

Cours Laravel 6 – les bases – installation et organisation

 laravel.sillo.org/cours-laravel-6-les-bases-installation-et-organisation/

bestmomo

August 28,
2019

Dans ce chapitre nous allons voir comment créer une application Laravel et comment le code est organisé dans une application.

Pour utiliser Laravel et suivre ce chapitre et l'ensemble du cours vous aurez besoin d'un serveur équipé de PHP avec au minimum la version 7.1.2 et aussi de MySQL. Nous avons vu dans le précédent chapitre les différentes possibilités. D'autre part plusieurs extensions de PHP doivent être activées.

Créer une application Laravel

Le serveur

Pour fonctionner correctement, Laravel a besoin de PHP :

- Version $\geq 7.2.0$,
- Extension PDO,
- Extension Mbstring,
- Extension OpenSSL,
- Extension Tokenizer,
- Extension XML.,
- Extension BCMath,
- Extension Ctype,
- Extension JSON

Laravel est équipé d'un serveur sommaire pour le développement qui se lance avec cette commande :

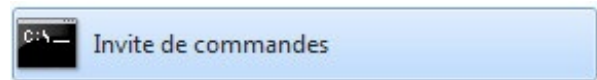
```
php artisan serve
```

On y accède à cette adresse : <http://localhost:8000>. Mais évidemment pour que ça fonctionne il faut que vous ayez PHP installé.

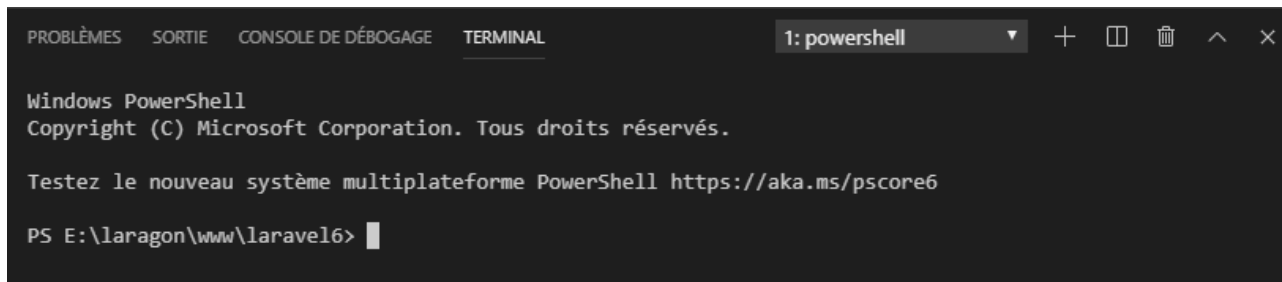
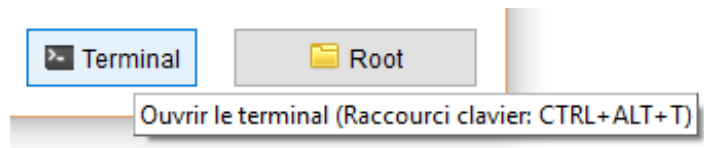
Prérequis

Composer fonctionne en ligne de commande. Vous avez donc besoin de la console (nommée Terminal ou Konsole sur OS X et Linux). Les utilisateurs de Linux sont très certainement habitués à l'utilisation de la console mais il en est généralement pas de même pour les adeptes de Windows. Pour trouver la console sur ce système il faut chercher l'invite de commande :

Si vous utilisez Laragon, comme je vous le conseille, vous avez une console améliorée (**Cmder**) accessible avec ce bouton.



