# 好客租房\_08

# 今日目标5

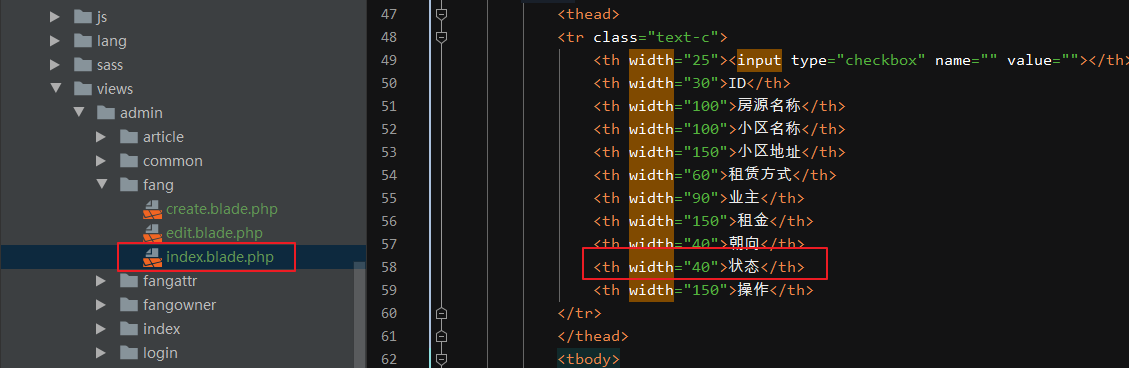
* 能够实现房源信息查看
* 能够实现房源列表搜索功能
* 能够接入echarts可视化图表显示
* 了解ORM模型事件定义
* 能够添加任务到队列中

