Cours Laravel 5.5 – les bases – présentation générale

Dans ce premier chapitre je vais évoquer PHP, son historique rapide et sa situation actuelle. Je vais aussi expliquer l'intérêt d'utiliser un framework pour ce langage et surtout pourquoi j'ai choisi Laravel. J'évoquerai enfin le patron MVC et la Programmation Orientée Objet.

Un framework ?

Approche personnelle

PHP est un langage populaire et accessible. Il est facile à installer et présent chez tous les hébergeurs. C'est un langage riche et plutôt facile à aborder, surtout pour quelqu'un qui a déjà des bases en programmation. On peut réaliser rapidement une application web fonctionnelle grâce à lui. Mais le revers de cette simplicité est que bien souvent le code créé est confus, complexe, sans aucune cohérence. Il faut reconnaître que PHP n'encourage pas à organiser son code et rien n'oblige à le faire.

Lorsqu'on crée des applications PHP on finit par avoir des routines personnelles toutes prêtes pour les fonctionnalités récurrentes, par exemple pour gérer des pages de façon dynamique. Une fois qu'on a créé une fonction ou une classe pour réaliser une tâche il est naturel d'aller la chercher lorsque la même situation se présente. Puisque c'est une bibliothèque personnelle et qu'on est seul maître à bord il faut évidemment la mettre à jour lorsque c'est nécessaire, et c'est parfois fastidieux.

En général on a aussi une hiérarchie de dossiers à laquelle on est habitué et on la reproduit quand on commence le développement d'une nouvelle application. On se rend compte des fois que cette habitude a des effets pervers parce que la hiérarchie qu'on met ainsi en place de façon systématique n'est pas forcément la plus

adaptée.

En résumé l'approche personnelle est plutôt du bricolage à la hauteur de ses compétences et de sa disponibilité.

(Re)découvrir PHP

Lorsque j'ai découvert PHP à la fin du dernier millénaire (ça fait plus impressionnant dit comme ça) il en était à la version 3. C'était essentiellement un langage de script en général mélangé au HTML qui permettait de faire du templating, des accès aux données et du traitement. La version 4 en 2000 a apporté plus de stabilité et une ébauche de l'approche objet. Mais il a fallu attendre la version 5 en 2004 pour disposer d'un langage de programmation à la hauteur du standard existant pour les autres langages.

Cette évolution incite à perdre les mauvaises habitudes si on en avait. Un site comme http://www.phptherightway.com offre des pistes pour mettre en place de bonnes pratiques. Donc si vous êtes un bidouilleur de code PHP je vous conseille cette saine lecture qui devrait vous offrir un nouvel éclairage sur ce langage et surtout vous permettre de vous lancer de façon correcte dans le code de Laravel.

Un framework

D'après Wikipedia un framework informatique est un « ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel ». Autrement dit une base homogène avec des briques toutes prêtes à disposition. Il existe des frameworks pour tous les langages de programmation et en particulier pour PHP. En faire la liste serait laborieux tant il en existe !

L'utilité d'un framework est d'éviter de passer du temps à développer ce qui a déjà été fait par d'autres souvent plus compétents et qui a en plus été utilisé et validé par de nombreux utilisateurs. On peut imaginer un framework comme un ensemble d'outils à disposition. Par exemple je dois faire du routage pour

mon site, je prends un composant déjà tout prêt et qui a fait ses preuves et je l'utilise : gain de temps, fiabilité, mise à jour si nécessaire…

Il serait vraiment dommage de se passer d'un framework alors que le fait d'en utiliser un présente pratiquement uniquement des avantages.

Pourquoi Laravel ?

Constitution de Laravel

Laravel, créé par Taylor Otwel, initie une nouvelle façon de concevoir un framework en utilisant ce qui existe de mieux pour chaque fonctionnalité. Par exemple toute application web a besoin d'un système qui gère les requêtes HTTP. Plutôt que de réinventer quelque chose, le concepteur de Laravel a tout simplement utilisé celui de **Symfony** en l'étendant pour créer un système de routage efficace. De la même manière, l'envoi des emails se fait avec la bibliothèque **SwiftMailer**. En quelque sorte Otwel a fait son marché parmi toutes les bibliothèques disponibles. Nous verrons dans ce cours comment cela est réalisé. Mais Laravel ce n'est pas seulement le regroupement de bibliothèques existantes, c'est aussi de nombreux composants originaux et surtout une orchestration de tout ça.

