LARAVEL

Table of Contents

Workshop Laravel

Introduction

Bienvenue sur ce workshop dédié au framework Laravel.

Vous n'avez pas besoin d'avoir de réelles connaissances en PHP afin de réaliser ce workshop. N'hésitez pas à nous poser des questions si vous en avez au cours du workshop, on vous aidera avec plaisir

Installation

Vous avez seulement besoin de Docker installé sur votre OS pour ensuite utiliser l'installateur automatique de Laravel. Ca permet de lancer quelques conteneurs automatiquement avec la bonne version de PHP, MySQL, etc. sans rien installer sur votre système.

But du workshop

Durant ces deux heures, vous allez apprendre à créer un blog avec Laravel.

Vous pourrez consulter les articles, en créer de nouveau à partir d'une page dédiée en quelques clics.

Sources et aller plus loin

L'intégralité de ce workshop et plus encore est disponible sur Github si vous êtes bloqué sur un exercice ou si vous êtes juste curieux d'avoir l'intégralité du site terminé !

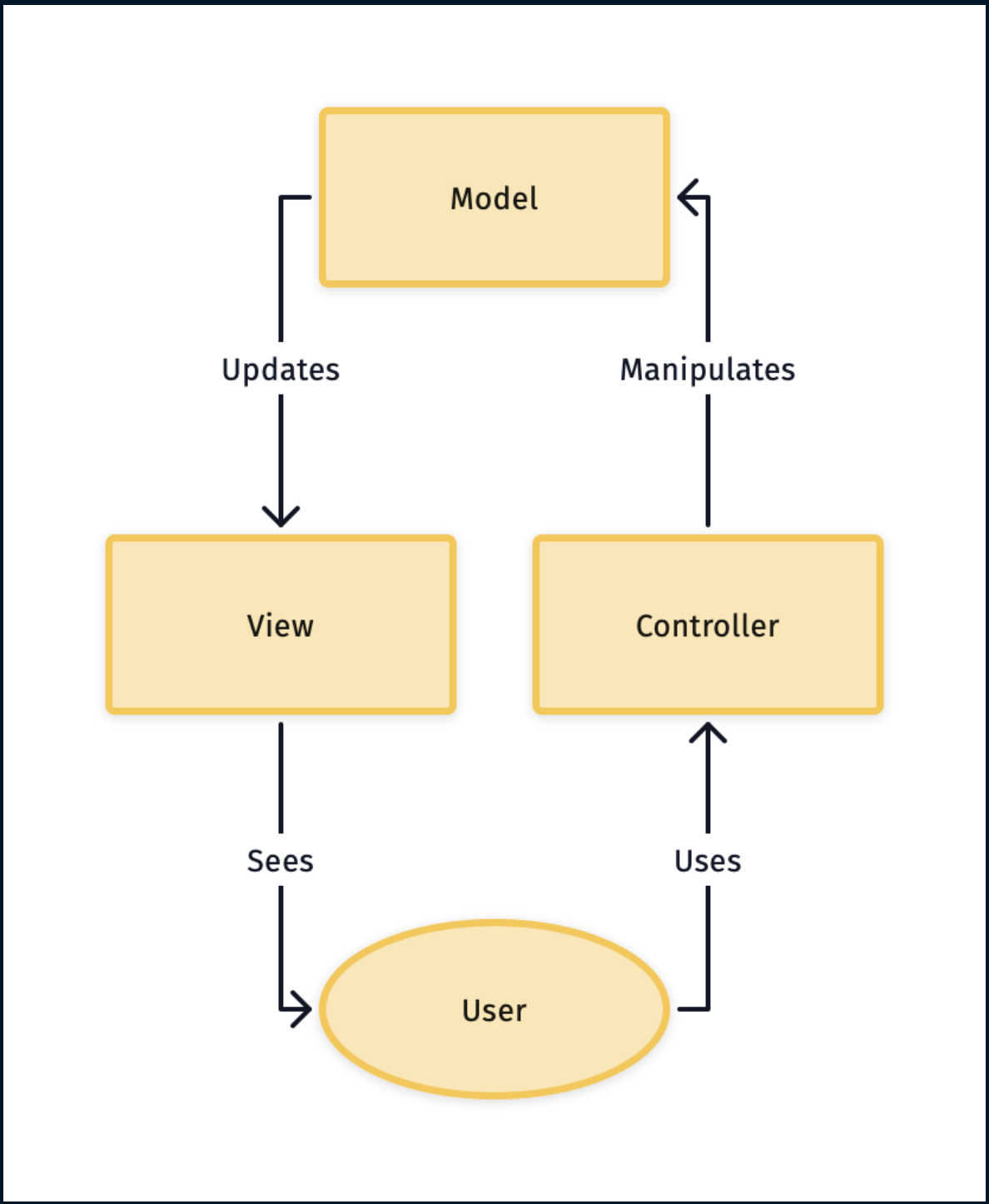
Si vous souhaitez aller plus loin après avoir terminé ce workshop, n'hésitez pas à visionner ce cours gratuit et complet sur Laravel 8 !

