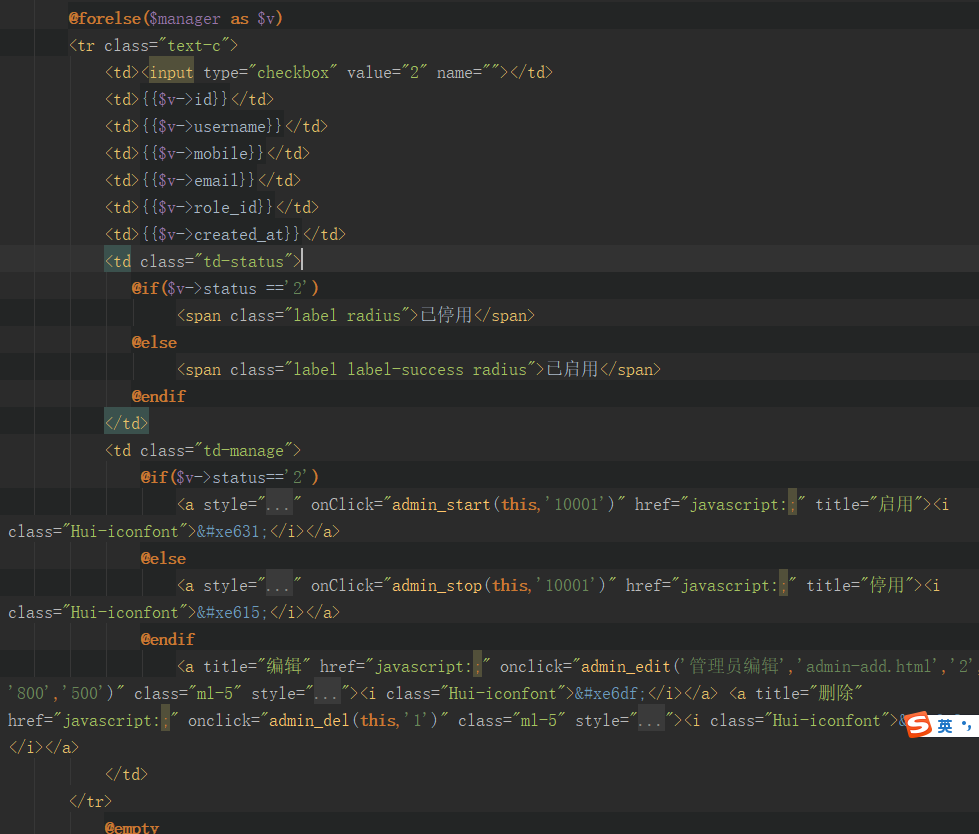
③ 创建需要的index 方法 （查询列表的数据、展示视图）



将admin\_list页面复制然后更改js 与css

/admin/lib/

⑤将数据输出 （循环+判断）



2、使用datatables插件实现无刷新分页

Datatables插件是一款基于jQuery框架进行开发的无刷新分页插件， 其中除了分页还有排序、搜索、当前表格直接编辑等功能

网址：<http://datatables.club/> 中文网 <https://datatables.net/> 原网

该分页插件有两种形式 ：客户端分页方式、服务端分页方式(limit)

