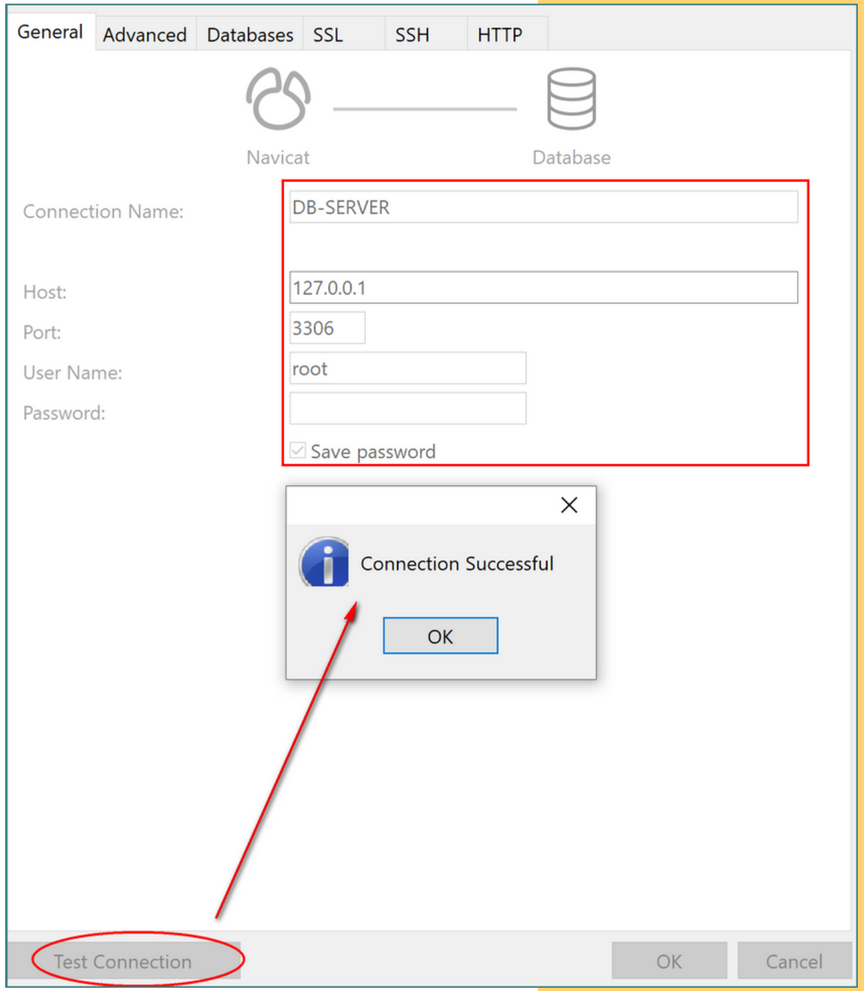
CONNECT TO DATABASE

by TRY VORN

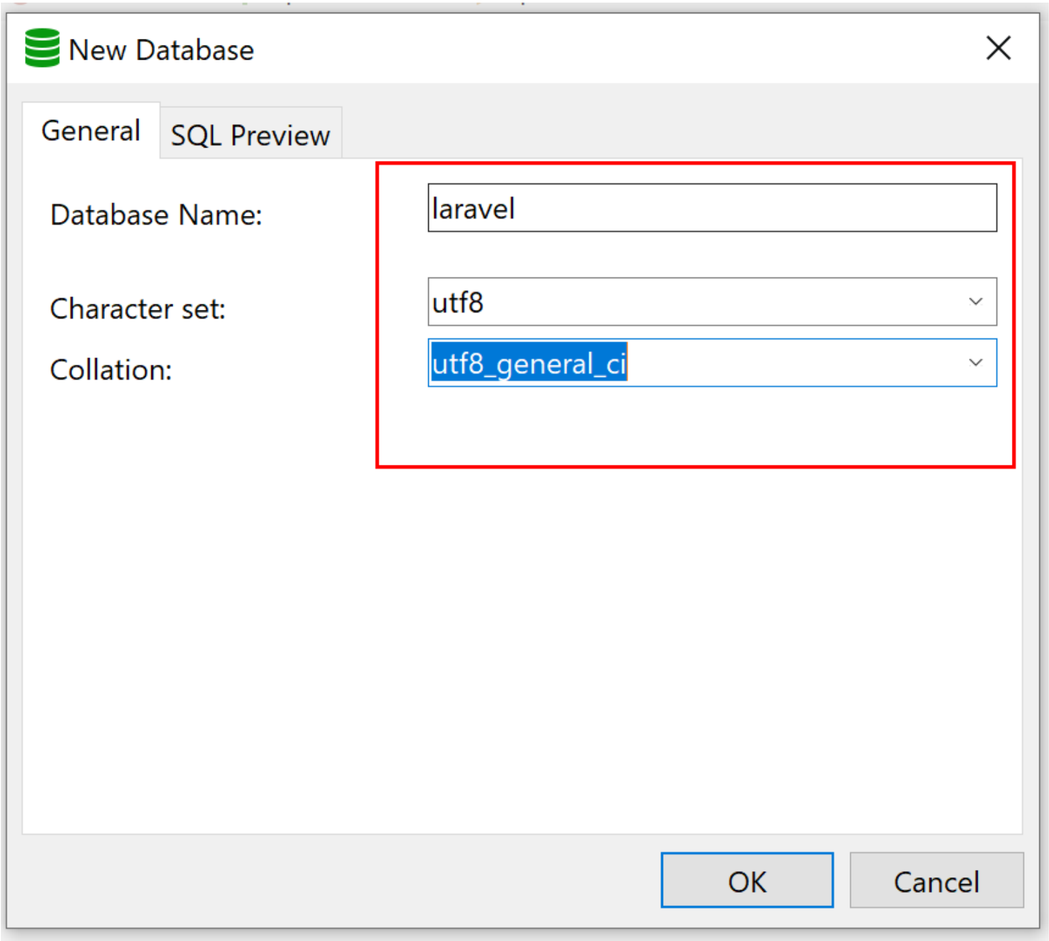
HOW TO CONNECT LARAVEL APPLICATION TO MYSQL DATABASE.

