F1 🡪 file keresése VS code-ban!

Post model létrehozása

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan make:model Post -m

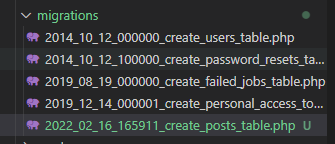
Model created successfully.

Created Migration: 2022\_02\_16\_165911\_create\_posts\_table

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216>



AB tábla:



public function up()

    {

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

            $table->id();

            $table->string("title");

            $table->text("description");

            $table->string("creator\_name");

            $table->string("category");

            $table->integer("rating");

            $table->boolean("active");

            $table->timestamps();

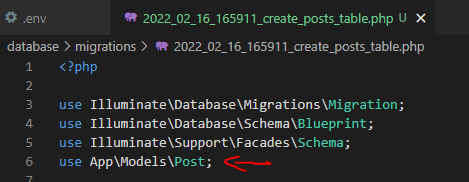
        });

    }

Migráció: php artisan migrate

Ha további adatra szükség van: rollback

Hivatkozunk a Post táblára:



Ezután létrehozzuk a Post:create fv-t:

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

use App\Models\Post;

class CreatePostsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->id();

            $table->string("title");

            $table->text("description");

            $table->string("creator\_name");

            $table->string("category");

            $table->integer("rating");

            $table->boolean("active");

            $table->timestamps();

        });

        Post::create([

            "title" => 'Title',

            "description" => 'Description',

            "creator\_name" => 'Creator Name',

            "category"=> 'general',

            "rating" => 5,

            "active" => true,

        ]); //asszociatív tömb, amely tart. az oszlopnevet és a hozzátart. értéket

        $post = new Post();

        $post -> title ='';

        $post -> description ='';

        $post -> save();

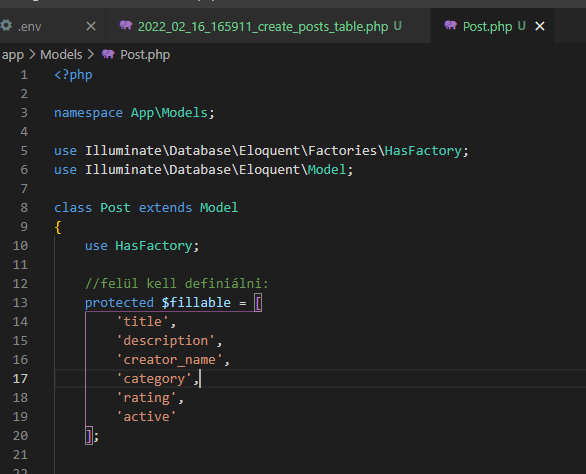
    }

Ha további Post::create sorokat írunk be, és php artisan migrate-t futtatunk, hibát ad:

Mass Assignment Exception

A Model-ben kell engedélyezni:

Megengedjük a tömeges adathozzárendelést



Ha beragad a tábla, manuálisan kell törölni MySQL-ben

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

use App\Models\Post;

class CreatePostsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->id();

            $table->string("title");

            $table->text("description");

            $table->string("creator\_name");

            $table->string("category");

            $table->integer("rating");

            $table->boolean("active");

            $table->timestamps();

        });

        Post::create([

            "title" => 'Title1',

            "description" => 'Description1',

            "creator\_name" => 'Creator Name1',

            "category"=> 'general1',

            "rating" => 5,

            "active" => true,

        ]); //asszociatív tömb, amely tart. az oszlopnevet és a hozzátart. értéket

        Post::create([

            "title" => 'Title3',

            "description" => 'Description2',

            "creator\_name" => 'Creator Name2',

            "category"=> 'general2',

            "rating" => 4,

            "active" => true,

        ]);

        Post::create([

            'title' => 'Title3',

            'description' => 'Description3',

            'creator\_name' => 'Creator Name3',

            'category'=> 'general3',

            'rating' => 3,

            'active' => true,

        ]);

        // $post = new Post();

        // $post -> title ='';

        // $post -> description ='';

        // $post -> creator\_name ='';

        // $post -> category ='';

        // $post -> rating ='';

        // $post -> save();

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

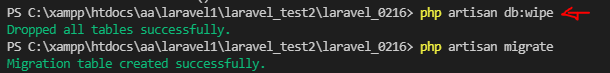
    {

        Schema::dropIfExists('posts');

    }

}

droppolja az összes táblát (wipe):



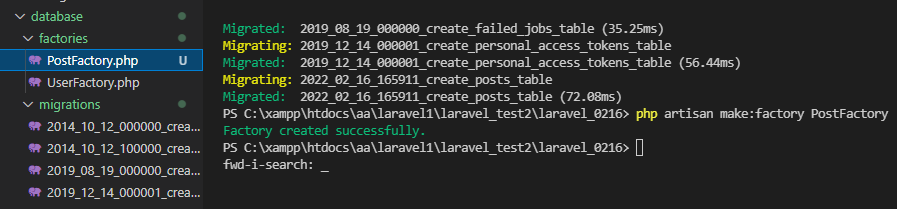
Seederek / Factoryk

Factoryk: az egyes táblákat felépítik

Seederek: A factoryban hányat kell létrehozni; a factory a seeder parancsára annyit hoz létre, amennyire szükség van

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan make:factory PostFactory

Factory created successfully.



