Documentation technique d'installation et d'utilisation de Laravel.

Créé par : HENRY Alexis,

Le *08/07/2022*.

Modifié par : HENRY Alexis,

Le *08/07/2022*.

Version du document : v1.

Sommaire

Installation

Pour plus d'informations sur l'installation de Laravel, vous trouverez cela sur la documentation officielle de Laravel : https://laravel.com/docs/9.x/installation.

Installation via composer

La première étape consiste en l'installation de la structure Laravel via composer. Cela s'effectue à l'aide de la commande :

- composer create-project laravel/laravel=9 example-app

```
ubuntu@dev:/var/www$ composer create-project laravel/laravel=9 example-app
```

Ci-dessous, une image du lancement de la création du projet Laravel :

```
ubuntu@dev:/var/www$ composer create-project laravel/laravel=9 example-app
Creating a "laravel/laravel=9" project at "./example-app"
Installing laravel/laravel (v9.0.0)
  - Downloading laravel/laravel (v9.0.0)
  - Installing laravel/laravel (v9.0.0): Extracting archive
Created project in /var/www/example-app
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
```

Si l’installation s’est correctement terminée, l’arborescence de votre dossier ressemblera à ça :

```
ubuntu@dev:/var/www$ ll example-app
total 384
drwxrwxr-x 13 ubuntu ubuntu   4096 Jul  6 15:42 ./
drwxr-xr-x  8 ubuntu ubuntu   4096 Jul  6 15:42 ../
-rw-rw-r--  1 ubuntu ubuntu    258 Feb  8 16:52 .editorconfig
-rw-rw-r--  1 ubuntu ubuntu    948 Jul  6 15:42 .env
-rw-rw-r--  1 ubuntu ubuntu    897 Feb  8 16:52 .env.example
-rw-rw-r--  1 ubuntu ubuntu    152 Feb  8 16:52 .gitattributes
-rw-rw-r--  1 ubuntu ubuntu    207 Feb  8 16:52 .gitignore
-rw-rw-r--  1 ubuntu ubuntu    175 Feb  8 16:52 .styleci.yml
-rw-rw-r--  1 ubuntu ubuntu   3958 Feb  8 16:52 README.md
drwxrwxr-x  7 ubuntu ubuntu   4096 Feb  8 16:52 app/
-rwxr-xr-x  1 ubuntu ubuntu   1686 Feb  8 16:52 artisan*
drwxrwxr-x  3 ubuntu ubuntu   4096 Feb  8 16:52 bootstrap/
-rw-rw-r--  1 ubuntu ubuntu   1746 Feb  8 16:52 composer.json
-rw-rw-r--  1 ubuntu ubuntu 289517 Jul  6 15:42 composer.lock
drwxrwxr-x  2 ubuntu ubuntu   4096 Feb  8 16:52 config/
drwxrwxr-x  5 ubuntu ubuntu   4096 Feb  8 16:52 database/
drwxrwxr-x  3 ubuntu ubuntu   4096 Feb  8 16:52 lang/
-rw-rw-r--  1 ubuntu ubuntu    473 Feb  8 16:52 package.json
-rw-rw-r--  1 ubuntu ubuntu   1175 Feb  8 16:52 phpunit.xml
drwxrwxr-x  2 ubuntu ubuntu   4096 Feb  8 16:52 public/
drwxrwxr-x  5 ubuntu ubuntu   4096 Feb  8 16:52 resources/
drwxrwxr-x  2 ubuntu ubuntu   4096 Feb  8 16:52 routes/
drwxrwxr-x  5 ubuntu ubuntu   4096 Feb  8 16:52 storage/
drwxrwxr-x  4 ubuntu ubuntu   4096 Feb  8 16:52 tests/
drwxrwxr-x 43 ubuntu ubuntu   4096 Jul  6 15:42 vendor/
-rw-rw-r--  1 ubuntu ubuntu    559 Feb  8 16:52 webpack.mix.js
```

Vous pourrez alors, lancer le serveur de php artisan, via la commande ci-dessous :