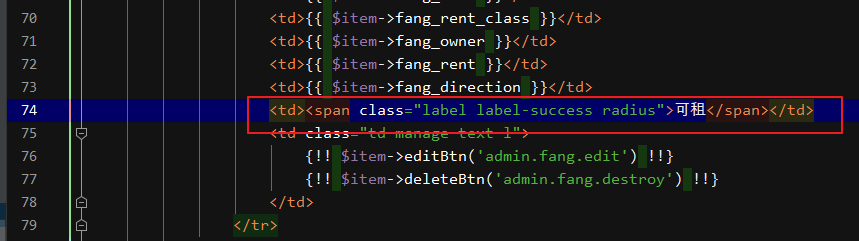
# 一、房源管理

## 1.1、已租和未租列表

在房源列表中添加一个字段来显示对应的房源信息状态，并且给此状态还可以添加点击事件，方便日后修改状态操作。

在模板页面中添加对应html字段

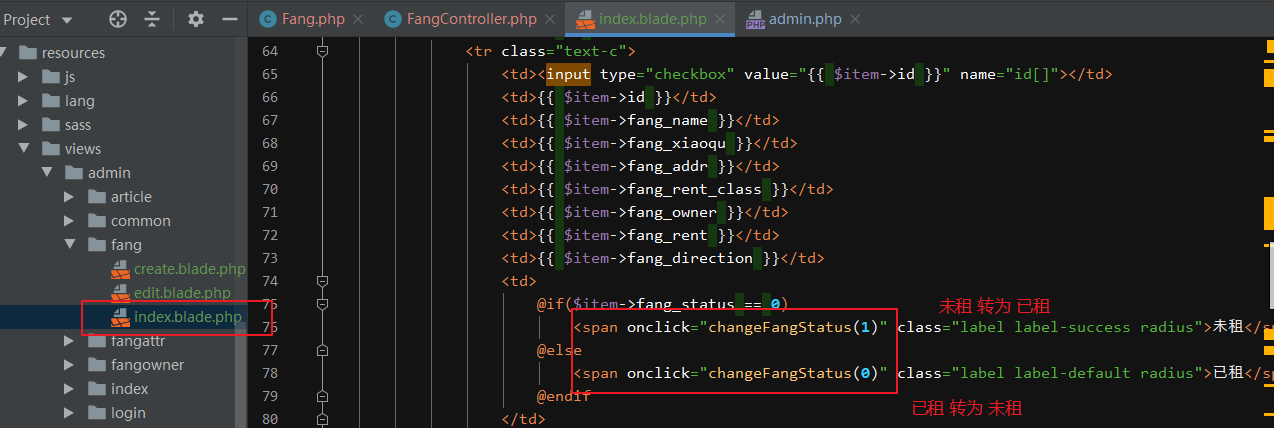




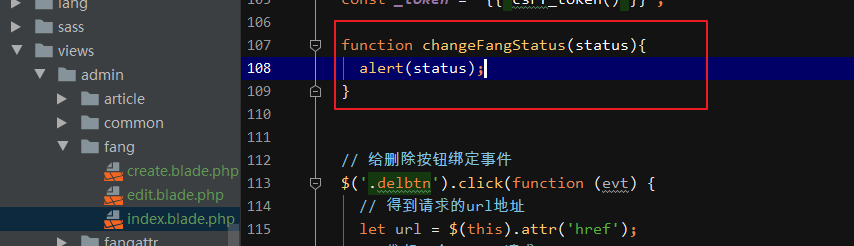
静态页面展示的效果



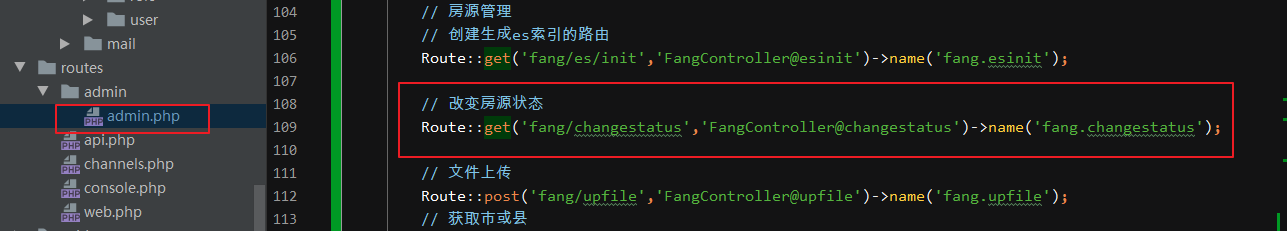
给状态列表中的按钮添加点击事件



js中实现对应的点击事件



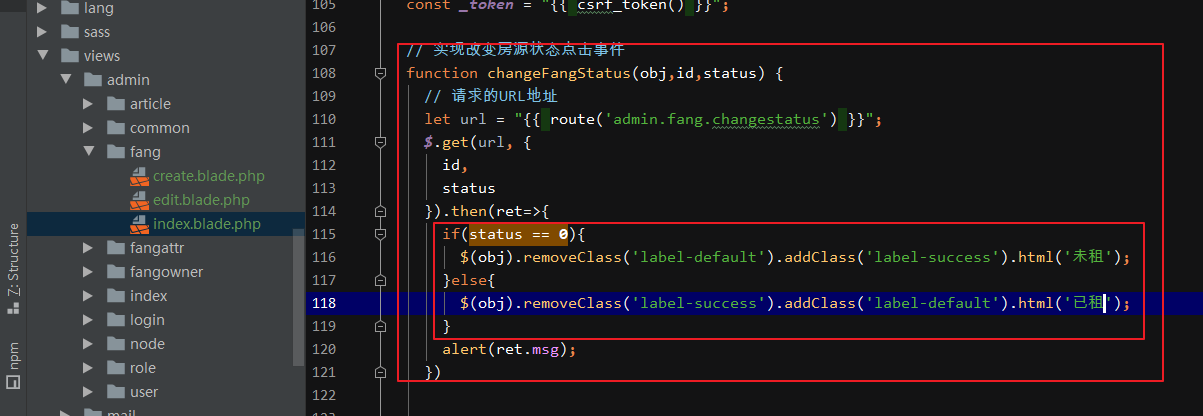
定义实现修改房源状态路由



在控制器中实现对应的改变状态方法



js实现点击修改状态和前端效果



效果



## 1.2、echarts可视显示已租未租

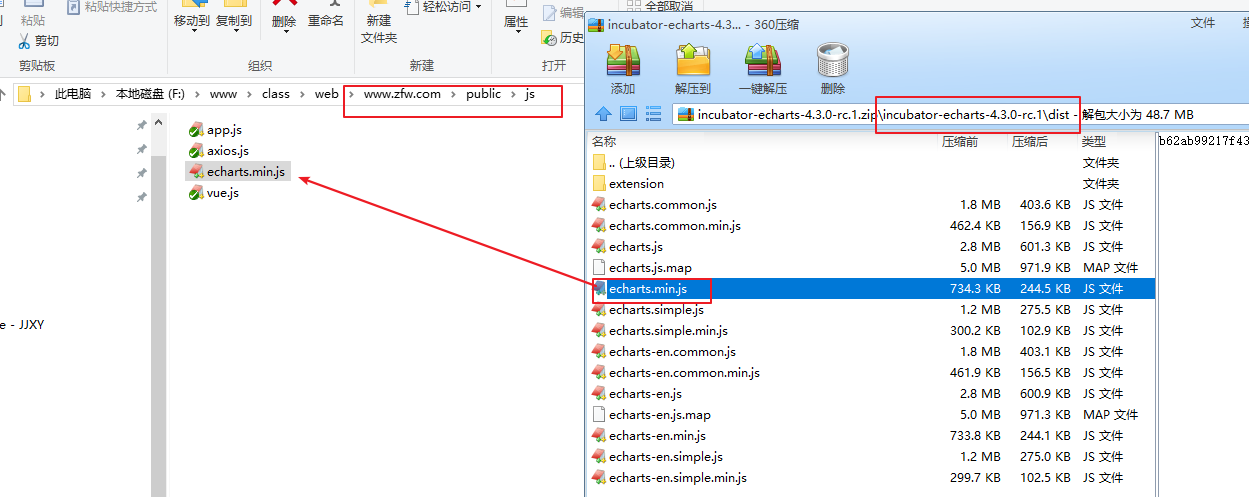
可视化的显示在登录成功后进入的欢迎页面中显示，使用echarts来显示图表。

官网：<https://echarts.baidu.com/>

下载对应的echarts源码包：<https://www.echartsjs.com/zh/download.html>



解压复制对应的js类库文件到项目中去。



简单上手：<https://www.echartsjs.com/zh/tutorial.html>

* 引入 Echarts

<!DOCTYPE html>

<html>

<head>

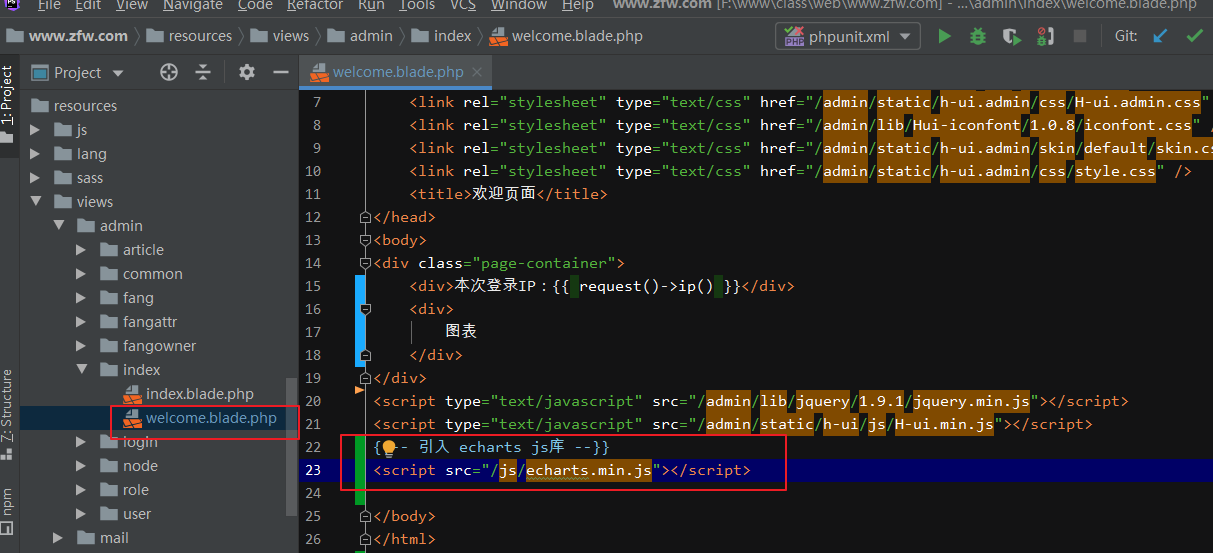
<meta charset="utf-8">

<!-- 引入 ECharts 文件 -->

<script src="echarts.min.js"></script>

</head>

</html>



* 在html中定义显示图表的容器

<body>

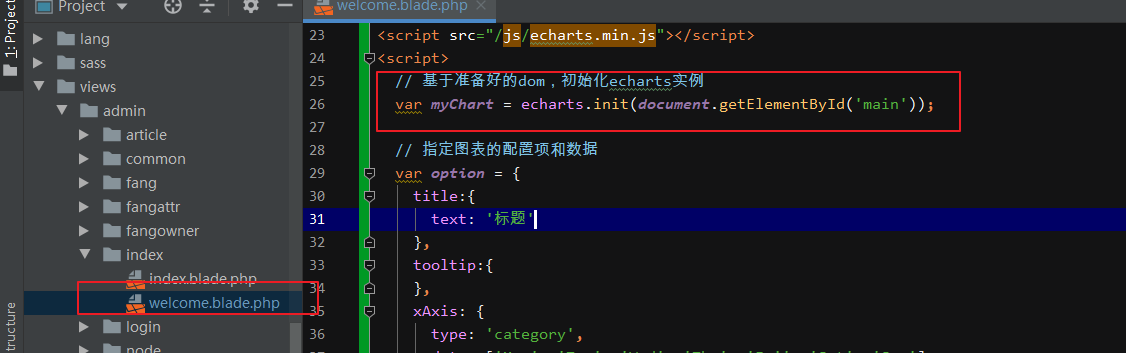
<!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->

