Développement d'applications Web avec le framework PHP Symfony 2

Mathieu Peltier
Mercator Océan (CNRS, Observatoire Midi-Pyrénées, UMS-831)
Parc Technologique du Canal - 8-10 rue Hermès
31520 Ramonville Saint-Agne

Résumé

Tout comme dans le cas d'autres langages, les frameworks de développement PHP, pour la plupart basés sur le modèle MVC (Modèle / Vue / Contrôleur), ont considérablement modifié la façon d'aborder le développement d'une application Web en fournissant une architecture prédéfinie, une méthode de conception et des outils permettant d'obtenir des programmes plus fonctionnels, maintenables, sécurisés et évolutifs.

La société Mercator Océan, dont le CNRS est un des membres fondateurs, est spécialisée dans le domaine de la prévision océanographique opérationnelle. Elle a choisi Symfony 2 pour développer des applications Intranet et Internet. Écrit en PHP 5, Symfony est un framework MVC libre comparable au Zend Framework ou à CakePHP intégrant plusieurs composants libres tiers. Il dispose d'une large communauté d'utilisateurs et d'une très bonne documentation. La version 2, sortie en juillet 2011, est une refonte complète des versions précédentes et dispose de fonctionnalités très complètes. Elle se présente sous la forme d'un assemblage de composants découplés, introduit un système de templates flexible et performant nommé Twig et utilise par défaut l'ORM (Object Relational Mapper) Doctrine. Symfony 2 promeut la modularité et la réutilisation des développements via le nouveau système de « bundles » et le « conteneur de services », et met également l'accent sur la facilité d'utilisation.

Malgré la durée de la phase d'apprentissage, l'utilisation d'un tel framework s'avère pertinente même pour des projets de petite taille et permet au développeur d'être plus productif et de progresser dans le domaine de la programmation Web.

Mots-clés

Développement, framework, MVC, PHP, Symfony, Web.

1 Introduction

Apparu en 1995, le langage PHP s'est rapidement imposé comme l'un des langages les plus populaires pour la programmation Web. Sa facilité d'apprentissage et sa souplesse l'ont paradoxalement sans doute desservi en cantonnant son image à celle d'un langage pour « amateurs », d'autant plus que des failles de sécurité peuvent être facilement introduites dans une application en l'absence d'une réelle maîtrise du langage. L'apparition des frameworks de développement Web, pour la plupart basés sur le modèle MVC (Modèle / Vue / Contrôleur) et initialement inspirés par l'emblématique projet Ruby on Rails[1], est en train de changer la situation (voir [1] et [2]). Un framework, parfois appelé cadre d'applications en français, est un ensemble de composants logiciels associé à une méthode de conception cohérente et éprouvée permettant de créer les fondations d'une application Web.

[1] Ruby on Rails est un framework de développement Web MVC libre écrit en Ruby et apparu en 2004.

Cet article détaillera d'abord les avantages de l'utilisation d'un framework et évoquera les raisons pour lesquelles le choix s'est porté sur Symfony pour certains développements au sein de la société Mercator Océan, spécialisée dans le domaine de la prévision océanographique opérationnelle. Il décrira ensuite les principaux concepts et fonctionnalités de Symfony 2, la dernière version stable du framework sortie en juillet 2011 qui est une refonte majeure des versions précédentes. Une courte démonstration illustrera enfin cette présentation.

2 Pourquoi Symfony ?

2.1 Intérêt d'un framework de développement

L'utilisation d'un framework de développement présente plusieurs avantages. En fournissant un certain nombre de briques logicielles permettant de résoudre les tâches récurrentes, elle permet au développeur de se concentrer sur le code métier de l'application. Mais le principal avantage est sans doute l'obtention d'applications plus fonctionnelles, maintenables, sécurisées et évolutives. Un framework fournit en effet une architecture prédéfinie et une méthode de conception cohérente et rigoureuse, respectant les bonnes pratiques en matière de génie logiciel. L'utilisation d'un framework MVC permet enfin de séparer les concepts et peut faciliter le travail en équipe sur les gros projets.

Cependant, quel que soit le framework choisi, il ne faut pas négliger le coût de la phase d'apprentissage, car l'adoption d'un framework modifie en profondeur les habitudes de développement.

