Développer en back-end

Seeders et Factories

Réalisé par M. Hamid Belyazidi

Développer en back-end

Introduction

A. Découvrir le Framework PHP Laravel

1. Découvrir les notions fondamentales des Frameworks PHP
2. Préparer l'environnement de Laravel

B. Programmer avec Laravel

1. Connaître les fondements du modèle MVC Laravel
2. Maîtriser le Framework Laravel

C. Approfondir la programmation Laravel

1. Gérer la sécurité
2. Interagir avec la base de données
3. Manipuler l'ORM Eloquent
4. Prendre en charge les tests

D. Administrer un site à l'aide d'un CMS

1. Manipuler les éléments essentiels d'un CMS
2. Personnaliser graphiquement un site à l'aide d'un CMS
3. Manipuler les outils avancés d'un CMS

Conclusion

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

- ✓Mise en route
- ✓Générateur de requêtes (Query Builder)
- ✓Pagination de la base de données
- ✓Gestion de migration (génération, structure, exécution, manipulation des tables, colonnes, indexeset events)
- **✓Création de Seeders (utilisation des modèles factories, appels de seeders additionnels,**
- ✓désactivation d'événements de modèles)
- ✓Insertion des données d'un formulaire dans une base de données
- ✓Utilisation de Redis

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

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.**

Nous allons voir que les deux travaillent ensemble et qu’ils portent bien leurs noms.

Surtout si les données à rentrer deviennent de plus en plus complexes et que nous sommes amenés à en enregistrer fréquemment lors de la phase de développement de notre application pour divers test.

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les 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/seeds**

Pour que notre seeder **StagiairesTableSeeder** s'execute nous devons utiliser la méthode **call** dans la méthode **run()** de notre Classe **DatabaseSeeder** .

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

.

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

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
class StagiairesTableSeeder extends Seeder
{
        public function run()
    {
        DB::table('stagiaires')->insert([
            'nom'=>'Barouni',
            'prenom'=>'Racha',
            'age'=>23
        ]);
        }
}
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

```
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),
        ]);
         }
}
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

Pour finir nous allons seeder la base de données mais avant cela assurer vous d'effacer correctement tous vos enregistrements déjà présents pour repartir d'une base vide :

```
php artisan migrate:fresh
```

Assurez-vous de n'avoir aucune donnée dans votre base et lancez la commande de seeding :

```
php artisan db:seed
```

Si vous avez bien suivi vous devriez voir dans votre base de données un nouvel enregistrement dans la table stagiaires.

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

Vous pouvez exécuter la commande artisan db:seed pour amorcer votre base de données. Par défaut, la commande db:seed exécute la classe Database\Seeders\DatabaseSeeder , qui peut à son tour invoquer d'autres classes de départ. Cependant, vous pouvez utiliser l'option –class pour spécifier une classe de départ à exécuter individuellement :

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

.

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Seeders

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
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Tinker

Toutes les applications Laravel incluent **Tinker** par défaut.

Vous pouvez installer Tinker à l'aide de Composer :

```
composer require laravel/tinker
```

Tinker vous permet d'interagir avec l'ensemble de votre application Laravel en ligne de commande :

```
PS F:\ISTA2022\TPSeeders> php artisan tinker
Psy Shell v0.11.10 (PHP 8.1.2 — cli) by Justin
Hileman

> DB::table('stagiaires')-
>insert(['nom'=>'Girari','prenom'=>'Nada','age'=>2
1])
= true
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Factories

Les Factories sont donc là pour nous permettre de créer des enregistrements en quantité et d'établir facilement diverses relations entre nos tables.

Nous allons donc créer le **StagiaireFactory**

```
php artisan make:factory StagiaireFactory
```

Vos Factories se situe dans le dossier **/database/factories.**

L'option **--model** de la commande permet de préciser de quel model il s'agit.

```
php artisan make:factory StagiaireFactory --
model=Stagiaire
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Factories

Nous allons donc créer Le model **Stagiaire** :

```
php artisan make:model Stagiaire
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

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

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Factories

Nous allons donc créer Le **StagiaireFactory:**

```
php artisan make:factory StagiaireFacory
```

```php
use Illuminate\Database\Eloquent\Factories\Factory;
\\Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Stagiaire>
class StagiaireFactory extends Factory
{
   public function definition()
    {
        return [
            'nom'=> fake()->firstName(40),
            'prenom'=>fake()->lastName(40),
            'age'=>fake()->numberBetween(18,30),
        ];
    }
}
```

C. Approfondir la programmation Laravel

2. Interagir avec la base de données

Seeders et Factories

Les Factories

La commande artisan pour remplir la base de donnée :

```
php artisan db:seed
```

DatabaseSeeder

On peu aussi ajouter des données directement dans la méthode run()

```
public function run()
    {
        \App\Models\Stagiaire::factory()->create([
            'nom' => 'Tahiri',
            'prenom' => 'Hassan',
            'age' => 24
        ]);
    }
```