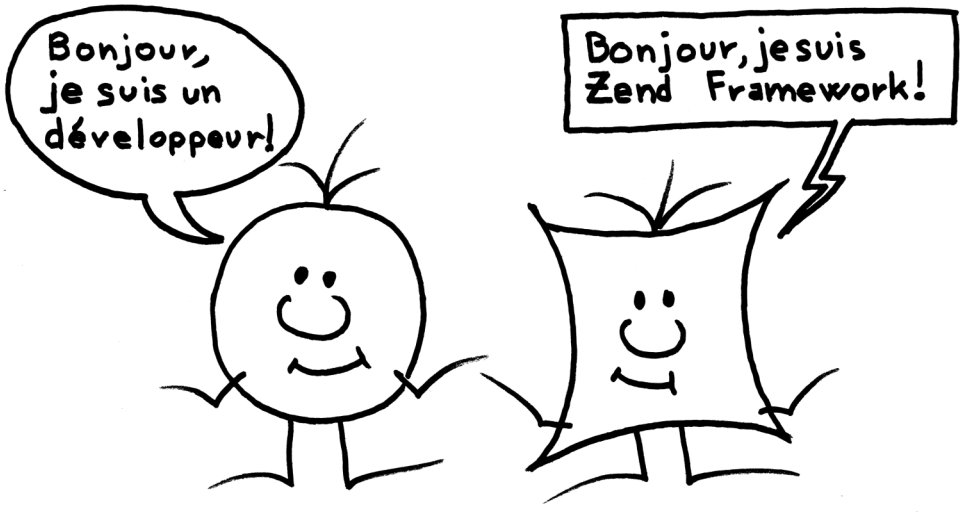
chapitre 1


Bonjour, je suis un développeur!
Bonjour, je suis Zend Framework!

Introduction à Zend Framework

Une entreprise, pour avancer, doit respecter des règles et disposer d'outils pour sa gestion et sa croissance. Pour les développements informatiques, c'est le framework qui joue ce rôle de cadre et de boîte à outils. Aujourd'hui, de nombreux frameworks et composants existent pour PHP, à tel point qu'il est difficile de faire un choix. Zend Framework, de plus en plus prisé, fait partie de ces alternatives. Est-il adapté à vos attentes et à vos méthodes de travail ?

SOMMAIRE

MOTS-CLÉS

- avantages
- inconvénients
- apports
- structure
- principe
- introduction

Ce chapitre résume l'intérêt de Zend Framework pour vos développements PHP. Son objectif est de s'assurer que cet outil est bien adapté à vos besoins et de vous donner les clés qui permettront de débuter efficacement.

Nous nous adressons ici aussi bien au technicien qui souhaite faire le choix d'un outil pour ses développements qu'au décideur qui souhaite en connaître les avantages stratégiques.

B.A.-BA Qu'est-ce qu'un framework ?

Framework signifie « cadre de travail » en français. Le principal objectif de cet outil est de proposer une démarche et des ressources pour mieux maîtriser les développements et gagner du temps. L'annexe A propose une introduction plus détaillée de ce qu'est un framework.

Avantages et inconvénients de Zend Framework

Il existe de nombreux frameworks pour PHP. Zend Framework se veut, comme PHP, simple et souple à utiliser, ce qui est plus ou moins le cas dans la réalité, comme nous le verrons par la suite. Comme tout framework, il propose des méthodes, des ressources et des outils. Il s'adapte à PHP pour améliorer la qualité et la fiabilité du code, dans une certaine mesure. De nombreuses solutions proposées aux problèmes courants sont simples, d'autres comportent une implémentation avancée qui nécessite un apprentissage préalable.

ALTERNATIVE Autres frameworks PHP

En France, les principaux autres frameworks que l'on trouve sur le marché des applications professionnelles sont les suivants :

- *Symfony* : un projet mûr qui propose une architecture solide, mais légèrement plus rigide. Il est appuyé par une grande communauté d'utilisateurs ainsi qu'une entreprise (Sensio).
- *Prado* : un framework sérieux qui propose une architecture intéressante et un fonctionnement très spécifique.
- *Copix* : un projet mûr à destination du monde professionnel, qui est capable de répondre à de nombreux besoins.
- *Jelix* : un framework français, comme Copix, de bonne qualité.
- *CodeIgniter* : un framework de plus en plus populaire pour sa simplicité et ses performances.