Si vous utiliser Visual Studio Code vous pouvez ouvrir directement un terminal (et même plusieurs) directement dans une fenêtre de l'éditeur :



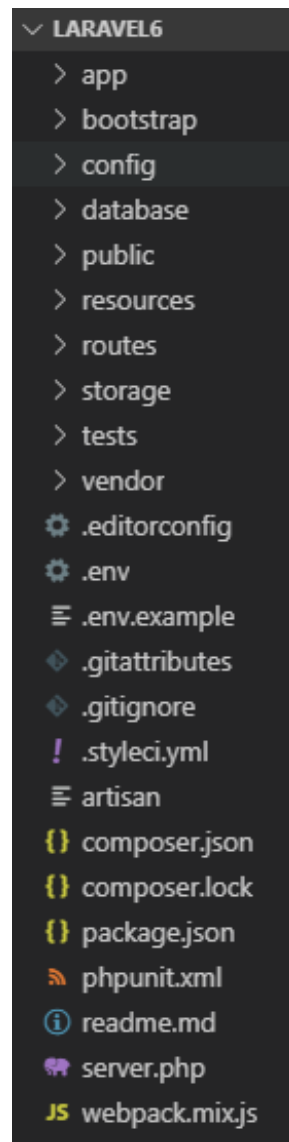
Installation avec composer

Il y a plusieurs façons de créer une application Laravel. La plus classique consiste à utiliser la commande **create-project** de composer. Par exemple je veux créer une application dans un dossier **laravel6** à la racine de mon serveur, voici la syntaxe à utiliser :

```
composer create-project --prefer-dist laravel/laravel laravel6
```

L'installation démarre et je n'ai plus qu'à attendre quelques minutes pour que composer fasse son travail jusqu'au bout. Vous verrez s'afficher une liste de téléchargements. Au final on se retrouve avec cette architecture :

On peut vérifier que tout fonctionne bien avec l'URL <http://localhost/laravel6/public>. Normalement on doit obtenir cette page très épurée :



Laravel

[DOCS](#)[LARACASTS](#)[NEWS](#)[BLOG](#)[NOVA](#)[FORGE](#)[VAPOR](#)[GITHUB](#)

Pour les mises à jour ultérieures il suffit encore d'utiliser composer avec la commande **update** :

```
composer update
```

Installation avec Laravel Installer

Une autre solution pour installer Laravel consiste à utiliser l'installateur. Il faut commencer par installer globalement l'installateur avec composer :

```
composer global require "laravel/installer"
```

Il faut ensuite informer la variable d'environnement **path** de l'emplacement du dossier **.../composer/vendor/bin**.

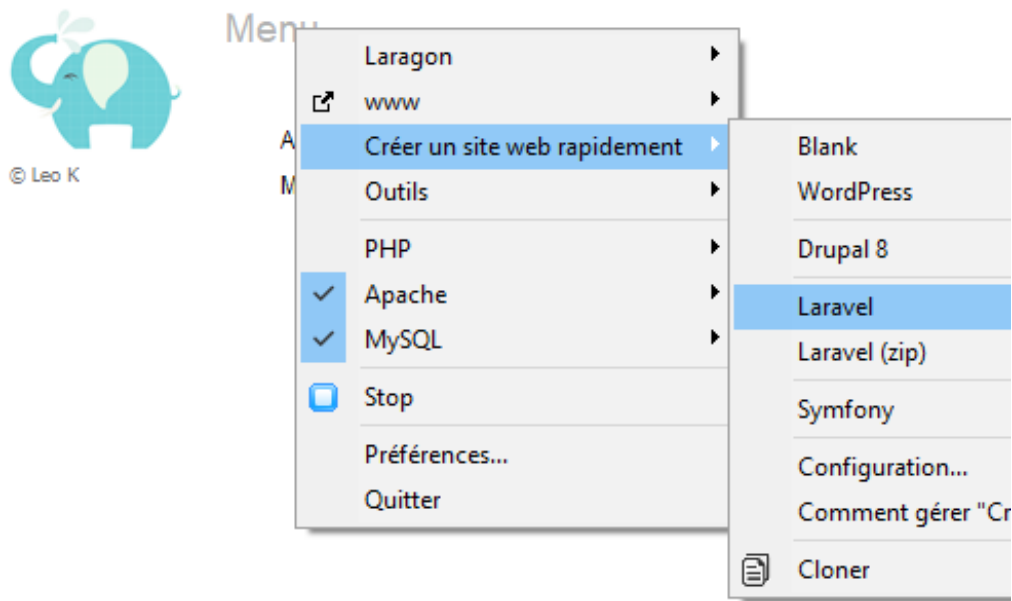
Pour créer une application il suffit de taper :

```
laravel new monappli
```

Laravel sera alors installé dans le dossier **monappli**.

