

Créer une application avec Laravel 5.5 – Installation et style

Dans ce chapitre nous allons commencer la création de notre application de photos. La première chose va être d'installer Laravel et ensuite de changer le style pour le faire mieux correspondre à une galerie photos.























Les articles ont été mis à jour pour la version définitive de Bootstrap 4.

L'installation

Pour installer Laravel c'est tout simple :

```
composer create-project laravel/laravel album --prefer-dist
```

Si tout se passe bien vous devez avoir ces dossiers :

- >  app
- >  bootstrap
- >  config
- >  database
- >  public
- >  resources
- >  routes
- >  storage
- >  tests
- >  vendor
-  .env
-  .env.example
-  .gitattributes
-  .gitignore
-  artisan
-  composer.json
-  composer.lock
-  package.json
-  phpunit.xml
-  readme.md
-  server.php
-  webpack.mix.js

Et cet aspect :

Laravel

[DOCUMENTATION](#)[LARACASTS](#)[NEWS](#)[FORGE](#)[GITHUB](#)

Pour compléter on va aussi installer l'authentification :

```
cd album  
php artisan make:auth
```

Il faut aussi créer une base de données, on va partir du principe qu'on utilise MySQL.

Il faut bien paramétrer dans le fichier `.env`, par exemple pour une installation locale par défaut :

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

Ensuite on lance les migrations :

```
php artisan migrate
```

```
λ php artisan migrate  
Migration table created successfully.  
Migrating: 2014_10_12_000000_create_users_table  
Migrated: 2014_10_12_000000_create_users_table  
Migrating: 2014_10_12_100000_create_password_resets_table  
Migrated: 2014_10_12_100000_create_password_resets_table
```

Et on a les 3 tables dans la base :

Table ▲
migrations
password_resets
users
3 tables

Et on gagne un barre de navigation pour l'authentification :

LOGIN REGISTER

Vérifiez que tous les formulaires fonctionnent :

Register

Name

E-Mail Address

Password

Confirm Password

Register

Et qu'on peut bien s'enregistrer et se connecter/déconnecter.

Quand on se connecte on aboutit à cette page :

Laravel

Dupont ▼

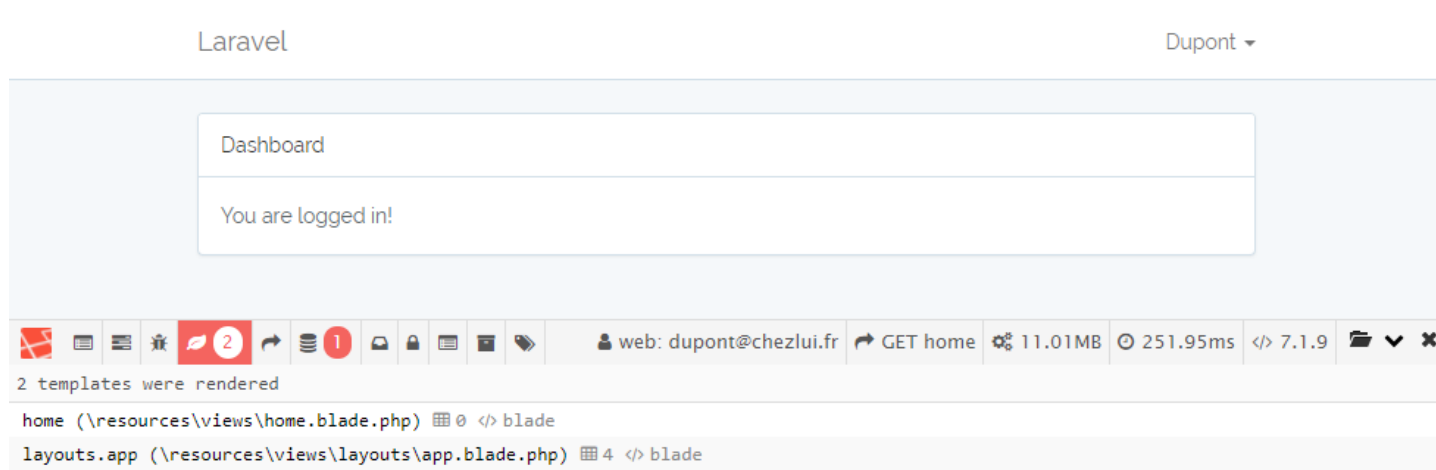
Dashboard

You are logged in!