<div **id="main"** style="width: 600px;height:400px;"></div>

</body>

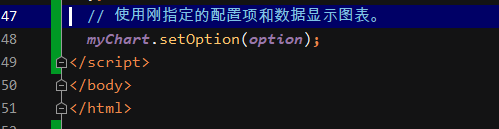


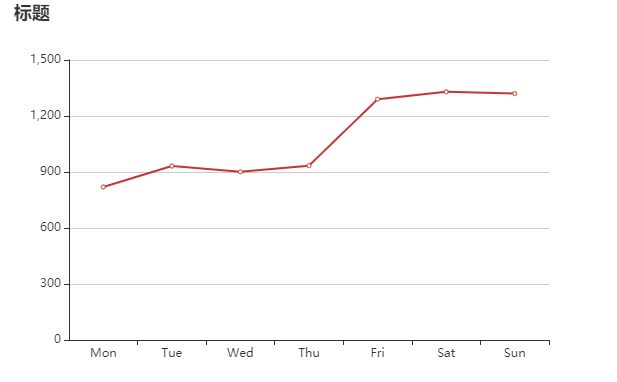
* 初始化数据显示



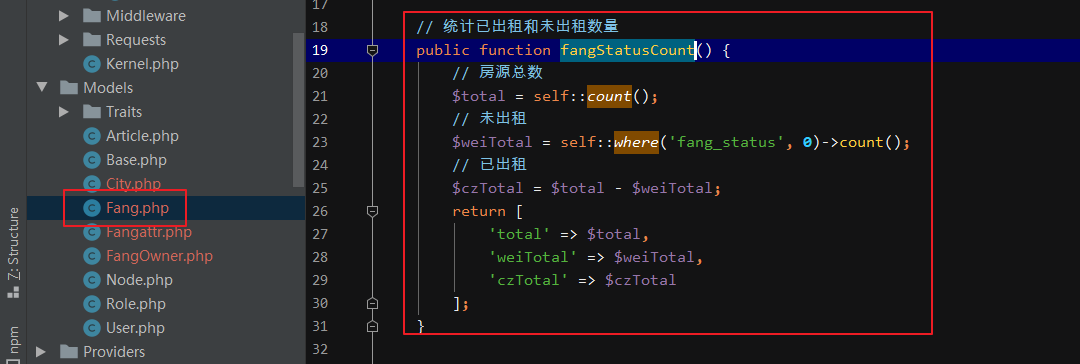
看官方示例，复制其代码



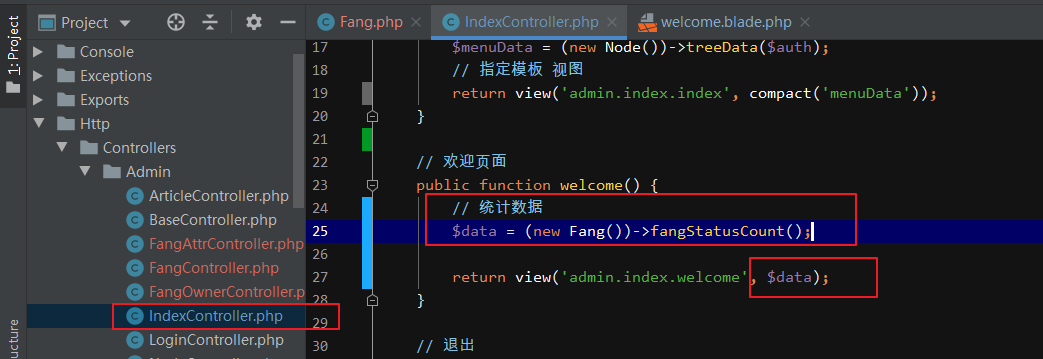




在模型中定义房源状态统计信息

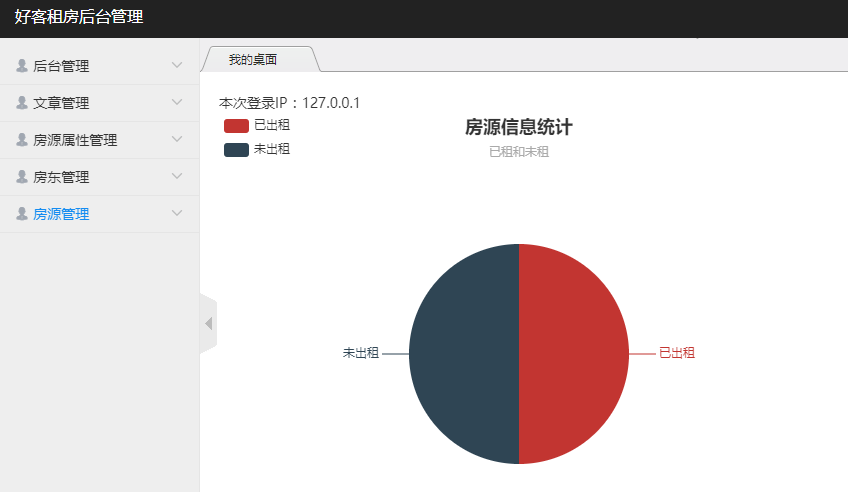


在欢迎页面中的控制器中调用模型得到统计数据



在欢迎页面前端模板中调用统计数据

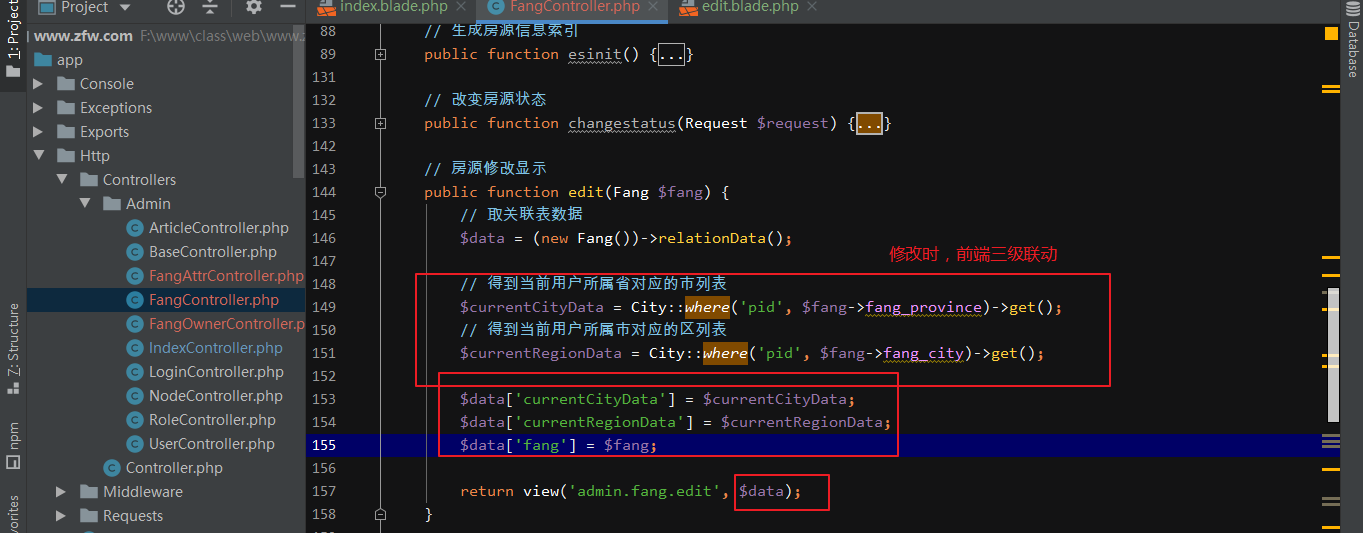




## 1.3、修改房源

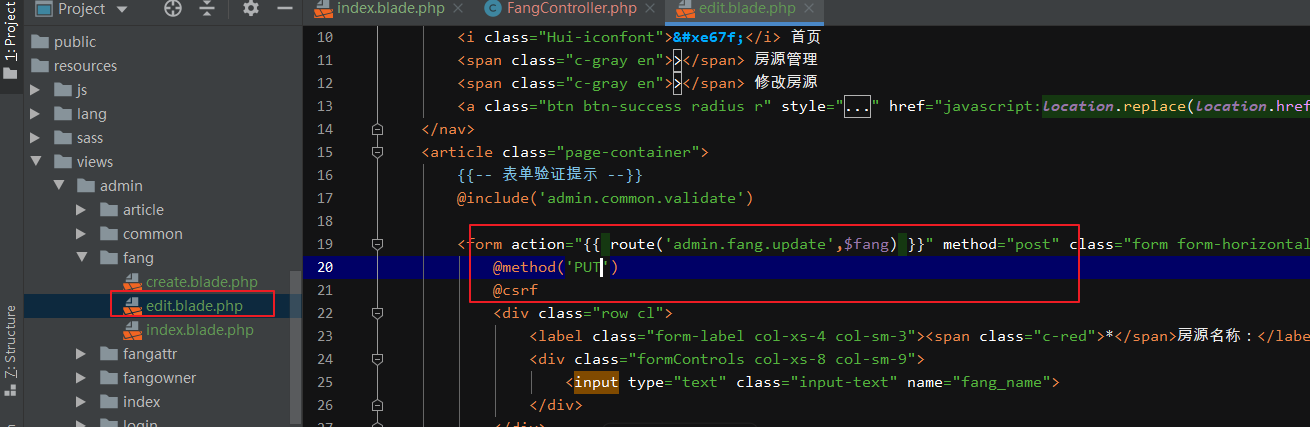
难点：三级联动的修改显示，表单选项多，容易出错。

在房源操作控制器中显示对应的数据

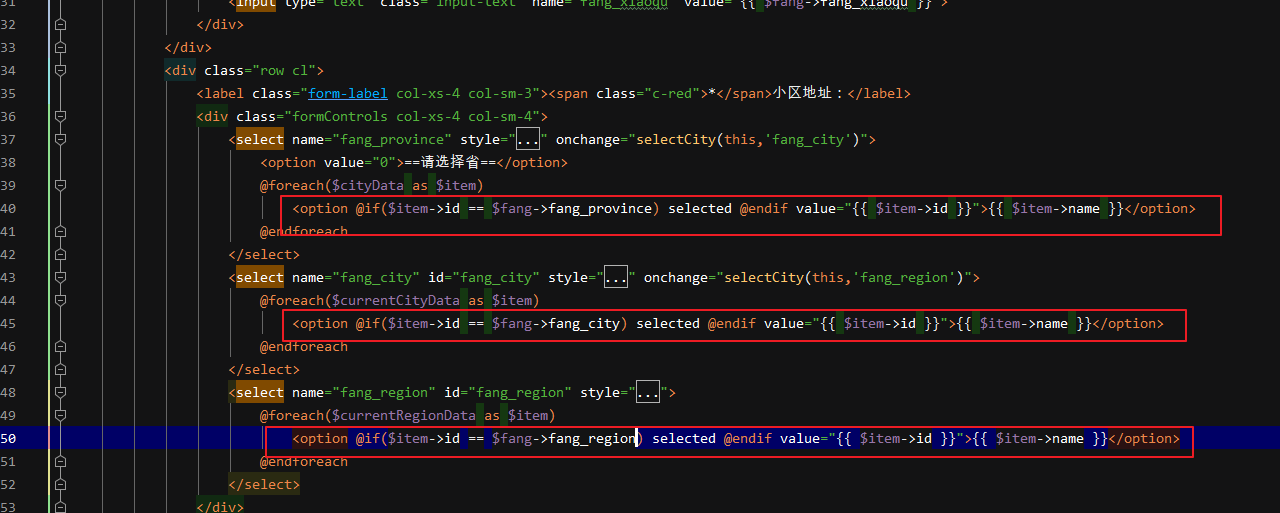


在模板中显示对应的数据

模板中修改提交地址和提交方式put

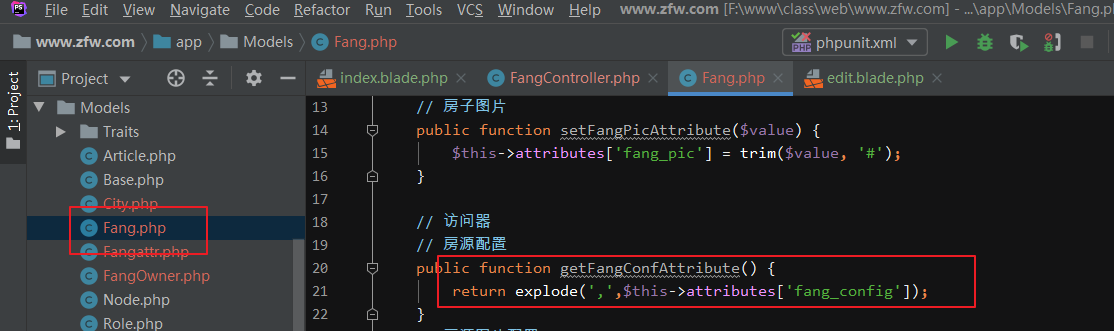


在表单中显示三级联动省市区数据



配套设施，使用了模型中的访问器

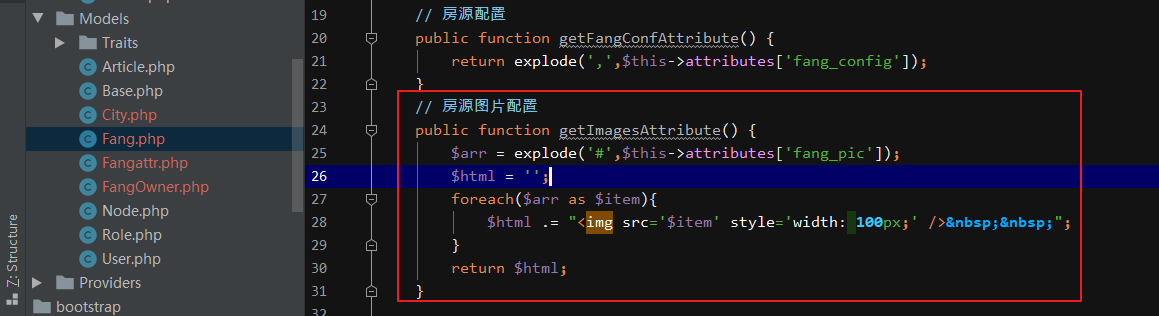
定义访问器



在模板中可以使用对应的定义好的访问来进行条件判断



房源图片的显示，使用了访问器

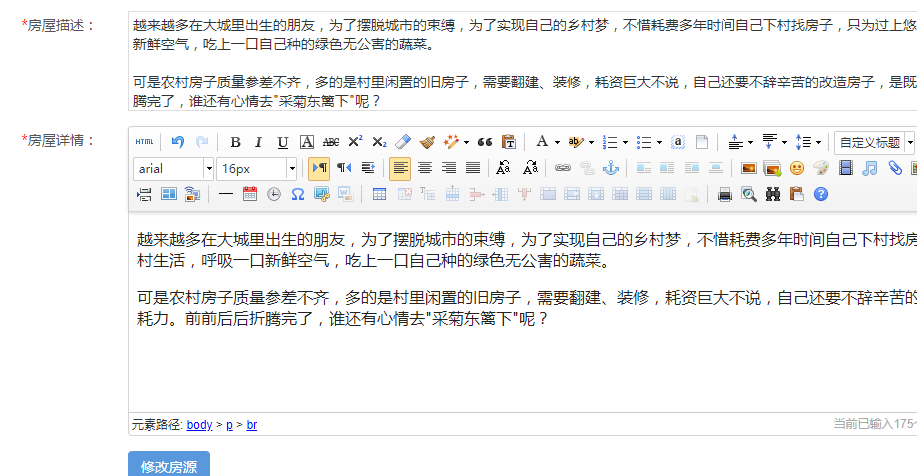


在模板中显示调用



效果

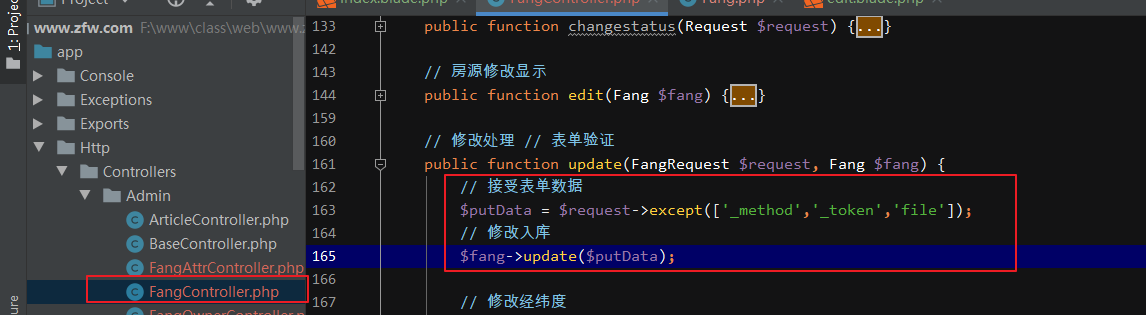




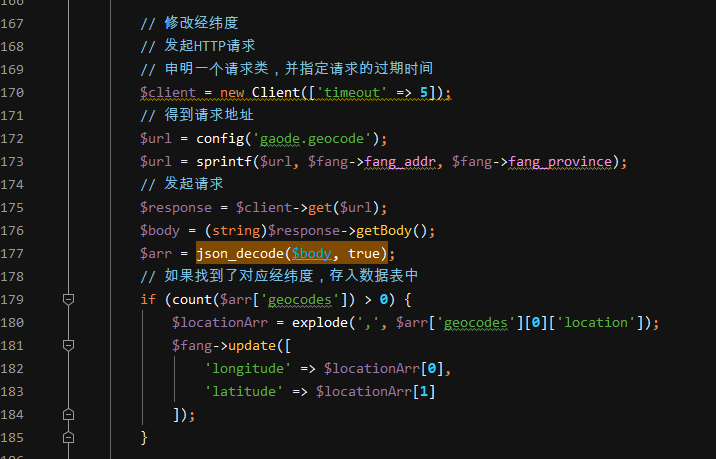
修改入库

* 接受表单数据
* 地址转为经纬度
* 修改es中的数据

接受表单数据入库



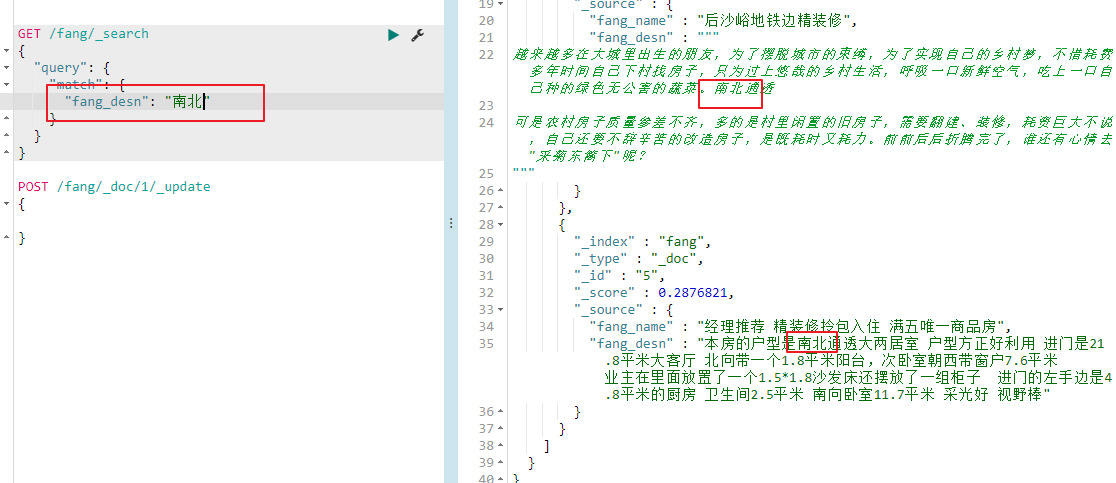
地址转换



更新ES数据源



效果

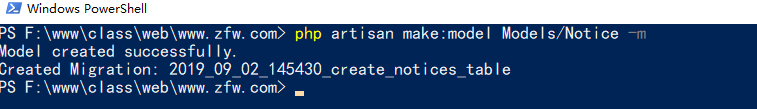


# 二、预约管理

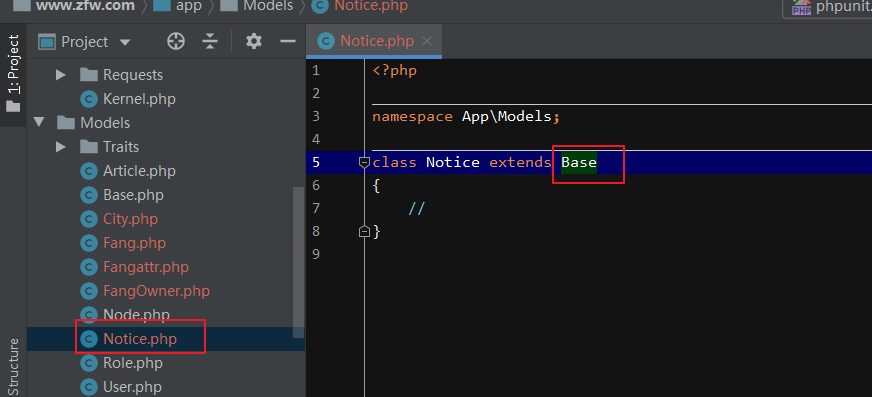
## 2.1、预约表设计

假设 租客表已经存在，前提来设计预约表

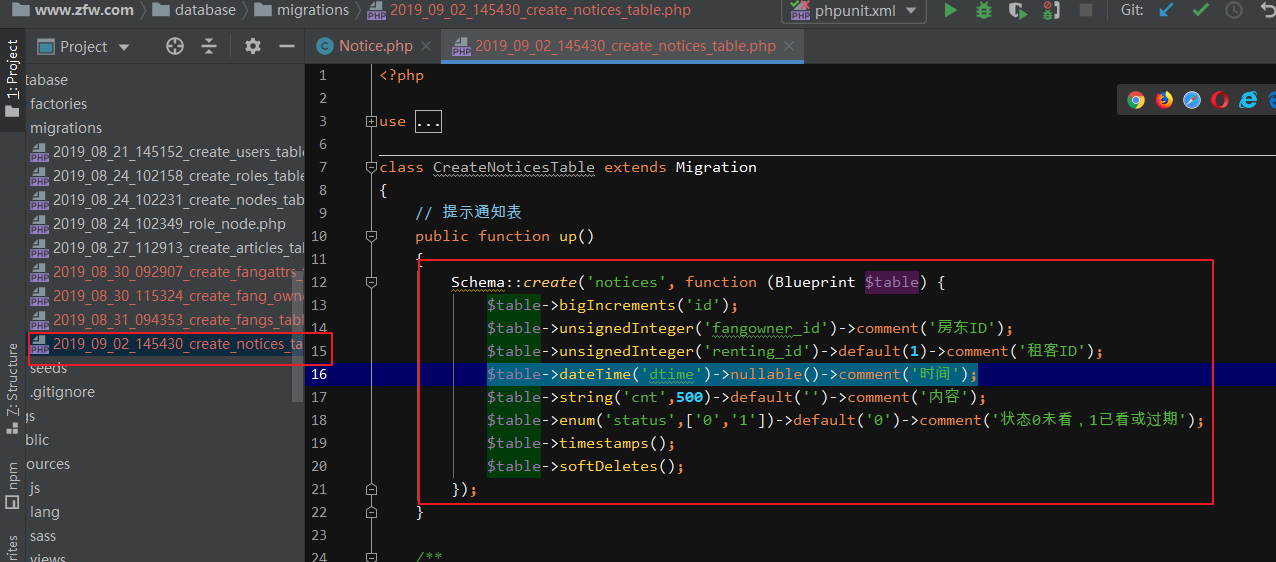
php artisan make:model Models/Notice -m



更换模型类的继承



迁移文件



Schema::create('notices', function (Blueprint $table) {

$table->bigIncrements('id');

$table->unsignedInteger('fangowner\_id')->comment('房东ID');

$table->unsignedInteger('renting\_id')->default(1)->comment('租客ID');

