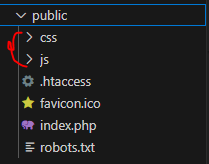
* Routing:

If we only want to show views from route then the shortcut is

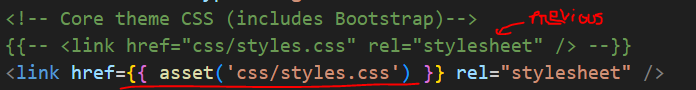


* Template:

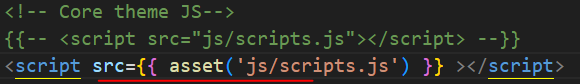
When we are work with blade file we have sometimes separate css/js file we need to put them in our public folder.



Include this to our blade template we need to use assert



Same way to include js file

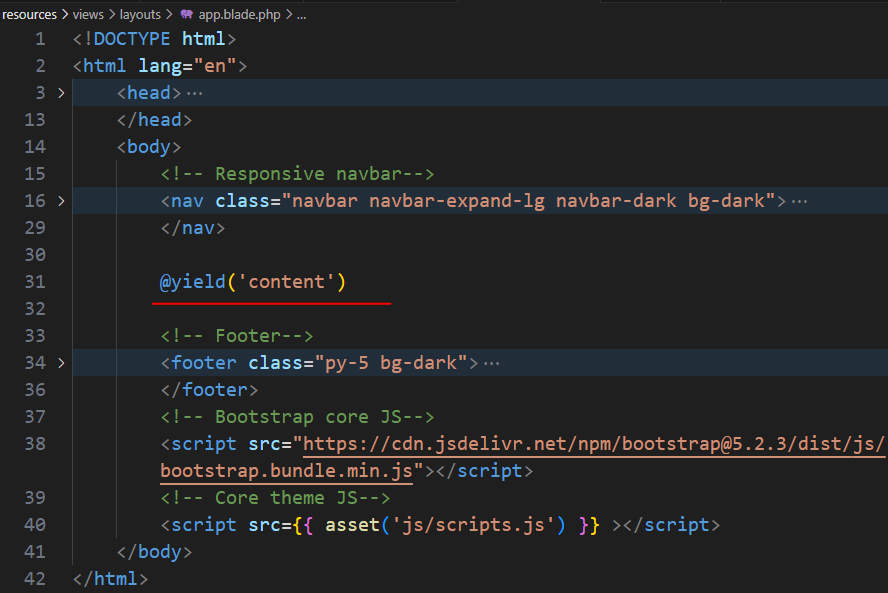


* Extend template:

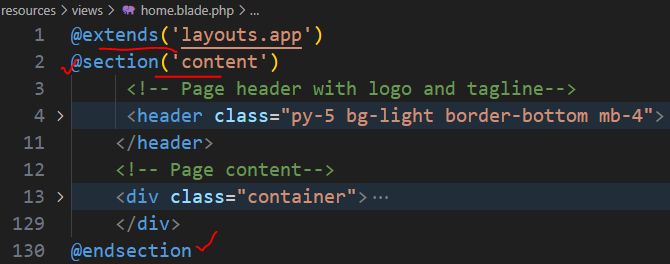
We make a layout page first and then extends it to other pages to remove same code repetition

We create a new folder name ‘layouts’ inside the ‘resource/views’ folder and then make a file name ‘app.blade.php’

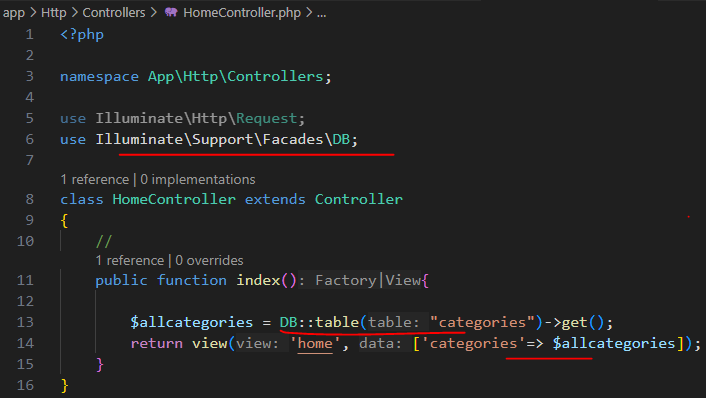
Here first we paste the code which is used in every pages, then in the main content section we write ‘@yield(‘content’) here ‘content’ is just a name, we can use any name here



Now for extends this layout in our home page,



* Fetch all the data from table

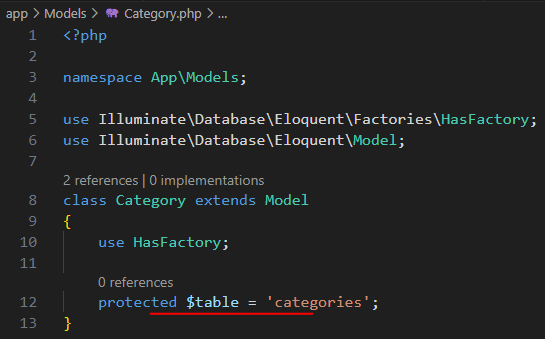


* Create model and connect it with database table



Typically, a Model name is a database name in a singular form.