2.2 Choix de Symfony

La société civile Mercator Océan, dont le CNRS est un des membres fondateurs, est spécialisée dans le domaine de la prévision océanographique opérationnelle. Les applications Web développées avec Symfony sont des applications de gestion (congés du personnel, saisie des temps sur projets pour le suivi de l'activité) s'interfaçant avec l'ERP (Enterprise Resource Planning) CEGID et la solution de courrier électronique collaborative OBM (Open Business Management), ainsi que des applications scientifiques (gestion du catalogue des produits diffusés).

De très nombreux frameworks PHP existent[2]. On peut citer par exemple CakePHP ou le Zend Framework. Le livre blanc [1] propose un ensemble de critères pour choisir le framework répondant aux exigences d'un projet donné. Le choix s'est porté sur Symfony au sein de Mercator Océan pour plusieurs raisons. Lancé sous la forme d'un logiciel libre en 2005 (license MIT), Symfony est sponsorisé par la société française Sensio Labs et intégre plusieurs composants libres tiers éprouvés. Connu pour ses performances, il est largement utilisé, y compris par de très gros sites tels que Yahoo! pour ses sites Yahoo! Bookmarks et Yahoo! Answers, ou encore Dailymotion pour son service d'hébergement et de partage de vidéos en ligne. Symfony est également utilisé par des applications comme par exemple phpBB, un moteur de forum libre très populaire écrit en PHP. Il dispose d'une communauté de développeurs, contributeurs et utilisateurs mondiale et active, ce qui est un gage de pérennité. Enfin c'est peut être par la qualité et la pédagogie de sa documentation que ce framework se distingue[3].

---

[2] Une liste plus ou moins exhaustive des frameworks PHP existants est disponible à l'adresse suivante : fr.wikipedia.org/wiki/Liste_de_frameworks_PHP.

[3] De nombreux livres et autres ressources sont disponibles pour Symfony 1 et l'on peut penser qu'il en sera de même pour Symfony 2.

3 Présentation de Symfony 2

3.1 Prérequis et installation

Symfony 2 peut être téléchargé sous la forme de « distributions » qui contiennent le cœur du framework constitué de composants modulaires, une sélection de « bundles »[4] additionnels et (de façon optionnelle) les librairies tierces utilisées (par exemple Doctrine ou SwiftMailer). Seule la distribution nommée « Symfony Standard Edition » est pour l'instant disponible. Elle contient le script PHP bin/vendors qui permet de télécharger ou de mettre à jour les librairies tierces (contenues dans le répertoire vendor) à partir des dépôts Git définis dans les fichiers deps et deps.lock. Symfony et la plupart des bundles et librairies tierces disponibles utilisent en effet le système de configuration Git et sont hébergés par le site github.com.

Symfony 2 peut être utilisé avec une base de données relationnelle (MySQL, PostgreSQL, SQLite, etc.) et s'installe sans difficulté par exemple sur une plateforme LAMP classique. La distribution standard inclut deux scripts exécutables en ligne de commande (app/check.php) et via le serveur Web (web/config.php) qui permettent de vérifier les prérequis nécessaires à l'installation, le plus important étant l'utilisation de PHP 5.3.2 ou supérieur[5]. Symfony fournit en effet une console d'administration (app/console) écrite en PHP et utilisable en ligne de commande nécessaire à l'utilisation du framework au jour le jour et la configuration de PHP en ligne de commande peut être différente de celle du serveur Web. Les répertoires app/cache et app/logs doivent être accessibles en lecture / écriture à l'utilisateur exécutant le serveur Web ainsi qu'à celui exécutant des commandes avec la console. Le second script web/config.php permet également de configurer la base de données. Enfin, pour des raisons de sécurité, il est important que seul le répertoire web de la distribution soit accessible publiquement.

