25 mars

Tutoriel sur Laravel

Préparé par : Lydiane Beaulne-Bélisle

Ceci est un tutorial qui montre comment débuter avec le Framework PHP Laravel.

Créé pour le cours de Projet de fin d'étude
Collège de Maisonneuve
25 mars 2014

Table des matières

Laravel

I. Introduction

Laravel est un Framework PHP libre de droits qui a fait son apparition en 2011. Il est peut-être jeune comparé aux autres de son genre, mais il se démarque par sa facilité, sa syntaxe élégante, et toutes sa documentation disponible à tous. De plus, Laravel utilise la toute dernière version de PHP 5.3 et a fréquemment des patches de disponibles avec de nouveaux éléments et des mises à jour qui règlent les problèmes, ce qui prouve qu'il est en constante évolution et amélioration. En ce moment, il se base sur « Composer », le meilleur outil de dépendance qui gère des projets en PHP jusqu'à maintenant.

II. MVC

Laravel se base effectivement sur le patron de conception MVC, c'est-à-dire modèle-vue-contrôleur.

Le modèle interagit avec la base de données, les regroupe, traite et gère les données. La vue s'occupe principalement de faire afficher ce que le modèle renvoie. Ensuite, elle s'occupe de recevoir toute interaction de l'utilisateur (hover, clic de souris, entrée de texte, etc.). Ce sont ces actions-là que le contrôleur gère. Celui-ci prend en charge de synchroniser le modèle et la vue. Il capte toutes les activités de l'utilisateur et, en fonction de, il actionne les changements à effectuer sur le site.

En séparant les composants d'un site internet en ces trois catégories, cela permet une clarté de l'architecture des dossiers et simplifie grandement la tâche aux développeurs.

Installation de Laravel

I. Composer

La première étape serait d'installer le Composer dont laravel utilise. Cependant, qu'est-ce qu'un composer? Et bien c'est assez simple, Composer trouve les fichiers PHP qu'on a besoin dans un projet. Il va les chercher et les installer à la bonne place, pour nous. Comme Laravel est un Framework PHP, il est très pratique de l'utiliser pour bien partir un projet.

II. Installation de Composer

On peut le télécharger sur le site getcomposer.org ou directement sur le site de laravel.com. Ce Composer a une entente avec Laravel et va chercher tous les fichiers que Laravel utilise pour bien fonctionner. En installant le Composer, on pourra installer beaucoup plus facilement et rapidement le Framework Laravel, sans faire d'erreur. Une fois le fichier « Composer-Setup.exe » est installé dans l'ordinateur, on peut maintenant installer Laravel.

III. Installation de Laravel

Maintenant, il faut savoir où installer le projet. Je vous conseille de travailler en localhost, avant de travailler sur le serveur lui-même, pour simple efficacité. Ouvrez une fenêtre Windows de où vous voulez mettre le dossier de projet et faites click-droit, si vous avez bien installé le Composer, vous devriez voir dans la liste « use Composer here ». Vous le sélectionnez et une fenêtre de commande va apparaître. Vous devez inscrire la ligne de commande ci-dessous :

```
composer create-project laravel/laravel your-project-name –prefer-dist
```

Composer : signifie qu'on utiliser le Composer

create-project : signifie qu'on crée un projet

laravel/laravel : va chercher les fichiers à installer pour le Framework Laravel

your-project-name : on met le nom qu'on veut donner à notre projet

--prefer-dist : il y a plusieurs sous branches de Laravel, et cette commande-là va chercher la version complète de Laravel.

Exemple :

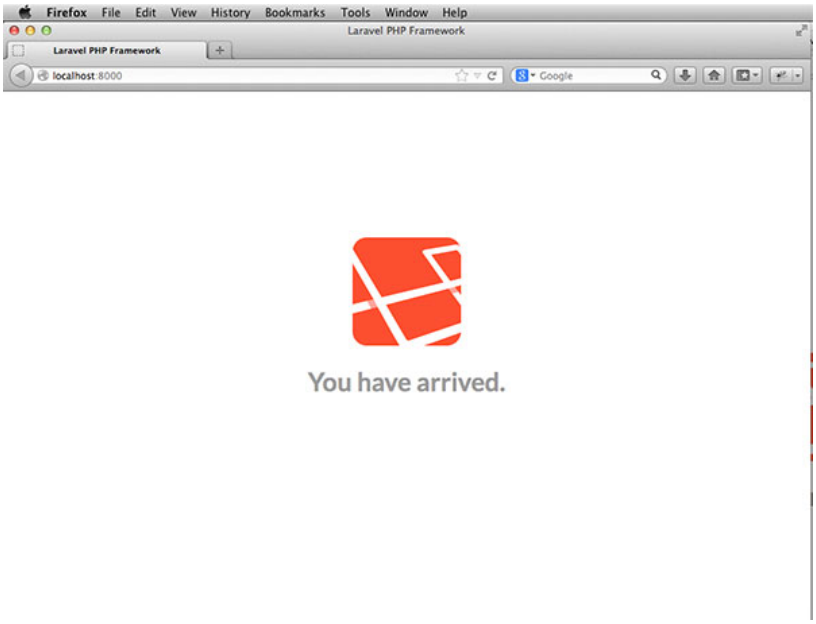
```
composer create-project laravel/laravel portfolio –prefer-dist
```

Quand vous avez fini, vous pesez sur la touche « enter » et cela va installer systématiquement Laravel au complet.