客户端分页：优点 ：只需要请求一次数据库就可以 ：缺点 如果数据量很大 页面加载不迅速。

服务端分页 优点： 页面加载速度不会根据数据的多少而增加 数据量少的时候服务器的压力相对较大 缺点 获取所有数据需要请求多次数据库

2.1、使用客户端分页方式实现分页

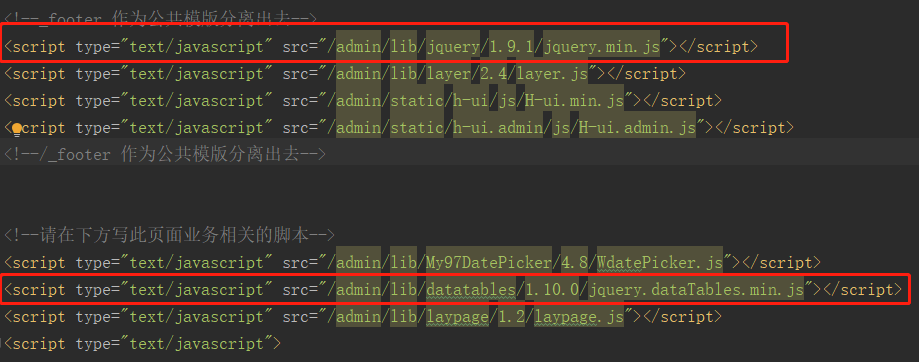
使用步骤：

① 先确保引入jQuery 之后 再去引入 datatables 的JavaScript 文件；

② 需要初始化datatables插件

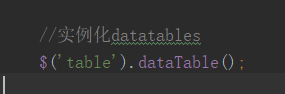
③ 【可选】databtables 支持一些扩展的配置

①引入对应的文件



② 初始化datatables 插件

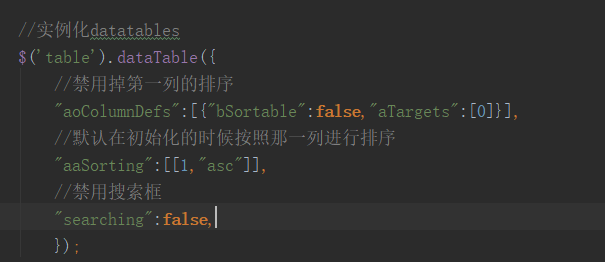
**语法：$(选择器).dataTable();**





可选操作： 针对当前页面做一些操作

1. 禁用掉第一列（复选框）排序
2. 更改默认的初始化排序操作



提示： 后期需要使用这些配置代码 不建议记忆 直接搜索即可

# RBAC 权限管理

## 介绍

RBAC： role-based access Control 基于角色的访问控制。 是一个比较完善的权限访问控制机制 目前有两套权限控制方案: 基于用户的权限控制、基于角色的权限控制

rbac 在后期权限维护上是很方便的

rbac 权限方式在项目开发的初始阶段就已经有了一个成型标准

RBAC方式体现

三表RBAC形式 五表 RBAC形式 原理是一致的

出现三表（用户表、角色表、权限表）和五表（用户表、用户与角色的关系表、角色表、角色与权限的关系表、权限表） 的原因就是设计规范要求(三范式)。

三表 ：

优点：表少，维护起来数据方便，理解起来也方便 ；缺点不遵循范式要求

五表 ：

优点：遵循范式要求 ；

## 数据表设计

三表：用户表 角色表 权限表

后台角色表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | role | | | | | |
| 序号 | 字段 | 类型 | 说明 | 空 | 默认值 | 主键 |
| 1 | id | int | 自增主键 | N | N/A | 是 |
| 2 | role\_name | var\_chat(20) | 角色名称，如超级管理员 | N |  |  |
| 3 | auth\_ids | text | 权限表主键id集合，如1,2,3 | Y |  |  |
| 4 | auth\_ac | text | 权限对应的控制器方法集合如adminController@index,indexController@index | Y |  |  |

后台权限表

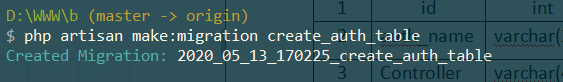
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | auth | | | | | |
| 序号 | 字段 | 类型 | 说明 | 空 | 默认值 | 主键 |
| 1 | id | int | 自增主键 | N | N/A | 是 |
| 2 | auth\_name | varchar(20) | 角色名称，如超级管理员 | N |  |  |
| 3 | Controller | varchar(40) | 权限表主键id集合，如1,2,3 | N |  |  |
| 4 | action | varchar(30) | 权限对应的控制器方法集合 | N |  |  |
| 5 | pid | tinyint | 父级权限id(2级权限),0顶级权限 | N | 0 |  |
| 6 | is\_nav | enum([1,2]) | 是否作为菜单显示,1=是，2=否 | N | 是 |  |