$table->dateTime('dtime')->nullable()->comment('时间');

$table->string('cnt',500)->default('')->comment('内容');

$table->enum('status',['0','1'])->default('0')->comment('状态0未看，1已看或过期');

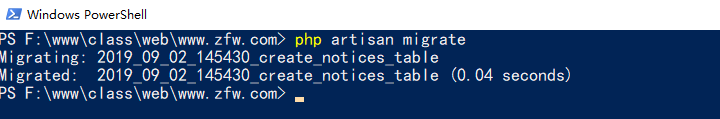
$table->timestamps();

$table->softDeletes();

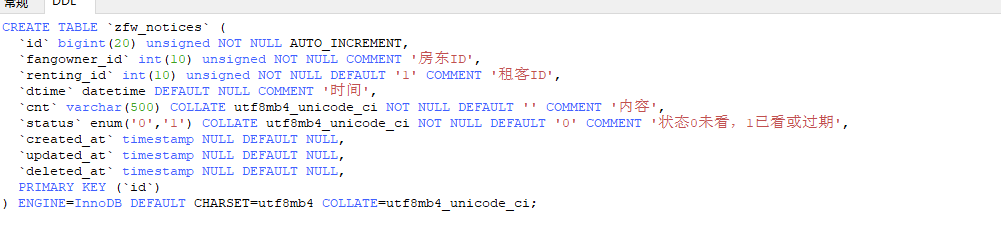
});

执行迁移

php artisan migrate



数据库中就有此表

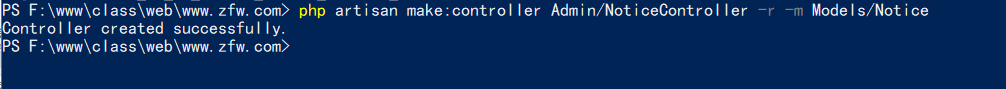


## 2.2、预约列表

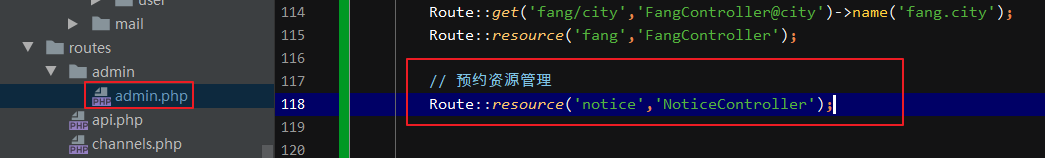
使用paginate进行分页显示，同时最好有搜索功能。

定义资源控制器和资源路由

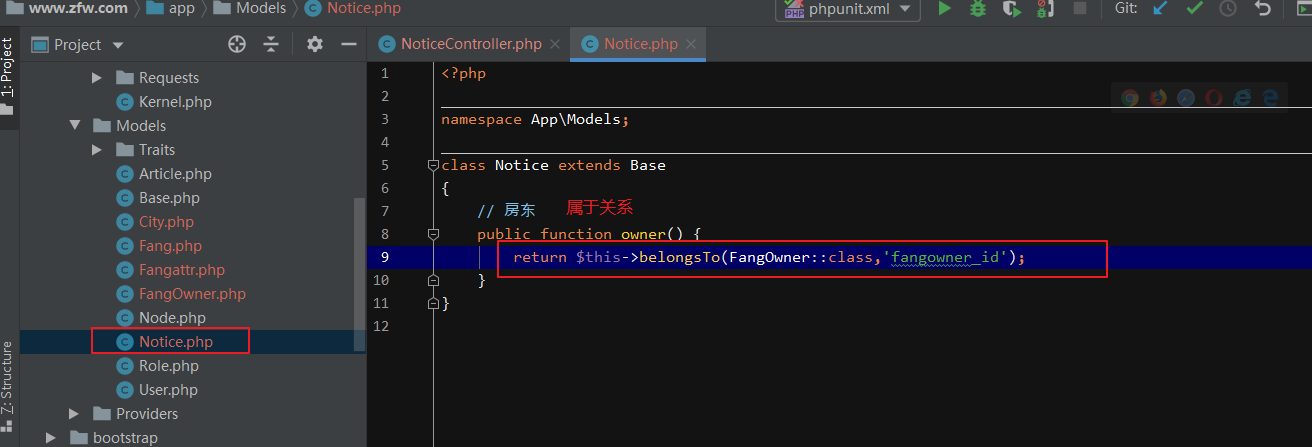
创建资源控制器



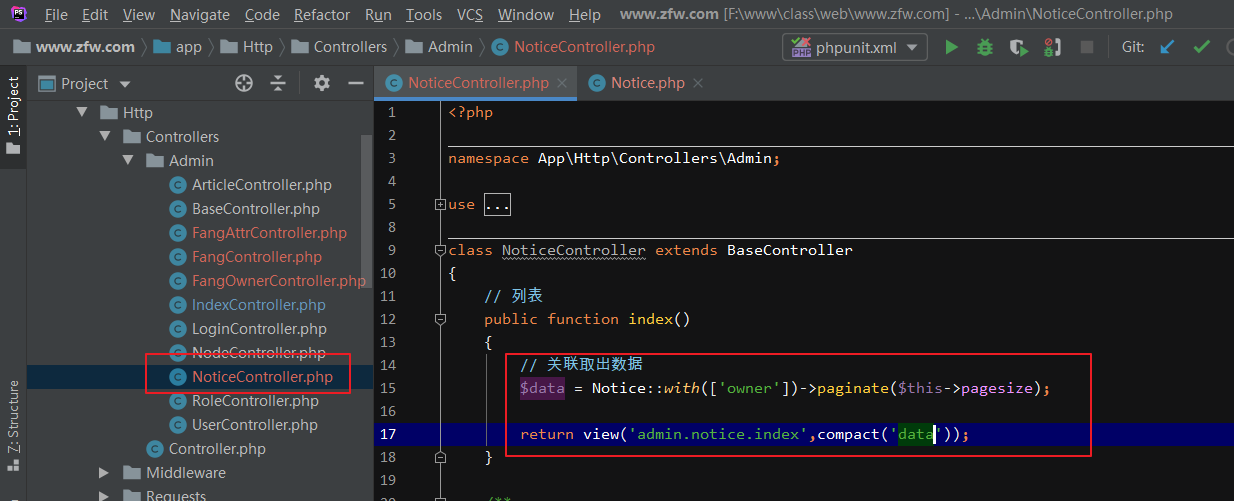
定义资源路由



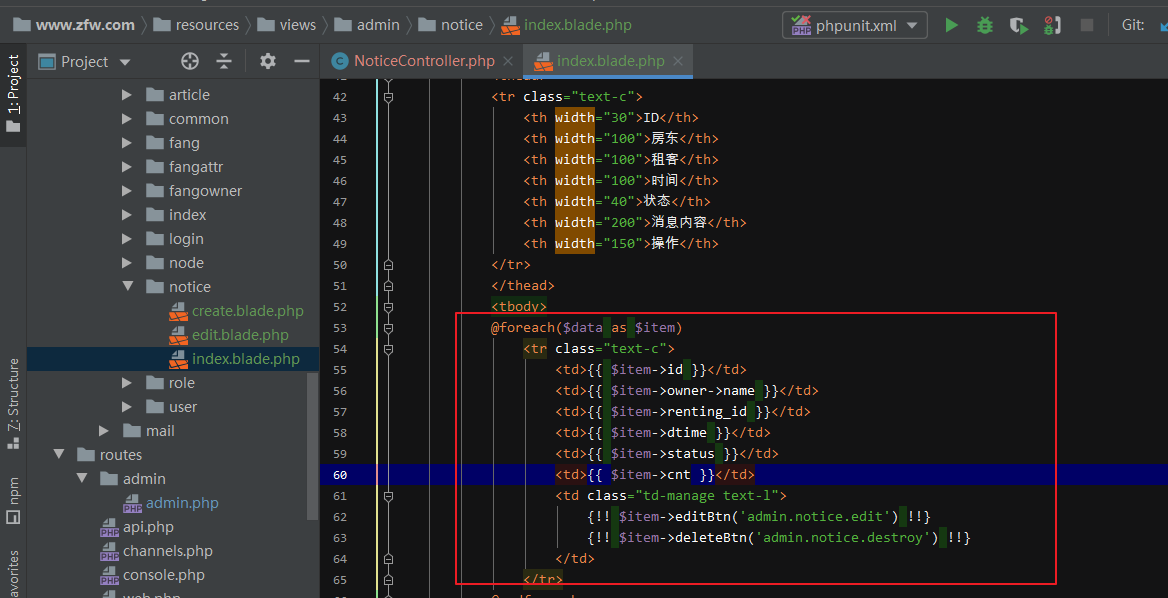
在模型中定义模型关联



在控制器调用关联关系，分页取出数据



模板中读取数据展示



效果



## 2.3、新增预约

实现添加预约步骤

* 接受数据入库
* 发送消息给房东和租客

什么时机发送消息给两个客户？

添加入库成功后，发送消息给两个客户

在laravel在写控制中，还是写在别的地方？

从业务代码可维护成本上来说，分离代码是是更好的维护，所以一般不写在控制器中.