Laravel 8 From Scratch

Premiers étapes

Comprendre la MVC

Ce schéma vous explique le fonctionnement d'une architecture de type MVC, que vous allez retrouver dans d'autres frameworks et particulièrement dans Laravel.



Créer votre premier projet Laravel

Pour créer un nouveau projet Laravel, il faut executer la commande :

```
curl -s "https://laravel.build/blog?with=mysql" | bash
```

```
cd blog
```

Maintenant vous pouvez lancer le projet pour qu'on puisse voir sur le navigateur

```
./vendor/bin/sail up -d
```

Pour exécuter des commandes artisan, on peut faire simplement :

```
./vendor/bin/sail artisan
```

et pour exécuter des scripts PHP

```
./vendor/bin/sail php -v
```

Petit tip ! Vous pouvez mettre en place un alias Bash pour que c'est plus facile d'écrire les commandes

```
alias sail='bash vendor/bin/sail'
```

Les termes techniques

Il existe plusieurs termes techniques que l'on retrouve sur l'architecture MVC et sur Laravel que l'on doit connaître.

- Model : modèle permettant de représenter un produit, une catégorie, un utilisateur, etc en leur définissant des propriétés comme un `id`, `first_name`, `email`, etc.
- View : une vue représente une page.
- Controller : Le logique entre le model et une vue
- Route : une instruction permettant de spécifier à Laravel une action à réaliser si l'utilisateur se rend sur une page.

Comprendre les routes et les views

Les routes permettent de spécifier au framework toutes les pages qui existent sur votre site internet.

Une route doit être liée un controller (afin de respecter la MVC). Vous pouvez comparer un controller à une intersection qui permet de dire au framework quelles actions réaliser en fonction de l'action demandé lors de la création de la route.

Afin de créer votre première route, rendez-vous dans le dossier `routes/`

Lors de l'initialisation de votre projet, Laravel vous génère automatiquement une route vers la page d'accueil de votre projet.

Voici le code afin de créer une route avec Laravel.

```
Route::get('/', function () {
    return view('welcome');
})->name('welcome');
```

Passons au crible ce morceau de code :

`Route` permet de dire au framework que l'on souhaite déclarer une route.

`get` représente la requête qui sera utilisée afin de réaliser l'action souhaitée. Il existe également d'autres requêtes HTTP qui existent, mais pour l'instant, on ne les abordera pas.

Cependant, si vous êtes curieux, vous pouvez retrouver la liste des requêtes disponible sur la documentation Mozilla.

`function()` permet de préciser au framework que vous souhaitez créer une fonction de type closure (logique)

Cette fonction, dans ce cas précis, aura pour seul but de vous afficher une vue, autrement dit, une page appelé welcome.

Et `->name('index.welcome')` nous permet de nommer notre route `welcome` pour l'utilisation facile plus tard quand on va générer des liens avec Laravel.

