Livre Blanc

Frameworks PHP pour l'entreprise

Définition, critères de choix et analyses

Emmanuelle Gouleau
Olivier Mansour
Tristan Rivoallan
Vincent Lemaire
Xavier Lacot

version 1.0
14/05/08

1. Introduction

Au moment d'industrialiser ses développements, une entreprise cherche habituellement la meilleure solution, ou en tout cas celle qui est la plus adaptée à ses besoins, ses contraintes, son environnement.

Dans le cadre bien précis du développement PHP pour le Web, la situation a longtemps été anarchique. Le faible coût d'apprentissage des bases du PHP a donné au langage une réputation de « langage amateur », ce qui l'a longtemps desservi. PHP devrait ainsi n'être réservé qu'aux petits projets, peu stratégiques et/ou peu critiques.

La donne a cependant peu à peu changé, avec l'arrivée des « framework MVC » et autres « framework RAD », des librairies ou ensemble de librairies conçues pour faciliter et accélérer les développements. Certes écrit dans un autre langage, RoR (Ruby on Rails) a apporté une nouvelle manière de voir le développement Web : plus rapide, plus simple, plus efficace. Quelques années après l'apparition de RoR, PHP à son tour a vu naitre quantités de frameworks de développement rapide.

Ce livre blanc présente l'analyse effectuée par Clever Age de la situation actuelle des frameworks PHP, du point de vue de l'entreprise :

- Quels sont les grands critères permettant de sélectionner un framework PHP ? Comment évaluer les multiples frameworks PHP disponibles sur le marché ?
- Quels sont les risques à choisir un framework donné ? A quel point cela engage-t-il l'entreprise ? Quels sont les impacts sur son infrastructure, son mode de fonctionnement ?
- Dans un contexte projet, quel est le framework PHP MVC le plus adapté à votre structure ? Quel est celui qui peut vous permettre de rationaliser plus aisément vos développements ? Quel est le plus à même d'accélérer vos travaux ?

2. Sommaire

3. Qu'est ce qu'un framework MVC ?

1. L'intérêt d'un framework ?

Le terme de « framework » se traduit littéralement par « cadre de travail » : dans le cas des frameworks Web, concrètement, il s'agit d'un ensemble de librairies et d'outils qui permettent d'améliorer la conception d'applications Web, en apportant des fonctionnalités supplémentaires tout en ajoutant de la rigueur dans leur développement. Un framework permet également une automatisation des tâches dans la mesure où il intègre un certain nombre de routines implémentées nativement.

Utiliser un framework, c'est donc avoir la garantie de disposer d'une architecture cohérente, où la rigueur de développement est primordiale. C'est aussi une réflexion sur l'avenir, puisqu'un code léger, optimisé et cohérent est bien plus simple à maintenir et améliorer que la trop célèbre « soupe de code »[1].

Cependant, utiliser un framework introduit de nombreuses exigences et de la complexité, dans la mesure où certaines habitudes de développement devront être mises de côté lors de son adoption. Ainsi, bien qu'il soit censé améliorer la productivité de vos développements, il existe toujours une phase au cours de laquelle le coût est plus important que le gain : ce choix dont donc être mûrement réfléchi.

2. Qu'est ce que MVC ?

MVC (Model-View-Controller, soit Modèle-Vue-Contrôleur) est une architecture qui intègre trois niveaux de conception suivants :

- **Le Modèle** : il s'agit du comportement de l'application. Ce niveau intègre l'ensemble des interactions avec la base de données et le traitement des données : il contient et manipule toutes les données, en gérant leur sélection, leur insertion, leur modification ou leur suppression (« CRUD »). Pour cela, il propose des méthodes spécifiques à la bonne tenue de ces actions.

- **La Vue** : il s'agit de l'interface que l'utilisateur va manipuler. Elle habille les données transmises par le modèle et reçoit toutes les actions effectuées par l'utilisateur, sans en assurer le traitement : les actions sont transférées au contrôleur.

- **Le Contrôleur** : il prend en charge la gestion des évènements pour mettre à jour la vue ou synchroniser des informations via le modèle. Il reçoit toutes les actions effectuées par l'utilisateur, et effectue la détection d'erreurs (vérification du remplissage correct des champs d'un formulaire, par exemple). Tout comme la vue, le contrôleur n'effectue aucune modification sur les données, il est uniquement chargé d'appeler le modèle et de renvoyer la vue concernée.

Un bon exemple valant mieux qu'un long discours, partons du principe que notre application doit afficher une liste de clients et permettre l'ajout, la suppression et la modification de ceux-ci.