laravel提供orm操作中，就有一个模型事件，可以利用此事件，添加对应的操作代码。

用前段话来说，可以理解orm事件为生命周期。

同步的发送还是异步的发送

同步： 添加入库成功后，就直接发送消息出去，消息发送成功后，才跳转到列表页面。

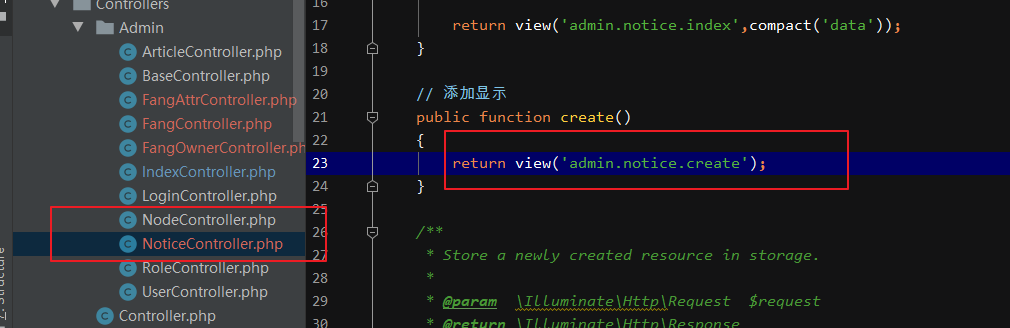
异步： 添加入库成功后，不一定直接就发送消息出去，等到服务器不太忙或者有一个别的程序监听，像这样发送消息的工作，交给监听程序来完成。

php中如何来实现工作的异步执行？

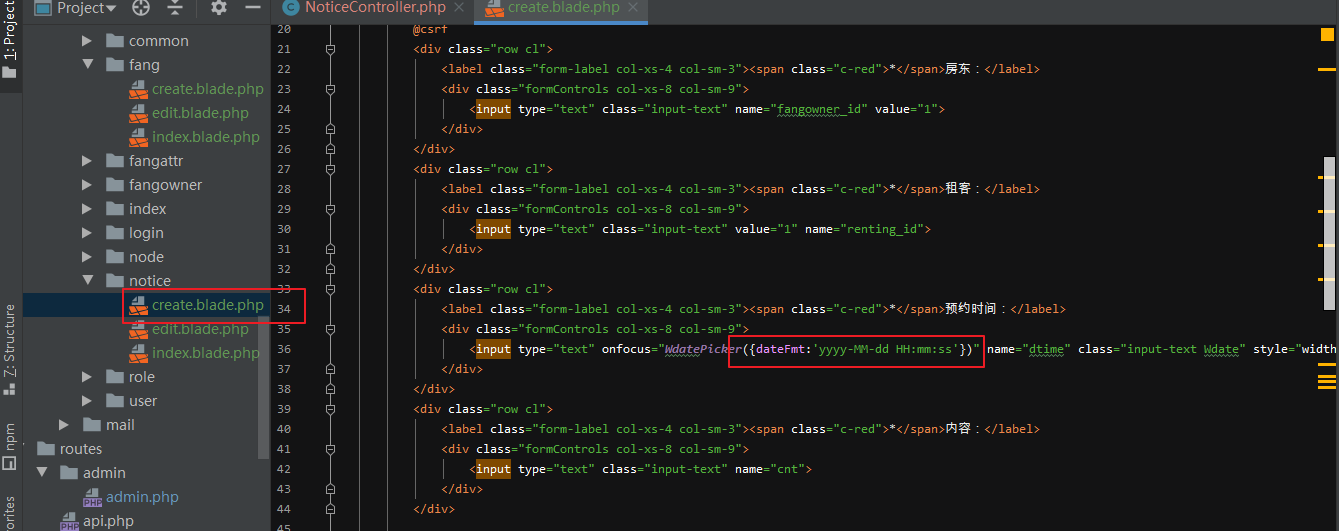
使用消息队列

### 2.3.1、显示添加界面

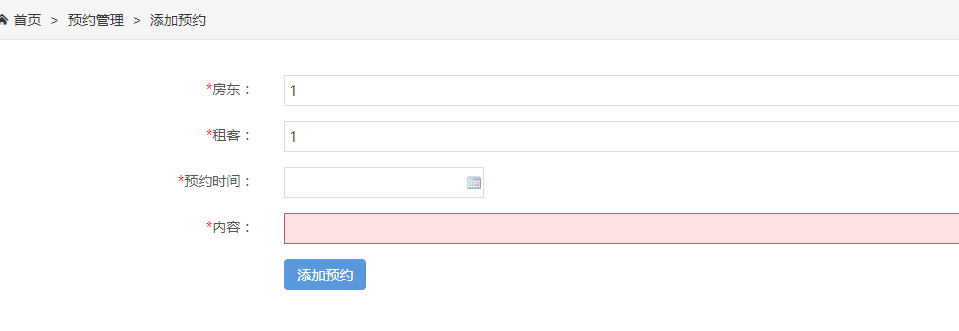
在预约控制器中指定模板显示



显示添加的模板页面

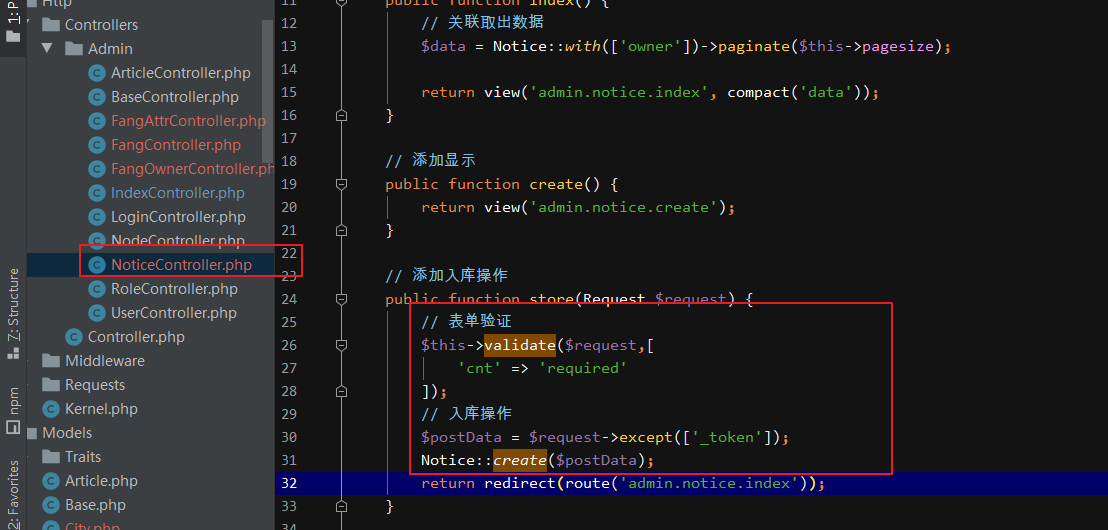


添加显示效果

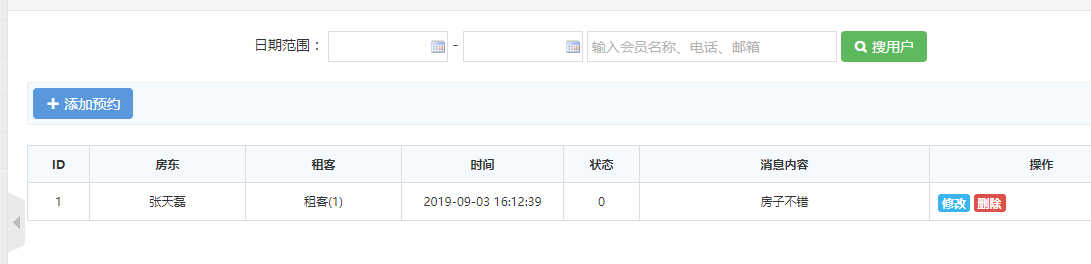


### 2.3.2、预约入库

首先需要对表单提交有服务器端验证，后使用create方法添加数据入库。



添加入库成功后，跳转到列表页面显示



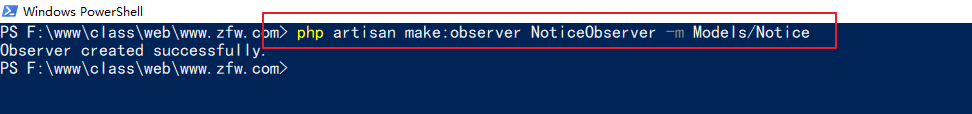
### 2.3.3、添加模型事件监听

为了解决，添加数据成功后，代码不在控制器中编写，让代码分层。

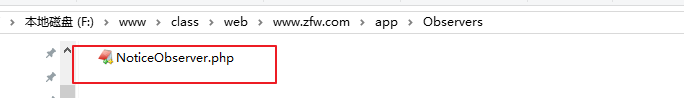
使用artisan命令来添加模型观察事件

# 此命令是laravel5.5之后才有的,之前想要有此功能，手动创建目录和文件

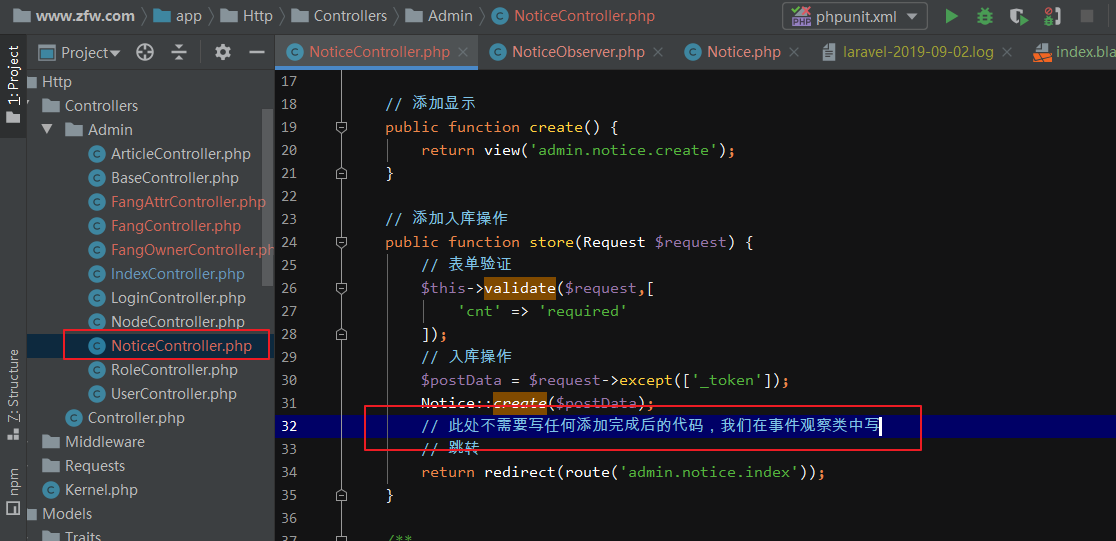
php artisan make:observer NoticeObserver -m Models/Notice