3.2 Contenu de la distribution standard de Symfony 2

La distribution standard contient les fichiers suivants :

```
Symfony/ <- archive décompressée (exemple Symfony_Standard_Vendors_2.0.0.tgz)
    app/ <- configuration globale de l'application, répertoire du cache et des logs, console d'administration, etc.
        cache/, config/, logs/, Resources/
        autoload.php, check.php, console, phpunit.xml.dist, ...
    bin/ <- utilitaires divers
    src/ <- répertoire à utiliser pour les développements de nouveaux bundles
    vendor/ <- code du framework et des librairies tierces (Assetic, Doctrine, SwiftMailer, etc.)
    web/ <- seul répertoire à exposer publiquement au sein du serveur Web
        bundles/ <- fichiers statiques (CSS, JavaScripts, images, etc.)
        app.php, app_dev.php, config.php, ... <- « front controllers »
    deps, deps.lock, ...
```

3.3 Architecture MVC

Symfony 2 est construit selon une architecture MVC (Modèle / Vue / Contrôleur) (voir [3]). Le modèle gère l'ensemble des interactions avec la base de données. Il fournit une abstraction des données et contient le code métier. Le contrôleur est responsable de la logique de contrôle de l'application. En interaction avec le modèle, il génère les vues qui représentent l'interface utilisateur.

---

[4] Voir le paragraphe 3.5 pour plus d'informations sur les bundles.

[5] L'utilisation d'un accélérateur PHP (APC, eAccelerator, etc.) est également fortement conseillée. Pour utiliser Doctrine, l'extension PHP PDO (PHP Data Objects) ainsi que le driver PDO associé à la base de données cible sont aussi requis.

La figure 1 décrit le flux de traitement de chaque requête HTTP. Le contrôleur en amont, unique, est appelé « front controller ». L'implémentation fournie par défaut web/app.php n'a que rarement besoin d'être modifiée. Selon la configuration du système de routage, il va déléguer la génération de la réponse à un contrôleur donné et plus précisément à une action donnée (fonction PHP). Chaque requête est ainsi associée à une « route » qui définit l'URL à utiliser pour accéder à la ressource et le contrôleur qui devra générer la réponse. Une route peut aussi inclure des paramètres et des conditions sur ces paramètres ou la méthode HTTP attendue. Les valeurs des paramètres sont automatiquement passées au contrôleur. Symfony supporte ainsi virtuellement n'importe quel type d'URL et permet de gérer facilement des URLs « propres » importantes pour le confort de l'utilisateur et le référencement. Les URLs ne sont en effet jamais « hardcodées » dans les vues et peuvent être ainsi facilement modifiées.

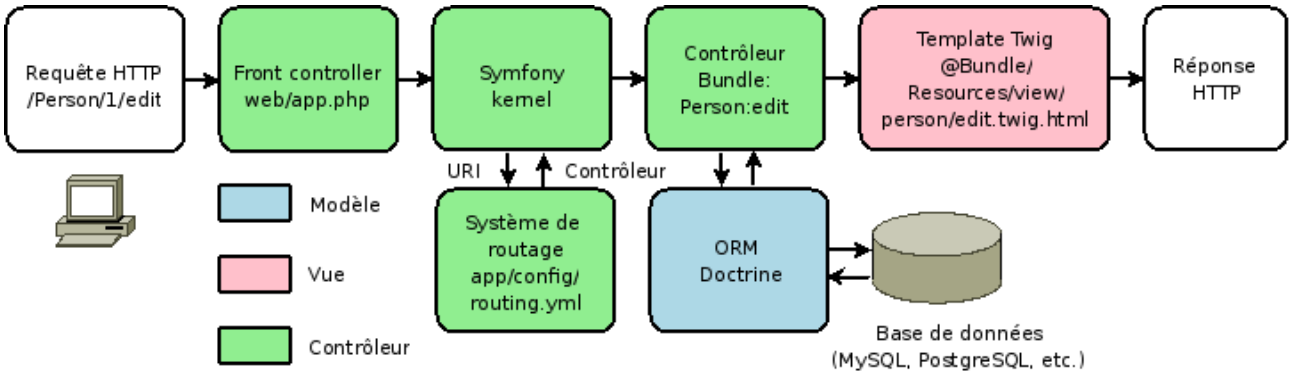

Figure 1 - Flux de traitement d'une requête HTTP dans Symfony 2.

Par défaut, Symfony 2 utilise l'ORM (Object Relational Mapper) Doctrine pour interagir avec la base de données relationnelle[6]. A partir de métadonnées décrivant les objets persistants de l'application, appelés « entités », fournies par le développeur par configuration, un ORM permet une correspondance entre la programmation orientée objet et la base de données relationnelle. Le développeur est ainsi déchargé des tâches de gestion de la persistance, toujours complexes, et l'application est plus portable : à tout moment la base de données utilisée peut être changée par configuration sans modifier le code. Doctrine est construit à partir d'une couche d'abstraction de la base de données (Doctrine Database Abstraction Layer) et fournit un dialecte SQL appelé Doctrine Query Language (DQL).

Enfin, même si les vues peuvent être toujours écrites en PHP, Symfony 2 utilise par défaut un nouveau système de templates plus flexible et performant nommé Twig. Ce système supporte par exemple l'héritage et un template peut ainsi redéfinir certaines parties d'un template parent. L'utilisation de Twig protège également par défaut le développeur contre les attaques de type CSS (Cross-Site Scripting)[7].

3.4 Fichiers de configuration

Les principaux fichiers de configuration de Symfony 2, app/config/config.yml, app/config/security.yml et app/config/routing.yml, permettent de spécifier respectivement le comportement du framework lui-même (moteurs de templates, validation des formulaires, sessions, internationalisation, paramètres d'accès à la base de données et d'envoi des mails, etc.), la sécurité et le système de routage. Chaque bundle peut également définir ses propres fichiers de configuration.

---

[6] Doctrine permet aussi d'utiliser les bases de données orientées documents MongoDB ou CouchDB. Il est également possible de n'utiliser que la couche d'abstraction de la base de données, afin d'écrire ses propres requêtes pour assurer la persistance des objets dans une base de données relationnelle ou bien simplement manipuler les données.

[7] Une attaque de type CSS (parfois appelée aussi XSS) consiste à injecter du code du côté client et à le faire exécuter par un utilisateur légitime à son insu.

Tous ces fichiers de configuration peuvent être écrits en PHP, XML ou YAML, le choix étant laissé à l'appréciation du développeur. YAML est un format standard de sérialisation de données simple et lisible basé sur Unicode. Des annotations peuvent être également insérées directement au niveau du code source PHP, par exemple pour décrire le mapping Doctrine et les contraintes de validations à positionner sur les entités[8]. Quel que soit le format choisi, ces informations sont analysées, traduites en PHP puis mises en cache lors de la première utilisation pour des raisons de performances (tout comme les vues Twig).

Symfony introduit également la notion d' « environnement de configuration », très utile pour travailler sur le même projet avec une configuration différente selon le contexte (développement, test, production). Chaque fichier de configuration peut être ainsi redéfini ou étendu : par exemple, le fichier app/config/config_dev.yml modifie le comportement du fichier app/config/config.yml. Un environnement donné est activé en sélectionnant le front controller nommé web/app_monenv.php (par exemple web/app_dev.php pour l'environnement dev).

3.5 Système de bundle

Symfony 2 met l'accent sur la modularité en introduisant la notion de « bundle ». Un bundle est un ensemble de fichiers respectant une organisation standardisée et implémentant une fonctionnalité donnée. En pratique, un bundle sera créé pour chaque application développée et certaines fonctionnalités du framework lui-même sont livrées de façon modulaire sous cette forme. Les bundles peuvent être utilisés de façon indépendante dans d'autres projets. Le site symfony2bundles.org propose un ensemble de bundles développés par la communauté. Le fichier app/autoload.php permet au framework de trouver automatiquement les classes associées aux namespaces des bundles utilisés sans avoir à inclure explicitement les fichiers PHP. Symfony 2 tire en effet parti des espaces de noms introduits par PHP 5.3.

3.6 Principales autres fonctionnalités

3.6.1 Sécurité

Le composant dédié à la sécurité de Symfony 2 permet de configurer l'authentification des utilisateurs (via un simple formulaire d'authentification, par HTTP ou encore via des mécanismes plus avancés comme les certificats X.509) et les autorisations.