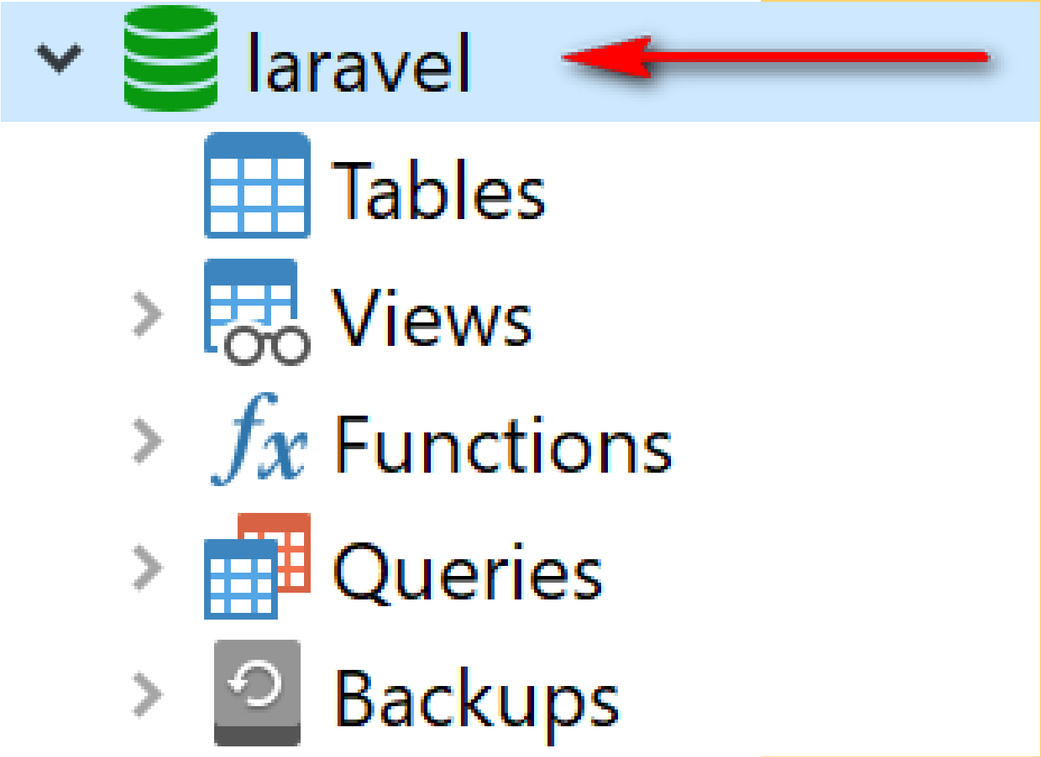
Creating the database

- You can create with Database Manager(phpMyAdmin) or Navicat Tool
- Check your connection



Database Name

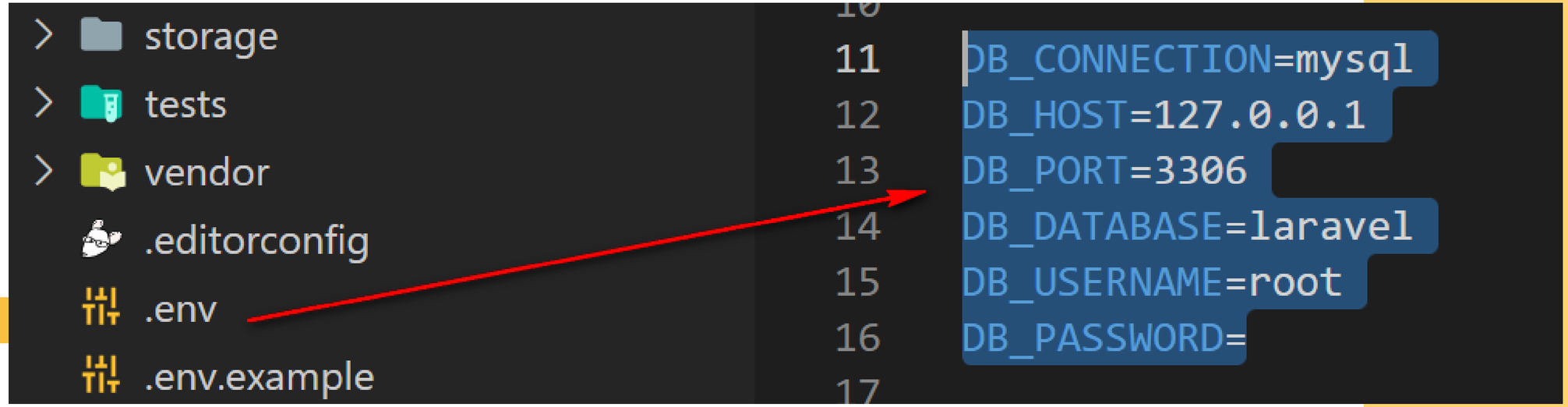




HOW TO CHECK DATABASE CONNECTIONS

Update .env configurations

The .env file has the environment configuration for your database as shown below

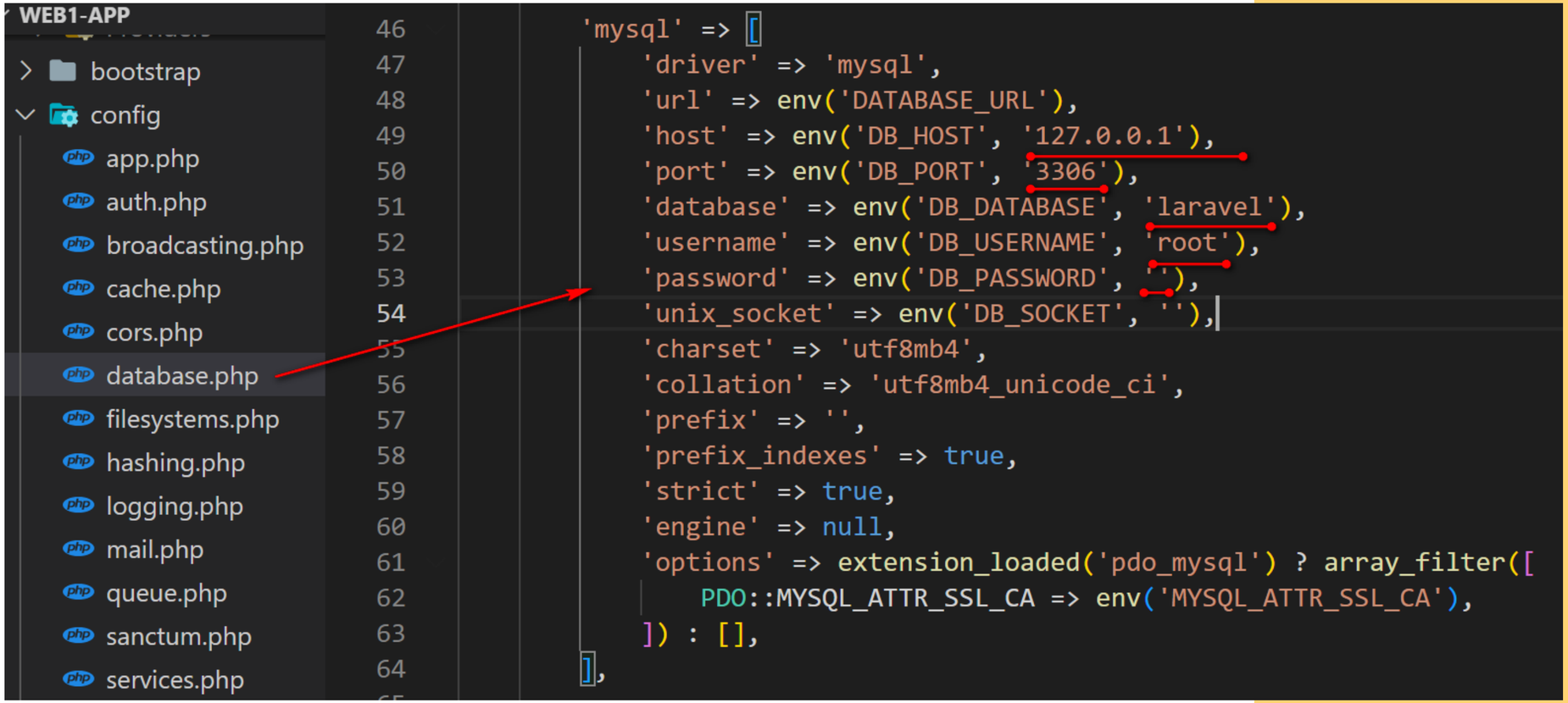


Laravel database configuration is stored inside config/database.php. The list of database configurations is listed inside this file.