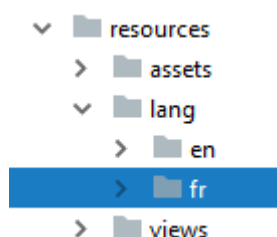
On va aussi installer la barre de débogage :

```
composer require barryvdh/laravel-debugbar --dev
```

Vérifiez qu'elle s'installe correctement :



Tant qu'on y est on va aussi ajouter la langue française. Téléchargez [ce package](#) et ajoutez le dossier du français ici :



Et changez cette ligne dans le fichier **config/app.php** :

```
'locale' => 'fr',
```

Vous ne verrez la différence que pour la validation :

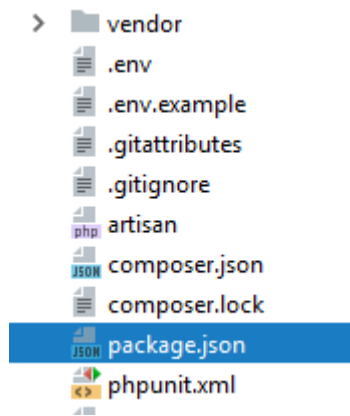


Pour le reste il faudra un peu relever les manches...

Le style

On va maintenant se consacrer au style parce que ce n'est pas très joli pour le moment...

On a à la racine un fichier **package.json** :



On va le prendre tel quel et on lance l'installation :

```
npm install
```

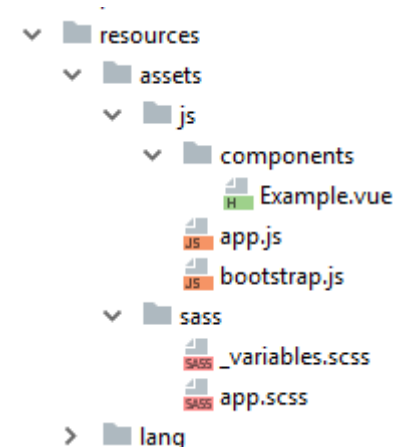
Voilà, comme ça on va pouvoir utiliser **mix**. On a un fichier de configuration à la racine :



Avec ce code :

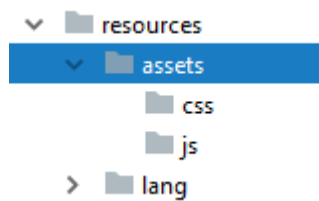
```
let mix = require('laravel-mix');  
  
mix.js('resources/assets/js/app.js', 'public/js')  
    .sass('resources/assets/sass/app.scss', 'public/css');
```

On a les assets ici :



Par défaut on utilise **Bootstrap 3** dans sa version sass avec des variables modifiées par le fichier `_variables.scss`. D'autre part pour le Javascript on utilise aussi **Bootstrap 3** évidemment et aussi **Vue.js**.

Pour notre application on va plutôt passer à **Bootstrap 4** qui est maintenant en version définitive. Pour le Javascript on va se contenter de JQuery parce qu'on a pas trop de travail côté client. En résumé on va faire un peu le ménage :



On va utiliser la version CSS de Bootstrap parce qu'on va très peu la modifier en se contentant de quelques surcharges (en plus la version **sass** n'est pas très stable).

