OFPPT

Etablissement : CFMNTIOE
Filière : DD/Full-Stack
Formateur : Zakaria KADDARI

Module : Développement backend
<< Travaux pratiques >>

Projet Laravel

Première partie

Parties abordées dans cette série :

- La relation avec une base de données
- Configuration de la connexion avec la base de données
- Le modèle : création, utilisation
- Les migrations : création, utilisation
- Les opérations CRUD (Create, Read, Update, et Delete)
- L'authentification (login) avec Laravel Breeze
- Utilisation de Bootstrap dans un projet Laravel

Travail à faire

1. Ouvrir **phpmyadmin** et créer une base de données portant le nom : **larabase**
2. Dans votre fichier **.env**, mettez le nom de votre base de données dans l'attribut DB_DATABASE. Modifier le nom d'utilisateur et/ou le mot de passe en cas de besoin.

   Exemple de configuration :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=larabase
DB_USERNAME=root
DB_PASSWORD=
```

3. Installer **Laravel Breeze** avec commande ci-dessous

```
composer require laravel/breeze --dev
```

4. Compléter l'installation de Laravel Breeze avec les commandes ci-dessous. (lancer les commandes une par une)

```
php artisan breeze:install
php artisan migrate
npm install
npm run dev
```

5. Vérifier le bon fonctionnement de Laravel Breeze en créant un nouveau compte sur /**register** puis de se connecter avec le compte créé sur /**login**
6. Créer le modèle Client avec la commande ci-dessous :

```
php artisan make:model Client -m
```

   L'option **m** permet de créer la classe migration correspondante au modèle.

7. Ouvrir le fichier de migration nouvellement crée database/migrations/ et ajouter les deux colonnes suivantes : nom, prenom, email, adresse

   Exemple :

```php
public function up()
{
    Schema::create('clients', function (Blueprint $table) {
        $table->id();
        $table->string('nom');
        $table->string('prenom');
        $table->string('email');
        $table->string('adresse');
        $table->timestamps();
    });
}
```

8. Dans votre modèle nouvellement créé situé dans **app/Models/Client.php**, ajouter la ligne suivante :

```php
protected $fillable = ['nom', 'prenom', 'email', 'adresse'];
```

9. Exécuter la migration nouvellement créée avec la commande ci-dessous, ceci va vous créer la table **Client**

```
php artisan migrate
```

10. Lancer la commande ci-dessous, pour créer un contrôleur pour gérer votre modèle

```
php artisan make:controller ClientController -r
```

    L'option **-r** permet d'initialiser automatiquement les actions CRUD du modèle

11. Compléter l'action **index** pour afficher la liste des clients depuis la base de données
12. Compléter l'action **create** pour pouvoir ajouter un nouveau client
13. Compléter l'action **store** pour pouvoir enregistrer le client ajouté
14. Compléter l'action **show** pour pouvoir consulter un client
15. Compléter l'action **edit** pour pouvoir éditer un client
16. Compléter l'action **update** pour pouvoir enregistrer le client modifié
17. Compléter l'action **destroy** pour pouvoir supprimer un client
18. Dans votre fichier de routage, ajouter une route de type **Resource** pour créer automatiquement les différentes routes CRUD
19. Créer un nouveau dossier **client** à l'intérieur du dossier **resources/views**

20. Créer les vues **index.blade.php**, **create.blade.php**, **edit.blade.php**, et **show.blade.php** mettez-les à l'intérieur du dossier client
21. Dans la vue **index**, afficher la liste des clients dans un tableau HTML.
22. Coder la vue **create.blade.php**, mettez dedans un formulaire pour ajouter un client, mettre en place l'affichage des erreurs de validation et l'affichage du message de succès après l'ajout.
23. Coder la vue **edit.blade.php**, mettez dedans un formulaire pour modifier un produit, mettre en place l'affichage des erreurs de validation et l'affichage du message de succès après la modification d'un client.
24. Créer et coder la vue **show.blade.php** qui permet d'afficher les informations d'un client dans des balises **p**, ajouter un bouton dans le tableau d'affichage des clients vers cette vue.
25. Créer un **layout** et lié le avec vos vue.
26. Styler les différentes pages à l'aide de Bootstrap
27. Ajouter la confirmation avant la suppression d'un client en JavaScript
28. Ajouter la pagination à la liste des clients (voir la documentation)