En effet, si on change la page lié à la view, si l'on utilise cette instruction, cela permettra de retrouver la bonne route.

Vous pouvez retrouver l'ensemble des vues de votre projet dans le dossier

`ressources/views`

Vous y retrouverez un fichier Blade appelé `welcome.blade.php`

Mais on n'a pas besoin de spécifier le nom complet de la vue lors de la création de la route ?

Laravel est assez intelligent pour savoir que l'ensemble de vos vues sont dans le dossier `ressources/views` . Vous avez juste à spécifier le nom, sans ajouter l'extension `.blade.php`

Cependant, si votre vue est stocké dans un dossier à l'intérieur de votre dossier de vues, il sera nécessaire de préciser le nom de votre dossier puis le nom de la vue.

Par exemple, si à l'intérieur de votre dossier `views`, vous disposez d'un autre dossier appelé `articles` afin d'y stocker l'ensemble de vos articles, vous devrez le spécifier dans la route.

```
Route::get('/', function () {
    return view('articles.welcome');
                // articles est un dossier, welcome est une page
contenue dans ce dossier
})->name('articles.index');
```

Comprendre les templates Blade

Laravel utilise le moteur de rendu Blade afin de rendre les pages à l'utilisateur.

C'est pour ça que les noms des vues se terminent généralement tous par `blade.php`.

Blade vous permettra de créer des pages afin de construire votre blog.

Suivez la documentation pour comprendre l'utilisation de Blade.

Blade Templates

Pourquoi utiliser des layouts Blade ?

Si vous souhaitez créer une page ayant une même structure qu'une autre page, par exemple, une top-bar avec un menu à l'intérieur ou encore un bas de page, afin d'éviter la répétition dans votre code, optez pour un layout.

Exercice 1 (Migrations)

Pour mettre en place la base de données, il faut lancer la commande `$ sail artisan migrate`

```
> sail artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (29.64ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (27.68ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (36.77ms)
```

Cette commande va "migrer" tout les structures de base de données. (les tables et leurs colonnes respective)

Pour notre cas, il faut créer un autre migration pour les articles de notre site web.

`$ sail artisan make:migration create_posts_table`

Ensuite le fichier crée ce retrouve dans `database/migrations/XXX_XX_XX_XXXXXX_create_posts_table.php`

Il faut ajouter a la table 2 colonnes

`title` de type `string`

`body` de type `text`

Database: Migrations

Exercice 2 (Models, Route, Controllers, Views)

Créer votre première route

Créez une route de type GET afin de charger une page accessible depuis `localhost/posts` et reliez cette route à un controller appelé `PostController` reliée à une méthode `index`. Le controller ainsi que la méthode sera créé lors de la prochaine étape.

Créer un contrôleur

Créez un contrôleur appelé `PostController` et créez une méthode `index` afin de pouvoir charger la vue correspondante.

Vous pouvez utiliser `sail artisan make:controller PostController` pour le créer.

La vue devra s'appeler `posts` afin de la créer lors de la prochaine étape.

Controllers

Créer un layout avec Blade

Comme expliqué ci-dessus, un layout vous permet d'avoir de définir une structure de page qui pourra être appliqué à vos pages.

Copiez le layout existant afin de le modifier puis créez un layout appelé `page.blade.php` afin de pouvoir l'appliquer à la view qui sera créé lors de la prochaine étape.