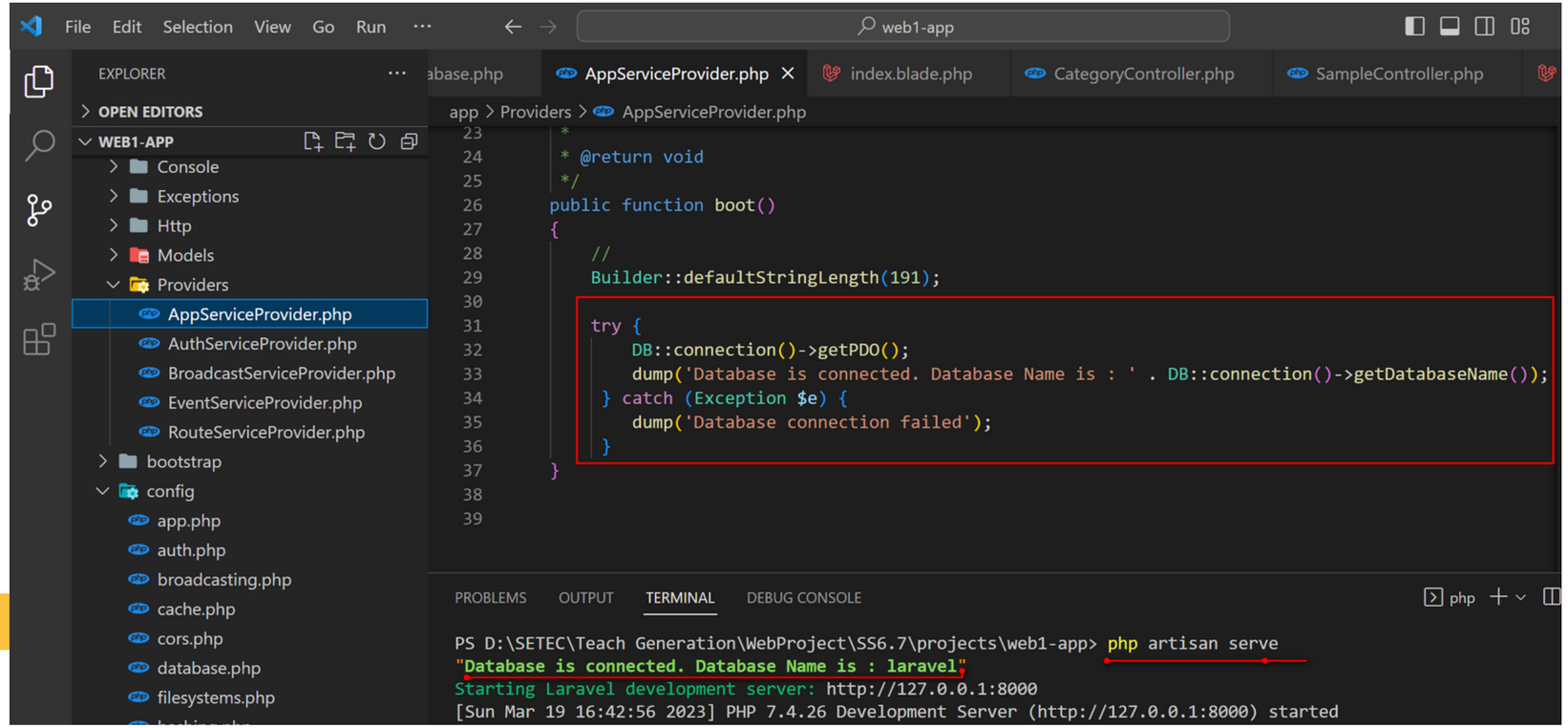
The default database used is **MySQL** and we are going to stick to it and check the database connection to **MySQL**.

```
*/

'default' => env('DB_CONNECTION', 'mysql'),

/*
```

Here are each of the database connections setup for your application. Of course, examples of configuring each database platform that is supported by Laravel is shown below to make development simple.

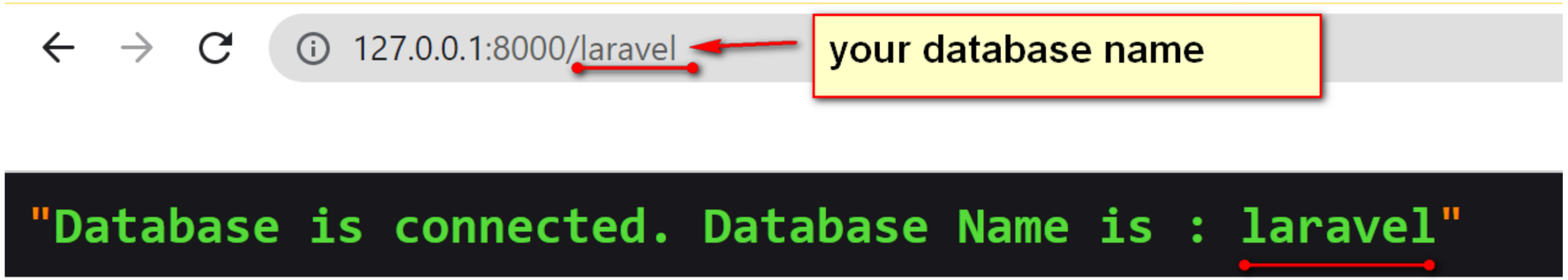


- Let us try a simple example that shows the database connection.
- Another is you can test the DB connection when you start Laravel. You can add the db connection code inside the boot() method in app/Providers/AppServiceProvider as shown below –