①创建需要 迁移的文件

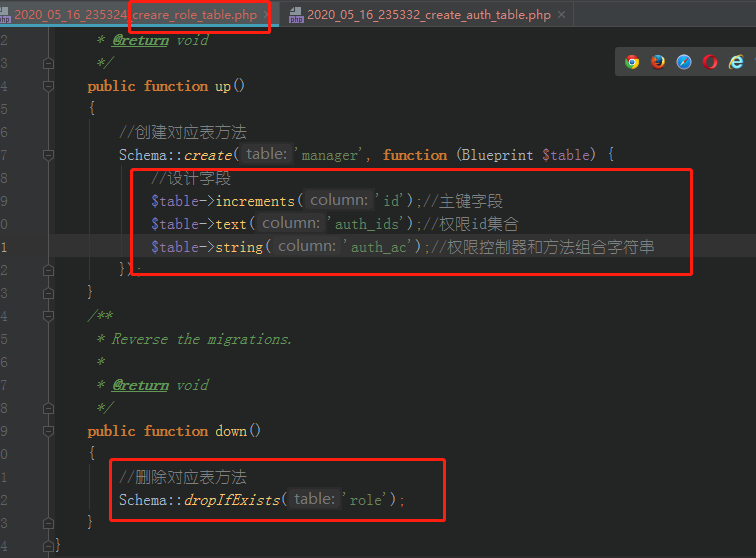
#php artisan make:migration creare\_role\_table