1 Voir « Coding Horror » : http://www.codinghorror.com/blog/

- Le Modèle lit en base de données toutes les informations sur les clients, et procède à l'enregistrement des modifications apportées à ceux-ci.

- La Vue se charge de décorer les données issues du modèle (mise en gras, code html, etc.), afficher des liens d'édition et de suppression ainsi que le formulaire de modification des fiches clients.

- Le Contrôleur, quand à lui, vérifie que les informations saisies dans le formulaire de modification d'un client sont bien formatées, et que tous les champs obligatoires ont été remplis par des données correctement formatées (que l'adresse email est bien conforme aux standards ou que le numéro de téléphone est valide, par exemple).

Pour résumer le principe de fonctionnement du MVC, lorsqu'un client effectue un appel à une application, la requête est analysée par le contrôleur qui demande au modèle concerné d'effectuer les opérations. Enfin, c'est ce même contrôleur qui va renvoyer la vue concernée au client.

4. Les grands critères de choix d'un framework pour l'entreprise

1. Risques pour l'utilisateur

Au-delà des considérations techniques qui entrent en jeu lors du choix d'un framework, il est essentiel d'évaluer les risques présentés par chacune des solutions retenues. Le framework devant à terme constituer la base de la majorité des développements réalisés pour l'entreprise, il est important que sa pérennité soit assurée : son développement doit être suffisamment dynamique, afin de limiter les risques d'abandon ou de fermeture du code.

Comment évaluer ces risques ? La méthode QSOS *(Qualification an Selection of Open Source software)* propose quelques critères intéressants, organisés selon quatre axes : Pérennité intrinsèque (maturité, adoption, etc.), Solution industrialisée (documentation, packaging, etc.), Adaptabilité technique (modularité, extensibilité, etc.) et Stratégie (direction des développements, licence, etc.).

2. Vues

MVC présente le grand intérêt de permettre aux utilisateurs non développeurs (comme les web designers ou intégrateurs) de pouvoir modifier les gabarits des pages produites. Typiquement, un web designer doit pouvoir modifier aisément la mise en forme des templates en utilisant une syntaxe simple et, si possible, standardisée.

En ce sens, il convient de se poser la question du fonctionnement du système de gestion de templates des frameworks, et l'adaptation nécessaire pour les personnes amenées à travailler sur ceux-ci. Il existe plusieurs cas :

- Soit le framework dispose de son propre système de gestion de templates,
- Soit il utilise un moteur de templating externe (comme Smarty[2] ou PHPTal[3]),
- Soit il permet de combiner les deux, en faisant un choix avant d'entrer en phase de développement.

En fonction des compétences internes de votre entreprise, il peut être judicieux de ne pas s'attacher à une technologie intégrant des conventions d'écritures propres (comme les moteurs de templates) mais plutôt de s'appuyer directement sur des technologies de plus bas niveau (par exemple PHP).

3. Performances

Bien évidemment, le critère de la performance est au premier plan dans l'évaluation d'un framework. Le propre du Web est la réactivité, la rapidité et la facilité d'accès à

2 http://www.smarty.com

3 http://phptal.motion-twin.com

l'information. Aussi, même s'il est extrêmement bien conçu en termes de structure et d'architecture, un framework aux performances déplorables ne permettra pas des développements et une maintenance d'une application Web satisfaisants.

Ce critère de la performance, bien qu'il semble a priori essentiel, a trop souvent tendance à être oublié par les développeurs soucieux de l'élégance de leur code et des technologies employées. On trouve sur le Web de nombreuses études de performances (« benchmarks ») qui se proposent de comparer l'efficacité des différents frameworks du marché, mais de telles études sont dans la plupart des cas à prendre avec précaution, car trop de facteurs influent sur le résultat final : directives de compilation de l'interpréteur, optimiseurs de cache et de performances activés, adéquation des tests effectués aux réalités du développement, soucis d'optimisation de l'environnement serveur.

En vérité, rien ne vaut une implémentation de référence, qui constitue la seule véritable mise en situation de production du framework. Oubliez les benchmarks qui prétendent mesurer au millième de seconde près l'affichage d'un texte statique, ce ne sont généralement que de désespérantes tentatives de masquer la réalité des choses.

Une présentation de Cal Henderson (Flickr), en 2005, résume très bien ce point de vue, en exprimant l'idée selon laquelle les frameworks ne passent pas à l'échelle : « frameworks don't scale »[4].