- php artisan serve –host=xxx

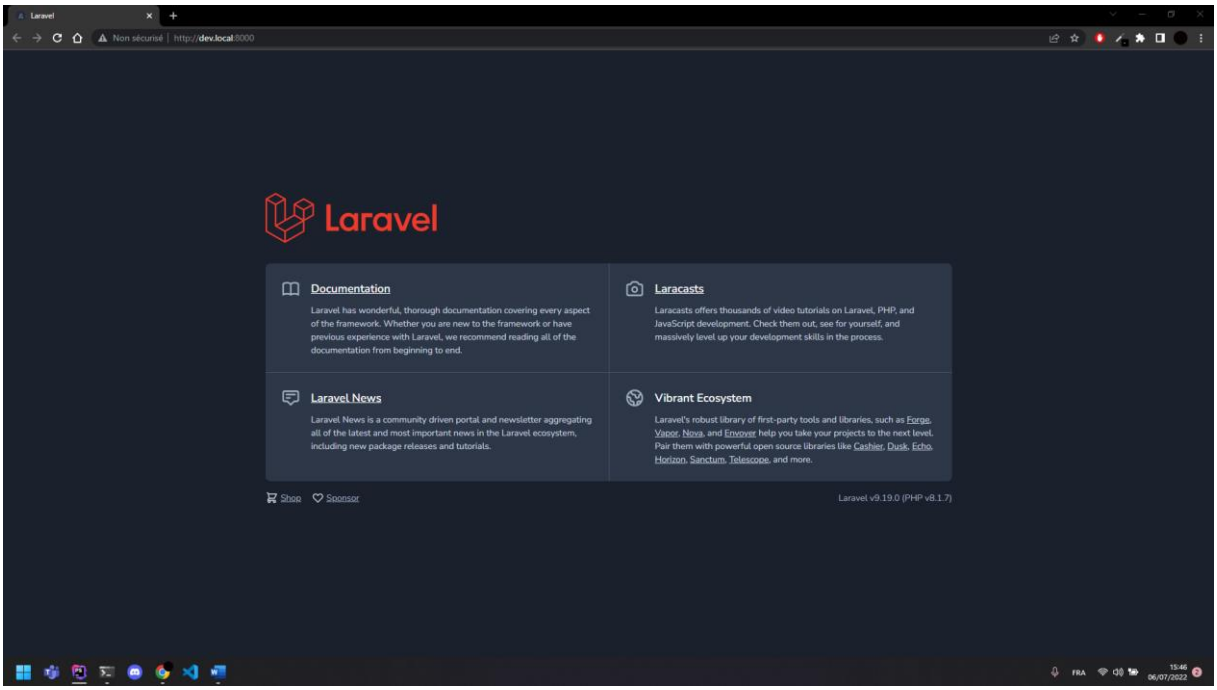
L’option : --host, vous permet de spécifier l’url sur lequel vous voulez que le serveur soit lancé (ici dev.local). Par défaut, le serveur est lancé sur le port 8000.

```
ubuntu@dev:/var/www/example-app$ php artisan serve --host=dev.local
```

Suite à ça, vous pourrez alors accéder via, l'url que vous retournera la commande « php artisan serve », à votre projet Laravel.