Installation avec Laragon

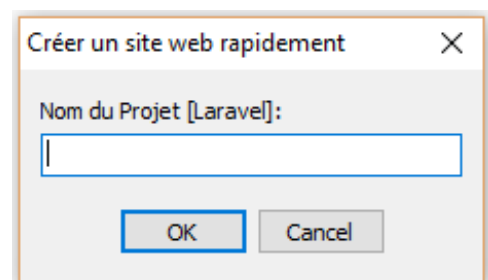
Une façon encore plus simple pour installer Laravel est d'utiliser Laragon avec le menu :



On vous demande alors le nom du projet (donc du dossier) :

Et ensuite vous n'avez plus qu'à attendre ! La console s'ouvre et vous pouvez suivre le déroulement des opérations. On vous rappelle la commande de composer et en plus une base de données et automatiquement créée avec le même nom !

Vous avez alors accès directement à l'application avec laravel6.oo.



*Si vous installez Laravel en téléchargeant directement les fichiers sur Github et en utilisant la commande **composer install** il vous faut effectuer deux actions complémentaires. En effet dans ce cas il ne sera pas automatiquement créé de clé de sécurité et vous allez tomber sur une erreur au lancement. Il faut donc la créer avec la commande **php artisan key:generate**. D'autre part vous aurez à la racine le fichier **.env.example** que vous devrez renommer en **.env** (ou en faire une copie) pour que la configuration fonctionne.*

Autorisations

Au niveau des dossiers de Laravel, les seuls qui ont besoin de droits d'écriture par le serveur sont **storage** (et ses sous-dossiers), et **bootstrap/cache**.

Des URL propres

Pour un serveur Apache il est prévu dans le dossier public un fichier **.htaccess** avec ce code :

```
<IfModule mod_rewrite.c>
  <IfModule mod_negotiation.c>
    Options -MultiViews -Indexes
  </IfModule>

  RewriteEngine On

  # Handle Authorization Header
  RewriteCond %{HTTP:Authorization} .
  RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]

  # Redirect Trailing Slashes If Not A Folder...
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteCond %{REQUEST_URI} (.+)/$
  RewriteRule ^ %1 [L,R=301]

  # Handle Front Controller...
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteRule ^ index.php [L]
</IfModule>
```

Le but est essentiellement d'éviter d'avoir **index.php** dans l'url. Mais pour que ça fonctionne il faut activer le module **mod_rewrite**.

Avec NGinx il faut ajouter ça dans la configuration du site :

```
location / {
    try_files $uri $uri/ /index.php?$query_string;
}
```