4. Routage

Le « routing », que l'on peut traduire en français par « routage », est le mécanisme par lequel la requête d'une URI est associée à la réalisation d'une action par le framework. Le routing présente plusieurs intérêts : filtrer les requêtes effectuées sur l'application, permettre l'association de jolies URLs à des actions côté serveur, mais également abstraire la notion d'URLs au sein de l'application.

On distingue généralement deux types de routage :

- le routage « ascendant », dont le rôle est d'effectuer l'acheminement d'une requête HTTP vers le module adéquat de l'application. Dans la plupart des frameworks modernes, le routage ascendant prend en charge la lecture des paramètres de la requête. Par exemple, une requête sur http://lacot.org/blog/2007/07 doit permettre d'activer le module de blog de l'application, mais en plus elle doit indiquer à ce module que les variables qu'il doit prendre en compte sont l'année « 2007 » et le mois « 07 ».

- le routage « descendant », lui, favorise la portabilité et l'indépendance de l'application vis-à-vis des URLs. Concrètement, il permet de générer des URLs depuis le contrôleur ou la vue, et favorise ainsi la séparation entre le monde extérieur (le « Web ») et le fonctionnement interne du framework. Par exemple, l'emploi de la route interne « blog » avec les paramètres « 2007 » et « 07 » doit générer l'url voulue, dépendante de la configuration globale de l'application (domaine de déploiement, etc.).

Le routage est un élément très intéressant par différents aspects. Tout d'abord, il favorise le référencement d'un site Web en facilitant la mise en place d'URLs propres, et ceci sans le recours systématique au mod_rewrite d'Apache2, alourdissant la charge des serveurs Web en cas de sur-utilisation. Il procure également d'autres gains, moins visibles et mesurables, mais tout aussi importants : sécurité de l'accès à l'application, établissement

4 Voir http://www.iamcal.com/talks/flickr_services.pps

de contrats entre l'application et les URIs d'appel, maîtrise du modèle d'adressage du projet.

5. Internationalisation et régionalisation

En quoi cela consiste-t-il ? Selon Wikipédia, « L'internationalisation d'un logiciel consiste à le préparer à la régionalisation (« localization » en anglais), c'est-à-dire à l'adaptation à des langues et des cultures différentes. Contrairement à la régionalisation, qui nécessite surtout des compétences en langues, l'internationalisation est un travail essentiellement technique, mené par des programmeurs »[5]). Le travail d'internationalisation intervient sur quatre parties d'un logiciel :

- les contenus en base de données,
- les textes de l'interface,
- les formats régionaux (devises, dates, etc.),
- la gestion des jeux de caractères.

Bien qu'étant cruciale, cette fonctionnalité d'i18n est souvent oubliée - ou écartée - par les développeurs de frameworks, et n'est que rarement disponible nativement. Si ce besoin est important pour votre projet, veillez donc à bien le valider lors de la phase d'évaluation de la solution.

6. Outillage

Le terme « outillage » englobe tous les éléments non nécessaires au bon fonctionnement d'un framework, mais qui fournissent de l'assistance au développement.

1. Journaux

Un framework doit être capable de produire des journaux d'exécution, qui doivent fournir par défaut suffisamment d'informations pour que le développeur soit en mesure de comprendre les étapes qui ont mené à une erreur. Le système de logs d'un framework doit permettre de journaliser des entrées relatives à l'exécution des développements spécifiques, et il doit en outre être flexible et extensible :

- filtrage des entrées en fonction d'un niveau d'erreur (debug, avertissement, erreur fatale, etc.),
- disponibilité de différents *backends* (fichier, email, xmpp, etc.),
- format des messages standards,
- possibilité de développer des *backends* personnalisés.

2. Debuggage

Le « debuggage » correspond à la recherche et à l'analyse des dysfonctionnements.

5 http://fr.wikipedia.org/wiki/Internationalisation_de_logiciel

Même si des outils de debuggage ne sont pas essentiels au fonctionnement d'un framework, leur présence favorise naturellement les conditions de développement, dans la mesure où ils permettent au développeur de comprendre plus rapidement l'origine d'un bug. Tous les développements PHP peuvent employer des outils de debugging comme Xdebug[6], mais leur emploi est généralement assez peu aisé, et devrait plutôt être réservé à des besoins de profiling avancés.

C'est la raison pour laquelle de plus en plus de frameworks Web proposent des outils de debuggage, utilisables par le développeur durant la phase de développement. Les fonctionnalités couvertes par ces outils peuvent être assez variées : affichage des exceptions rencontrées, liste des requêtes de base de données effectuées, cheminement de la requête, etc.

