Laravel 2 - Starting an MVC Project using Laravel

1 - Utilisation d'un modèle Bootstrap

1. Téléchargez un exemple de modèle Bootstrap gratuit sur le lien suivant: https://startbootstrap.com/previews/clean-blog/

> Bootstrap est un framework web frontal gratuit et open-source. Il contient des modèles de conception HTML et CSS pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface, ainsi que des extensions JavaScript en option. Contrairement à de nombreux frameworks Web antérieurs, il ne concerne que le développement frontal.

Vous pouvez télécharger bootstrap sur: https://getbootstrap.com/

2. Décompressez le fichier téléchargé

3. Ouvrez le dossier et copiez les fichiers de modèle dans xampp / htdocs / blog / public

---

2 - Travailler avec Laravel View

Les vues contiennent le code HTML servi par votre application et séparent votre logique contrôleur / application de votre logique de présentation

Les vues sont stockées dans le répertoire **resources/views**.

Toutes les vues doivent être enregistrées avec la **suffixe blade**.

c'est-à-dire qu'un fichier nommé "home.php" doit être enregistré en tant que **"home.blade.php**"

Ouvrez le fichier public / index.html copiez tout le contenu et collez-le dans home.blade.php

---

3 - Routes et contrôleurs

Toutes les routes Laravel sont définies dans vos fichiers de route, qui se trouvent dans le répertoire routes.

Ces fichiers sont automatiquement chargés par le framework.

Le fichier **routes/web.php** définit les routes qui sont destinées à votre interface Web.

Ces routes sont affectées au groupe de middleware Web, qui fournit des fonctionnalités telles que l'état de session et la protection CSRF

Ouvrez le fichier web.php et ajoutez le code suivant:

```
Route::get('/home', function () {
    return view('home');
```

```
});
```

---

4 - Passer la route à travers le contrôleur

Au lieu de définir toute votre logique de traitement des demandes comme des fermetures dans les fichiers de routage, vous souhaiterez peut-être organiser ce comportement à l'aide de classes Controller.

Les contrôleurs peuvent regrouper la logique de traitement des demandes associée dans une seule classe.

Les contrôleurs sont stockés dans le répertoire **app/Http/Controllers** .

Into the file Route/web.php change the previous route with function to a route with a controller, code bellow:

Laravel Version < 8

```
Route::get('/home', 'BlogController@index');
```

Laravel Version >=8

```
use App\Http\Controllers\BlogController;

Route::get('/home', [BlogController::class,'index'])
```

Aide-mémoire Laravel 8: https://learninglaravel.net/cheatsheet/

Dans le répertoire du contrôleur, créez un fichier nommé BlogController.php

```
namespace App\Http\Controllers;

class BlogController extends Controller

    {

    public function index(){

    return view('home');

    }

}
```

Ou vous pouvez créer le modèle de contrôleur en utilisant Php Artisan

```
php artisan make:controller [Nom]Controller
```

---

5 - Création d'une vue principale

La vue principale stockera dans un seul fichier les codes, qui sont répétés dans de nombreux autres fichiers comme CSS, JS, en-tête, pied de page, menu, etc.

Dans le répertoire **resource / views** créez un fichier nommé **master.blade.php**

Modifiez les références CSS et JS du dossier public :

de:

```
<link href="css/styles.css" rel="stylesheet" />
```

à:

```
<link href="{{asset('css/styles.css')}}" rel="stylesheet" />
```

1. Découpez le code <Html> <Head> ... </head> de Home.blade.php et collez-le dans master.blade.php

2. Entre les balises <title>, créez une variable Laravel @yield ('title'). Cela vous permettra de changer le titre de la page dans les autres pages.

Par exemple: <title> @yield('title')</title>

3. Insérez le code suivant en haut de la page home.blade.php:

```
@extends('master')
```

Cette commande chargera la vue master.blade.php

4. Après cela, tapez:

```
@section('title', 'Name of the page')
```

Cette commande utilisera le titre de variable créé précédemment et attribuera le nom de la page souhaitée.

5. Coupez les balises <body> et <nav> Jusqu'à </ nav> de home.blade.php et collez-les dans master.blade.php

6.Insérez le code suivant après la balise </ nav>:

```
@yield('content')
```

Il réservera cet espace au contenu principal de la page.

7. Dans le fichier home.blade.php, remplacez les codes coupés par:

```
@section ('content')
```

8. Couper le <footer> .... Jusqu'au code </ html> de home.blade.php et collez-le dans master.blade.php après @yield ('content')

9. Dans home.blade.php, remplacez les codes coupés par:

```
@endsection
```

---

6 - Exercise

Maintenant, pour vous entraîner, créez les vues about.blade.php (about.html), contact.blade.php (contact.html) et post.blade.php (post.html) à l'aide de master.blade.php pour remplacer les codes répétés.

Créez les routes et les contrôleurs pour accéder à ces pages.

---

7 - Laravel Jeton (Token)

Laravel valide la demande de publication à l'aide d'un jeton.

Laravel facilite la protection de votre application contre les attaques de falsification de requêtes intersites (CSRF). Les falsifications de requêtes intersites sont un type d'exploit malveillant par lequel des commandes non autorisées sont exécutées au nom d'un utilisateur authentifié. Laravel génère automatiquement un «token» CSRF pour chaque session utilisateur active gérée par l'application. Ce jeton est utilisé pour vérifier que l'utilisateur authentifié est celui qui fait réellement les demandes à l'application.

Chaque fois que vous définissez un formulaire HTML dans votre application, vous devez inclure un champ de jeton CSRF masqué dans le formulaire afin que le middleware de protection CSRF puisse valider la demande.

Référence: https://laravel.com/docs/5.8/csrf

Le jeton doit être inséré à l'intérieur des balises de formulaire html.

```
<form action = "insert" method = "post">
    <input type = "hidden" name = "_token" value = "{{csrf_token()}}" />
```

or

{{csrf_field()}}

or
@csrf

---

8 - Route - Post

Créez une nouvelle Route en utilisant la méthode post.

```
Route::post('/contact-form', [BlogController::class, 'contactForm']);
```

Dans le contrôleur, utilisez la bibliothèque "request" pour recevoir le tableau de publication, puis créez une variable à l'aide du résultat "request" pour renvoyer les données à la vue.

```
public function contactForm(Request $request){

  return view('blog-contact', ['data'=> $request]);

}
```

Afficher le résultat dans la vue à l'aide d'une condition isset variable.

```
@if(isset($data))

                Name: {{ $data->name }}

@else

 ... insert the form

@endif
```