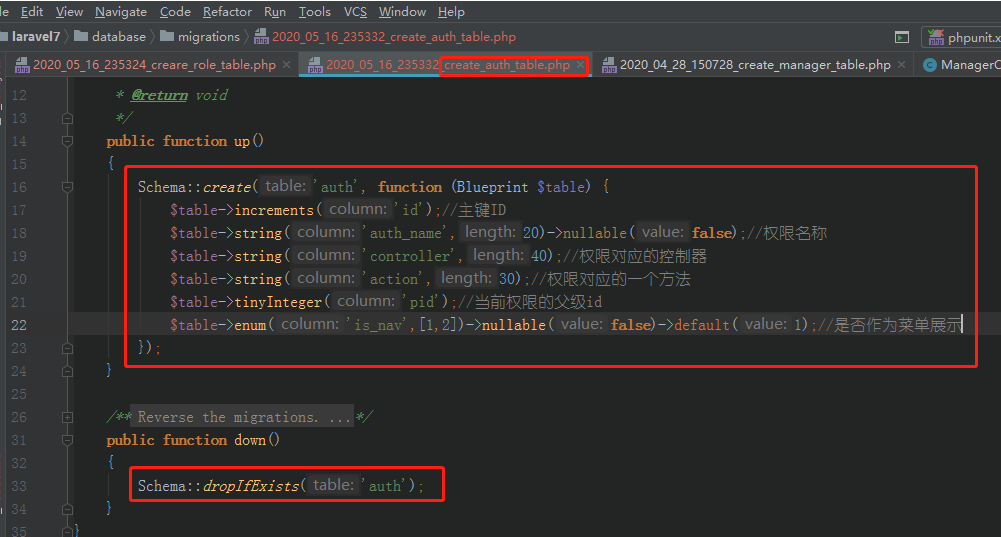


#php artisan make:migration create\_auth\_table



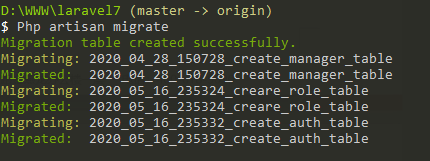
Create\_r





准备迁移文件生成对应的数据表

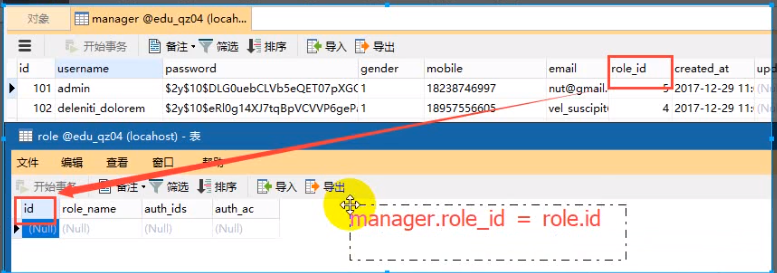
# PHP artisan migrate



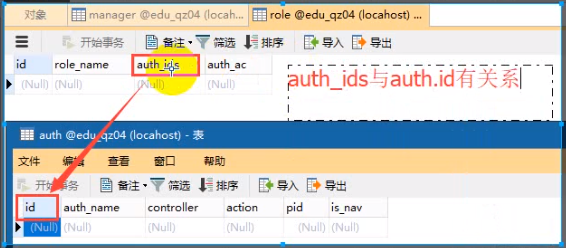
## 理清三表之间的关系【重点】

三表： 用户表、角色表、权限表

用户表-角色表



角色表-权限表



注意：用户表与权限表之间没有直接关系

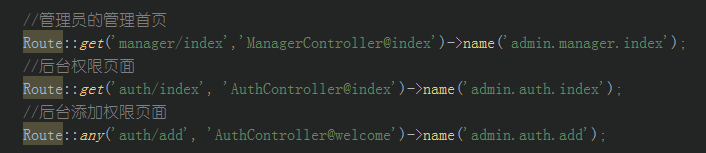
## 权限管理操作

### 4.1 展示相关视图

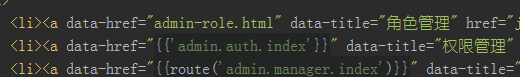
①确定需要的路由

添加页面： admin/auth/add any 类型

列表页面： admin/auth/index get 类型



同时修改左侧菜单的链接地址，将对应的文字连接到列表页面

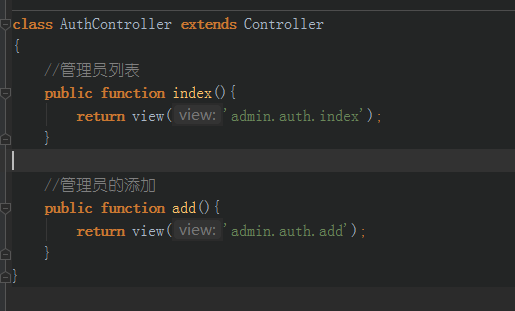


②创建对应的控制文件

# Php artisan make:controller Admin/AuthController



③创建2个方法 Auth控制器下的index方法 和add方法



④创建需要的视图文件

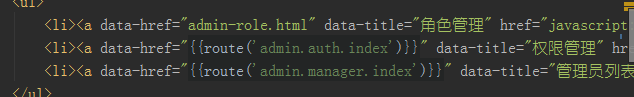
需要：admin.auth.index（admin-permission.html）和 admin.auth.add (不存在)文件

修改其他的添加模板得到添加权限的模板

D:\WWW\laravel7\resources\views\admin\auth\index.blade.php

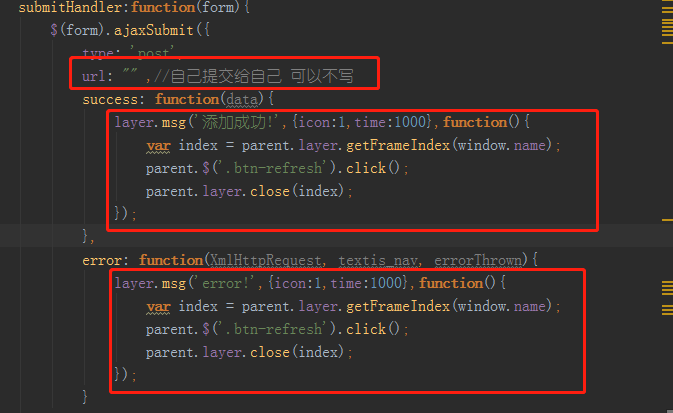


D:\WWW\laravel7\resources\views\admin\index\index.blade.php

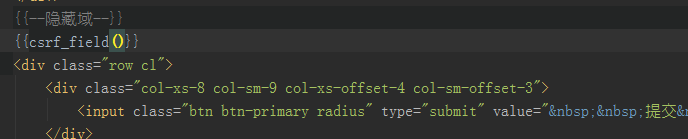


### 4.2、实现权限的添加功能

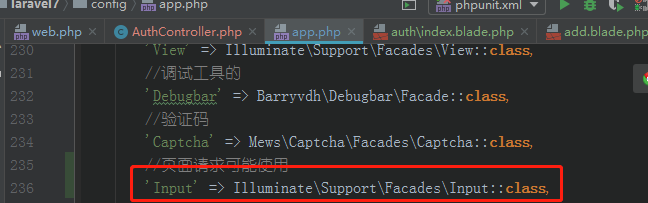
①修改ajax提交部分的代码



②添加csrf\_field（）隐藏域



③根据需求完善ajax的提交的方法



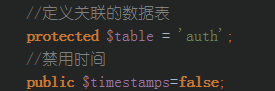
在页面 use Input;即可使用

1. 创建需要的auth模型文件

#php artisan make:model Admin/Auth



同时定义模型的基本属性



### 4.3、实现权限的列表功能