Avant-propos

Après plus de trois ans d'existence en tant que projet Open Source, Symfony est devenu l'un des frameworks incontournables de la scène PHP. Son adoption massive ne s'explique pas seulement par la richesse de ses fonctionnalités ; elle est aussi due à l'excellence de sa documentation – probablement l'une des meilleures pour un projet Open Source.

La sortie de la première version officielle de Symfony a été célébrée avec la publication en ligne du tutoriel *Askeet*, qui décrit la réalisation d'une application sous Symfony en 24 étapes prévues pour durer chacune une heure. Publié à Noël 2005, ce tutoriel devint un formidable outil de promotion du framework. Nombre de développeurs ont en effet appris à utiliser Symfony grâce à Askeet, et certaines sociétés l'utilisent encore comme support de formation.

Le temps passant, et avec l'arrivée de la version 1.2 de Symfony, il fut décidé de publier un nouveau tutoriel sur le même format qu'Askeet. Le tutoriel *Jobeet* fut ainsi publié jour après jour sur le blog officiel de Symfony, du 1er au 24 décembre 2008 ; vous lisez actuellement sa version éditée sous forme de livre papier.

COMMUNAUTÉ

Une étude de cas communautaire

Pour Askeet, il avait été demandé à la communauté des utilisateurs de Symfony de proposer une fonctionnalité à ajouter au site. L'initiative eut du succès et le choix se porta sur l'ajout d'un moteur de recherche. Le vœu de la communauté fut réalisé, et le chapitre consacré au moteur de recherche est d'ailleurs rapidement devenu l'un des plus populaires du tutoriel.

Dans le cas de Jobeet, l'hiver a été célébré le 21 décembre avec l'organisation d'un concours de design où chacun pouvait soumettre une charte graphique pour le site. Après un vote communautaire, la charte de l'agence américaine *centre{source}* fut choisie. C'est cette interface graphique qui sera intégrée tout au long de ce livre.

Découvrir l'étude de cas développée

Cet ouvrage décrit le développement d'un site web avec Symfony, depuis ses spécifications jusqu'à son déploiement en production, en 21 chapitres d'une heure environ. Au travers des besoins fonctionnels du site à développer, chaque chapitre sera l'occasion de présenter non seulement les fonctionnalités de Symfony mais également les bonnes pratiques du développement web.

L'application développée dans cet ouvrage aurait pu être un moteur de blog – exemple souvent choisi pour d'autres frameworks ou langages de programmation. Nous souhaitions cependant un projet plus riche et plus original, afin de démontrer qu'il est possible de développer facilement et rapidement des applications web professionnelles avec Symfony. C'est au chapitre 2 que vous en découvrirez les spécificités ; pour le moment, seul son nom de code est à mémoriser : *Jobeet*...

En quoi cet ouvrage est-il différent ?

On se souvient tous des débuts du langage PHP 4. C'était la belle époque du Web ! PHP a certainement été l'un des premiers langages de programmation dédié au Web et sûrement l'un des plus simples à maîtriser.

Mais les technologies web évoluant très vite, les développeurs ont besoin d'être en permanence à l'affût des dernières innovations et surtout des bonnes pratiques. La meilleure façon d'effectuer une veille technologique efficace est de lire des blogs d'experts, des tutoriels éprouvés et bien évidemment des ouvrages de qualité. Cependant, pour des langages aussi variés que le PHP, le Python, le Java, le Ruby, ou même le Perl, il est décevant de constater qu'un grand nombre de ces ouvrages présentent une lacune majeure... En effet, dès qu'il s'agit de montrer des exemples de code, ils laissent de côté des sujets primordiaux, et pallient le manque par des avertissements de ce genre :

- « Lors du développement d'un site, pensez aussi à la validation et la détection des erreurs » ;
- « Le lecteur veillera bien évidemment à ajouter la gestion de la sécurité » ;
- « L'écriture des tests est laissée à titre d'exercice au lecteur. »

Or chacune de ces questions – validation, sécurité, gestion des erreurs, tests – est primordiale dès qu'il s'agit d'écrire du code professionnel. Comment ne pas se sentir, en tant que lecteur, un peu abandonné ? Si ces contraintes – de surcroît les plus complexes à gérer pour un développeur – ne sont pas prises en compte, les exemples perdent de leur intérêt et de leur exemplarité !

Le livre que vous tenez entre les mains ne contient pas d'avertissement de ce type : une attention particulière est prêtée à l'écriture du code nécessaire pour gérer les erreurs et pour valider les données entrées par l'utilisateur. Du temps est également consacré à l'écriture de tests automatisés afin de valider les développements et les comportements attendus du système.

BONNE PRATIQUE **Réutilisez le code libre quand il est exemplaire !**

Le code que vous découvrirez dans ce livre peut servir de base à vos futurs développements ; n'hésitez surtout pas à en copier-coller des bouts pour vos propres besoins, voire à en récupérer des fonctionnalités complètes si vous le souhaitez.