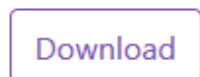
On va donc [télécharger Bootstrap 4](#) :

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v4.0.0** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins

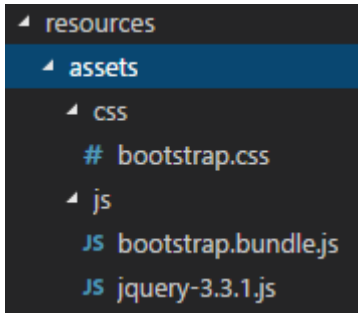
This doesn't include documentation, source files, or any optional JavaScript dependencies (jQuery and Popper.js).



Pour le Javascript on va aussi [récupérer JQuery](#) :

[Download the uncompressed, development jQuery 3.3.1](#)

Vous copiez tout ça ici :



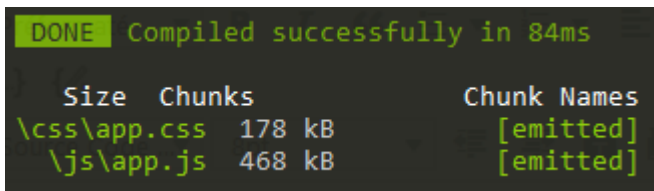
Ensuite on modifie le fichier **webpack.mix.js** :

```
mix.styles([
    'resources/assets/css/bootstrap.css'
], 'public/css/app.css')
.mscripts([
    'resources/assets/js/jquery-3.3.1.js',
    'resources/assets/js/bootstrap.bundle.js'
], 'public/js/app.js');
```

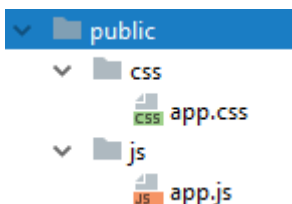
*Le fichier **bootstrap.bundle.js** contient la librairie **popper.js** utile pour certains plugins de Bootstrap.*

Et on lance mix :

```
npm run dev
```



On se retrouve avec tout ça rassemblé dans le dossier **public** :



Bon évidemment on a mis un peu le bordel dans les vues :

Laravel

Login

E-Mail Address

Password

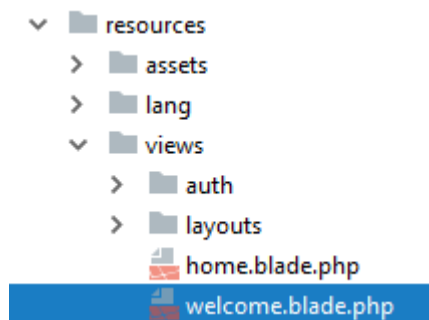
☐ Remember Me

Login

[Forgot Your Password?](#)

On va arranger tout ça maintenant...

On va supprimer la vue **welcome** :



On va arriver directement sur la vue **home**. Pour que ça marche on va enlever la déclaration du middleware **auth** dans le contrôleur **HomeController**. Il va donc rester ce code :

```
<?php
```

```
namespace App\Http\Controllers;
```

```
class HomeController extends Controller  
{
```

```
    /**
```

```
     * Show the application dashboard.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
    */
```



```

    public function index()
    {
        return view('home');
    }
}

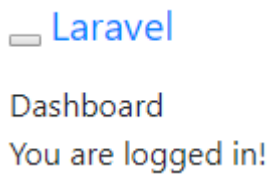
```

Et pour les routes (**routes/web.php**) :

```
Auth::routes();
```

```
Route::get('/', 'HomeController@index')->name('home');
```

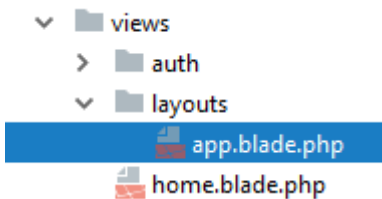
On arrive donc maintenant ici :



C'est devenu du n'importe quoi ☹

Le layout

Le layout de base est ici :



Changez le code pour celui-ci :

```

<!DOCTYPE html>
<html lang="{{ app()->getLocale() }}">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ config('app.name', 'Album') }}</title>
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">