La souplesse de Zend Framework est telle que, quelle que soit la base choisie, une collaboration cohérente peut être mise en place avec d'autres frameworks ou composants.

Commençons par quelques avantages essentiels :

- une *communauté forte* qui assure une durabilité exceptionnelle autant que nécessaire. La pérennité des développements est fortement dépendante de celle du framework ;
- des *concepteurs expérimentés* et un *code source testé* pour une qualité de code garantie. Utiliser un outil fiable réduit considérablement les risques engagés ;
- un *support commercial et technique*, assuré par la société Zend, qui reste maîtresse des développements, un gage de pérennité et de fiabilité ;
- des *conventions claires et complètes* qui vont dans le sens du travail en équipe. Cela permet d'augmenter la vitesse de développement et de faciliter la reprise du projet à long terme ;
- des *composants souples*, de plus en plus nombreux et complets, avec peu d'interdépendance. Ces ressources couvrent tous les développements redondants et communs aux projets web, qui, grâce au framework, ne sont plus à redévelopper ;
- un *principe de fonctionnement simple* qui n'impose pas une structure rigide. Le développeur est guidé dans sa démarche, sans être contraint ni laissé pour compte ;

- une *installation et une prise en main simples et rapides*. Cette caractéristique réduit les risques dus au *turn-over*, c'est-à-dire le développement d'un projet par plusieurs développeurs qui se relaient ;
- la possibilité de s'adapter à n'importe quelle application ou d'adapter n'importe quelle ressource dans les composants. Cette absence de limitation est une véritable porte ouverte à l'innovation.

Les inconvénients que nous pouvons noter à l'heure actuelle sont les suivants :

- Zend Framework propose des ressources dites *de bas niveau*. En d'autres termes, le framework ne se considère pas comme un L4G qui permettrait de construire une application presque sans code, ce qui limiterait les possibilités d'innovation et de personnalisation ;
- rejoignant le point précédent, Zend Framework est *orienté composants*. Monter une application complète est complexe et requiert de bonnes notions de développement logiciel ;
- ce framework demande *un minimum de connaissances en programmation orientée objet* (POO) et *ne fonctionne pas avec PHP 4*. Il est important en particulier de comprendre les aspects théoriques de certains composants, ce qui est en partie le but de cet ouvrage.

L4G

L4G signifie « langage de 4e génération ». Le principe d'un L4G est de pouvoir développer quasiment sans toucher au code grâce à des outils, souvent graphiques, dits de *haut niveau*. Les langages de 3e génération permettent une meilleure compréhension du code par l'homme grâce à une syntaxe et des mots réservés. Il s'agit des langages procéduraux et objets : C, Java, PHP, etc. La deuxième génération comprend les langages de type assembleur et la première, le langage machine avec des 0 et des 1.

Structure et principe

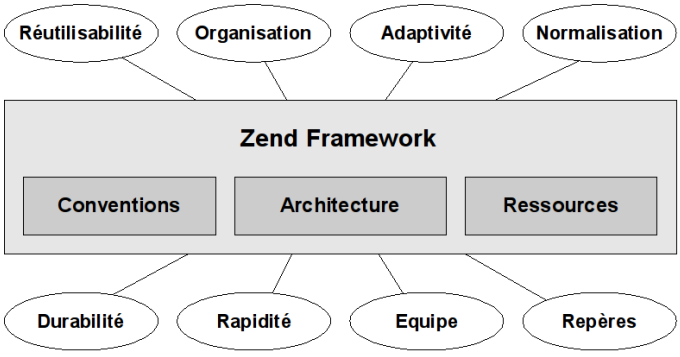


Figure 1–1
Apport de Zend Framework dans le développement web

D'un point de vue technique, Zend Framework apporte à PHP ce que les rails apportent à la locomotive : un support qui permet d'avancer efficacement dans les développements ; la figure 1-1 illustre ce principe. Les grandes lignes d'un framework sont les suivantes :

- fournir des règles de développement claires et précises (conventions) ;
- fournir des composants réutilisables (ressources) ;
- proposer un cadre technique pour le développement (architecture).