Symfony fournit en standard des outils permettant au développeur de tenir compte de ces contraintes plus facilement et en étant parcimonieux en quantité de code. Une partie de cet ouvrage est consacrée à ces fonctionnalités car encore une fois, la validation des données, la gestion des erreurs, la sécurité et les tests automatisés sont ancrés au cœur même du framework – ce qui lui permet d'être employé y compris sur des projets de grande envergure.

Dans la philosophie de Symfony, les bonnes pratiques de développement ont donc part égale avec les nombreuses fonctionnalités du framework. Elles sont d'autant plus importantes que Symfony est utilisé pour le développement d'applications critiques en entreprise.

Organisation de l'ouvrage

Cet ouvrage est composé de vingt-et-un chapitres qui expliquent pas à pas la construction d'une application web professionnelle Open Source avec le framework Symfony. L'objectif de cette série de chapitres est de détailler une à une les fonctionnalités qui font le succès de Symfony, mais aussi et surtout de montrer ce qui fait de Symfony un outil professionnel, efficace et agréable à utiliser.

Le **chapitre 1** ouvre le bal avec l'installation et l'initialisation du projet Jobeet. Ces premières pages sont jalonnées en cinq parties majeures : le téléchargement et l'installation des librairies de Symfony, la génération de la structure de base du projet ainsi que celle de la première application, la configuration du serveur web et enfin l'installation d'un dépôt Subversion pour le contrôle du suivi du code source.

Le **chapitre 2** dresse le cahier des charges fonctionnelles de l'application développée au fil des chapitres. Les besoins fonctionnels majeurs de Jobeet y seront décrits un à un à l'aide de cas d'utilisation illustrés.

Le **chapitre 3** entame véritablement les hostilités en s'intéressant à la conception du modèle de la base de données, et à la construction automatique de cette dernière à partir de l'ORM Doctrine. L'intégralité du chapitre sera ponctuée par de nombreuses astuces techniques et bonnes pratiques de développement web. Ce chapitre s'achèvera enfin avec la génération du tout premier module fonctionnel de l'application à l'aide des tâches automatiques de Symfony.

Le **chapitre 4** aborde l'un des points clés du framework Symfony : l'implémentation du motif de conception Modèle Vue Contrôleur. Ces quelques pages expliqueront tous les avantages qu'apporte cette méthodologie éprouvée en termes d'organisation du code par rapport à une autre, et sera l'occasion de découvrir et de mettre en œuvre les couches de la Vue et du Contrôleur.

Le **chapitre 5** se consacre quant à lui à un autre sujet majeur de Symfony : le routage. Cet aspect du framework concerne la génération des URLs propres et la manière dont elles sont traitées en interne par Symfony. Ce chapitre sera donc l'occasion de présenter les différentes types de routes qu'il est possible de créer et de découvrir comment certaines d'entre elles sont capables d'interagir directement avec la base de données pour retrouver des objets qui leur sont liés.

Le **chapitre 6** est dédié à la manipulation de la couche du Modèle avec Symfony. Ce sera donc l'occasion de découvrir en détail comment le framework Symfony et l'ORM Doctrine permettent au développeur de manipuler une base de données en toute simplicité à l'aide d'objets plutôt que de requêtes SQL brutes. Ce chapitre met également l'accent sur une autre bonne pratique ancrée dans la philosophie du framework Symfony : le remaniement du code. Le but de cette partie du chapitre est de sensibiliser le lecteur à l'intérêt d'une constante remise en question de ses développements – lorsqu'il a la possibilité de l'améliorer et de le simplifier.

Le **chapitre 7** est une compilation de tous les sujets abordés précédemment puisqu'il y est question du modèle MVC, du routage et de la manipulation de la base de données par l'intermédiaire des objets. Toutefois, les pages de ce chapitre introduisent deux nouveaux concepts : la simplification du code de la Vue ainsi que la pagination des listes de résultats issus d'une base de données. De la même manière qu'au sixième chapitre, un remaniement régulier du code sera opéré afin de comprendre tous les bénéfices de cette bonne pratique de développement.

Le **chapitre 8** présente à son tour un sujet encore méconnu des développeurs professionnels mais particulièrement important pour garantir la qualité des développements : les tests unitaires. Ces quelques pages présentent tous les avantages de l'ajout de tests automatiques pour une application web, et expliquent de quelle manière ces derniers sont parfaitement intégrés au sein du framework Symfony via la librairie Open Source Lime.

Le **chapitre 9** fait immédiatement suite au précédent en se consacrant à un autre type de tests automatisés : les tests fonctionnels. L'objectif de ce chapitre est de présenter ce que sont véritablement les tests fonctionnels et ce qu'ils apportent comme garanties au cours du développement de l'application Jobeet. Symfony est en effet doté d'un sous-framework de tests fonctionnels puissant et simple à prendre en main, qui permet au développeur d'exécuter la simulation de l'expérience utilisateur dans son navigateur, puis d'analyser toutes les couches de l'application qui sont impliquées lors de ces scénarios.

Pour ne pas interrompre le lecteur dans sa lancée et sa soif d'apprentissage, le **chapitre 10** aborde l'importante notion de gestion des formulaires. Les formulaires constituent la principale partie dynamique d'une application web puisqu'elle permet à l'utilisateur final d'interagir avec le système. Bien que les formulaires soient faciles à mettre en place, leur gestion n'en demeure pas moins très complexe puisqu'elle implique des notions de validation de la saisie des utilisateurs, et donc de sécurité. Heureusement, Symfony intègre un sous-framework destiné aux formulaires capable de simplifier et d'automatiser leur gestion en toute sécurité.

Le **chapitre 11** agrège les connaissances acquises aux chapitres 9 et 10 en expliquant de quelle manière il est possible de tester fonctionnellement des formulaires avec Symfony. Par la même occasion, ce sera le moment idéal pour écrire une première tâche automatique de maintenance, exécutable en ligne de commande ou dans une tâche planifiée du serveur.

Le **chapitre 12** est l'un des plus importants de cet ouvrage puisqu'il fait le tour complet d'une des fonctionnalités les plus appréciées des développeurs Symfony : le générateur d'interface d'administration. En quelques minutes seulement, cet outil permettra de bâtir un espace complet et sécurisé de gestion des catégories et des offres d'emploi de Jobeet.

L'utilisateur est l'acteur principal dans une application puisque c'est lui qui interagit avec le serveur et qui récupère ce que ce dernier lui renvoie en retour. Par conséquent, le **chapitre 13** se dédie entièrement à lui et montre, entre autres, comment sauvegarder des informations persistantes dans la session de l'utilisateur, ou encore comment lui restreindre l'accès à certaines pages s'il n'est pas authentifié ou s'il ne dispose pas des droits d'accès nécessaires et suffisants. D'autre part, une série de remaniements du code sera réalisée pour simplifier davantage le code et le rendre testable.

Le **chapitre 14** s'intéresse à une puissante fonctionnalité du sous-framework de routage : le support des formats de sortie et l'architecture RESTful. À cette occasion, un module complet de génération de flux de syndication RSS/ATOM est développé en guise d'exemple afin de montrer avec quelle simplicité Symfony est capable de gérer nativement différents formats de sortie standards.

Le **chapitre 15** approfondit les connaissances sur le framework de routage et les formats de sortie en développant une API de services web destinés aux webmasters, qui leur permet d'interroger Jobeet afin d'en récupérer des résultats dans un format de sortie XML, JSON ou YAML. L'objectif est avant tout de montrer avec quelle aisance Symfony facilite la création de services web innovants grâce à son architecture RESTful.

Toute application dynamique qui se respecte comprend spontanément un moteur de recherche, et c'est exactement l'objectif du **chapitre 16**. En seulement quelques minutes, l'application Jobeet bénéficiera d'un moteur de recherche fonctionnel et testé, reposant sur le composant `Zend_Search_Lucene` du framework Open Source de la société Zend. C'est l'un des nombreux avantages de Symfony que de pouvoir accueillir simplement des composants tiers comme ceux du framework Zend.

Le **chapitre 17** améliore l'expérience utilisateur du moteur de recherche créé au chapitre précédent, en intégrant des composants JavaScript et Ajax non intrusifs, développés au moyen de l'excellente librairie jQuery. Grâce à ces codes JavaScript, l'utilisateur final de Jobeet bénéficiera d'un moteur de recherche dynamique qui filtre et rafraîchit la liste de résultats en temps réel à chaque fois qu'il saisira de nouveaux caractères dans le champ de recherche.

Le **chapitre 18** aborde un nouveau point commun aux applications web professionnelles : l'internationalisation et la localisation. Grâce à Symfony, l'application Jobeet se dotera d'une interface multilingue dont les contenus traduits seront gérés à la fois par Doctrine pour les informations dynamiques des catégories, et par le biais de catalogues XLIFF standards.

Le **chapitre 19** se consacre à la notion de plug-ins dans Symfony. Les plug-ins sont des composants réutilisables à travers les différents projets, et qui constituent également un moyen d'organisation du code différent de la structure par défaut proposée par Symfony. Par conséquent, les pages de ce chapitre expliquent pas à pas tout le processus de transformation de l'application Jobeet en plug-in complètement indépendant et réutilisable.

Le **chapitre 20** de cet ouvrage se consacre au puissant sous-framework de mise en cache des pages HTML afin de rendre l'application encore plus performante lorsqu'elle sera déployée en production au dernier chapitre. Ce chapitre est aussi l'occasion de découvrir de quelle manière de nouveaux environnements d'exécution peuvent être ajoutés au projet, puis soumis à des tests automatisés.

Enfin, le **chapitre 21** clôture cette étude de cas par la préparation de l'application à la dernière étape décisive d'un projet web : le déploiement en production. Les pages de ce chapitre introduisent tous les concepts de configuration du serveur web de production ainsi que les outils d'automatisation des déploiements tels que rsync.

Pour conclure, trois parties d'**annexes** sont disponibles à la fin de cet ouvrage pour en savoir plus sur la syntaxe du format YAML et sur les directives de paramétrage de deux fichiers de configuration de Symfony présents dans chaque application développée.