Work

TEST WITH URL

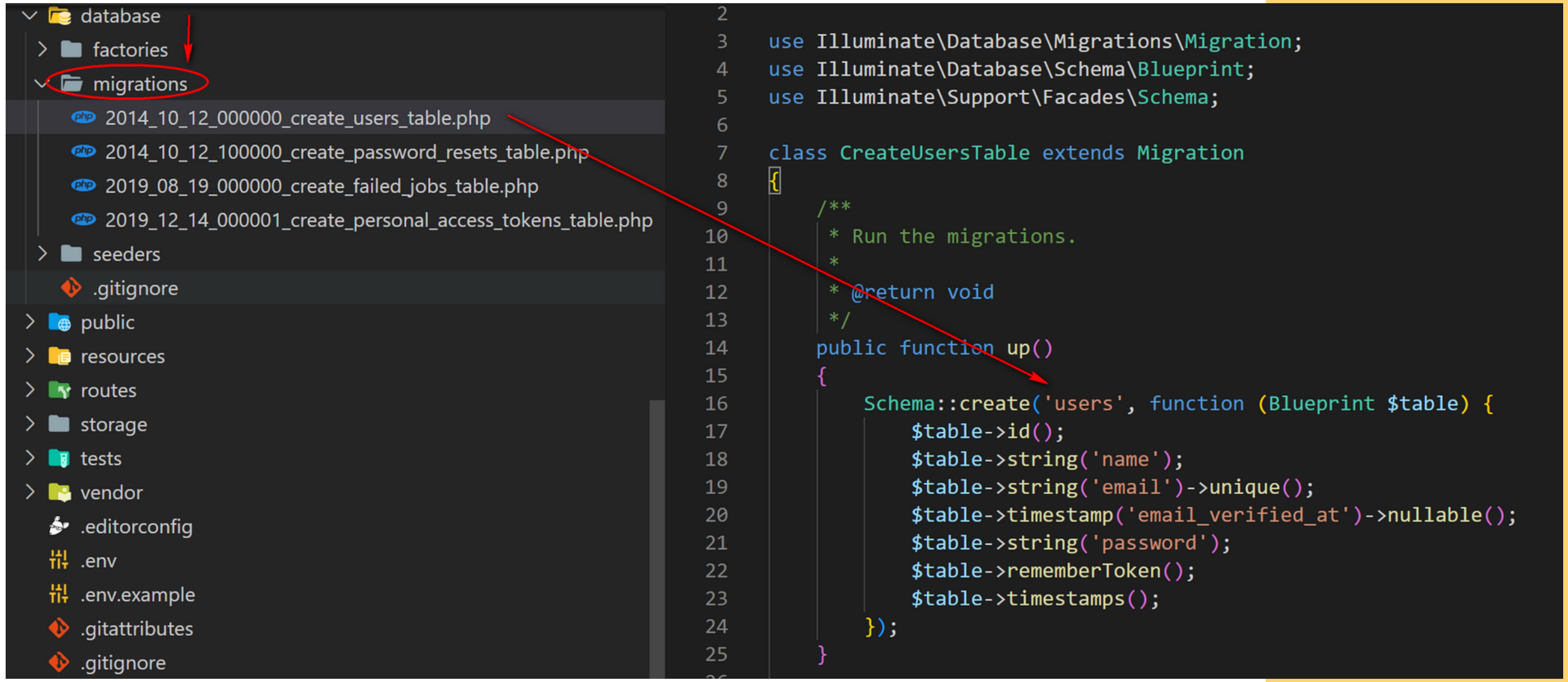


http://127.0.0.1:8000/laravel

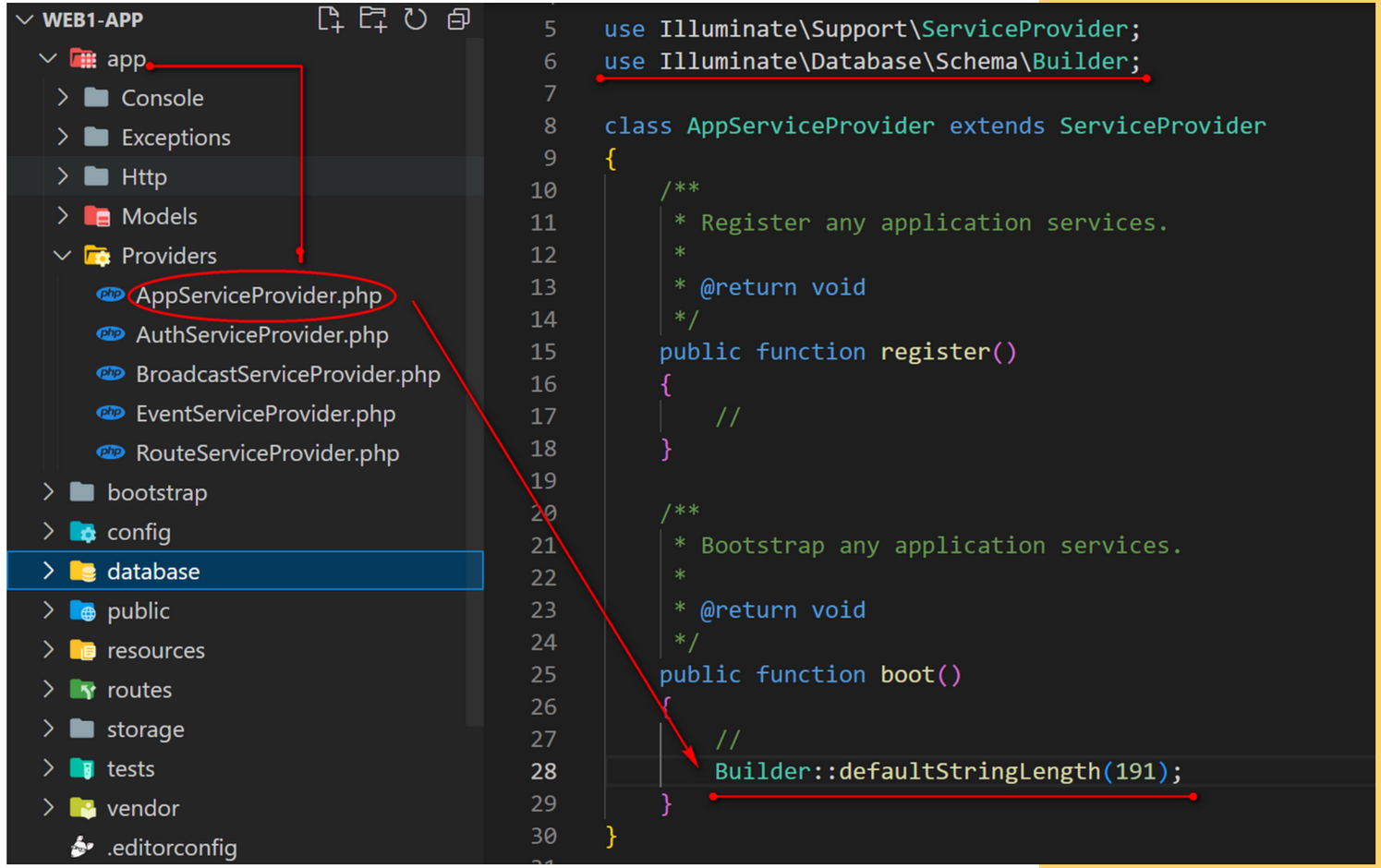
MIGRATIONS(OPTIONAL)

Migrations are found in database/migrations directory.

Laravel application comes default with user migration and it looks like this:



Note: As outlined in the Migrations guide to fix this all you have to do is edit your AppServiceProvider.php file and inside the boot method set a default string length:



Every command also includes a "help" screen which displays and describes the command's available arguments and options. To view a help screen, precede the name of the command with help:

php artisan help make

Note: To run the command create table make sure you are running MySQL Service

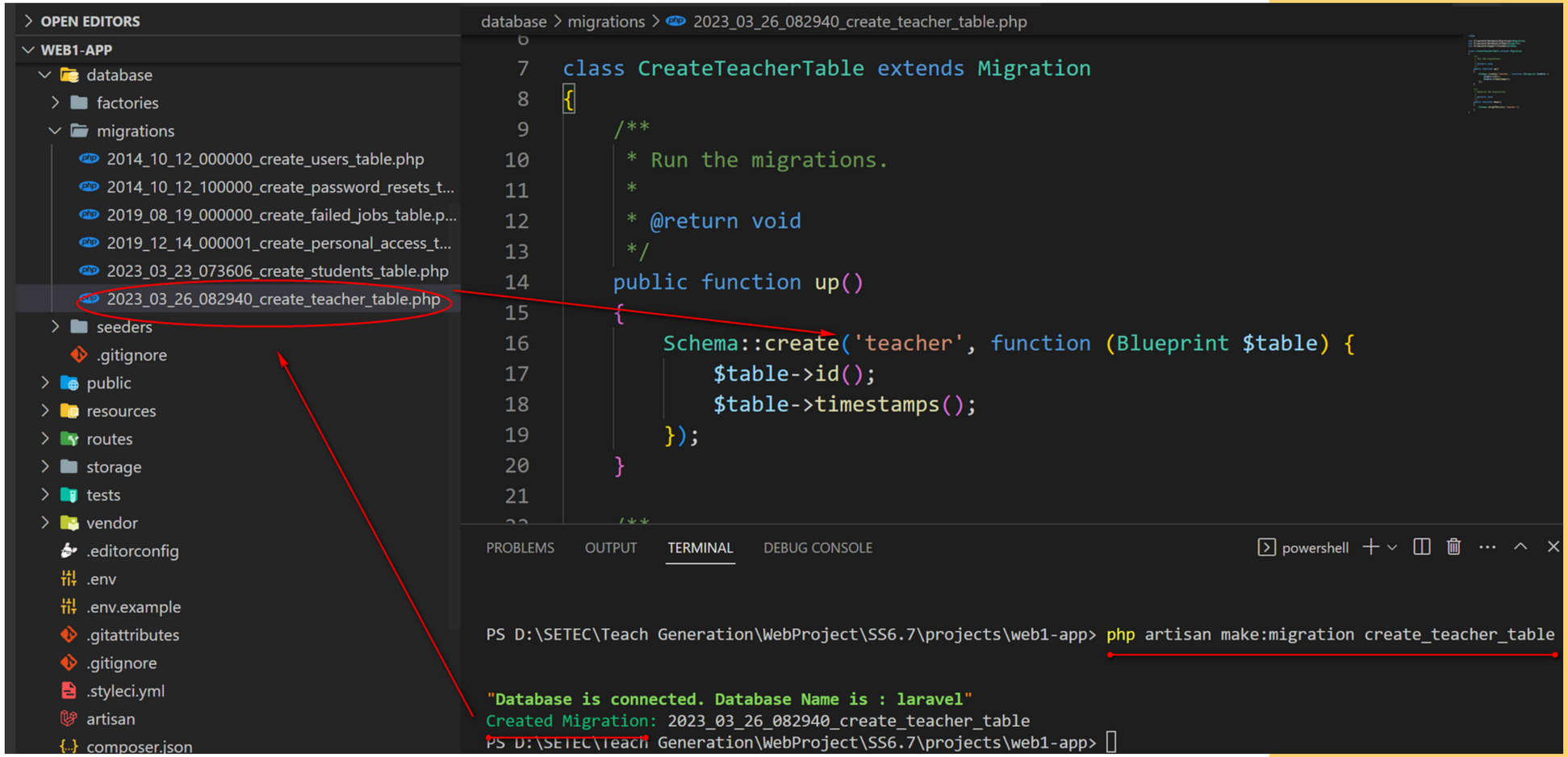
```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\web1-app> php artisan help make
"Database is connected. Database Name is : laravel"

  Command "make" is not defined.

  Did you mean one of these?
      make:cast
      make:channel
      make:command
      make:component
      make:controller
      make:event
      make:exception
      make:factory
      make:job
      make:listener
      make:mail
      make:middleware
      make:migration
      make:model
      make:notification
      make:observer
      make:policy
      make:provider
      make:request
      make:resource
      make:rule
      make:seeder
      make:test
```

