Laravel Seeding & Migrations

Often times when you start a project. You may want to add in some example content, for many obvious reasons. You may want to show a client or potential client what the end product could look like. You may need it to gauge if your UI will work with the content and so on.

There are a couple different methods we can use in Laravel to do this. We will need to use command line again. While you may be used to doing this using a web interface. There are some advantages too doing this via command line. It's less time consuming for one. It also creates less confusion for all the devs working on the same project.

https://laravel.com/docs/4.2/migrations

Seeder

To make a seeder, we have to make a seeder file. We can do so in the CLI with the following command:

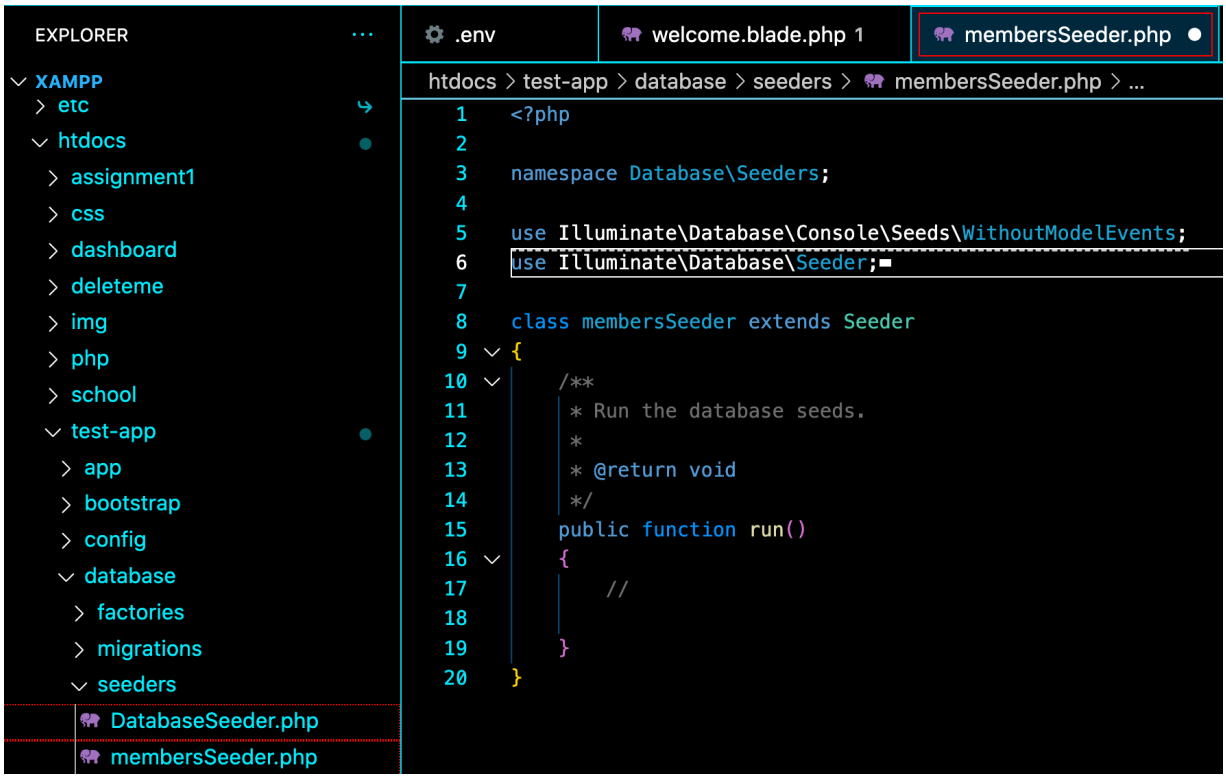
php artisan make:seeder seederName

```
⊗ →  test-app php artisan serve

   INFO  Server running on [http://127.0.0.1:8000].

   Press Ctrl+C to stop the server

   2023-07-25 16:21:31 ..............................
   2023-07-25 16:21:32 /favicon.ico .................
   2023-07-25 16:21:49 ..............................
   2023-07-25 16:21:49 /favicon.ico .................
  ^C
○ →  test-app php artisan make:seeder productsSeeder
```

Hit enter and it will make the seeder php file.

```
^C
● →  test-app php artisan make:seeder productsSeeder

   INFO  Seeder [database/seeders/productsSeeder.php] created successfully.

○ →  test-app
```

This file can be found in appfolder>database>seeders>the-name-you-gave-seeder.php



Now with this file you can enter the seed data. Before we do, let's add a couple more lines in at line 7:

```
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
```

Note* In Laravel, "Illuminate" refers to the set of core components and classes that form the foundation of the framework. Illuminate is a collection of elegant and powerful PHP libraries that handle various aspects of web application development, including database interaction,

routing, caching, validation, and more.