Organisation de Laravel

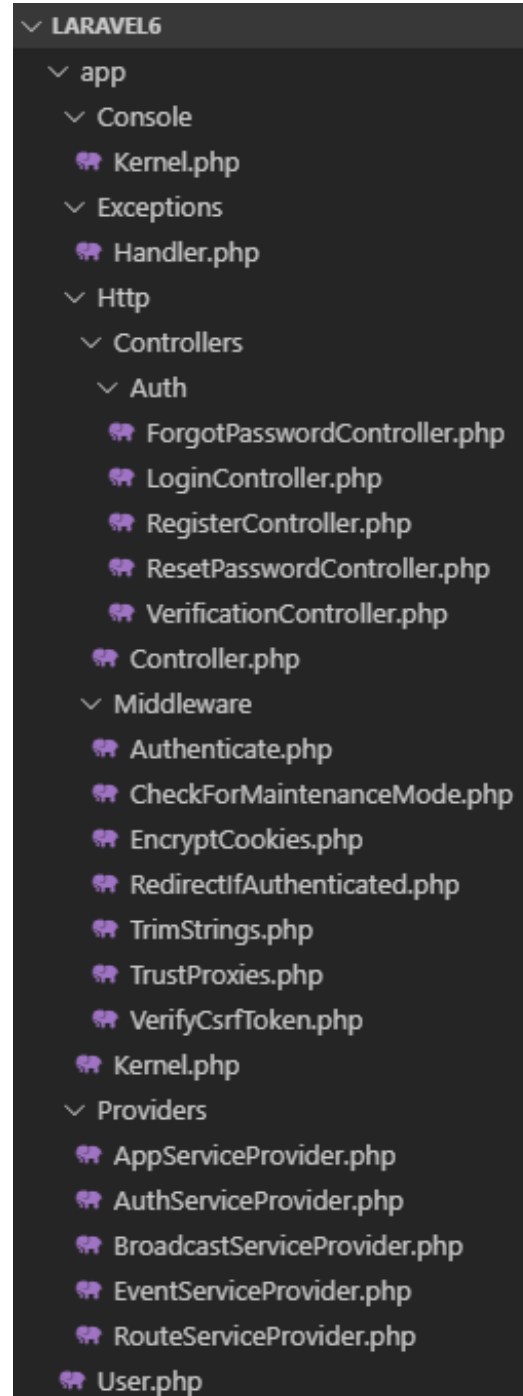
Maintenant qu'on a un Laravel tout neuf et qui fonctionne voyons un peu ce qu'il contient...

Dossier app

Ce dossier contient les éléments essentiels de l'application :

- **Console** : toutes les commandes en mode console,
- **Http** : tout ce qui concerne la communication : contrôleurs, middlewares (il y a 7 middlewares de base qui servent à filtrer les requêtes HTTP) et le kernel,
- **Providers** : tous les fournisseurs de services (providers), il y en a déjà 5 au départ. Les providers servent à initialiser les composants.
- **User** : un modèle qui concerne les utilisateurs pour la base de données.

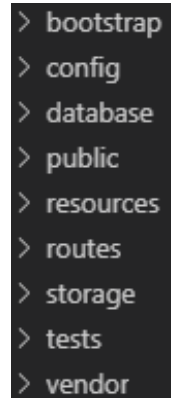
Évidemment tout cela doit vous paraître assez nébuleux pour le moment mais nous verrons en détail ces éléments au fil du cours. Et on verra d'ailleurs que seront créés bien d'autres dossiers selon nos besoins.



Autres dossiers

Voici une description du contenu des autres dossiers :

- **bootstrap** : scripts d'initialisation de Laravel pour le chargement automatique des classes, la fixation de l'environnement et des chemins, et pour le démarrage de l'application,
- **public** : tout ce qui doit apparaître dans le dossier public du site : images, CSS, scripts...
- **config** : toutes les configurations : application, authentification, cache, base de données, espaces de noms, emails, systèmes de fichier, session...
- **database** : migrations et populations,
- **resources** : vues, fichiers de langage et assets (par exemple les fichiers Sass),
- **routes** : la gestion des urls d'entrée de l'application,
- **storage** : données temporaires de l'application : vues compilées, caches, clés de session...
- **tests** : fichiers de tests unitaires,
- **vendor** : tous les composants de Laravel et de ses dépendances (créé par composer).