Vous allez trouver dans Laravel :

- un système de routage (RESTFul et ressources),
- un créateur de requêtes SQL et un ORM,
- un moteur de template,
- un système d'authentification pour les connexions,
- un système de validation,
- un système de pagination,
- un système de migration pour les bases de données,
- un système d'envoi d'emails,
- un système de cache,
- un système de gestion des événements,

- un système d’autorisations,
- une gestion des sessions,
- un système de localisation,
- un système de notifications…

Et d’autres choses encore que nous allons découvrir ensemble. Il est probable que certains éléments de cette liste ne vous évoquent pas grand-chose, mais ce n’est pas important pour le moment, tout cela deviendra plus clair au fil des chapitres.

Le meilleur de PHP

Plonger dans le code de Laravel, c’est recevoir un cours de programmation tant le style est clair et élégant et le code bien organisé. La version actuelle de Laravel est la 5.5, elle nécessite au minimum la version 7 de PHP. Pour aborder de façon efficace ce framework, il serait souhaitable que vous soyez familiarisé avec ces notions :

- **les espaces de noms** : c’est une façon de bien ranger le code pour éviter des conflits de nommage. Laravel utilise cette possibilité de façon intensive. Tous les composants sont rangés dans des espaces de noms distincts, de même que l’application créée.
- **les fonctions anonymes** : ce sont des fonctions sans nom (souvent appelées closures) qui permettent d’améliorer le code. Les utilisateurs de Javascript y sont habitués. Les utilisateurs de PHP un peu moins parce qu’elle y sont plus récentes. Laravel les utilise aussi de façon systématique.
- **les méthodes magiques** : ce sont des méthodes qui n’ont pas été explicitement décrites dans une classe mais qui peuvent être appelées et résolues.
- **les interfaces** : une interface est un contrat de constitution des classes. En programmation objet c’est le sommet de la hiérarchie. Tous les composants de Laravel sont fondés sur des interfaces.
- **les traits** : c’est une façon d’ajouter des propriétés et méthodes à une classe sans passer par l’héritage, ce qui permet de passer outre certaines limitations de l’héritage

simple proposé par défaut par PHP.

Un framework n'est pas fait pour remplacer la connaissance d'un langage mais pour assister celui (ou celle) qui connaît déjà bien ce langage. Si vous avez des lacunes il vaut mieux les combler pour profiter pleinement de Laravel.

La documentation

Quand on s'intéresse à un framework il ne suffit pas qu'il soit riche et performant, il faut aussi que la documentation soit à la hauteur. C'est le cas pour Laravel. Vous trouverez la documentation sur le site officiel. Mais il existe de plus en plus de sources d'informations dont voici les principales :

- https://laravel.fr : site d'entraide francophone avec un forum actif.
- http://laravel.io : le forum officiel.
- http://cheats.jesse-obrien.ca : une page bien pratique qui résume toutes les commandes.
- http://www.laravel-tricks.com : un autre site d'astuces.
- http://packalyst.com : le rassemblement de tous les packages pour ajouter des fonctionnalités à Laravel.
- https://laracasts.com : de nombreux tutoriels vidéo en anglais dont un certain nombre en accès gratuit.
- https://github.com/chiraggude/awesome-laravel : une page comportant une multitude de liens intéressants.

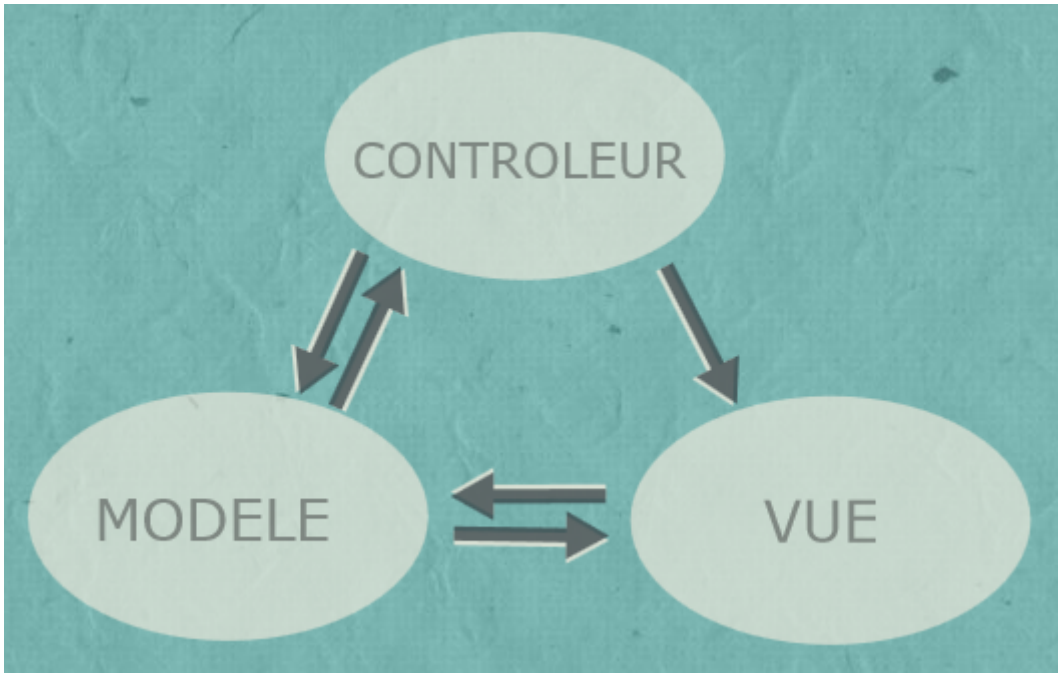
Il existe aussi de bon livres mais pratiquement tous en anglais.