Les règles de développement

Elles décrivent comment organiser et écrire le code PHP :

- le *formatage des fichiers* – comment les fichiers sont-ils organisés et que contiennent-ils ?
- les *conventions de nommage* – comment nomme-t-on les fonctions, les classes, les variables, etc. ?
- le *style de codage* – toutes les règles qui décrivent la forme du code PHP : espaces, indentations, casse, etc.

À quoi servent les règles de développement ?

- à *travailler en équipe* – il est plus facile de relire du code dont on connaît les règles d'organisation et d'écriture ;
- à *gagner du temps* – en sachant où se situent les fonctionnalités et comment les trouver, de manière immédiate ;
- à *aller plus loin* – en ayant la maîtrise globale d'une application même lorsqu'il y a beaucoup de code source et de fonctionnalités dont on n'est pas forcément l'auteur.

MÉTHODE Règles de développement

Les règles de développement complètes de Zend Framework sont décrites dans la documentation officielle à l'adresse suivante :

- http://framework.zend.com/manual/fr/coding-standard.html

Si vous n'avez pas de règles précises, nous vous conseillons alors fortement de vous baser sur celles-ci.

Les composants réutilisables

Ils mettent à disposition des fonctionnalités courantes que l'on retrouve dans la plupart des applications web. Cela permet :

- d'éviter d'avoir à les développer ;
- d'avoir à disposition des fonctionnalités fiables, utiles et testées ;
- de minimiser la densité du code source, donc gagner du temps et ainsi optimiser le temps de mise sur le marché (*time to market* ou délai de lancement).

Composant

La majorité des chapitres de cet ouvrage est consacrée aux composants réutilisables.
Si Zend Framework en propose de nombreux prêts à l'emploi, il est toujours possible de les personnaliser ou de les compléter, mais aussi d'en créer de nouveaux, très facilement.

L'architecture

Elle est le squelette de l'application PHP –la base technique sur laquelle le développeur peut construire. L'architecture permet en particulier :

- d'avoir des repères essentiels pour s'organiser et gérer à terme un code source dense, avec des fonctionnalités nombreuses et complexes ;
- d'éviter de partir d'une page blanche grâce à la présence d'un socle technique et d'une méthodologie ;
- de favoriser l'adoption du même modèle pour le développement de plusieurs applications.

Nous verrons par la suite que l'architecture proposée par Zend Framework est assez souple pour disposer de plusieurs configurations possibles, notamment lorsque l'on doit gérer plusieurs applications ou une application qui doit être séparée en modules.

CULTURE Architecture et MVC

On entend souvent parler de ce modèle très populaire qu'est MVC (Modèle-Vue-Contrôleur). L'architecture est directement liée à cette notion qui est largement détaillée, en théorie dans l'annexe E et, en pratique, dans les chapitres 6 et 7.

Conseils pour bien démarrer avec Zend Framework

Avant de commencer avec Zend Framework, il est important d'être conscient de certains prérequis et de l'état d'esprit à adopter pour apprendre dans de bonnes conditions.

Prérequis

Une bonne compréhension de Zend Framework passe par la maîtrise d'un bon nombre de notions.

Afin de garantir l'acquisition rapide de ces notions par les débutants et d'assurer un contenu le plus précis et concis possible, voici une liste de ces notions complémentaires traitées en annexes.