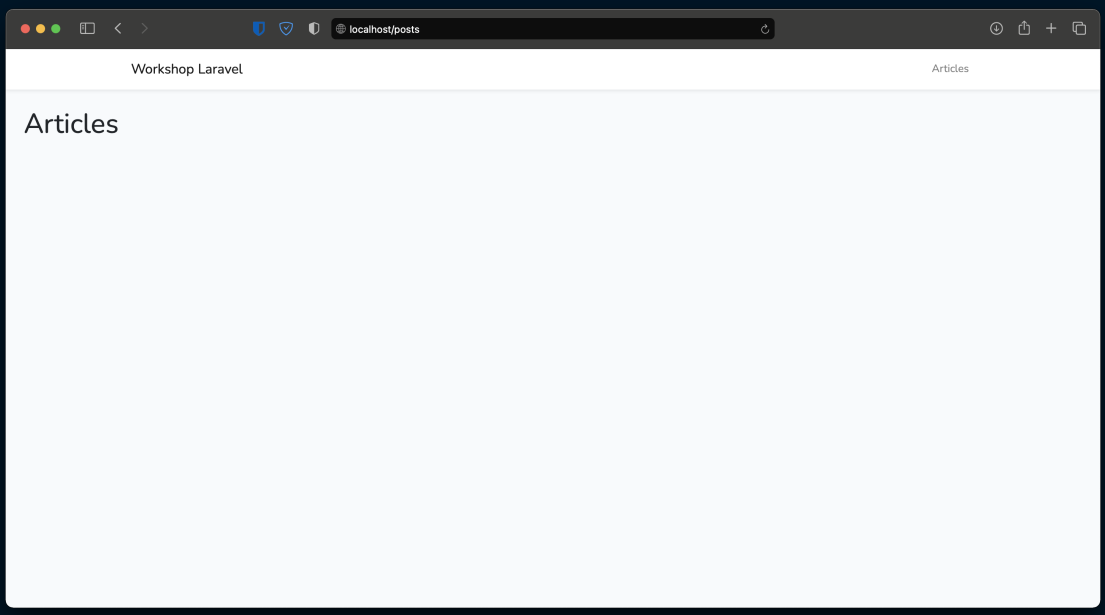
Blade Templates

Créer un view avec Blade

Créez une view appelé `posts.blade.php` comme nom de fichier puis appliquez le layout créé lors de l'étape précédente. Ajoutez-y un titre pour représenter le début de la page dédiée aux articles.

Blade Templates

Résultat à obtenir



Votre page devrait être accessible en vous rendant sur `localhost/posts` et devrait ressembler à peu près à ceci. Pour l'instant, la page est un peu vide, mais à la fin de ce workshop, vos articles seront affichés !

Les models

Notion des models

Dans le cadre MVC, la lettre "M" signifie "Model". Le modèle est un moyen de gérer la logique commerciale dans toute application basée sur le cadre MVC. Dans Laravel, le modèle est une classe qui représente la structure logique et la relation des tables de données sous-jacentes. Dans Laravel, chaque table de la base de données a un "modèle" correspondant qui nous permet d'interagir avec cette table. Les modèles vous permettent de récupérer, d'insérer et de mettre à jour des informations dans votre tableau de données. Tous les modèles Laravel sont stockés dans le répertoire principal de l'application.

Créer un model

```
sail artisan make:model Post
```

Cette commande va créer un model qui s'appelle Post et se retrouve dans le dossier `app/Models/`

Quelques explications sur CRUD

CRUD - Wikipédia

Etablir les opérations CRUD dans le controller

Créez les différentes méthodes afin de pouvoir interagir en créant, affichant, mettant à jour et supprimant des articles.

Create

```php
// soit
$post = new Post(request(['title', 'body']));
$post->save();

// ou si vous souhaitez forcer un titre et un contenu
$post = Post::create([
        'title' => 'My new blog',
        'body' => 'This is my new blog, enjoy',
]);
```

Read

```php
// Trouver l'instance d'un model
$post = Post::find(1); // cherche le post avec l'id 1

// Afficher le titre du post
echo $post->title;
```

Update

```php
// Trouver l'instance d'un model
$post = Post::find(1);

// soit
$post->title = 'New Title';
$post->body = 'New Body';
$post->save();

// ou
$post->update([
        'title' => 'New Title';
        'body' = 'New Body',
]);
```

Destroy

```php
// Trouver l'instance d'un model
$post = Post::find(1);

// Le supprimer
$post->delete();
```

Etablissez les 4 opérations (Create, Read, Update, Delete) dans le PostController.