```

```

        @yield('css')
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-dark bg-
dark">
    <a class="navbar-brand" href="{{ route('home') }}">{{
config('app.name', 'Album') }}</a>
    <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
        <div class="collapse navbar-collapse"
id="navbarSupportedContent">
            <ul class="navbar-nav ml-auto">
                @guest
                    <li class="nav-item"><a class="nav-link" href="{{
route('login') }}">@lang('Connexion')</a></li>
                    <li class="nav-item"><a class="nav-link" href="{{
route('register') }}">@lang('Inscription')</a></li>
                @else
                    <li class="nav-item">
                        <a id="logout" class="nav-link" href="{{
route('logout') }}">@lang('Déconnexion')</a>
                        <form id="logout-form" action="{{
route('logout') }}" method="POST" class="hide">
                            {{ csrf_field() }}
                        </form>
                    </li>
                @endguest
            </ul>
        </div>
</nav>
@yield('content')
<script src="{{ asset('js/app.js') }}"></script>
@yield('script')
<script>
    $(function() {
        $('#logout').click(function(e) {
            e.preventDefault();
            $('#logout-form').submit()
        })
    })

```

```
    })  
</script>  
</body>  
</html>
```

C'est une adaptation pour Bootstrap 4, ce qui donne cet aspect de la barre :



Avec le bon effet « responsive » :



Remarquez que les textes ont déjà été préparés pour le multi-langue avec la directive **@lang** :

```
@lang('Connexion')
```

On pourra ainsi ensuite facilement ajouter une langue.

D'autre part comme on dispose de JQuery j'ai un peu remanié le code pour la déconnexion.

La barre est belle mais pour les formulaires c'est pas encore gagné :

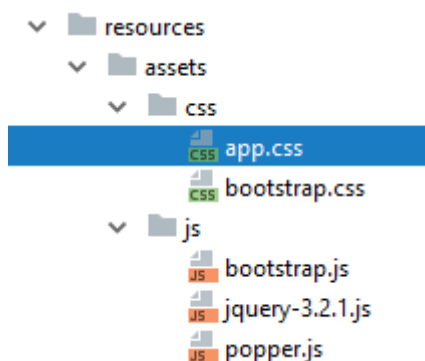
Password

☐ Remember Me[Login](#)[Forgot Your Password?](#)

Ce qui est logique puisqu'ils sont prévus pour Bootstrap 3. Il faudra s'en occuper, on verra ça dans le prochain chapitre.

Un peu de CSS

Pour le moment on va continuer un peu notre stylisation. Pour ça on va ajouter un fichier CSS pour nos surcharges :



Et évidemment on l'ajoute dans **webpack.mix.css** :

```
mix.styles([
    'resources/assets/css/bootstrap.css',
    'resources/assets/css/app.css'
], 'public/css/app.css')
.scripts([
    'resources/assets/js/jquery-3.3.1.js',
    'resources/assets/js/bootstrap.bundle.js'
], 'public/js/app.js');
```

On va faire deux choses :