Illuminate is the backbone of Laravel and provides the essential functionalities that make Laravel a robust and feature-rich PHP framework. The name "Illuminate" comes from the idea of shedding light on the development process, making it easier and more enjoyable for developers to work on their projects.

. `use Illuminate\Support\Facades\DB;`
. This line imports the `DB` facade from the `Illuminate\Support\Facades` namespace. In Laravel, the `DB` facade provides a simple and convenient way to interact with the database without directly using the underlying database connections or raw SQL queries. With this "use" statement, you can use the `DB` facade anywhere in your current PHP file without having to write the full namespace every time. For example, you can now use `DB::table('users')->get()` instead of `Illuminate\Support\Facades\DB::table('users')->get()`.

. `use Illuminate\Support\Str;`
. This line imports the `Str` class from the `Illuminate\Support` namespace. The `Str` class provides various string manipulation methods that are commonly used in Laravel projects. With this "use" statement, you can directly use the `Str` class and its methods in your current PHP file without specifying the full namespace. For example, you can now use `Str::slug('Hello World')`

`instead of Illuminate\Support\Str::slug('Hello World').`

Now we can enter some seed data:

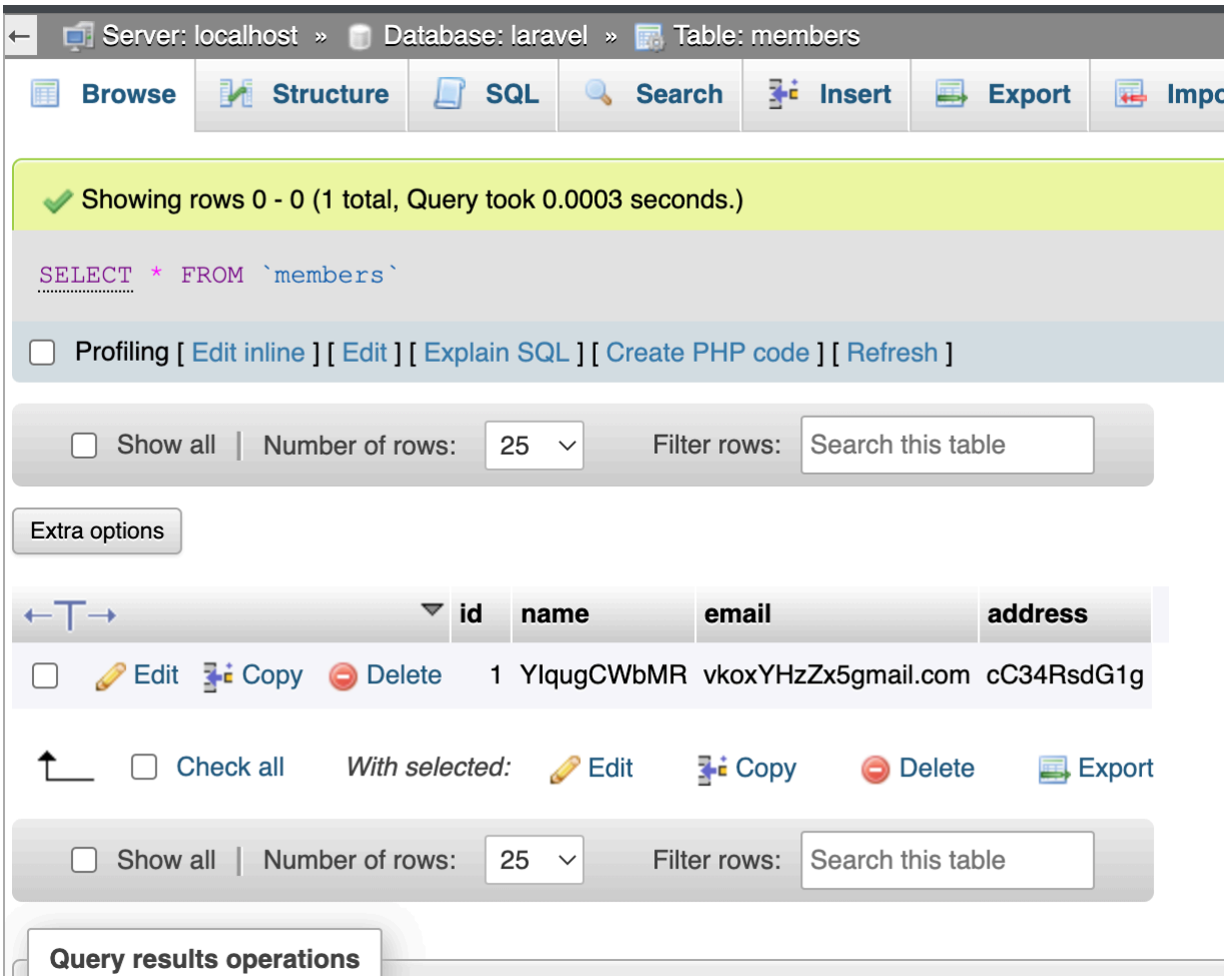
```
public function run()
    {
        //
        DB::table('members')->insert([

'name'=>Str::random(10),

'email'=>Str::random(10) . 'gmail.com',

'address'=>Str::random(10)
        ]);
    }
```