By default, if we use Laravel naming, the ‘Category’ Model will know to use the ‘categories’ table, as a plural form of the word "category". If we have another migration naming, we can specify the table name in the Model with the protected $table property.



So we use it if the model is not auto recognize the table name

* Create model and migration file together by

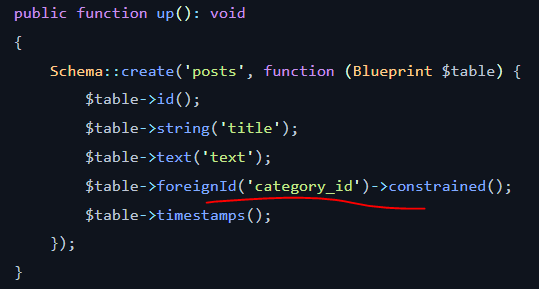


* Connect two table together,

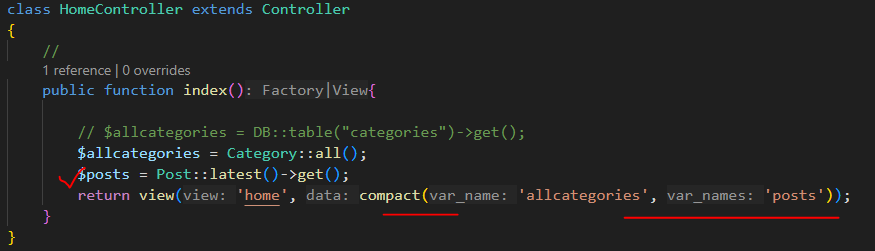
For example, we want to connect post, category table together. Every post should belong to one of the categories

Laravel has a syntax to create foreign keys. There are a few options, but I prefer the method foreignId() with constrained(). It will create both the DB column and the foreign key.

The name of the relation column should have a format of "xxxxx\_id", where "xxxxx" is a singular form of the relations table. And then, we define a constrained(), which is a shorter Laravel method for ->references('id')->on('categories').



* To see latest post,



We can do this by writing order by

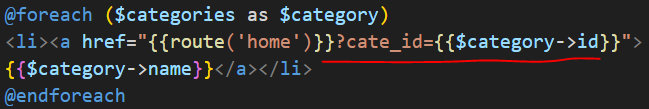


* Click category to view it’s related post

In controller



In blade file

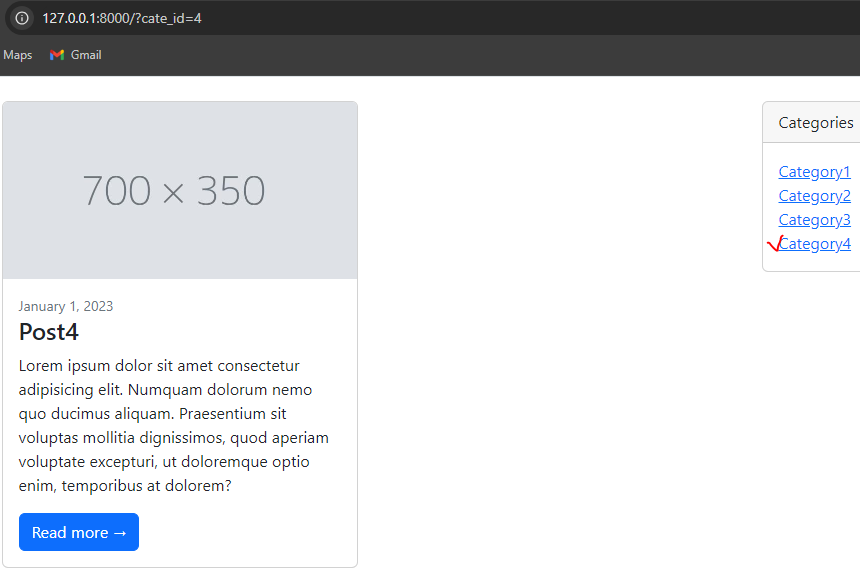
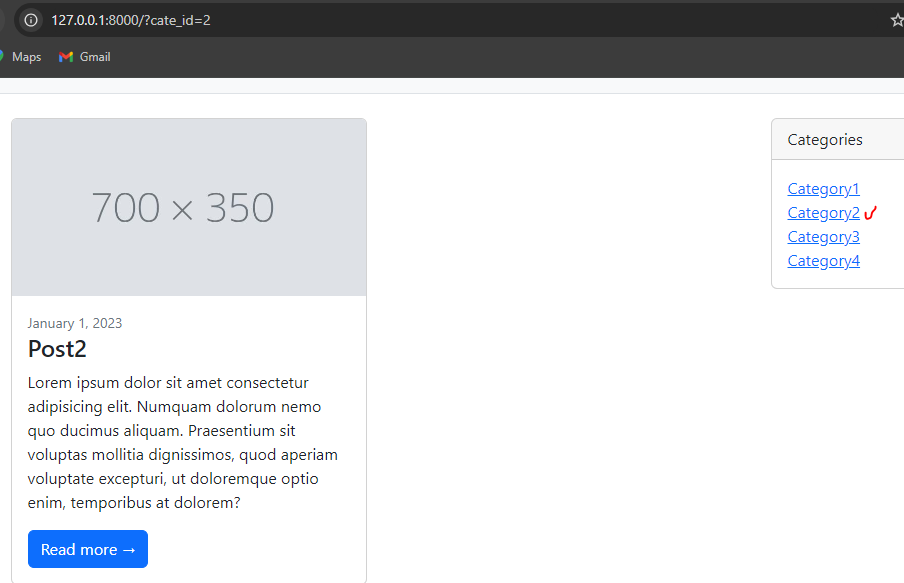


