Seeder & Factory

Introduction

Laravel inclut la possibilité de peupler la base de données de manière beaucoup plus "propre" mais surtout beaucoup plus rapide et réutilisable grâce aux **Seeders** et aux **Factories.** En effet, les Seeders sont utilisés pour *insérer* des données dans la base de données, tandis que les Factories sont utilisées pour *générer des données* pour remplir vos tables. Les deux outils sont utiles pour les tests et les développements.

Seeders :

Il est possible de n'utiliser que les **Seeders** pour peupler notre base de données de manière très basique.

Pour créer un nouveau seeder pour notre table **stagiaires** lancez la commande :

```
php artisan make:seeder StagiairesTableSeeder
```

Ce nouveau fichier se trouvera dans le dossier **/database/seeders**

Exemples des seeders :

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
class StagiairesTableSeeder extends Seeder
{
        public function run()       {
        DB::table('stagiaires')->insert([
            'nom'=>'Alaoui',
            'prenom'=>'Aya',
            'age'=>19
        ]);          }
}
```

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
class StagiairesTableSeeder extends Seeder
{
        public function run()      {
         DB::table('stagiaires')->insert([
            'nom'   => Str::random(20),
            'prenom' => Str::random(20),
            'age' => rand(18,30),
        ]);          }
}
```

Pour que notre seeder **StagiairesTableSeeder** s'execute, on peut l'appeler via la méthode **call** à l'intérieure de la méthode **run()** de notre Classe **DatabaseSeeder** .

```php
use Illuminate\Database\Seeder;
class DatabaseSeeder extends Seeder {
    public function run() {
        $this->call([
            StagiairesTableSeeder::class,
            ]);
    } }
```

Pour finir, on lance la commande :

```
php artisan db:seed
```

Ainsi, dans votre base de données un nouvel enregistrement est inséré dans la table stagiaires.

Remarque 1 : Par défaut, la commande db:seed exécute la classe Database\Seeders\DatabaseSeeder. Cependant, on peut utiliser l'option –class pour spécifier une classe de départ à exécuter individuellement :

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

Remarque 2 : Vous pouvez également amorcer votre base de données en utilisant la commande migrate:fresh en combinaison avec l'option –seed , qui supprimera toutes les tables et réexécutera toutes vos migrations. Cette commande est utile pour reconstruire complètement votre base de données. L'option –seeder peut être utilisée pour spécifier un seeder spécifique à exécuter :

```
php artisan migrate:fresh -seed
php artisan migrate:fresh --seed -seeder=StagiaireSeeder
```

Les Factories

Les Factories sont des classes qui peuvent générer des données aléatoires. Ils peuvent être utilisés pour remplir la base de données avec des données de test plus complètes et plus précises que celles générées par les Seeders.

Exemple : nous avons déjà une migration qui crée la table user. Aussi, nous avons un model User et une factorie créée UserFactory .

Ainsi, on peut créer un seeder qui inclut la méthode run suivante :

```
use App\Models\User;
...
public function run()
{
User::factory()
->count(50)
->create();
}
```

Testez !

Remarque importante 1 :

Faker est une bibliothèque PHP utilisée pour générer des données aléatoires. Elle est souvent utilisée avec Laravel pour générer des données fictives pour les tests ou pour remplir la base de données avec des données de test. Faker peut générer des noms, des adresses, des numéros de téléphone, des adresses e-mail, des dates, des textes aléatoires, etc. Il est très utile pour les développeurs qui veulent tester leur application avec des données réelles sans avoir à entrer manuellement chaque enregistrement.

Etapes à suivre

1- Créer un modèle via la commane : php artisan make:model Stagiaire

2- Ajouter au modèle le code suivant :

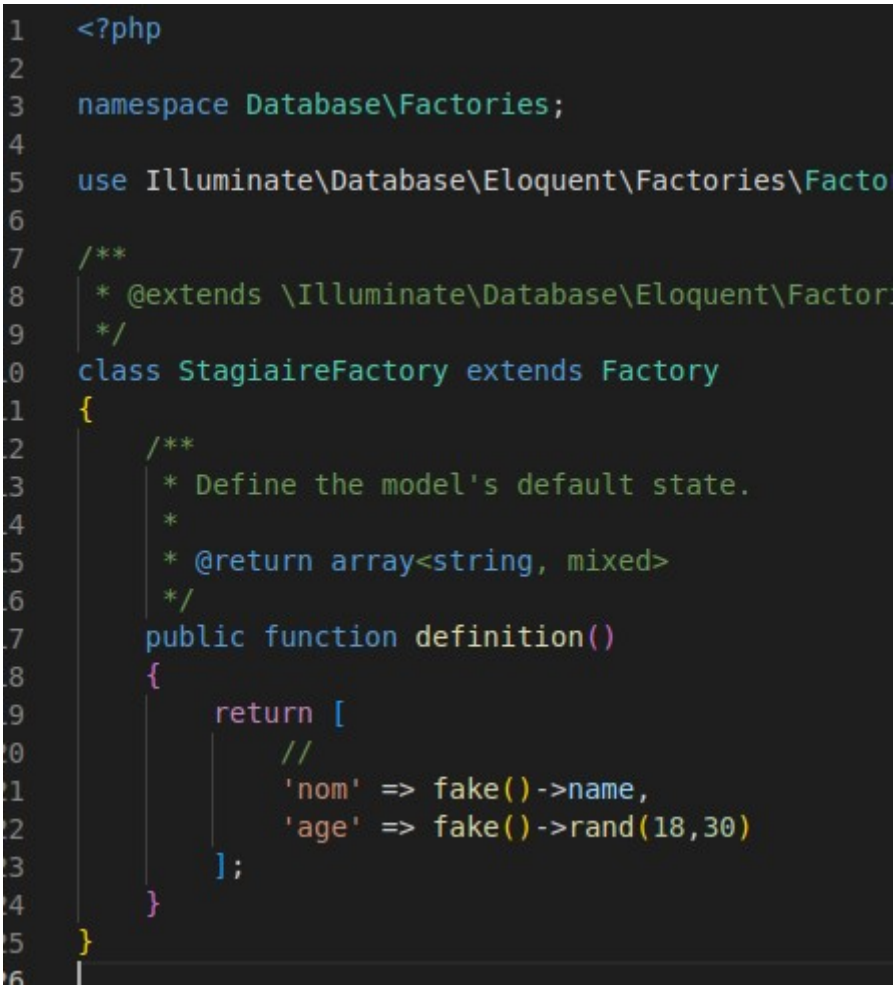
```php
use Illuminate\Database\Eloquent\Model

class Stagiaire extends Model
{
    use HasFactory;
protected $fillable = ['nom', 'age'];
}
```