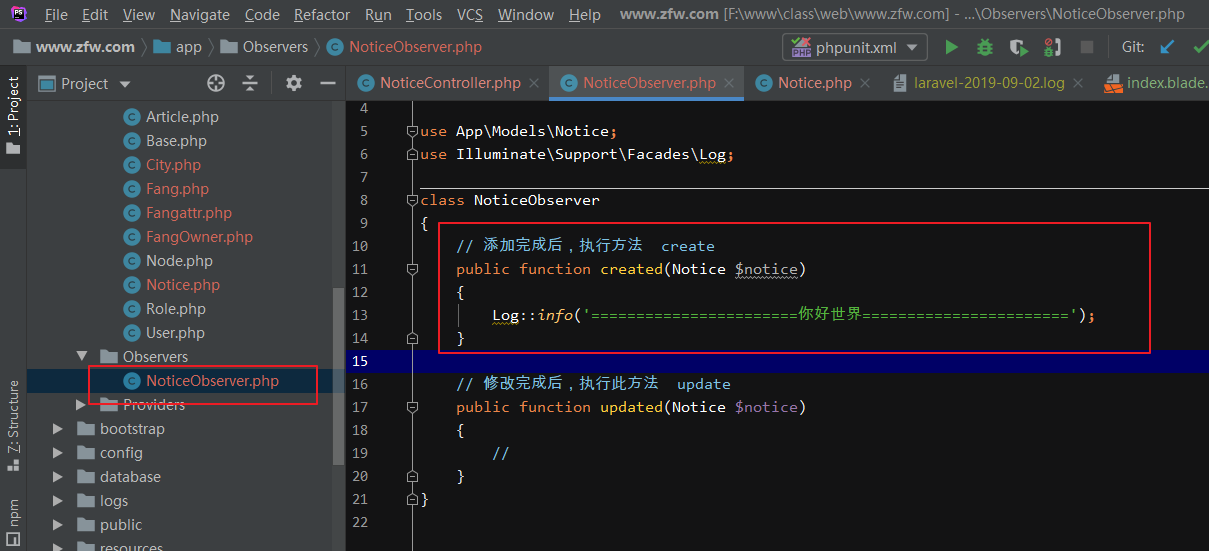




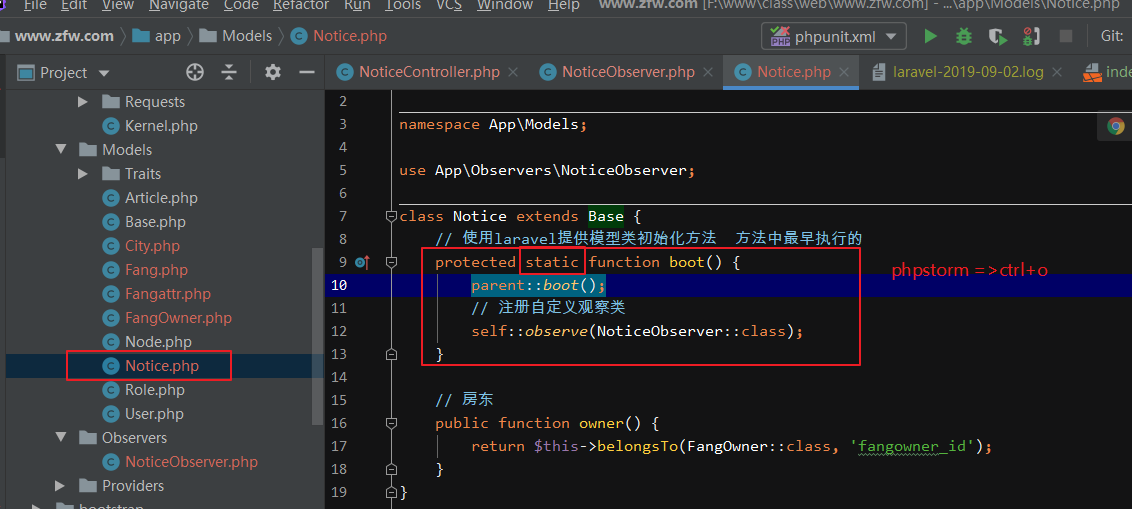
不需要在原来控制器添加成功后写代码，我们在自定义的事件观察类中对应的生命周期方法中写对应的功能。



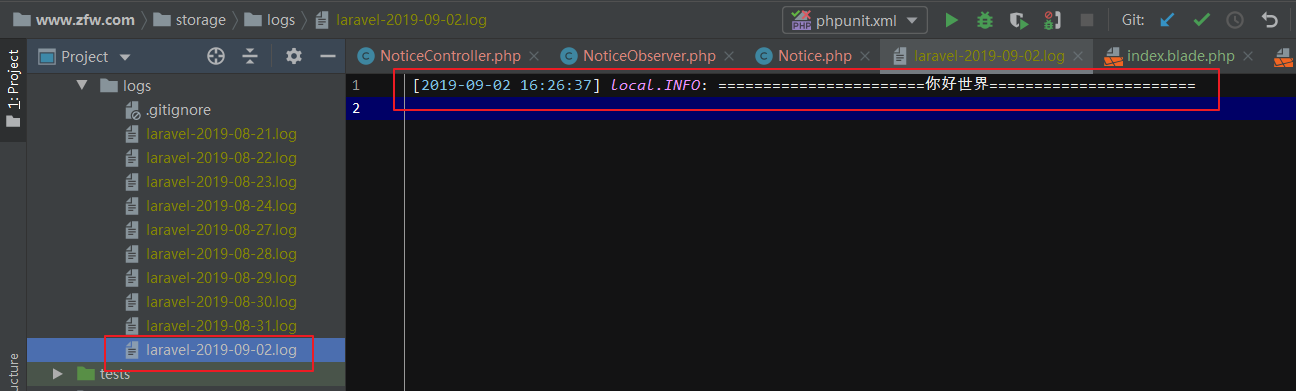
在事件观察类中添加如下的事件，名写上日志



还需要在模型中注册一下

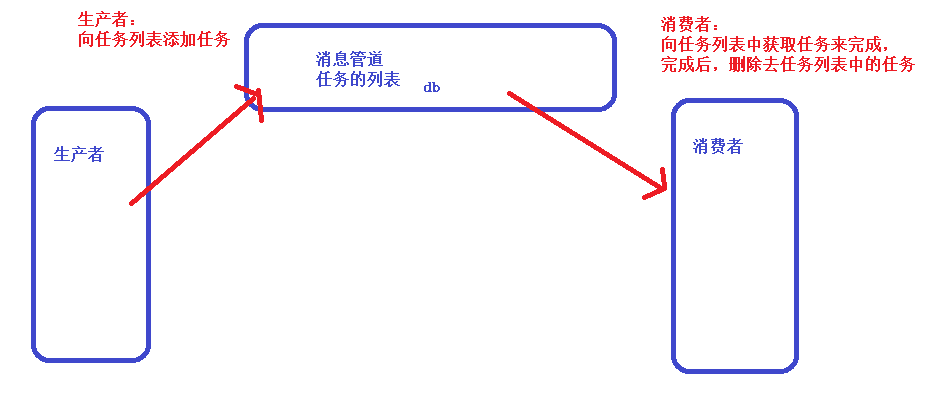


此时表单提交数据后，就会在触发模型事件，就会写入日志



### 2.3.4、消息队列

没有一个任务，分给另一个程序就完成。



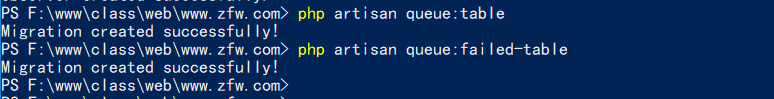
根据上图，首先创建消息管道，来存储任务。本次使用mysql表来当消息管道。

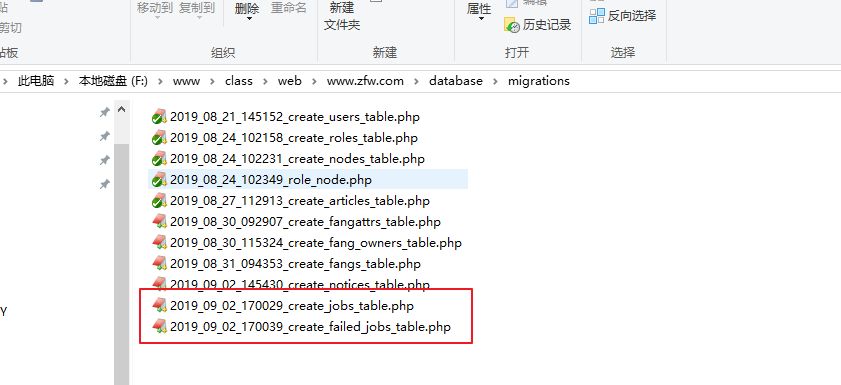
创建容器数据表

php artisan queue:table 任务队列表

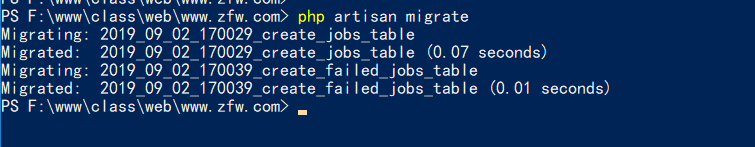
php artisan queue:failed-table 执行任务失败后，存储的表