After which we can go back to the CLI and run the command:

➜ test-app php artisan db:seed --class=membersSeeder

After that your data should be seeded.

```
test-app php artisan db:seed --class=membersSeeder

  INFO  Seeding database.

test-app
```



If you wanted to make it run multiple times you could run something like this:

```
// Use a loop to insert multiple
records
        for ($i = 0; $i < 20; $i++)
{
            DB::table('members')-
>insert([
                'name' =>
Str::random(10),
                'email' =>
Str::random(10) . 'gmail.com',
                'address' =>
Str::random(10)
            ]);
        }
    }
}
```

Migrations

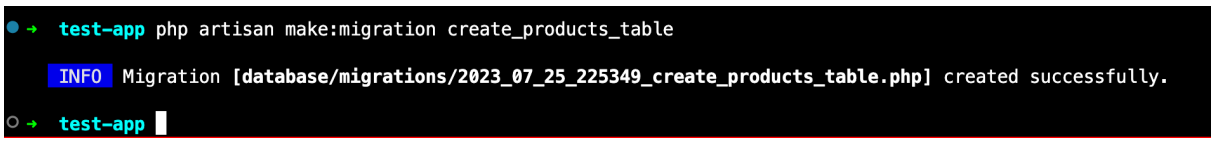
Migration in Laravel allows you to create/ manage/delete tables in your database, it also allows you to version control the DB as well. So that when you have a team, all working on the

same project, you can avoid conflict changes in the DB. It's a convenient way to interact with the database without having to manually go into phpMyAdmin to do things.

Migrations are written in PHP and stored in the database/migrations directory.

We can use this command to make a migration:

```
php artisan make:migration create_products_table
```



This will make a php file (not a db table yet). Load the new file.

We can add the schema of the table we want:

```
<?php

use
Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

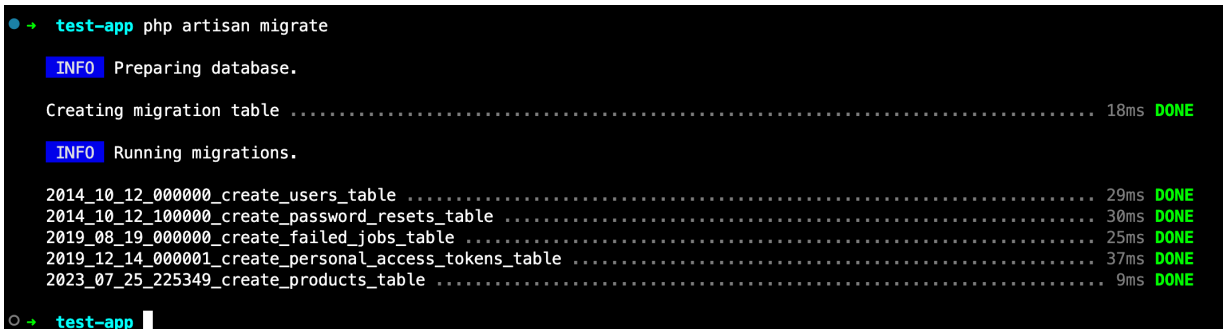
```php
class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('products', function
(Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description');
            $table->decimal('price', 10,
2); // Decimal with 10 digits and 2 decimal
places
            $table->integer('stock')-
>default(0);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('products');
```