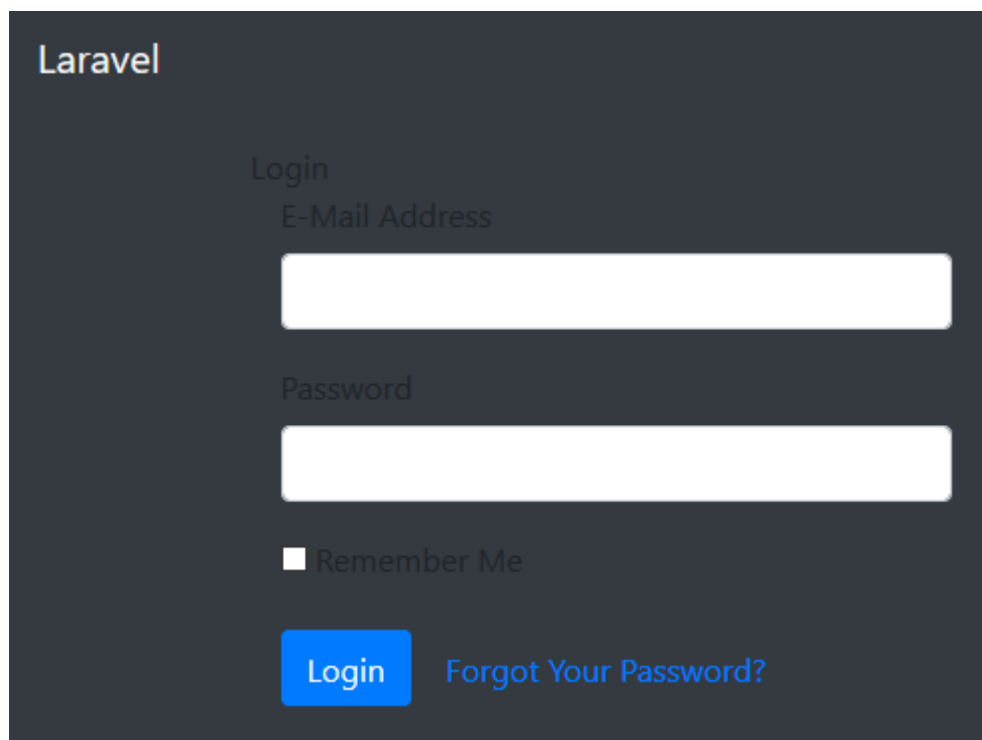
- harmoniser la couleur du fond de la page avec celle de la barre de navigation
- réserver un peu d'espace en haut de la page pour éviter que le contenu soit caché par la barre

```
body {  
  background-color: #343a40;  
  padding-top: 70px;  
}
```

On relance **npm** (vous pouvez aussi passer en mode **watch** pour que toutes les modifications soient automatiquement effectuées) :

```
npm run dev
```

Ce qui donne :

A screenshot of a web application's login page. The page has a dark, charcoal-colored background. In the top left corner, the word "Laravel" is displayed in a light blue font. The login form is centered and consists of the following elements: the word "Login" in a light gray font; a label "E-Mail Address" in a light gray font above a white rectangular input field; a label "Password" in a light gray font above another white rectangular input field; a checkbox with a small white square followed by the text "Remember Me" in a light gray font; a blue rectangular button with the word "Login" in white text; and a link "Forgot Your Password?" in a light blue font to the right of the login button.

*Si ça ne s'actualise pas dans votre navigateur lancez un rafraîchissement (en général **CTRL F5**)*

La couleur est bien harmonisée et le formulaire décalé vers le bas. On perd des textes à cause de la couleur mais on s'en occupera bientôt.

Il reste deux petits soucis dans notre application :

- elle s'appelle encore **Laravel** alors qu'on voudrait **Album**
- lorsqu'on se connecte on est renvoyés sur l'url **/home** qui n'existe plus.

Pour le premier point on règle ça dans **.env** :

```
APP_NAME=Album
```

Pour le second point on change cette propriété dans **LoginController** :

```
protected $redirectTo = '/';
```

Et tant qu'on y est on fait la même chose dans **RegisterController** et **ResetPasswordController**.

Conclusion

Dans ce chapitre on a vu :

- l'installation de Laravel avec l'authentification
- les migrations de base
- l'installation de la barre de débogage
- l'installation du français
- l'utilisation de mix pour installer JQuery et Bootstrap 4
- le remaniement des routes
- la modification du Layout pour l'adapter à Bootstrap 4 et au multi-langue
- l'utilisation du CSS pour styliser la page
- le changement du nom de l'application
- le remaniement des redirections dans les contrôleurs de l'authentification

Pour vous simplifier la vie vous pouvez [charger le projet](#) dans son état à l'issue de ce chapitre.