- Le *framework* : il est important de bien comprendre les avantages à utiliser un framework ; si ce n'est pas encore le cas, reportez-vous à l'annexe A.
- Les *bases de données* : notamment, les notions de couche d'abstraction, de passerelles, d'ORM et de CRUD. Si vous ne les maîtrisez pas, alors l'annexe B vous permettra de comprendre leurs principes, nécessaires à l'apprentissage du composant `Zend_Db`.
- La *programmation orientée objet* : sujet aussi vaste que nécessaire ! Si vous ne maîtrisez pas les fondements de la POO, vos possibilités avec Zend Framework seront très limitées. L'annexe C permet à ceux qui ne maîtrisent pas cette notion d'aborder les bases nécessaires et d'aller plus loin si besoin.
- Les *design patterns* : ils permettent d'approfondir la compréhension de certains composants et de répondre à des solutions courantes de POO. Cette notion est abordée dans l'annexe D.
- Le *pattern MVC* : l'un des plus importants design patterns mérite un chapitre théorique. Comprendre MVC est nécessaire pour aborder l'architecture d'une application web dans son ensemble. MVC est traité dans l'annexe E.
- Le *langage PHP* : savoir comment fonctionne PHP est un gage de confort indéniable lorsqu'on l'utilise tous les jours, ne serait-ce que pour l'optimisation et la qualité de vos développements. Pour revoir ces aspects, lisez l'annexe F.
- Le *gestionnaire de sources Subversion* : le code source de Zend Framework est stocké dans un dépôt de données Subversion. Vos projets devraient eux aussi utiliser un gestionnaire de version, car cela présente des avantages précieux, détaillés dans l'annexe G.

CONSEIL Comprendre le fonctionnement du framework

Il est possible, en suivant des tutoriels, d'arriver à développer du code Zend Framework correct. Nous insistons par contre sur le fait que la compréhension du fonctionnement (interne) de Zend Framework est un avantage indéniable lorsqu'il s'agit d'écrire une application, quelle qu'en soit la complexité.

- Les *tests unitaires avec PHPUnit* : au cœur de la gestion de la qualité et de la maintenance du code se trouvent les tests unitaires. À quoi servent-ils et comment fonctionnent-ils ? À la lecture de l'annexe H, vous aurez l'essentiel en main pour aborder vos développements, muni de cette notion devenue essentielle.

État d'esprit

Loin de nous l'idée d'imposer ici une conduite restrictive, il s'agit juste de préciser une chose essentielle : les concepteurs de Zend Framework ont spécialement conçu cet outil pour être à l'image de PHP. Il est donc important de prendre en considération les faits suivants :

- *Zend Framework n'impose rien, il propose*. Libre à vous de partir dans la direction que vous souhaitez, tant au niveau de l'architecture que de l'utilisation des composants, ou même avec vos propres normes...
- *Zend Framework est simple*. Qu'il s'agisse de son architecture ou de la plupart de ses composants, l'idée n'est pas d'avoir à faire à une usine à gaz, mais à un outil qui permet de développer plus vite et plus facilement.

CULTURE Simplicité et souplesse

Ces deux notions, si elles ont l'avantage de permettre d'aller vite et de favoriser fortement l'innovation, ne sont pas totalement sans inconvénient. Se perdre dans une architecture brouillon est sans aucun doute le premier travers à éviter : il est nécessaire d'être organisé et rigoureux ! Dans cet ouvrage, nous mettons un point d'honneur à maintenir coûte que coûte la rigueur nécessaire à la production de développements cohérents.

En résumé

Zend Framework est un outil qui adopte la souplesse et, dans une certaine mesure, la simplicité de PHP. Mais comme tout outil, pour en récolter les meilleurs fruits, il est nécessaire de le maîtriser et de respecter ses règles. Nous attirons donc votre attention sur les points suivants :

- Zend Framework est un outil puissant, soutenu par une large communauté et prêt pour la mise en œuvre d'applications stratégiques.
- Cet outil comporte des conventions, une architecture modulaire qui favorise la réutilisabilité, autant de principes à ne pas perdre de vue !
- Enfin, une bonne connaissance de PHP 5 et de la programmation orientée objet, entre autres, est indispensable. Les annexes de cet ouvrage vous aideront à acquérir les prérequis nécessaires à la maîtrise de Zend Framework.