When we click category->name it takes us home view + category id, here ‘cate\_id’ is just a variable, we can put any name here, and also we need to pass it to our controllers ‘request’ parameter

When I click category2 it redirect this page

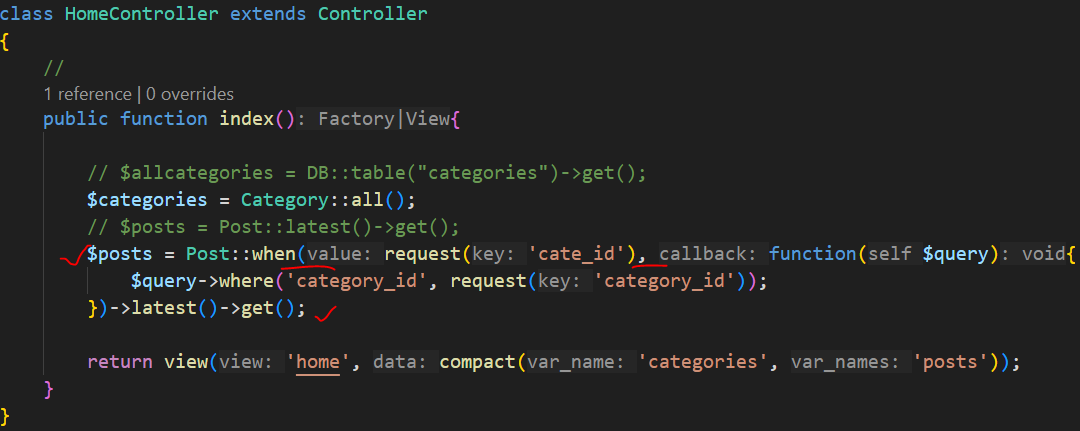


And the output is like



let's fix the page to show all categories when the category isn't selected

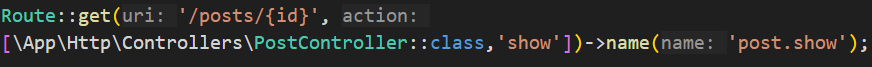
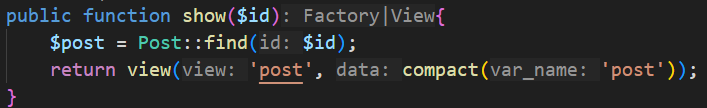
Eloquent has a when() method, which accepts the condition as a first parameter. If a condition is true, Eloquent will execute a closure function, a second parameter of the when() method.



When it gets request parameter then ‘where’ query is executed

So now by default showing all the post when select a category then only show that post which belongs to that category.

* Dynamic URL by id:

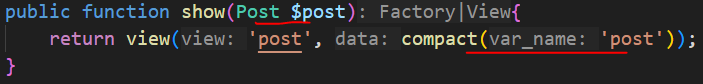


We can do it in more simple way,

if we pass ‘post’ instead of ‘id’, it will be treated as an object

Then, in the Controller, we need to type ‘Post’ that it is a post Model object and define the parameter as $post. And then, the DB search is done automatically by Laravel under the hood. So we don't need to call find().





this way of doing routes is called “Route Model Binding” because it binds the Model by type hinting the Model object type with the parameter in the Route. And it will automatically show the 404 page if the record doesn't exist.

Now we need to declare route in this way,



But we can also define it in this way like below



* Laravel Breeze:

We will create a simple admin panel to manage categories and posts. For that, we will create a separate Laravel project based on the same Models and database structure. Why do we need a separate project? Two reasons:

1. Starter Kits. Laravel has starter kits to help with authentication and authorization. Features like login, logout, and register come out of the box with Laravel ecosystem packages. This course will use Laravel Breeze because it is simpler. Starter kits must be installed on a fresh Laravel project because they overwrite some files. That's why we can't use our existing project.
2. Visual Design. We have used a simple Bootstrap template for the front page. Starter kits use Tailwind, which would also conflict with our current project code.

Next lecture -> Generate Admin User: Factories and Seeds