WEB разработка

Faker
DB Seeding

Съдържание

DbSeeding
Faker

Database Seeding

Database Seeding

Laravel разполага с класове/seed classes/, предназначени за пълнене на базата данни с тестови данни /seeding/.

Тези класове се запазват в database/seeds директорията.

Seed-класовете могат да приемат произволни имена, но е добре да следвате някаква конвенция като например - UsersTableSeeder и т.н.

По подразбиране класът DatabaseSeeder е дефиниран.

От този клас използвате метода call за да стартирате другите seed класове. Така контролирате реда на пълнене на базата данни.

Дефиниране на seed класове

Дефиниране на seed класове

За да създадете нов seeder клас, изпълнете командата

php artisan make:seeder

Всички seeder класове след създаването им се разполагат в директорията database/seeds

php artisan make:seeder UsersTableSeeder

Дефиниране на seed класове

seeder класът по подразбиране има само един метод - run.

Този метод се изпълнява при изпълнение на артизан командата db:seed

С метода run, може да въвеждате в БД данните, които искате.

Пример. Да променим класът DatabaseSeeder class и да добавим insert заявка към метода run.

```
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('users')->insert([
            'name' => str_random(10),
            'email' => str_random(10).'@gmail.com',
            'password' => bcrypt('secret'),
        ]);
    }
}
```

Използване на Model Factories

Using Model Factories

Описването на атрибутите на всеки seed модел е трудоемки.

Вместо това използвайте model factories, с които удобно се генерират огромни количества от записи в базата данни.

Разгледайте документацията - model factory documentation.

След като дефинирате нужните factory модели, с factory helper функцията може да въведете записите в базата данни.

Например - да създадем 50 потребителя и за всеки потребител да добавим по 50 поста.

```
public function run()
{
    factory(App\User::class, 50)->create()->each(function ($u) {
        $u->posts()->save(factory(App\Post::class)->make());
    });
}
```

Изпълнение на Seeder класоете

Running Seeders

След като сте описали нужните seeder класове, артизан командата db:seed записва данните в БД.
По подразбиране db:seed изпълнява класът DatabaseSeeder, чрез който може да изпълните други seed класове.
Може да използвате и --class опцията, за да конкретизирате конкретен seeder клас, който да изпълните индивидуално.

php artisan db:seed //изпълнява DatabaseSeeder класът
и
php artisan db:seed --class=UsersTableSeeder //изпълнява
UserTableSeeder

Faker

Faker

Model Factories

Когато създавате тестови данни /seed data/ може да използвате Eloquent или синтаксиса на Laravel за генериране на заявки /query builder/.
С въвеждането на model factory в Ларавел имате и друга възможност за генериране на тестови данни.

Faker

Отворете database/factories/ModelFactory.php в него има дефиниран по подразбиране -

```
$factory->define(App\User::class, function (Faker\Generator $faker) {
        return [
            'name' => $faker->name,
            'email' => $faker->email,
            'password' => bcrypt(str_random(10)),
            'remember_token' => str_random(10),
        ];
    });
```

Faker

1. “App\User::class” моделът е подаден като първи параметър
2. Следва callback функция, дфинираща данните, които са предназначени за колоните на таблицата.
3. Тази callback функция вкарва в БД т.нар.Faker данни.

Faker е мощна PHP библиотека, генерираща данни от различен тип.

Някои примери

- $fake->name – “John Smith”
- $faker->email – “tkshlerin@collins.com”

Обърнете внимание:
Ако приложението ви ще изпаща истински email-и, използвайте **$faker->safeEmail** вместо просто ->**email**.
->email може да генерира реален адрес, докато safeEmail използва example.org, който е запазен от RFC2606 за тестови цели.

Faker

Създаване на нов factory за модела Issues.

```
$factory->define(App\Issues::class, function (Faker\Generator $faker) {
        return [
            'subject' => $faker->sentence(5),
            'description' => $faker->text(),
        ];
    });
```

С тази дефиниция създаваме изречение с 5 думи за колоната subject и безчислен текст за description.

Да се върнем към seed класовете и да използваме дефинираните factories за генериане на данни.

Faker

В класът UserTableSeeder променяме run метода:

```
public function run()
    {
      factory(App\User::class, 2)->create()->each(function($u) {
        $u->issues()->save(factory(App\Issues::class)->make());
      });
    }
```

factory(App\User::class, 2)->create() - създава 2 обекта от клас User със съответните свойства и ги записва в базата данни.

Faker

След изпълнение на командата

$ php artisan migrate --seed

```
/*Migrated: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrated: 2015_10_03_141020_create_issues_table
Seeded: UserTableSeeder*/
```

В базата данни са създадени таблиците, които са запълнени с тестови данни генерирани от model factory/

Повече информация на https://scotch.io/tutorials/generate-dummy-laravel-data-with-model-factories