Une fois l'installation complété, testez si ça a bel et bien fonctionné. Pour ce faire, ouvrer votre fenêtre de navigation, google chrome ou firefox, et entrer votre lien pour le site soit :

```
localhost/nom-de-votre-dossier/public
```

Il faut absolument rajouter /public à la fin parce que le fichier index.php se retrouve dans ce dossier-là. Si tout a bien fonctionné, vous devriez voir le logo de Laravel en plein centre de la page.



Maintenant, vous êtes prêts à commencer votre site!

Utilisation de Laravel

I. Structure des dossiers

Il est important de d'abord analyser la structure des dossiers pour savoir comment la hiérarchie fonctionne. Voici les dossiers important à retenir :

Projet	app	config
		controllers
		models
		views
		routes.php
	public	css
		img
		js
		less

Il y a d'autres dossiers mais moins important quand on commence à utiliser Laravel.

Le dossier **app** contient tous les éléments nécessaires à la programmation « back-end » du site.

Le dossier **public**, lui, contient les médias et les autres langages de programmation, soit le CSS, le JS et il y a même un dossier **less** pour ceux qui veulent s'aventurer avec ce langage, ce que je conseil très fortement, less est très facile à utiliser et simplifie beaucoup l'apparence du css!

Il y a également un dossier pour **boostrap** si jamais on veut l'utiliser pour le site internet.

On peut voir ici que Laravel utilise les dernières nouveautés dans la programmation pour apporter la meilleure expérience aux utilisateurs de ce Framework.

II. Relier la base de données

Maintenant, il faut relier la base de données au projet. Dans le dossier **config**, il existe le fichier **database.php**. En l'ouvrant, on peut voir plusieurs array, dont un qui est associé à mysql.

```
'mysql' => array(
    'driver'    => 'mysql',
    'host'      => 'localhost',
    'database'  => '',
    'username'  => '',
    'password'  => '',
    'charset'   => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix'    => '',
),
```

Pour 'database' => il faut mettre ensuite le nom de la base de données, soit 'portfolio' dans mon cas.

Pour 'username' => notre nom d'utilisateur de la connexion à phpmyadmin, qui est 'root' pour moi.

Pour 'password' => le mot de passe.

Et pour le charset, il est par défaut en utf8.

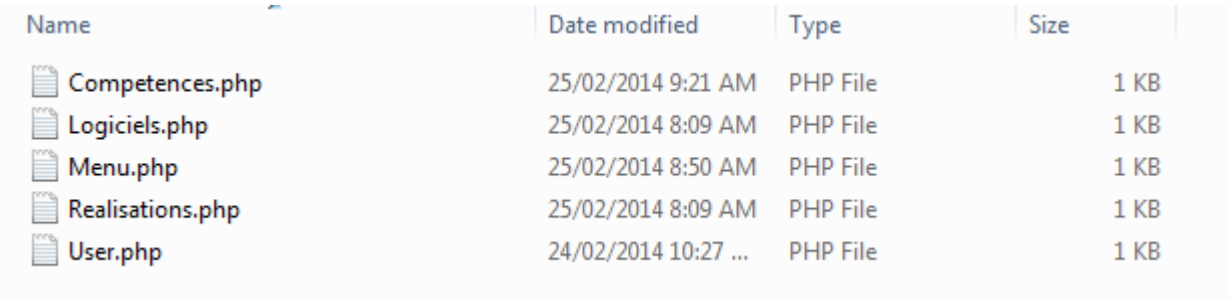
Comme il y a plusieurs type de connexion à une base de données disponible, il ne faut pas oublier de mettre par défaut le type de connexion que l'on va utiliser, soit 'mysql' comme ci-dessous, toujours dans le même fichier :

```
/*
|--------------------------------------------------------------------------
| Default Database Connection Name
|--------------------------------------------------------------------------
|
| Here you may specify which of the database connections below you wish
| to use as your default connection for all database work. Of course
| you may use many connections at once using the Database library.
|
*/

'default' => 'mysql',
```

III. Créer les models

Pour chaque table de notre base de données que l'on veut utiliser pour notre site, il faut créer un modèle pour chacun. Dans le dossier **app/models**, il y a User.php qui est un fichier de base.

Dans mon portfolio, j'ai plusieurs tables. Alors j'ai créé un fichier php à partir de User.php comme suit :



Name	Date modified	Type	Size
Competences.php	25/02/2014 9:21 AM	PHP File	1 KB
Logiciels.php	25/02/2014 8:09 AM	PHP File	1 KB
Menu.php	25/02/2014 8:50 AM	PHP File	1 KB
Realisations.php	25/02/2014 8:09 AM	PHP File	1 KB
User.php	24/02/2014 10:27 ...	PHP File	1 KB

Dépendamment des éléments et de ce qu'on veut faire avec, chaque fichier peut différer avec le niveau de complexité. Cependant, moi je veux seulement faire afficher mes données sur mon portfolio.

Dans mon **Menu.php**, je dois absolument déclarer deux variables. La première, je dois définir dans quelle table je vais chercher mes informations. Et, par la suite, je dois définir une clé première par défaut. Pour le moment, c'est tout ce que cette *class* a besoin.

```php
<?php


class Menu extends Eloquent {

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'menu';
    protected $primaryKey = 'idMenu';

}
```

Et on fait de même pour toutes les autres tables dans la base de données.

IV. Controllers

```php
<?php

class HomeController extends BaseController {

    /*
    |--------------------------------------------------------------------------
    | Default Home Controller
    |--------------------------------------------------------------------------
    |
    | You may wish to use controllers instead of, or in addition to, Closure
    | based routes. That's great! Here is an example controller method to
    | get you started. To route to this controller, just add the route:
    |
    |   Route::get('/', 'HomeController@showWelcome');
    |
    */

    public function showAccueil()
    {
        $data = array();
        $menu = Menu::all();
        $data["menu"] = array();


        //MENU
        foreach ($menu as $key => $value) {
            $data["menu"][] = $value->nomMenu."<br>";
        }

        return View::make('accueil')->with('data', $data);
    }

}
```

Dans le dossier **app/controllers**, il y a le fichier de base **HomeController.php**.

- On créer une fonction showAccueil, qui va contenir toutes les données que l'on veut afficher dans la page Accueil.
- On crée un array qui va contenir nos données de la base de données.
- On crée une variable qui va aller prendre toutes les données de la table 'Menu'

- On crée une autre variable $data['menu'] qui va être un array et qui va contenir les informations qu'on est allé chercher.
- Ensuite, ce qu'on veut retourner, ce sont les données qu'on a récupérer de la base de données. `return View ::make('accueil')->with('data', $data);`

  make('accueil') parce qu'on veut afficher les information dans la page **accueil**.

Ensuite, comme ce sont les données d'un tableau que l'on veut afficher dans la page, on doit faire un *foreach* qui va parcourir notre tableau, celui de $menu.

$value->nomMenu : nomMenu est le titre d'une catégorie dans la table Menu. On veut afficher chaque nom dans la table Menu.

Si ça avait été le idMenu qu'on aurait voulu faire afficher, ça aurait été :

```
$value->idMenu
```

V. Routes

Dans le fichier **Homecontroller.php**, il nous donne déjà la ligne à mettre dans le **routes.php**.

```
13
14  //Va chercher mon fichier HomeController.php et appel la fonction showAccueil.
15  Route::get('/', 'HomeController@showAccueil');
16
```

Comme je fais un exemple rapide, il n'y a qu'une seule route ici. Cependant, on peut en mettre plusieurs quand on a plusieurs pages.

VI. Créer la vue

Il nous reste maintenant seulement à créer la vue!

Dans le dossier **app/views**, on se créer un fichier .php qui contiendra du html5 dedans. On commence par créer une page bien simple, avec un *head* et un *body*.

Il ne faut pas oublier de rajouter le <!DOCTYPE html> pour que le html5 soit reconnu.

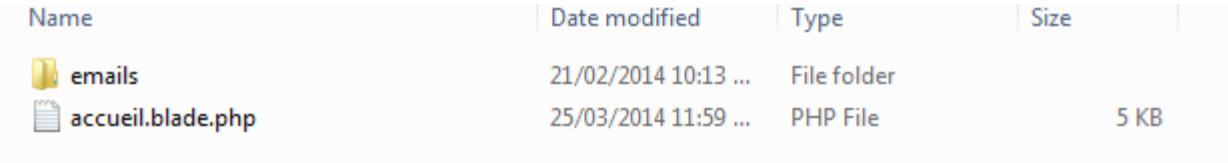
Comme c'est un menu que je veux faire afficher, je vais le mettre dans un <ul> et chaque donnée dans un <li>.

Il faut également parcourir notre tableau qui contient les noms du menu pour l'affichage. Ça nous prend donc un autre *foreach*.

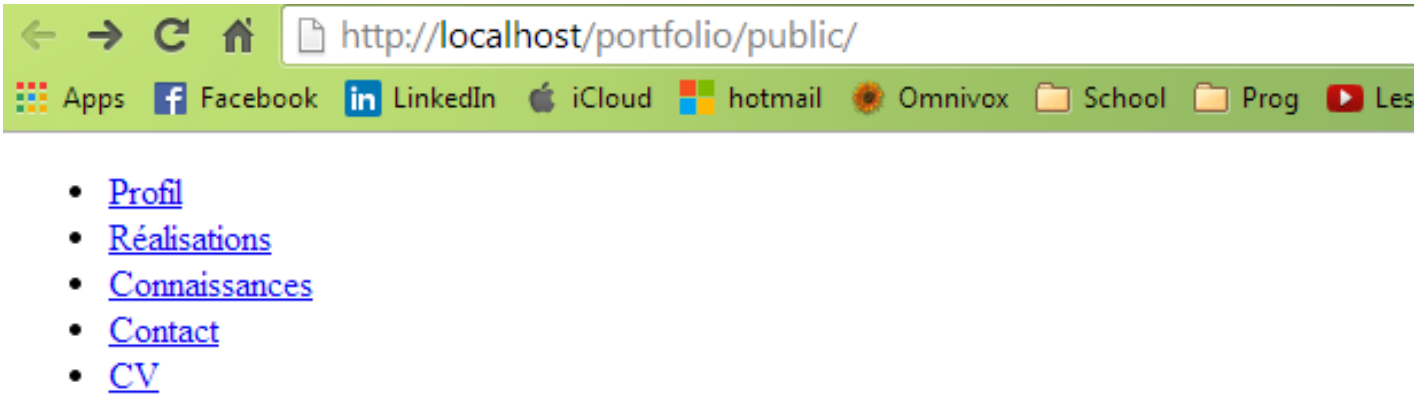
```
<!-- ____________________________ EN-TÊTE ____________________________ -->
<header>
    <section id="menu">

        <article id="menuList1">
            <ul>
                @foreach ($data["menu"] as $menus)
                    <li><a href="#">{{ $menus }}</a></li>
                @endforeach
            </ul>
        </article>

</header>
```

On peut remarquer qu'ici, le *foreach* n'est pas traditionnel. Effectivement, Laravel a sa propre syntaxe qui facilite la visibilité des informations dans le code. Cependant, pour pouvoir bénéficié ce système, il faut utiliser les « blades »

Pour ce faire, il suffit de rajouter au fichier **.blade.php** et le tour est joué!



Et maintenant, si on retourne sur notre site, on devrait voir la liste de notre menu!



Maintenant, votre site est prêt à être personnalisé!

Autres informations

Laravel permet de faire beaucoup plus mais ceci est seulement un tutoriel pour vous aider à débuter avec Laravel. Cependant, je vous conseille fortement de regarder :

- **Blade template**
- **Système d'utilisateur**
- **Système de connexion**
- **Less (à utiliser avec pour le CSS)**

Sources d'informations

Voici les sites qui m'ont vraiment aidé dans ma recherche :

1. C'est le site officiel de Laravel. Il est le plus important parce qu'il contient toutes les documentations nécessaires pour utiliser à plein potentiel ce Framework. Il nous montre également comment démarrer avec Laravel.
   Laravel, *Introduction*, [En ligne] http://laravel.com/docs/introduction (Page consultée le 25 mars 2014).
2. C'est le forum officiel de Laravel. On peut retrouver beaucoup d'aide et ça nous aide à comprendre mieux.
   Laravel.io, *Forum* , [En ligne] http://laravel.io/forum (Page consultée le 25 mars 2014).
3. Ce site contient plusieurs tutoriels. Certains tutoriels de Jeffrey Way sont pour aider à mieux comprendre Laravel et ils sont très bien fait et il explique très bien.
   Tutsplus, *Jeffrey Way*, [En ligne] https://tutsplus.com/author/jeffreyway/ (Page consultée le 25 mars 2014).
4. Pour bien comprendre Laravel, il faut que je comprenne d'abord ce qu'est un « Composer » puisqu'il se base sur ça pour gérer les projets.
   Getcomposer, *Getting Started*, [En ligne] https://getcomposer.org/doc/00-intro.md (Page consultée le 7 février 2014).
5. Ce site montre comment utiliser Blade de Laravel.
   Runnable, *Introducting to the Blade template engine [PHP and Laravel]*, [En ligne] http://runnable.com/UnApmfCNV4lmAAAz/-introducting-to-the-blade-template-engine-for-php-and-laravel (Page consultée le 25 mars 2014).
6. Comme Laravel se base sur le modèle MVC, il est fondamental de bien comprendre ce que c'est.
   Wikipédia, *Modèle-vue-contrôleur*, [En ligne] http://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur (Page consultée le 7 février 2014).