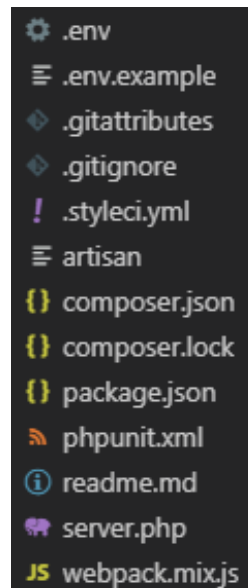
```
> bootstrap
> config
> database
> public
> resources
> routes
> storage
> tests
> vendor
```

Fichiers de la racine

Il y a un certain nombre de fichiers dans la racine dont voici les principaux :

- **artisan** : outil en ligne de Laravel pour des tâches de gestion,
- **composer.json** : fichier de référence de composer,
- **package.json** : fichier de référence de npm pour les assets,
- **phpunit.xml** : fichier de configuration de phpunit (pour les tests unitaires),
- **.env** : fichier pour spécifier l'environnement d'exécution.

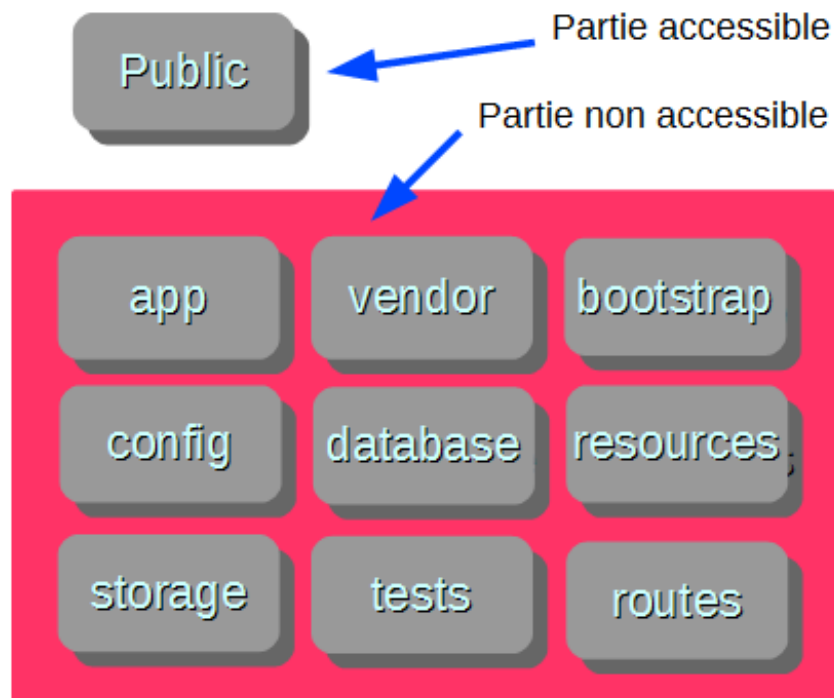
Nous verrons tout cela progressivement dans le cours, ne vous inquiétez pas !



```
⚙ .env
≡ .env.example
💎 .gitattributes
💎 .gitignore
! .styleci.yml
≡ artisan
{} composer.json
{} composer.lock
{} package.json
🔥 phpunit.xml
📖 readme.md
🐘 server.php
JS webpack.mix.js
```

Accessibilité

Pour des raisons de sécurité sur le serveur seul le dossier **public** doit être accessible :



Cette configuration n'est pas toujours possible sur un serveur mutualisé, il faut alors modifier un peu Laravel pour que ça fonctionne; j'en parlerai dans le chapitre sur le déploiement.

Environnement et messages d'erreur

Par défaut lorsque vous installez Laravel, celui-ci est en mode « debug » et vous aurez une description précise de toutes les erreurs. Par exemple ouvrez le fichier **routes/web.php** et changez ainsi le code :

```
Route::post('/', function () {  
    return view('welcome');  
});
```

Ouvrez l'url de base, vous obtenez une page d'erreur avec en particulier cette information :

 E:\Jargon\www\laravel6\

Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException
The GET method is not supported for this route. Supported methods: POST.

<http://laravel6.oo/>

Avec la version 6 Laravel change de système d'affichage des erreurs et a fait le choix d'Ignition qui est bien plus élaboré que le précédent. Par exemple il nous informe que notre base de données est mal renseignée :

Database name seems incorrect

You're using the default database name `laravel`. This database does not exist.
Edit the `.env` file and use the correct database name in the `DB_DATABASE` key

[READ MORE](#) [Database: Getting Started docs](#)

Ce qui est effectivement le cas comme nous le verrons bientôt.