php artisan help migrate

```
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\web1-app> php artisan help migrate
"Database is connected. Database Name is : laravel"
Description:
  Run the database migrations

Usage:
  migrate [options]

Options:
      --database[=DATABASE]        The database connection to use
      --force                      Force the operation to run when in production
      --path[=PATH]                The path(s) to the migrations files to be executed (multiple values allowed)
      --realpath                   Indicate any provided migration file paths are pre-resolved absolute paths
      --schema-path[=SCHEMA-PATH]  The path to a schema dump file
      --pretend                    Dump the SQL queries that would be run
      --seed                       Indicates if the seed task should be re-run
      --seeder[=SEEDER]            The class name of the root seeder
      --step                       Force the migrations to be run so they can be rolled back individually
  -h, --help                       Display help for the given command. When no command is given display help for the list command
  -q, --quiet                      Do not output any message
  -V, --version                    Display this application version
      --ansi|--no-ansi             Force (or disable --no-ansi) ANSI output
  -n, --no-interaction             Do not ask any interactive question
      --env[=ENV]                  The environment the command should run under
  -v|vv|vvv, --verbose             Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\web1-app>
```

- Example: create new table (create_teacher_table)

php artisan make:migration create_teacher_table



- After created table name
- It can be executed to the database with the following command:

php artisan migrate

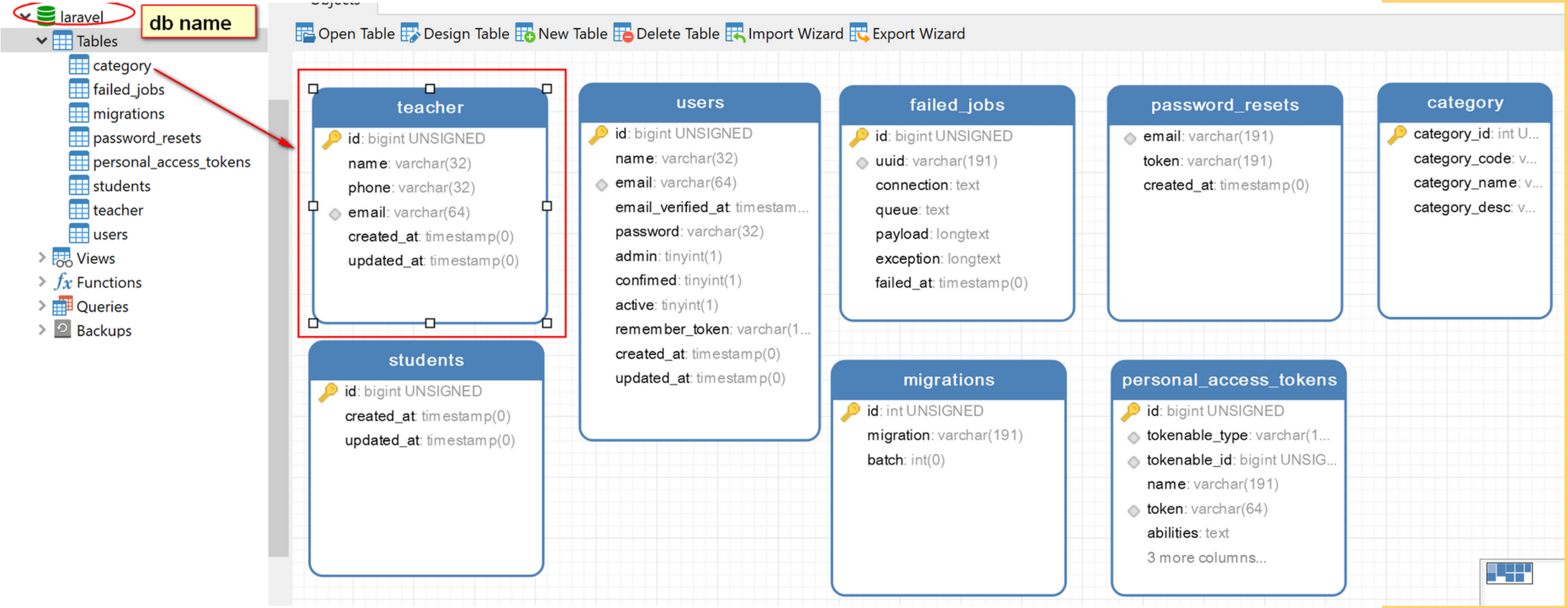
database > migrations > 2023_03_26_082940_create_teacher_table.php

```php
{
    Schema::create('teacher', function (Blueprint $table) {
        $table->id();

        $table->string('name', 32);
        $table->string('phone', 32);
        $table->string('email', 64)->unique();

        $table->timestamps();
    });
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE powershell