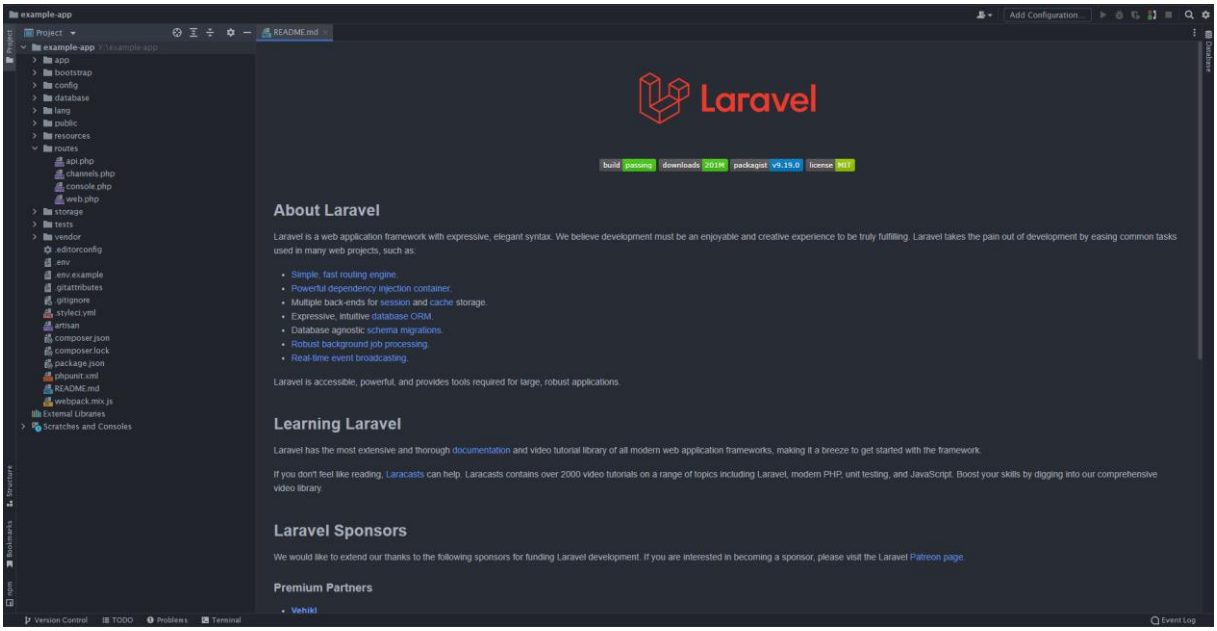
```
ubuntu@dev:/var/www/example-app$ php artisan serve --host=dev.local
Starting Laravel development server: http://dev.local:8000
[Wed Jul  6 15:48:39 2022] PHP 8.1.7 Development Server (http://dev.local:8000) started
```

Si l'installation s'est bien déroulée, vous accéderez donc à la page par défaut de Laravel :



Utilisation

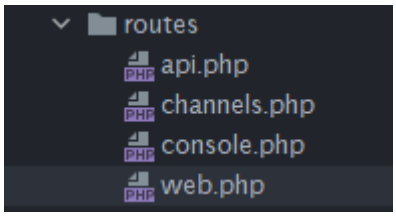
Pour débuter votre projet, vous pouvez alors, ouvrir dans un éditeur de code (type PHP Storm, VS Code, …), votre projet :



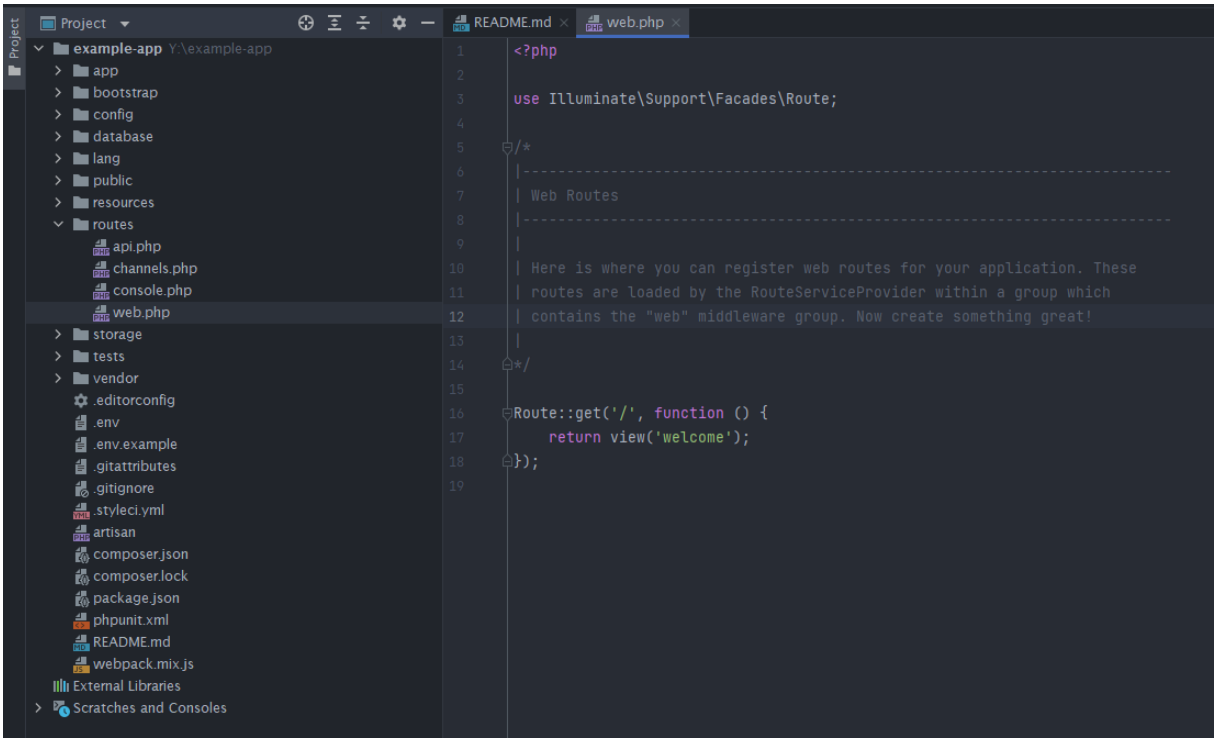
Pour débuter l'apprentissage de Laravel, nous allons créer les éléments nécessaires pour faire afficher un Hello World à l'aide de blade.