Pendant la phase de développement on a besoin d'obtenir des messages explicites pour traquer les erreurs inévitables que nous allons faire. En mode « production » il faudra changer ce mode, pour cela ouvrez le fichier **config/app.php** et trouvez cette ligne :

```
'debug' => env('APP_DEBUG', false),
```

Autrement dit on va chercher la valeur dans l'environnement, mais où peut-on le trouver ? Regardez à la racine des dossiers, vous y trouvez le fichier **.env** avec ce contenu

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:1KWREr1t3kmRTWq/dxD7SKgMQUtdvjleOdRxQ0COVSQ=
APP_DEBUG=true
APP_URL=http://localhost
```

```
LOG_CHANNEL=stack
```

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

```
BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

```
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
```

```
PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1
```

```
MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Vous remarquez que dans ce fichier la variable **APP_DEBUG** a la valeur **true**. On va la conserver ainsi puisqu'on veut être en mode « debug ». Vous êtes ainsi en mode débogage avec affichage de messages d'erreur détaillés. Si vous la mettez à **false** (ou si vous la supprimez), avec une URL non prévue vous obtenez maintenant juste :

Whoops, looks like something went wrong.

Il ne faudra évidemment pas laisser la valeur **true** lors d'une mise en production ! On en reparlera lorsqu'on verra la gestion de l'environnement. Vous ne risquerez ainsi plus d'oublier de changer cette valeur parce que Laravel saura si vous êtes sur votre serveur de développement ou sur celui de production.

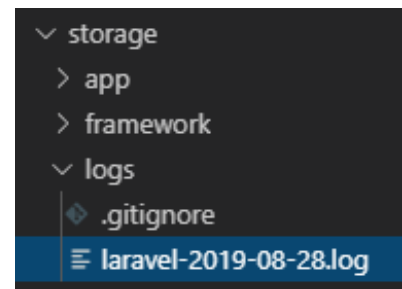
D'autre part il y a un fichier qui collecte les erreurs (Log), il n'existe pas à l'installation mais se crée à la première erreur signalée. Alors on va en créer une. Toujours dans le fichier **routes/web.php** modifiez ainsi le code :

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

On a changé le nom de la vue, du coup elle n'existe pas (on parlera plus tard des vues évidemment). Maintenant un fichier de Log a été créé ici :

On peut y lire :

```
[2019-08-28 12:32:47] local.ERROR: View [welcofme] not found.  
{ "exception": "[object] (InvalidArgumentException(code: 0): View  
[welcofme] not found. at
```



```
E:\\laragon\\www\\laravel6\\vendor\\laravel\\framework\\src\\Illuminate\\View\\FileViewFinder.php:137)
```

Par défaut il n'y a qu'un fichier mais si vous préférez avoir un fichier par jour par exemple il suffit d'ajouter cette ligne dans le fichier **.env** :

```
APP_LOG=daily
```

De la même manière par défaut Laravel stocke toutes les erreurs. C'est pratique dans la phase de développement mais en production vous pouvez limiter le niveau de sévérité des erreurs retenues (mode debug), par exemple si vous vous contentez des warning :

```
APP_LOG_LEVEL=warning
```

*La valeur de **APP_KEY** qui sécurise les informations est automatiquement générée lors de l'installation avec **create-project**.*

En résumé

- Pour son installation et sa mise à jour Laravel utilise le gestionnaire de dépendances **composer**.
- La création d'une application Laravel se fait à partir de la console avec une simple ligne de commande.
- Laravel est organisé en plusieurs dossiers.
- Le dossier **public** est le seul qui doit être accessible pour le client.

- L'environnement est fixé à l'aide du fichier **.env**.
- Par défaut Laravel est en mode « debug » avec affichage de toutes les erreurs.