```
    migrate:rollback
    migrate:status

PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\web1-app> php artisan migrate
"Database is connected. Database Name is : laravel"
Migrating: 2023_03_23_073606_create_students_table
Migrated:  2023_03_23_073606_create_students_table (11.63ms)
Migrating: 2023_03_26_082940_create_teacher_table
Migrated:  2023_03_26_082940_create_teacher_table (51.11ms)
PS D:\SETEC\Teach Generation\WebProject\SS6.7\projects\web1-app>
```

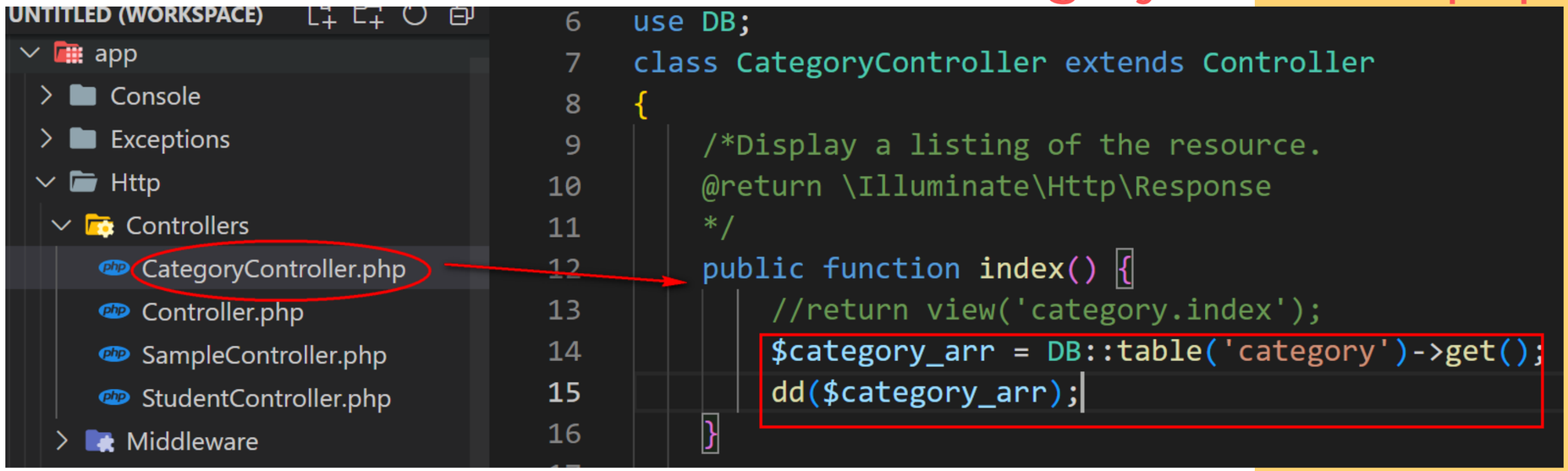
Go to check your database: *laravel*



Retrieve data from the database

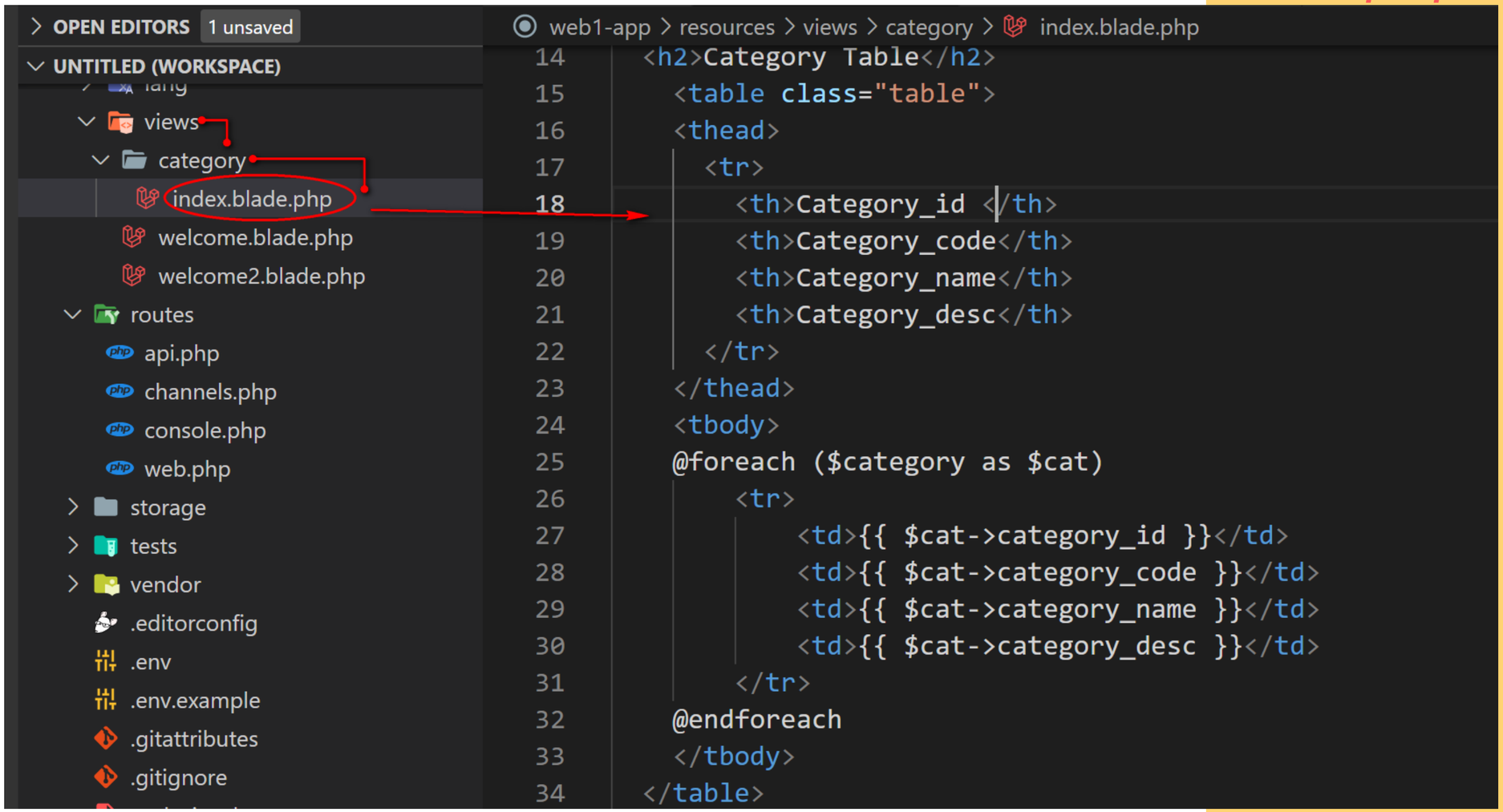
- Create **3** files for retrieving data ：
  - CategoryController.php (app/Http/Controllers/CategoryController.php)
  - index.blade.php (resources/views/category/index.blade.php)
  - web.php (routes/web.php)

1.CategoryController.php



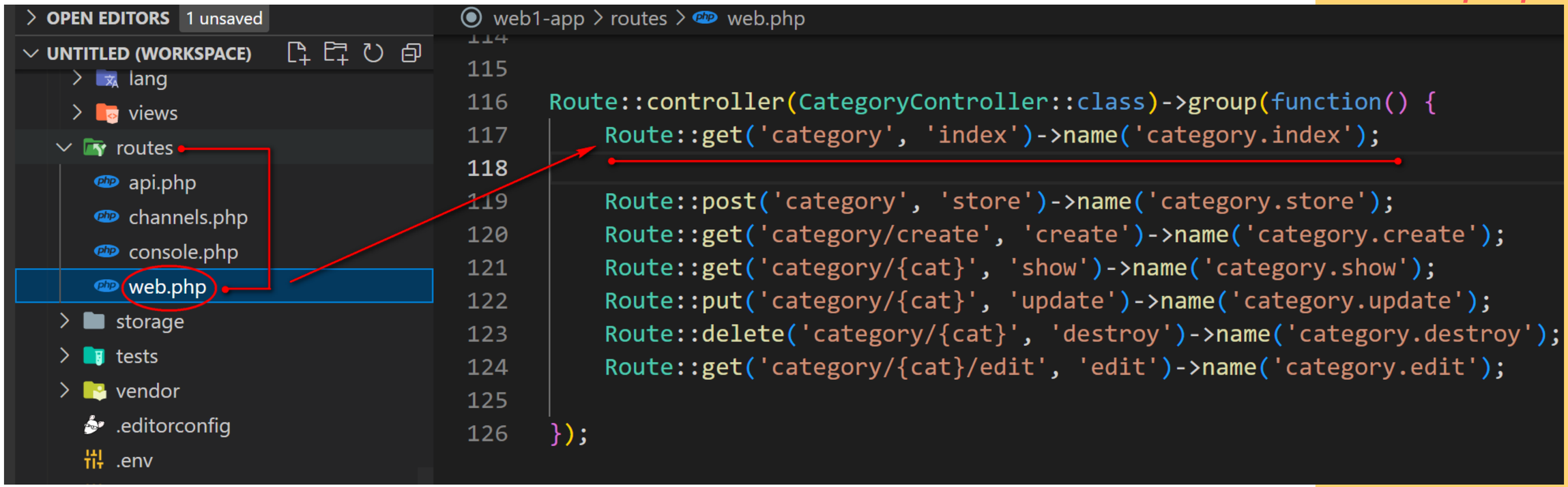
Retrieve data from the database (count...)

2. index.blade.php



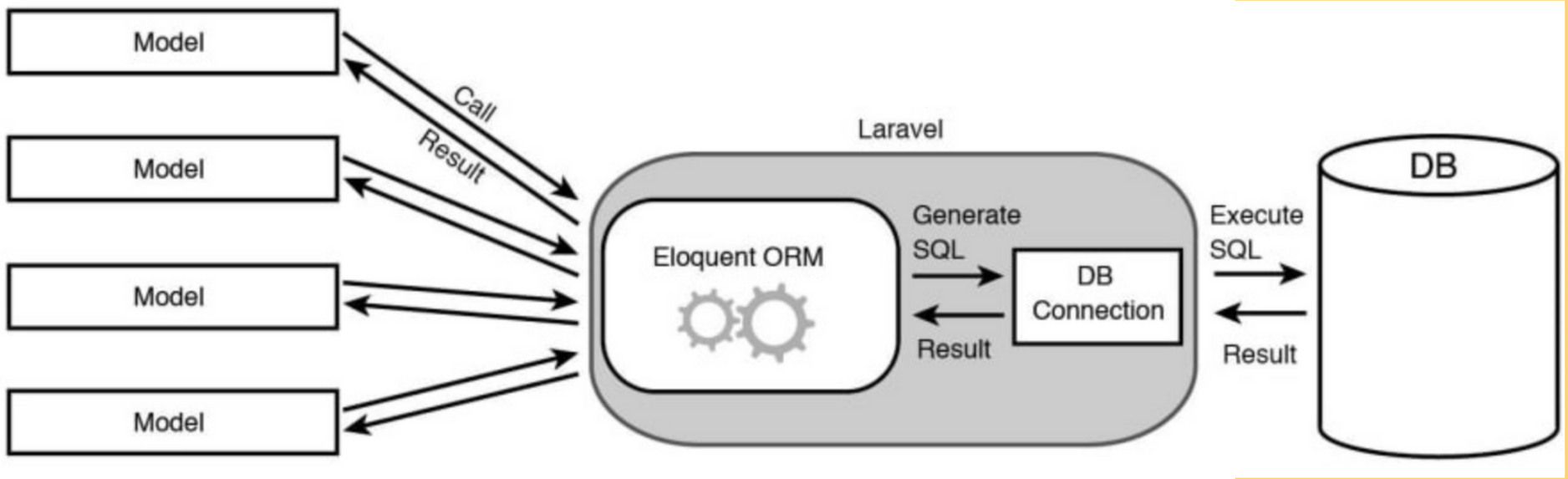
Retrieve data from the database (count...)

3. web.php



ELOQUENT ORM

Eloquent is an object relational mapper (**ORM**) that is included by default within the Laravel framework. An **ORM** is software that facilitates handling database records by representing data as objects, working as a layer of abstraction on top of the database engine used to store an application's data.