Pour avoir la structure du code de départ, consultez cet article ou aidez-vous des exemples donnés ci-dessus!

Simple Laravel CRUD with Resource Controllers | DigitalOcean

Exercice 3 : Créer un formulaire

Avant tout, créez une route de type `GET` afin de relier la page de création des articles au controller des articles.

Créez un formulaire afin de créer un nouvel article.

Pour en savoir plus sur les formulaires sur Laravel, consultez la documentation.

Blade Templates

Pour ce workshop, copiez le code ci-dessus puis créez une view `create.blade.php`.

```
@extends('layout')

@section('content')
<div id="wrapper">
    <div id="page" class="container">
        <h1>New article</h1>

        <form method="POST" action="/articles">
            @csrf <!-- Obligatoire :
https://laravel.com/docs/8.x/csrf#csrf-introduction -->
            <div class="field">
                <label class="label" for="title">Title</label>

                <div class="control">
                    <input class="input" type="text" name="title"
id="title" value="{{ old('title') }}">
                </div>
                @error('title')
                    {{ $message }}
                @enderror

            </div>

                <div class="field">

                    <label class="label" for="body">Body</label>
```

```
                    <div class="control">

                        <textarea class="textarea" name="body"
id="body">{{ old('body') }}</textarea>
                        @error('body')
                        <br>{{ $message }}
                        @enderror

                    </div>

                </div>

                <div class="field is-grouped">
                <div class="control">
                    <button class="button is-link" type="submit">
Submit</button>
                </div>

            </div>

            </div>
        </form>
    </div>
</div>
@endsection
```

Exercice 4 (Validation & Gestion d'erreurs)

Maintenant qu'on peux stocker les informations donnés par l'utilisateur, les prochaines étapes sont de valider les données envoyées, et afficher les messages d'erreurs si il faut.

Validation

Validation

```
class PostController extends Controller {

    public function create(Request $request)
    {
        // Definir les contraintes
        $validated = $request->validate([
            'title' => 'required|min:2|max:20', // le champ title est
requis, et le contenu doit être compris entre 2 et 20 caractères.
            'body'  => 'required|min:5|max:255',
        ]);

        /*
        * Si l'un des contraintes échoue, la page retournera au
formulaire
        * avant automatiquement, donc Laravel va meme pas toucher le
code en bas.
        */

        // $validated est de type array, qui contient tout les attributs
validé
        $post = Post::create($validated);

        return redirect()->route('posts.show', $post);
    }
}
```

Voici le rendu que vous devriez avoir : N'hésitez pas à customiser le formulaire à votre guise en ajoutant un peu de CSS

Testez le formulaire puis créez des articles !

Exercice 5 (Affichage des articles)

Afin d'afficher l'ensemble de vos articles sur la page, ajoutez à votre view index.blade.php ce code !

```
@extends('layout')

@section('content')

    <div id="wrapper">
        <div id="page"class="container">
            <h1> Articles </h1>
                <ul class="style1">
<!-- Une boucle qui récupère les articles depuis la base de données -->
                    @forelse($articles as $article)
                        <li class="first">
                            <h3>{{$article->title}}</h3>
                        </li>
                        <p>{{$article->body}}</a></p>
                        @empty
                        <p>Your blog is empty! <a
href="http://localhost:8000/create">Create your first article!</a></p>
                    @endforelse
                </ul>
        </div>
    </div>

@endsection
```

Créez un article puis admirez !

N'hésitez pas à customiser le blog comme bon vous semble !

Conclusion

Bravo d'avoir réussi à arriver jusqu'à là !

Toutes les sources de ce workshop sont disponibles sur le GitHub ci-dessous, si vous avez apprécié ce workshop, mettez-y une etoile !

github.com/Noubouille/laravel-workshop

Si vous souhaitez aller plus loin et que vous souhaitez aller plus loin sur Laravel, n'hésitez pas à visionner ce cours gratuit et complet sur Laravel 8, vous ne le regretterez pas !

Laravel 8 From Scratch