<https://laravel.com/docs/9.x/database-testing#generating-factories>

A Definition fv-t írjuk felül.

Assz. tömb lesz, a kulcs mindig az attr.

Utána meg kell adni, milyen adatok kerülnek oda 🡪 faker library: mindenféle fake adatot tud generálni

Számkombinációkat ill. regexet is tud.

Minta: <https://laravel.com/docs/9.x/database-testing#generating-factories> 🡪 CTRL+F 🡪 „definition”

<https://github.com/fzaninotto/Faker>

<https://github.com/FakerPHP/fakerphp.github.io>

public function definition()

    //milyen adatok kellenek nekünk (az egyes attribútumainkhoz milyen adat lesz szükséges)

    {

        //this: class-ra vonatkozik, annak attr-ait lehet kikérni

        //faker class fv-eit el tudjuk érni

        return [

            'title' => $this->faker->sentence(), //mondatot rak bele

            'description' => $this->faker->text(),

            'creator\_name' =>  $this->faker->name(),

            'category' =>  $this->faker->randomElement(['a','b','c','d']),

            'rating' => $this->faker->numberBetween(1,5), //randomDigit() is jó lehet

            'active' => $this->faker->boolean(),

        ];

    }

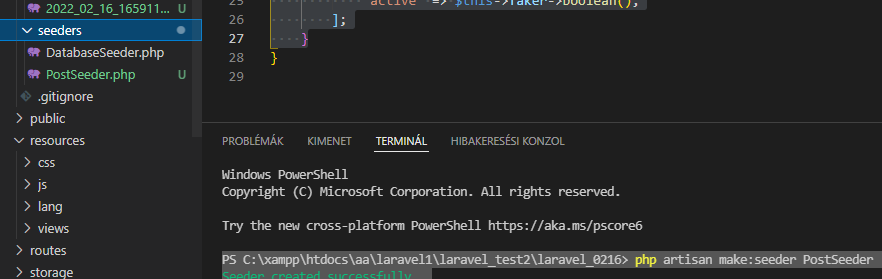
Idegen kulcs esetén (pl. user ID) a User modellemet el tudnám érni; kikérem az összeset, csak neki van egy random fv-e, amelyet idegen kulcsként be tud húzni: User::all()-> random()

pl. ha egy post egy user-hez tartozik. Ez bizt. az idegen kulcs helyes működését.

A factorykat a Seederek menedzselik.

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan make:seeder PostSeeder

Seeder created successfully.



Nem a Factory-t használjuk.

A Modelnek van HasFactory 🡪 ez engedélyez egy plussz fv-t, amelyet tudunk használni

A Modelt használjuk, amelynek lesz egy factory fv-e

class PostSeeder extends Seeder

{

    /\*\*

     \* Run the database seeds.

     \*

     \* @return void

     \*/

    public function run()

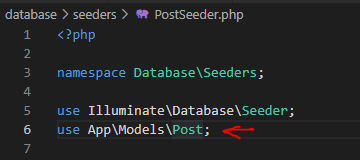
    {

        //posthoz létrehozott factoryból hozzon létre 50 darabot:

        Post::factory()->count(50)->create();

    }

}



Specifikus seeder futtatása

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan db:seed --class=PostSeeder

Database seeding completed successfully.

DatabaseSeederbe lehet betenni a futtatni kívánt seedereket a $this-> call után.

class DatabaseSeeder extends Seeder

{

    /\*\*

     \* Seed the application's database.

     \*

     \* @return void

     \*/

    public function run()

    {

        // \App\Models\User::factory(10)->create();

        $this->call([

            PostSeeder::class

        ]);

    }

}

Futtatás:

összes migr. visszavonása és felépítése:

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan migrate:refresh

seedelés:

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan db:seed

Az AB-ban rengeteg adat létrejött!

Összes migráció + összes seeder lefuttatása

db wipe migrate

PS C:\xampp\htdocs\aa\laravel1\laravel\_test2\laravel\_0216> php artisan migrate:refresh --seed

Magyarra állítás:

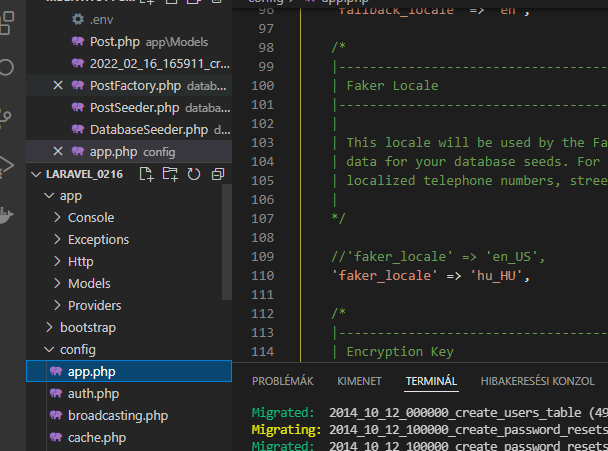
app.php-ba: 'faker\_locale' => 'hu\_HU',

<https://fakerphp.github.io/>

You may also seed your database using the migrate:refresh command, which will also rollback and re-run all of your migrations:

php artisan migrate:refresh –seed

Magyar locale-ra váltás:



AB tartalmának törlése és újraépítése:

