TP 5 : introduction à *Laravel*

Dans ce sujet, toutes les instructions concernant l'installation de *Laravel* et des applications tierces sont basées sur *Windows*. Pour les étudiants utilisant un autre système ou d'autres applications, merci de vous référez aux guides correspondants.

Pour ce TP, nous avons besoin des applications suivantes :

- *uWamp* : une distribution contenant un serveur Web, le moteur PHP, un SGBD
- *composer* : un gestionnaire de dépendances

I. Configuration du système

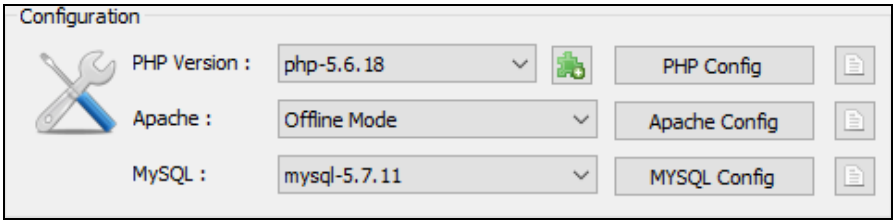
Pour pouvoir utiliser *Laravel*, il faut déjà s'assurer que l'ensemble des éléments nécessaires sont installés et configurés sur votre machine.

1. Installation de *uWamp*

Laravel nécessite l'utilisation de PHP en invite de commandes et un SGBD. Installez *uWamp* à l'adresse suivante :

https://www.uwamp.com/fr/?page=download

Par défaut, c'est la version PHP 5.6 qui est démarrée, comme illustré sur la capture ci-dessous :



Cliquez sur le bouton à côté de la version pour installer la dernière version possible. Sélectionnez le dépôt « *UwAmp PHP dépôt* », cochez la version php-7.2.7 et cliquez sur le bouton « *Installer* ». Une fois cette version installée, cliquez sur le bouton « *Fermer* ». Maintenant, vous pouvez sélectionner cette version dans la fenêtre précédente. Pour finir, vérifier que les erreurs PHP sont bien affichées. Cliquez sur le bouton « *PHP Config* ». Dans la nouvelle fenêtre, dans l'onglet « *PHP Setting* », cliquez sur *Display errors*. Vérifiez que la valeur est bien *On*, sinon modifiez-la.

2. Configuration de PHP

Jusqu'à présent, les scripts PHP étaient exécutés par le serveur Web. Pour *Laravel*, nous avons besoin du CLI PHP (la commande *php* dans l'invite de commandes). Pour cela, il faut que le chemin de `php.exe` soit renseigné dans la variable d'environnement *Path*. Avec une installation par défaut, il faut ajouter le chemin suivant : `C:\UwAmp\bin\php\php-7.2.7`

Ouvrez une invite de commandes et vérifiez que la commande *php* est fonctionnelle. Tapez la commande suivante qui affiche la version de PHP :

```
php -v
```

Maintenant, il faut configurer PHP. Sous *uWamp*, aucun fichier `php.ini` n'est spécifié pour le CLI. Vous pouvez en créez-un en copiant le fichier `php.ini-development` présent dans le répertoire de PHP et en le renommant `php.ini`. Éditez ensuite ce fichier et vérifiez les éléments suivants :

- La ligne `extension_dir = "ext"` ne doit pas être commentée pour *Windows* (retirez le ' ;' devant la ligne)
- Dans la rubrique « *Dynamic Extensions* », vérifiez que les extensions suivantes ne sont pas commentées :
    - bz2
    - curl
    - fileinfo
    - gd2
    - mbstring

- mysqli
- openssl
- pdo_mysql

Une fois le fichier créé et modifié, vérifiez dans l'invite de commande qu'il est bien pris en compte en tapant la commande suivante :

```
php --ini
```

Dans la ligne « *Loaded Configuration File* », le nom de votre fichier `php.ini` doit être indiqué.

3. Installation de *composer*

L'installation de *Laravel* utilise le logiciel *composer* qui est un gestionnaire de dépendances. Pour l'installer, suivez le guide sur la page suivante : https://getcomposer.org/download/

L'installation est effectuée directement en ligne de commandes à l'aide de PHP. Une fois l'installation terminée, vérifiez que *composer* est bien installé en tapant la commande suivante qui affiche la version :

```
composer -V
```

Si un des précédents éléments n'est pas fonctionnel correctement, il est inutile de passer à la suite.

II. Premier projet *Laravel*

1. Création du projet

Pour créer un projet *Laravel*, rendez-vous dans le répertoire `www` de *uWamp* et tapez la commande suivante :

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

L'installation de *Laravel* démarre, ce qui peut prendre plusieurs minutes. Une fois terminée, vous pouvez accéder à l'application créée en tapant l'URL suivante : http://localhost/exemple/public/ (pour rappel, le routeur est situé dans le répertoire `public` de votre projet). Vous devriez obtenir la page d'accueil par défaut de *Laravel* :

2. Configuration d'un hôte virtuel *Apache*

Avec la configuration actuelle, dès que vous spécifiez un lien dans vos scripts, vous devrez spécifier le chemin complet (le répertoire `exemple/public`). Pour simplifier, il est possible de définir des hôtes virtuels sous *Apache*. Nous ne décrirons par leur fonctionnement dans ce TP, mais ils permettent, dans notre cas, d'accéder directement à l'application en tapant l'URL : http://localhost/

Ouvrez *uWamp* et cliquez sur le bouton « *Apache config* ». Cliquez sur la liste des hôtes et sélectionnez `main-serveur *:80`. Dans le champ *Document Root*, ajoutez `exemple/public/` après `{DOCUMENTPATH}/`. En cliquant sur OK, le serveur *Apache* redémarre. À partir de maintenant, http://localhost/ doit afficher la page principale de votre application.

3. Installation de la barre de *debug*

Par défaut, dès que des erreurs d'exécution sont détectées, plusieurs fenêtres s'affichent dans le navigateur indiquant l'endroit où l'erreur a été générée. Parfois, il peut être compliqué de trouver le chemin exact du script où l'erreur a été générée (parfois, les erreurs entraînent des erreurs dans les scripts de *Laravel*). Il est donc conseillé d'installer la

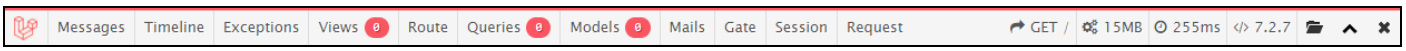
barre de *debug* dans votre application. Pour cela, avec l'invite de commandes, allez dans le répertoire de l'application (`www/exemple/`) et tapez la commande suivante :

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

Cela installe la barre de *debug*. Pour qu'elle soit prise en compte dans *Laravel*, tapez la commande suivante :

```
php artisan vendor:publish
```

Sélectionnez le numéro correspondant à la barre de *debug* (généralement 1) ou 0 pour tout publier. Maintenant, rechargez la page principale de votre application. Vous devriez voir apparaître la barre ci-dessous :



III. Jouer avec les routes de *Laravel*

1. Premières vues

Pour rappel, les routes sont spécifiées dans le script `web.php` situé dans le répertoire `routes`. Par défaut, vous avez le code suivant qui permet de spécifier la vue affichée par défaut (URL `/`) :

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

Il est possible de spécifier une autre vue à la place de *welcome*.

1. Créez une vue dans le répertoire `ressources/views/` nommée `index.php`. Dans ce fichier, placez-y un document HTML quelconque.
2. Modifiez le script `web.php` pour que votre page s'affiche par défaut. Vérifiez dans votre navigateur.
3. Ajouter du code PHP (n'oubliez pas les balises) et tapez du code incorrect. Vérifiez ce qui est affiché dans votre navigateur et dans la barre de *debug*.
4. Supprimez le code PHP incorrect. Ajoutez dans le document HTML un lien vers la page `page1.php`. Créez une nouvelle vue et modifiez le script `web.php` en conséquence. Vérifiez que vous pouvez bien accéder à cette page depuis la page d'accueil.

2. Vues paramétrées

Nous pouvons spécifier des vues paramétrées. Pour cela, ajoutez le code suivant dans `web.php` :

```
Route::get('article/{n}', function($n) {
        return view('article')->with('numero', $n);
    })->where('n', '[0-9]+');
```

Il est possible maintenant d'accéder à la vue `article.php` en saisissant l'URL suivante : http://localhost/article/X (où *X* est un entier).

1. Écrivez la vue `article.php`. Avec du code PHP, affichez le contenu de la variable `$numero`.
2. Vérifiez maintenant que vous pouvez accéder à cette vue et que le numéro correspond bien au numéro saisit dans l'URL. Vérifiez également le comportement de *Laravel* si vous saisissez autre chose qu'un entier.
3. Spécifiez maintenant votre page d'erreur à l'aide de la route suivante :

```
Route::fallback(function () {
        return view('erreur');
    });
```

3. Introduction à *Blade*

Pour rappel, *Laravel* propose un moteur de *template* nommé *Blade*. Il permet de créer des *templates* qui peuvent ensuite être utilisés dans des vues.

1. Renommez la vue `article.php` en `article.blade.php` (pour que le code soit interprété par *Blade*). Modifiez la vue et affichez le contenu de la variable `numero` en utilisant la notation {{ }}.

2. Créez maintenant un *template* nommé `template.blade.php`. Définissez 2 champs : *titre* (le titre de la page) et *contenu* (le contenu de la page).
3. Faites hériter `article.blade.php` de ce *template*. Vérifiez que la page s'affiche de la même manière.
4. Définissez maintenant un menu dans un script `menu.blade.php` (utilisez un menu *Bootstrap* classique). Incluez ce menu dans le *template* (la directive est `@include`).

4. Les contrôleurs

Comme vu en cours et en TD, il est préférable de créer des contrôleurs pour structurer votre code. Pour cela, il est possible d'utiliser *artisan* en tapant la commande suivante :

```
php artisan make:controller ArticleController
```

Vous devriez avoir maintenant un fichier `ArticleController.php` dans le répertoire `app/Http/Controllers`.

1. Créez ce contrôleur.

Par défaut, il n'y a aucune méthode.

2. Ajoutez la méthode *index* comme suit :

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

3. Modifiez la route comme suit :

```
Route::get('/', 'ArticleController@index');
```

4. Ajoutez la méthode *article* qui prend en paramètre un entier $n. Elle appelle la vue paramétrée `article.blade.php` comme vu dans le 2).
5. Modifiez la route comme suit :

```
Route::get('article_template/{n}', 'ArticleController@show')->where('n', '[0-9]+');
```

5. Pour ceux qui ont fini

Consultez la documentation *Laravel* pour voir les autres possibilités offertes par *Blade*.