Introduction

Laravel có một phương thức đơn giản để seed database với dữ liệu test sử dụng các seed class. Tất cả các seed class được lưu trong thư mục `database/seeds`. Các seed class có thể được đặt tên tuỳ ý, nhưng tốt nhất là nên đặt theo một nguyên tắc dễ nhận biết, ví dụ như là `UsersTableSeeder`, ... Mặc định, một class `DatabaseSeeder` được định nghĩa sẵn cho bạn. Từ class này, bạn có thể sử dụng phương thức `call` để gọi tới các seed class khác, cho phép bạn điều khiển thứ tự seed dữ liệu vào trong database.

Writing Seeders

Để sinh ra một seeder, bạn có thể gọi lệnh make:seeder Artisan command. Tất cả các seeder được sinh ra bởi framework sẽ được đặt trong thư mục database/seeds:

```
php artisan make:seeder UsersTableSeeder
```

Một seeder class chỉ chứa một phương thức mặc định là: `run`. Phương thức này được gọi khi mà `db:seed` Artisan command được thực thi. Bên trong `run`, bạn có thể chèn thêm dữ liệu vào database như bạn muốn. Bạn cũng có thể sử dụng query builder để thêm dữ liệu thủ công hoặc sử dụng Eloquent model factories.

Cùng thử làm một ví dụ, hãy cùng nhau sửa class DatabaseSeeder có sẵn sau khi cài đặt Laravel. Cùng nhau thêm một mệnh đề chèn dữ liệu bên trong hàm run:

```
<?php

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'),
]);
}
}
```

Sử dụng Model Factories

Dĩ nhiên là việc khai báo các thuộc tính cho các model seed một cách thủ công sẽ khá vướng víu. Thay vì thế, bạn có thể sử dụng model factories để sinh ra lượng lớn các dữ liệu vào trong database một cách tiện lợi. Đầu tiên, hãy xem tài liệu về model factory để biết cách khai báo các factories. Sau đó, bạn có thể sử dụng hàm `factory` để thêm dữ liệu vào trong database:
Ví dụ, hãy cùng nhau tạo 50 user và liên kết mối quan hệ cho mỗi user:

```
/**
 * Run the database seeds.
 *
 * @return void
 */
public function run()
{
    factory(App\User::class, 50)->create()->each(function($u) {
        $u->posts()->save(factory(App\Post::class)->make());
    });
}
```

Gọi các Seeders bổ sung

Bên trong class `DatabaseSeeder`, bạn có thể sử dụng hàm `call` để thực thi các seed class bổ sung. Sử dụng hàm `call` cho phép bạn phân tách cấu trúc seed vào database thành nhiều file, vì thế, sẽ không có seed class nào trở nên quá lớn. Đơn giản là chỉ cần tên của seeder class mà bạn muốn thực thi:

```
/**
 * Run the database seeds.
 *
 * @return void
 */
public function run()
{
    $this->call(UsersTableSeeder::class);
    $this->call(PostsTableSeeder::class);
    $this->call(CommentsTableSeeder::class);
}
```

Thực thi Seeders

Khi bạn đã viết các seeder class, bạn có thể sử dụng câu lệnh Artisan `db:seed` để seed vào database. Mặc định, câu lệnh `db:seed` thực thi `DatabaseSeeder`, mà các bạn có thể sử dụng để gọi các seed class khác. Tuy nhiên, bạn cũng có thể sử dụng tuỳ chọn `--option` để chỉ định thực hiện một seed class nào đó:

```
php artisan db:seed

php artisan db:seed --class=UsersTableSeeder
```

Bạn cũng có thể seed database sử dụng câu lệnh `migrate:refresh`, ngoài ra có thể sử dụng rollback và thực thi lại tất cả các migrations. Câu lệnh này cũng khá hữu ích trong việc thiết lập lại toàn bộ cấu trúc database:

```
php artisan migrate:refresh --seed
```