Routes

Laravel fonctionne avec un système de Routing, comme la plupart des frameworks. Pour ce qui est de celui-ci, le fichier contenant les routes se situe dans le dossier routes.



Ce fichier web.php, contient donc la route par défaut de Laravel. Qui nous renvoie sur la page de bienvenue sur le framework.



Nous allons supprimer cette route et en créer une qui nous redirigera vers une page disant « Hello World ».

La route réalisant cela, ressemblera donc à ceci :

```
17  Route::get( uri: '/', function () {
18      return "Hello, World !";
19  });
```

De ce fait, lorsque l'on essayera d'accéder à la page web, Laravel nous retournera le fameux « Hello, World ! ».

---

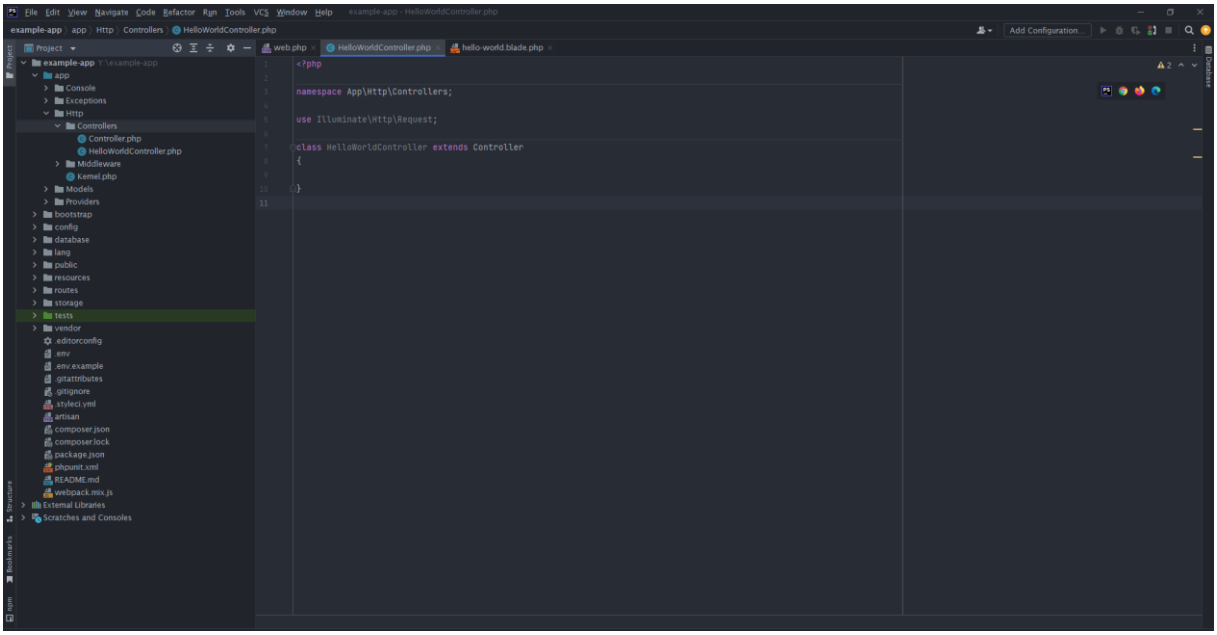
*

Controllers

Sur Laravel, nous pouvons créer des controllers à l'aide de la commande ci-dessous :

```
ubuntu@dev:/var/www/example-app$ php artisan make:controller HelloWorldController
Controller created successfully.
```

Cette commande va donc générer une class prenant le nom indiqué, et ce dans le fichier HelloWorldController.php, situé dans « App\Http\Controllers ».



Nous allons compléter ce Controller, pour qu'il retourne une vue blade affichant « Hello, World ! ». Il faut donc insérer, dans la classe, le code suivant :

```
public function __construct()
{

}

public function HelloWorld()
{
    return view('hello-world', [
        'title' => "Hello World !",
        "h1" => "Hello, World !",
        "p" => "Learn to use Laravel with me !"
    ]);
}
```

Les Controllers sont appelés dans les routes, comme ceci :

```
Route::get( uri: '/', [HelloWorldController::class, 'HelloWorld'])->name( name: 'hi_world');
```

Cela signifie que l'on utilise, pour la route « / », la fonction « HelloWorld « du controller « HelloWorldController ».