ELOQUENT ORM

Eloquent is an object relational mapper (ORM) that is included by default within the Laravel framework. An ORM is software that facilitates handling database records by representing data as objects, working as a layer of abstraction on top of the database engine used to store an application's data.

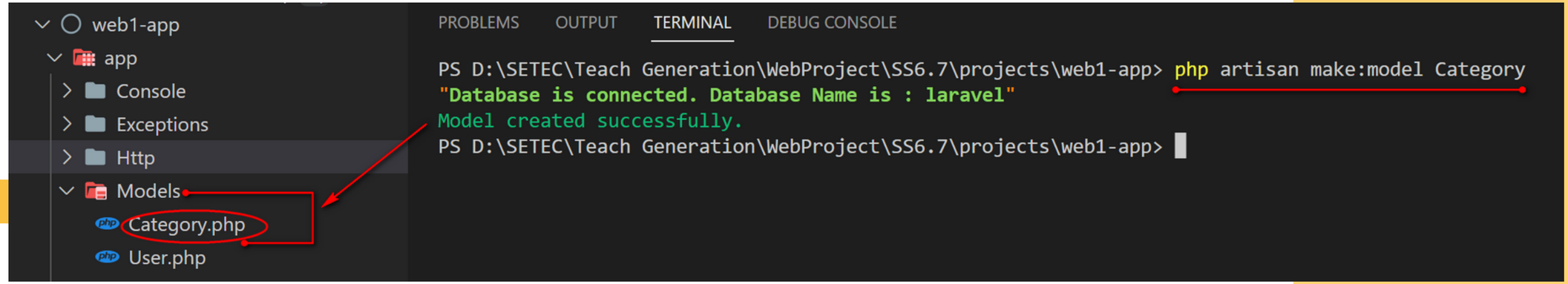
When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the database table,Eloquent models allow you to insert, update, and delete records from the table as well.

ELOQUENT ORM (COUNT...)

Generating Model Classes

Models typically live in the app\Models directory and extend the Illuminate\Database\Eloquent\Model class.

use command to generate a new model: *php artisan make:model Category*



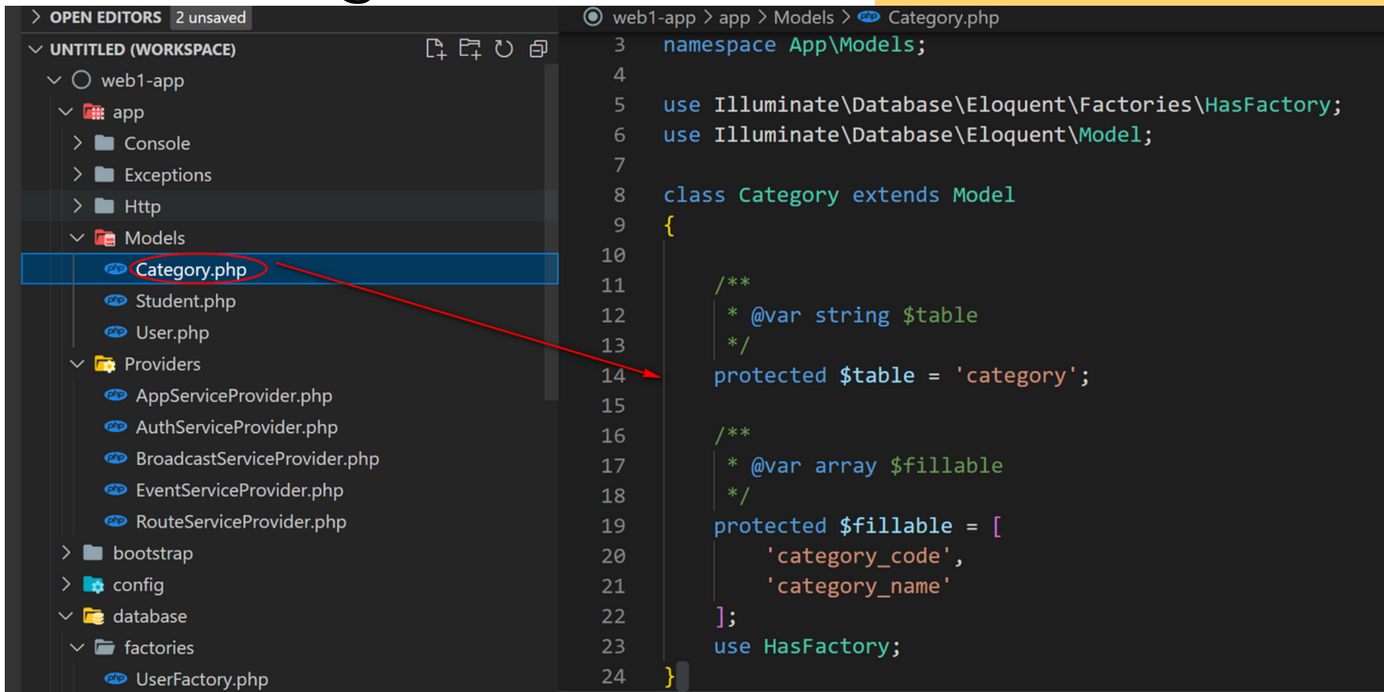
ELOQUENT ORM (COUNT...)

Setup Model Properties

Now let's, add protected $table = ""; and protected $fillable = []; to our model and defined each values.

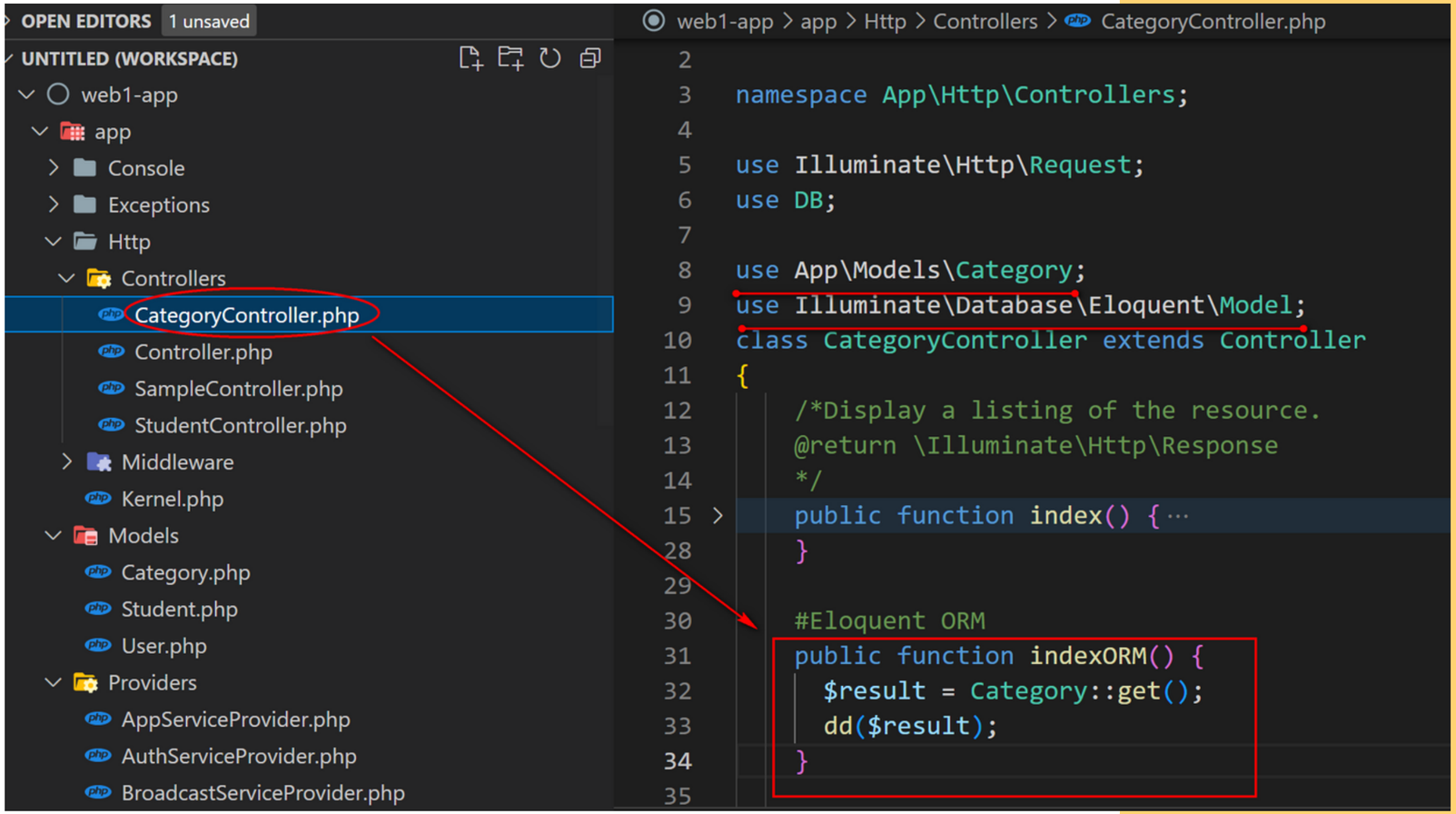
protected $table: Define model table name

protected $fillable: Fields that are mas-assignable



ELOQUENT ORM (COUNT...)

Model with Controller (CategoryController.php)



ELOQUENT ORM (COUNT...)

See the result

127.0.0.1:8000/categoryORM