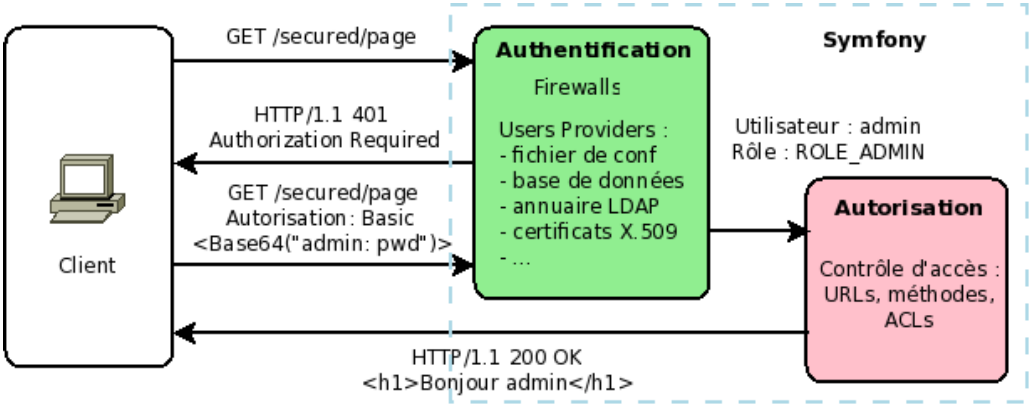

Figure 2 - Accès sécurisé à une page restreinte aux administrateurs d'un site.

Chaque mécanisme d'authentification est appelé firewall et le contrôle d'accès (autorisation) à appliquer en fonction des rôles affectés à chaque utilisateur est configurable par URLs, par contrôleurs ou encore via les ACLs (Access Control Lists). Les ACLs permettent un contrôle plus fin au niveau des instances d'un objet (par exemple pour donner aux utilisateurs d'un blog les

[8] Le bundle SensioFrameworkExtraBundle introduit également d'autres types d'annotations (par exemple pour définir les routes directement dans le code des contrôleurs).

droits d'édition sur leur propres articles sans leur donner les droits d'édition sur tous les articles). La figure 2 illustre le processus d'accès à une page restreinte aux administrateurs d'un site.

3.6.2 Formulaires

Symfony 2 est livré avec un nouveau composant destiné à construire les formulaires. Un grand nombre de types (nombres, dates, listes de choix, etc.) est géré par défaut et de nouveaux types peuvent être créés. Un système de « thèmes » permet de contrôler l'apparence du code HTML généré. La validation des formulaires est gérée par un autre composant à partir des règles associées à chaque objet entité fournies par le développeur par configuration. Enfin, les formulaires sont protégés par défaut des attaques de type CSRF (Cross-Site Request Forgery)[9].

3.6.3 Tests et Web Debug Toolbar

Symfony 2 utilise PHPUnit pour l'écriture des tests unitaires et fonctionnels d'une application[10]. Pour les tests fonctionnels, le framework fournit des objets permettant de simuler un client HTML et d'auditer chaque page générée (via une classe appelée `Crawler`).

L'utilisation de l'environnement `dev` (front controller `web/app_dev.php`) active la « Web Debug Toolbar » et le « Profiler », qui permettent de « débugger » de façon efficace une application en affichant par exemple des informations sur la requête HTTP, la trace d'exécution complète en cas d'erreur, le temps d'exécution, les requêtes SQL exécutées pour générer la page, etc. De plus, le cache est automatiquement désactivé dans cet environnement.

3.6.4 Internationalisation

L'internationalisation d'une application, appelée aussi « i18n», consiste à abstraire l'ensemble des chaînes de caractères et autres données (dates, monnaies, etc.) afin de pouvoir gérer plusieurs langages et cultures. Symfony 2 dispose d'un composant gérant l'internationalisation et supportant des catalogues de messages écrits en XLIFF (XML Localisation Interchange File Format, le format recommandé), PHP ou YAML.

3.6.5 Support de JavaScript et d'Ajax

L'utilisation de JavaScript et d'Ajax ne nécessite aucun prérequis particulier dans Symfony 2 et n'importe quelle librairie (jQuery, Prototype, etc.) peut être utilisée. Symfony encourage l'utilisation des techniques dites de « JavaScript discret » (unobtrusive JavaScript) qui consistent principalement à découpler les codes JavaScript et HTML. Dans le cas d'Ajax, le système de routage utilise un paramètre spécial nommé `_format` permettant de générer la réponse dans le bon format (par exemple XML ou JSON) en sélectionnant la vue ad hoc.

Symfony 2 intègre la librairie PHP Assetic qui permet de contrôler l'importation des scripts JavaScript utilisés par une application. Tous les scripts utilisés peuvent être ainsi regroupés en un seul fichier pour de meilleurs performances et des filtres peuvent être par ailleurs appliqués, comme par exemple le compresseur JavaScript de Yahoo! YUI Compressor[11].

3.6.6 Mécanismes de cache

Symfony 2 utilise les mécanismes de cache définis par le protocole HTTP. La distribution standard contient un reverse proxy écrit en PHP. Ce gestionnaire de cache supporte le standard

---

[9] Une attaque de type CSRF (parfois appelée aussi XSRF) consiste à faire exécuter du code existant à l'insu d'un utilisateur légitime pour par exemple lui faire soumettre des données.

[10] PHPUnit version 3.5.0 ou supérieure est requis et doit être installé indépendamment de Symfony 2.

[11] Voir le site developer.yahoo.com/yui/compressor pour plus d'informations. Assetic permet en fait également de contrôler l'importation des feuilles de style ou encore des images utilisées par la page de la même façon.

Edge Side Includes (ESI) qui permet de configurer un temps d'expiration en cache d'un fragment de page indépendant de celui de la page (par exemple pour gérer l'affichage des dernières nouvelles sur un site). Pour l'activer, il suffit de modifier le front controller utilisé afin d'utiliser la classe AppCache. Pour des performances optimales, d'autres gestionnaires de cache comme Varnish ou Squid peuvent être aussi utilisés, sans qu'aucune modification du code ne soit nécessaire.

3.6.7 Générateurs de code

Tout comme la plupart des frameworks, Symfony 2 propose des générateurs de code. La distribution standard inclut le bundle SensioGeneratorBundle qui permet d'obtenir une structure de base simple de création, mise à jour et suppression des données dans la base, que le développeur peut ensuite personnaliser et adapter. Ce type de structure de type « CRUD » (Create, Read, Update, Delete) est appelée « scaffolding » (échafaudage en français). D'autres bundles tiers peuvent être aussi utilisées[12].

3.6.8 Services et conteneur de services

Symfony 2 encourage la création de « services » afin d'améliorer la modularité et la ré-utilisabilité du code. Un service est en fait une simple classe PHP implémentant une fonctionnalité donnée (par exemple l'envoi de notifications par mail). Le framework fournit un conteneur de services permettant de gérer les services utilisés et leur configuration. Le patron de conception appelé injection de dépendances est utilisé afin de découpler les dépendances entre services. Celles-ci sont ainsi définies dynamiquement à l'exécution dans des fichiers de configuration et non de manière statique dans le code.

4 Démonstration

Afin d'illustrer la présentation précédente, on se propose dans cette section d'utiliser le générateur de code fourni par le bundle SensioGeneratorBundle afin de créer une application minimaliste permettant de gérer un carnet d'adresses. On suppose que la distribution standard de Symfony 2 a été installée et qu'une base de données a été configurée. La première étape consiste à créer le bundle MercatorOceanDemoBundle qui contiendra cette application :

```
$ php app/console generate:bundle --namespace=MercatorOcean/DemoBundle --dir=src  \
--bundle-name=MercatorOceanDemoBundle --format=annotation --structure --no-interaction
```

Cette commande, qui peut aussi fonctionner de manière interactive, génère l'arborescence suivante dans le répertoire src/MercatorOcean/DemoBundle/ :

```
Controller/ -> contrôleurs du bundle
DependencyInjection/ ->configuration avancée des dépendances entre services
Ressources/config/ -> configuration (routes, services, etc.)
        doc/ -> documentation (au format reStructuredText par défaut)
        public/ -> fichiers statiques (images, CSS, JavaScripts, etc.)
        translations/ -> fichiers nécessaires à l'internationalisation (XLIFF, YAML ou PHP)
        views/ -> vues utilisées (templates Twig ou PHP)
Tests/ -> tests unitaires et fonctionnels associés au bundle
```