Nous allons voir, dans la partie suivante, comment établir la view que le Controller retourne.

Views

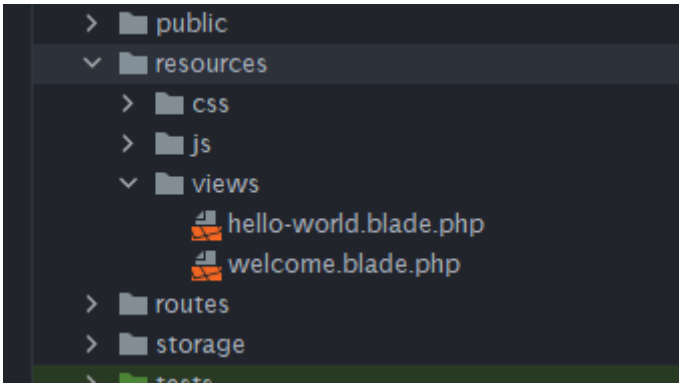
Laravel utilise le système de template Blade (Symfony par exemple utilise Twig). Ce système de template permet énormément de choses (insérer des conditions, des boucles, utiliser des variables, …) au sein d'une template.
Les données reçues par la vue proviennent du Controller qui appellent cette même vue. Par exemple, dans le Controller ci-dessous, lorsque j'appelle ma vue, je lui fais parvenir un certain nombre de données :

- title, qui contient « Hello World ! »,
- h1, qui contient « Hello, World ! »,
- p, qui contient « Learn to use Laravel with me ! ».

```
public function HelloWorld()
{
    return view( view: 'hello-world', [
        'title' => "Hello World !",
        "h1" => "Hello, World !",
        "p" => "Learn to use Laravel with me !"
    ]);
}
```

Je vais donc me servir de ces variables pour afficher les données dans ma vue.

Sur Laravel, on créer les vues dans « Ressources\Views » :



Lorsque l'on appelle des views dans une Route ou un Controller, on utilise uniquement ce qui est devant « .blade.php ».

De ce fait, ma vue « hello-world », ressemblera à ceci :

web.php × HelloWordController.php × hello-world.blade.php ×

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{{ $title }}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor" crossorigin="anonymous">
</head>
<body>

<div class="container mt-5">

    <h1>{{ $h1 }}</h1>

    <p>{{ $p }}</p>

</div>

</body>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2" crossorigin="anonymous"></script>
</html>
```

Voici donc le résultat attendu :