php artisan migrate 执行迁移



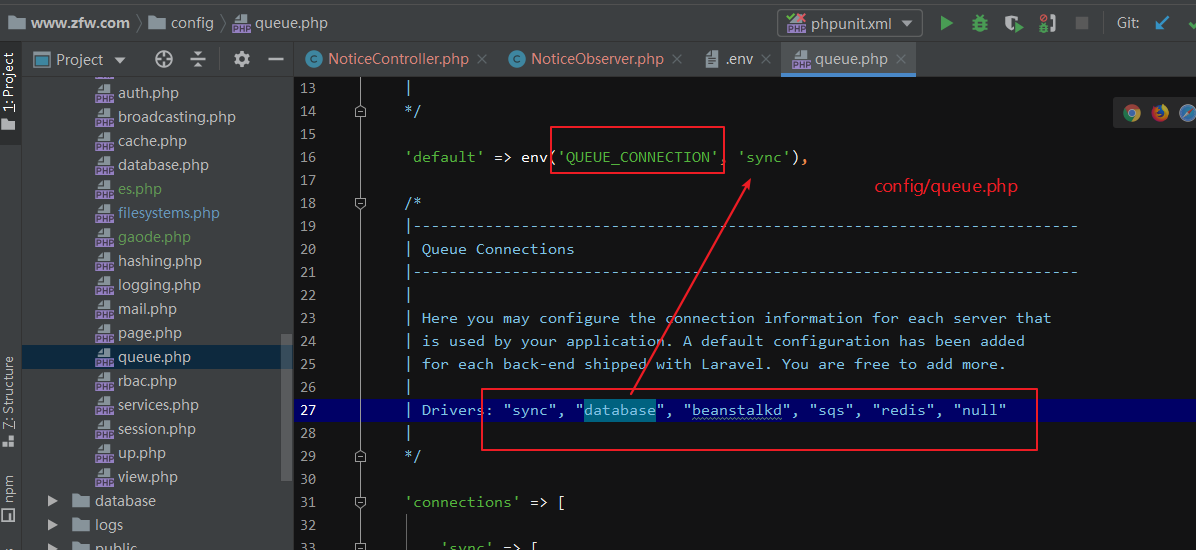


执迁迁移

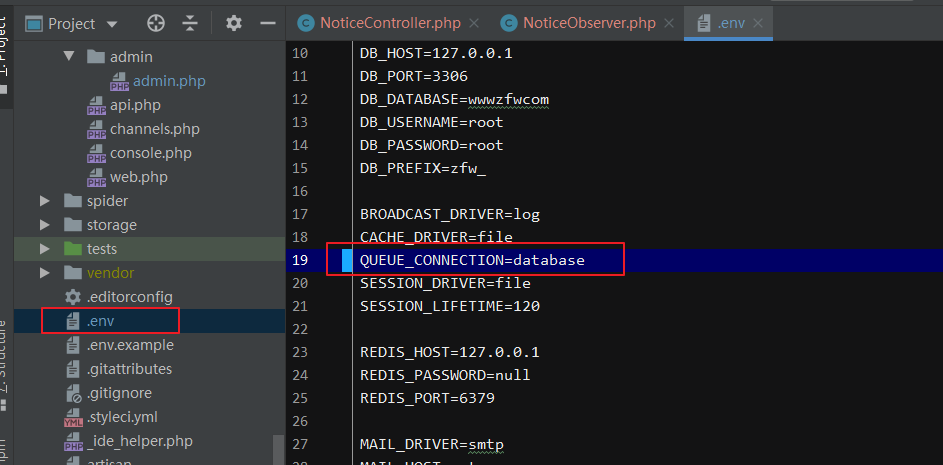


在.env文件中设置框架异步

在config目录中查看laravel默认支持的消息队列驱动

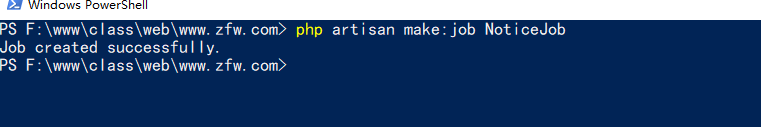


修改.env



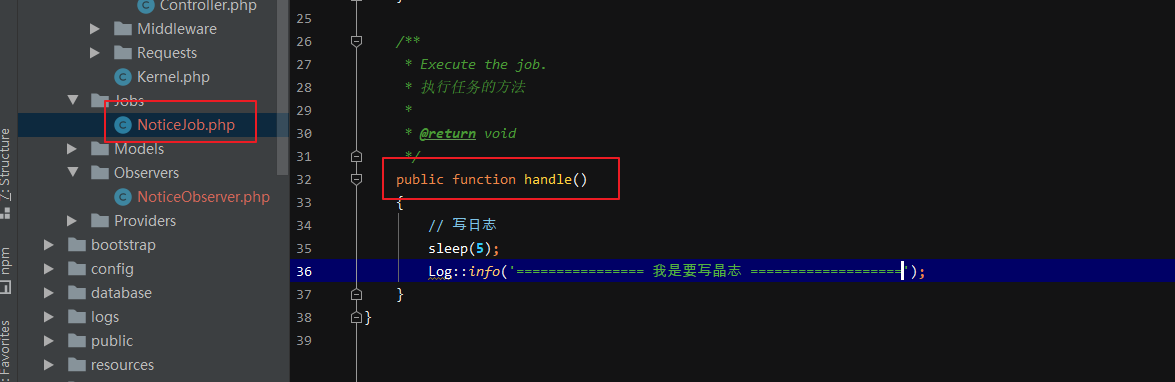
#### 2.3.4.1、创建任务

php artisan make:job xxx

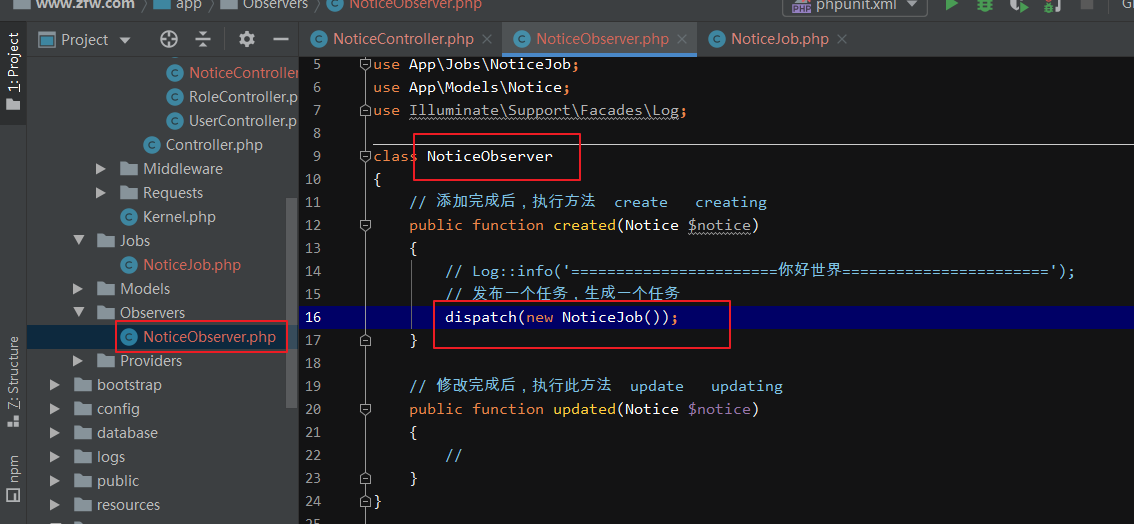




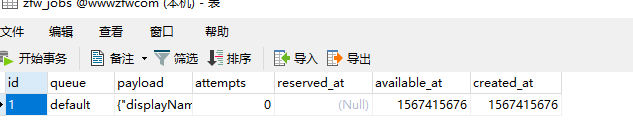
在任务文件中添加要干的事情



在模型事件中添加一个任务，生成一个任务



此时如果添加预约，添加成功后，就会在消息容器中有一个任务待执行。

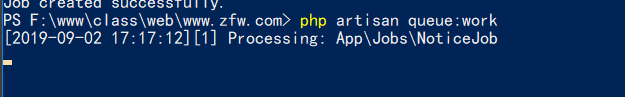


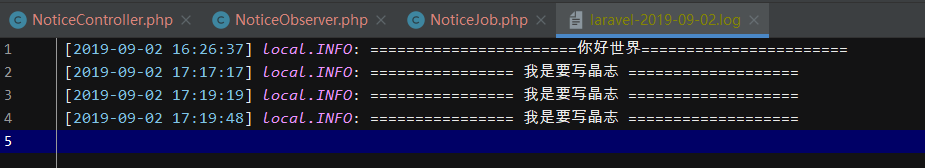
#### 2.3.4.2.、消费者

因为使用了laravel自带消息队列，所以消费者不需要写任务代码。

使用php artisan来消费它

php artisan queue:work





# 三、接口账号管理

## 3.1、设计账号表

## 3.2、添加接口账号

## 3.3、模拟登录