Elle modifie également les fichiers app/AppKernel.php et app/config/routing.yml afin de créer une instance du nouveau bundle lors de l'initialisation du framework et de rajouter la route

[12] Le bundle SonataAdminBundle téléchargeable sur le site symfony2bundles.org permet par exemple de générer une interface d'administration configurable et extensible avec des fonctionnalités avancées (pagination, critères de recherche, support des relations entre entités de type un ou plusieurs à plusieurs, etc.).

nécessaire pour accéder à son contrôleur par défaut. La seconde étape est de créer la classe MercatorOcean\DemoBundle\Entity\Person, dite classe entité, incluant les informations que l'on souhaite renseigner dans le carnet d'adresses, ainsi que la table qui servira à stocker les enregistrements dans la base de données :

```
$ php app/console generate:doctrine:entity –entity=MercatorOceanDemoBundle:Person --no-interaction  \
--fields="name:string(255) email:string(255) address:string(1024) birthdate:date" --format=annotation
$ php app/console doctrine:schema:update --force
```

On obtiendra enfin grâce à la commande suivante une application permettant de lister tous les enregistrements du carnet d'adresses, d'afficher un enregistrement donné correspondant à sa clé primaire et enfin de créer, modifier ou supprimer un enregistrement. On utilise ici des annotations pour la configuration du routage et pour spécifier le mapping Doctrine.

```
$ php app/console generate:doctrine:crud --entity=MercatorOceanDemoBundle:Person --format=annotation  \
--with-write --no-interaction
```

Même s'il s'agit bien sûr d'un exemple simpliste, cette courte démonstration montre qu'on peut obtenir une interface fonctionnelle en quelques commandes.

5 Conclusion

Symfony est actuellement un des frameworks PHP les plus en vue. Ce court article a permis de présenter les principales fonctionnalités de la version 2, lancée en juillet 2011, qui semble être très prometteuse. Symfony 2 figure ainsi actuellement parmi les projets les plus populaires sur le site github.com. L'avenir dira si elle sera largement adoptée par la communauté.

L'utilisation d'un framework modifie complétement la façon d'aborder le développement. Même si elle ne résout pas tous les problèmes, elle se révèle pertinente pour des projets de petite taille, du type de ceux en cours de développement au sein de la société Mercator Océan. Cependant la durée de la phase d'apprentissage, qui dépend évidemment du niveau de programmation initial du développeur, est un point à prendre en compte très sérieusement. Dans tous les cas, la maîtrise réelle d'un framework et des bonnes pratiques de développement associées ne peut s'acquérir en quelques jours et nécessitera plusieurs mois de pratique, d'autant que, dans le cas de Symfony 2, la documentation n'est par la force des choses pas encore au niveau de celle existante pour Symfony 1[13].

Il existe bien sûr beaucoup d'autres frameworks de développement et le choix peut se révéler très problématique. Du point du vue du développeur, le point positif est que la maîtrise d'un framework comme Symfony permettra sans nul doute d'aborder n'importe quel autre framework MVC beaucoup plus rapidement, mais aussi et surtout d'être plus productif et de progresser dans le domaine de la programmation Web en général.

6 Bibliographie

[1] E. Gouleau, O. Mansour, T. Rivoallan, V. Lemaire, X. Lacot. *Frameworks PHP pour l'entreprise : définitions, critères de choix et analyses*. Livre blanc Clever Age, 14 mai 2008.

[2] D. Seguy, J.-M. Fontaine. *Industrialisation PHP*. Livre blanc Alter Way, 2009.

[3] *Model–view–controller*. Encyclopédie Wikipédia, en.wikipedia.org/wiki/Model-view-controller.

[4] *The Symfony Book and Cookbook*. Documentation du projet Symfony, symfony.com/doc.

[13] Il faut espérer par exemple que l'excellent tutoriel Jobeet qui décrit l'implémentation d'une application réelle (un site de recherche d'emploi) soit porté sur Symfony 2. Voir le site www.jobeet.org pour plus d'informations.