```
        }
}
```

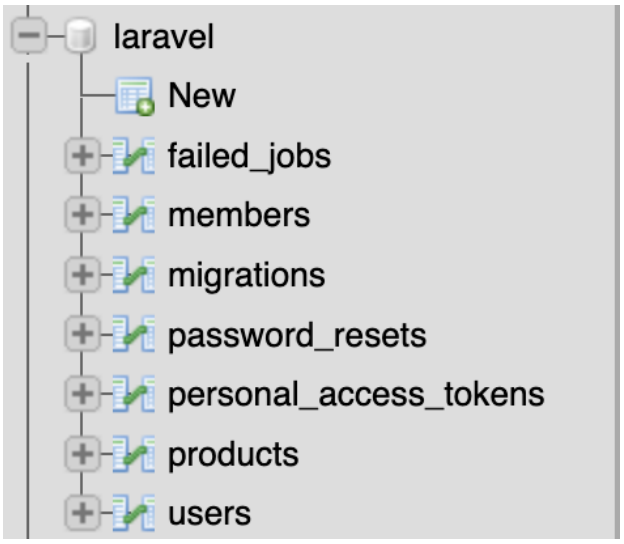
Save the file.

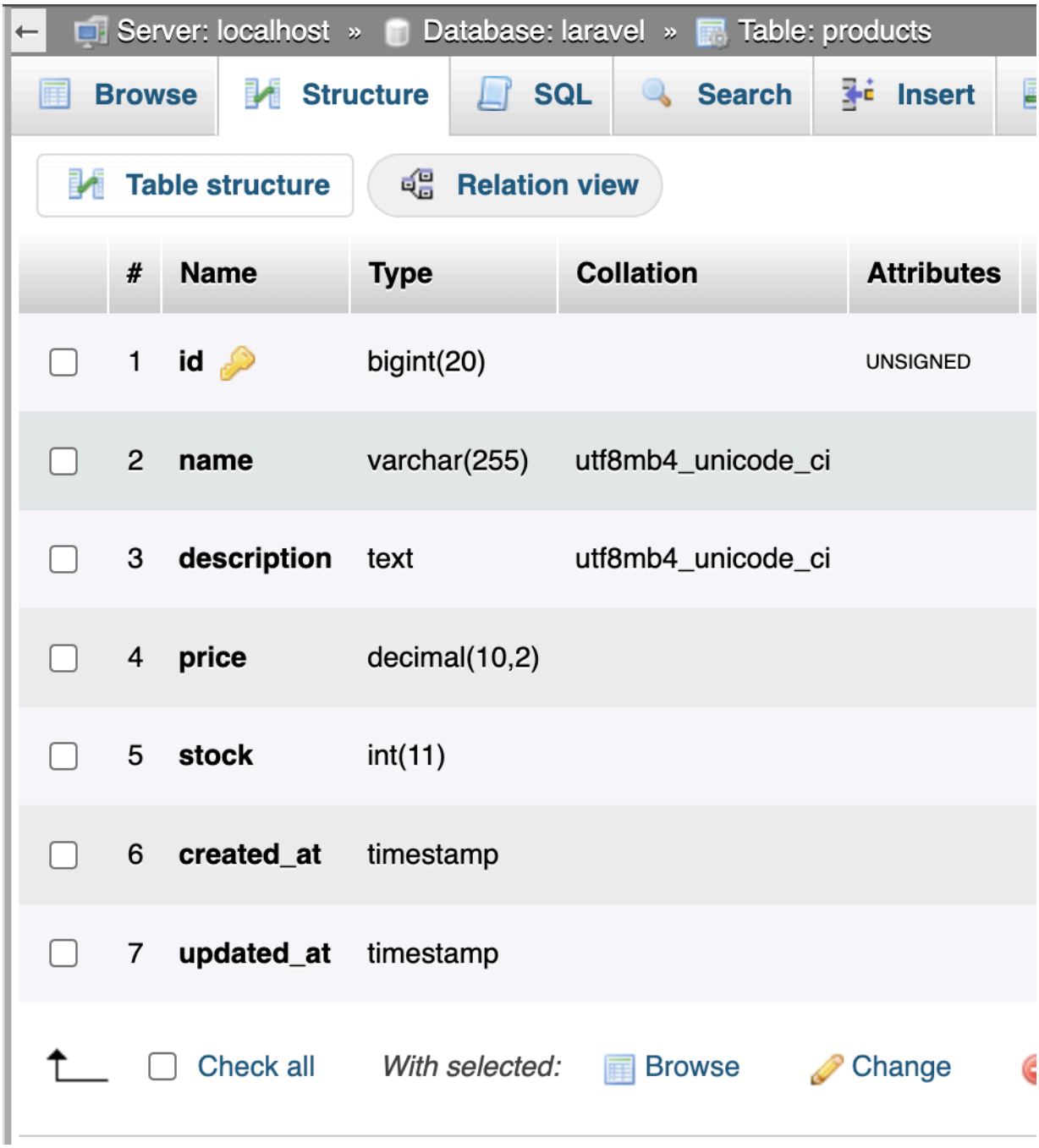
To run it you can use the command:

php artisan migrate



In this case I had some other migrations in there so they all ran.


laravel
New
failed_jobs
members
migrations
password_resets
personal_access_tokens
products
users



Additionally…

If you ever need to rollback the migration, you can use:

```
php artisan migrate:rollback
```

If you ever need to reset the DB, you can use:

```
php artisan migrate:reset
```