```
Illuminate\Database\Eloquent\Collection {#366 ▼
  #items: array:4 [▼
    0 => App\Mode…\Category {#367 ▼
      #table: "category"
      #fillable: array:2 [▼
        0 => "category_code"
        1 => "category_name"
      ]
      #connection: "mysql"
      #primaryKey: "id"
      #keyType: "int"
      +incrementing: true
      #with: []
      #withCount: []
      +preventsLazyLoading: false
      #perPage: 15
      +exists: true
      +wasRecentlyCreated: false
      #escapeWhenCastingToString: false
      #attributes: array:4 [▶]
      #original: array:4 [▶]
      #changes: []
      #casts: []
      #classCastCache: []
      #attributeCastCache: []
      #dates: []
      #dateFormat: null
      #appends: []
      #dispatchesEvents: []
      #observables: []
      #relations: []
      #touches: []
      +timestamps: true
      #hidden: []
      #visible: []
      #guarded: array:1 [▶]
    }
    1 => App\Mode…\Category {#368 ▶}
    2 => App\Mode…\Category {#369 ▶}
    3 => App\Mode…\Category {#370 ▶}
  ]
  #escapeWhenCastingToString: false
}
```

If you would like to generate a database migration when you generate the model, you may use the --migration or -m option:

php artisan make:model Flight --migration

```
# Generate a model and a FlightFactory class...
php artisan make:model Flight --factory
php artisan make:model Flight -f

# Generate a model and a FlightSeeder class...
php artisan make:model Flight --seed
php artisan make:model Flight -s

# Generate a model and a FlightController class...
php artisan make:model Flight --controller
php artisan make:model Flight -c

# Generate a model, FlightController resource class, and form request classes...
php artisan make:model Flight --controller --resource --requests
php artisan make:model Flight -crR

# Generate a model and a FlightPolicy class...
php artisan make:model Flight --policy

# Generate a model and a migration, factory, seeder, and controller...
php artisan make:model Flight -mfsc

# Shortcut to generate a model, migration, factory, seeder, policy, controller, a
php artisan make:model Flight --all

# Generate a pivot model...
php artisan make:model Member --pivot
```

Work

ADDING AND REMOVING COLUMNS FROM EXISTING TABLES USING LARAVEL MIGRATIONS.

https://dev.to/roxie/how-to-connect-laravel-application-to-mysql-database-5han

https://dev.to/roxie/adding-and-removing-columns-from-existing-tables-using-laravel-migrations-389g