MVC ? POO ?

MVC

On peut difficilement parler d'un framework sans évoquer le patron **Modèle-Vue-Contrôleur**. Pour certains il s'agit de la clé de voûte de toute application rigoureuse, pour d'autres c'est une contrainte qui empêche d'organiser judicieusement son code. De

quoi s'agit-il ? Voici un petit schéma pour y voir clair :



C'est un modèle d'organisation du code :

- le modèle est chargé de gérer les données,
- la vue est chargée de la mise en forme pour l'utilisateur,
- le contrôleur est chargé de gérer l'ensemble.

En général on résume en disant que le modèle gère la base de données, la vue produit les pages HTML et le contrôleur fait tout le reste. Dans Laravel :

- le modèle correspond à une table d'une base de données. C'est une classe qui étend la classe **Model** qui permet une gestion simple et efficace des manipulations de données et l'établissement automatisé de relations entre tables,
- le contrôleur se décline en deux catégories : contrôleur classique et contrôleur de ressource (je détaillerai évidemment tout ça dans le cours),
- la vue est soit un simple fichier avec du code HTML, soit un fichier utilisant le système de template **Blade** de Laravel.

Laravel propose ce patron mais ne l'impose pas. Nous verrons d'ailleurs qu'il est parfois judicieux de s'en éloigner parce qu'il y a des tas de chose qu'on n'arrive pas à caser dans cette organisation. Par exemple si je dois envoyer des emails où vais-je

placer mon code ? En général ce qui se produit est l'inflation des contrôleurs auxquels on demande des choses pour lesquelles ils ne sont pas faits.

POO

Laravel est fondamentalement orienté objet. La POO est un design pattern qui s'éloigne radicalement de la programmation procédurale. Avec la POO tout le code est placé dans des classes qui découlent d'interfaces qui établissent des contrats de fonctionnement. Avec la POO on manipule des objets.

Avec la POO, la responsabilité du fonctionnement est répartie dans des classes, alors que dans l'approche procédurale tout est mélangé. Le fait de répartir la responsabilité évite la duplication du code qui est le lot presque forcé de la programmation procédurale. Laravel pousse au maximum cette répartition en utilisant l'injection de dépendance.

L'utilisation de classes bien identifiées, dont chacune a un rôle précis, pilotées par des interfaces claires, dopées par l'injection de dépendances : tout cela crée un code élégant, efficace, lisible, facile à maintenir et à tester. C'est ce que Laravel propose. Alors vous pouvez évidemment greffer là dessus votre code approximatif, mais vous pouvez aussi vous inspirer des sources du framework pour améliorer votre style de programmation.

L'injection de dépendance est destinée à éviter de rendre les classes dépendantes et de privilégier une liaison dynamique plutôt que statique. Le résultat est un code plus lisible, plus facile à maintenir et à tester. Nous verrons ce mécanisme à l'œuvre dans Laravel.

En résumé

- Un framework fait gagner du temps et donne l'assurance de disposer de composants bien codés et fiables.
- Laravel est un framework novateur, complet, qui utilise les possibilités les plus récentes de PHP et qui est

impeccablement codé et organisé.

- La documentation de Laravel est complète, précise et de nombreux tutoriels et exemples sont disponibles sur la toile.
- Laravel adopte le patron MVC mais ne l’impose pas, il est totalement orienté objet.