3- Créer une factorie nommée StagiaireFactory via la commande :

php artisan make : factory Stagiairefactory

4- Modifier la factorie pour avoir :



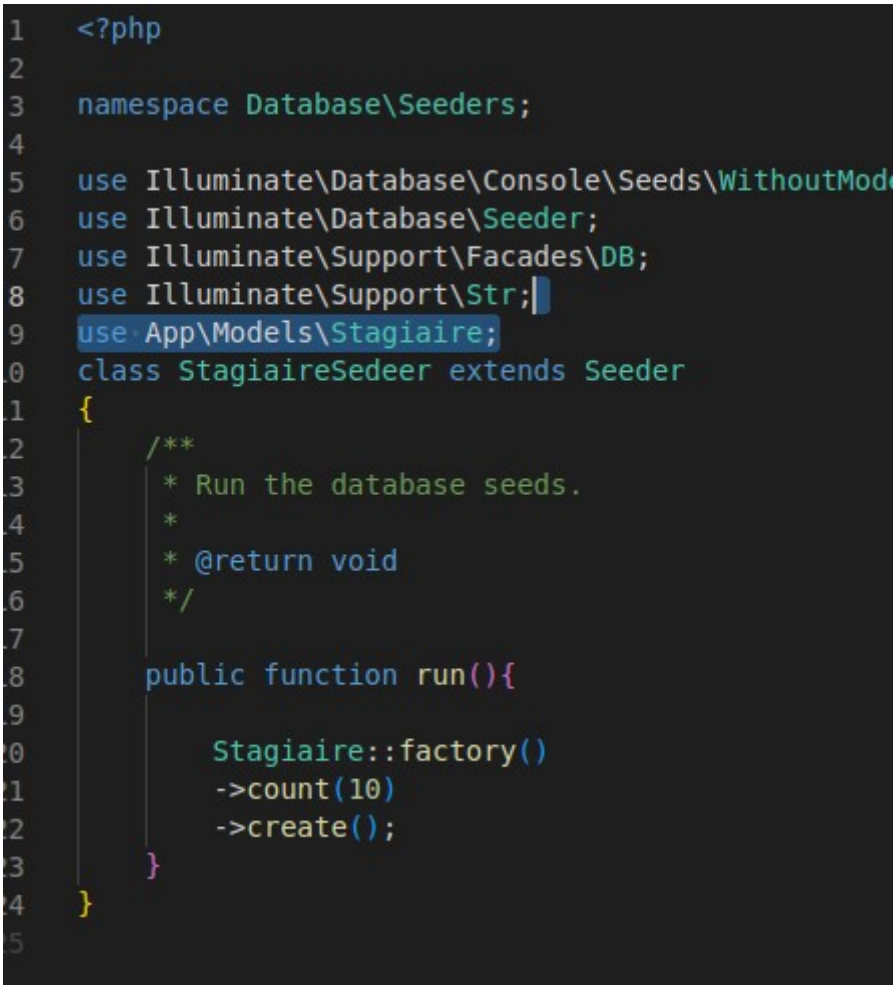
```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Facto

/**
 * @extends \Illuminate\Database\Eloquent\Factor
 */
class StagiaireFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition()
    {
        return [
            //
            'nom' => fake()->name,
            'age' => fake()->rand(18,30)
        ];
    }
}
```

5- créer un seeder Stagiaireseeder qui contient la méthode run :



```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutMod
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
use App\Models\Stagiaire;
class StagiaireSedeer extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */

    public function run(){

        Stagiaire::factory()
        ->count(10)
        ->create();
    }
}
```

Testez !

TP :

1. Créez une nouvelle application Laravel 9
2. Créez un modèle "Post" avec un titre, un contenu et une date de création. Vous pouvez utiliser la commande `php artisan make:model Post -m` pour générer automatiquement le modèle et la migration.
3. Créez une table "posts" dans votre base de données en exécutant la migration.
4. Créez un factory pour le modèle "Post" pour générer des données aléatoires pour la table "posts".
5. Créez un seeder pour alimenter la table "posts" avec des données fictives en utilisant le factory "Post".
6. Exécutez le seeder pour peupler la table "posts" avec des données et vérifiez les résultats dans la base de données.
7. Créez une vue pour afficher tous les posts et testez-la.