En général, méfiez-vous des frameworks qui ne proposent pas de tels outils ; cela peut être le signe d'un manque d'aboutissement, et ce sera à coup sûr un élément pénalisant au cours des développements effectués.

3. Scaffolding

Le scaffolding, que l'on pourrait littéralement traduire par « échafaudage », est une structure de base permettant d'effectuer des opérations de type CRUD *(Create, Retrieve, Update, Delete)* sur une structure de données particulière.

Certains frameworks proposent de générer du code effectuant ces opérations - charge ensuite au développeur d'adapter ce code à ses besoins spécifiques. D'autres frameworks possèdent des capacités d'inspection de la base de données, pour permettre la génération « à la volée » d'un scaffolding.

Il est important de bien comprendre le mécanisme de scaffolding adopté par un framework, afin de pouvoir en tirer le maximum. Peut-on aisément modifier les pages générées, leur ajouter des fonctionnalités ou modifier leur présentation ? Existe-t-il des extensions ou plugins de personnalisation ?

4. Command Line Interface

L'interface en ligne de commande, encore appelée « CLI » *(Commande Line Interface)* est un outil que ne proposent pas tous les frameworks. Comme son nom l'indique, il s'agit d'un utilitaire qui permet d'effectuer en ligne de commande différentes opérations au cours du développement ou au cours de la vie d'une application : génération de modules, vidage du cache, rotation des fichiers de log, synchronisation de plusieurs instances d'une application, création des tables, sauvegarde, etc.

Dans certains cas, l'interface en ligne de commande est même extensible, et autorise la création de taches propres au développeur, dans l'objectif de répondre aux besoins plus spécifiques d'un projet.

5. Environnements de développement

Il est important de réfléchir aux capacités d'intégration du framework avec l'environnement de développement utilisé au sein de l'entreprise, si celui-ci est standardisé dans votre structure :

6 http://xdebug.org

- niveau d'intégration du langage de gabarits : la coloration syntaxique et l'auto-complétion sont-elles supportées pour ce framework ? Existe-t-il un plugin pour votre éditeur permettant de profiter de ces fonctionnalités ?

- auto-complétion du code PHP : existe-t-il un plugin compatible avec votre IDE ajoutant l'auto-complétion des classes, méthodes et fonctions du framework ?

- exécution de tâches récurrentes : existe-t-il une gestion des automatismes tels que le vidage des fichiers de cache ou de rotation des logs, l'auto-génération de code, ... ?

- exécution de tests unitaires : le framework propose-t-il la réalisation de tests de fonctionnement de tout ou partie de votre application ?

- auto-génération de code : le framework met-il à disposition un dispositif de type *CRUD (Create, Retrieve, Update, Delete)* ou d'auto-génération de back-office permettant d'accélérer la conception de certaines étapes contraignantes et rébarbatives du projet ?

7. Intégration avec des briques externes

Aussi complet soit-il, un framework ne fournira jamais l'intégralité des briques nécessaires à la réalisation de toutes les applications. Il peut également arriver que certains composants du framework ne soient pas d'une qualité suffisante. Un bon framework doit donc permettre d'intégrer des briques externes, afin de remplacer sans difficulté certains de ses composants par d'autres. Cette intégration peut se faire de différentes manières, et peut être plus ou moins poussée. Certains frameworks se contenteront ainsi d'inclure des librairies externes par le biais de la fonction « require() », tandis que d'autres prendront à leur compte le chargement automatique de ces librairies.

Si on omet les deux ou trois frameworks leaders du marché, qui proposent déjà de nombreuses librairies pour répondre à la plupart des besoins courants du développeur, cette capacité d'extension du framework est primordiale, car elle permettra de profiter de composants issus d'autres frameworks. Évidemment, si vos développements font appel à un trop grand nombre de librairies externes, il sera peut être temps de reconsidérer votre choix de framework initial, fragilisé par la diversité des briques employées.

8. Respect des standards

On le sait bien, la question des standards revient souvent lorsque l'on s'intéresse au développement Web et, plus largement, au développement informatique tout court. Essentiels, surtout dans un environnement aussi hétérogène que le Web, les standards sont cependant souvent mis de côté lors du développement d'une application, car leur respect intégral constitue une tâche fastidieuse à intégrer. C'est regrettable, car leur support offre souvent des avantages extrêmement importants en terme d'accessibilité, de performances et d'intégration. Un bon framework devrait donc respecter nativement et complètement les standards d'Internet (XHTML, HTTP, email, etc.), et encourager le développeur à les mettre en œuvre, lorsque cela est possible.

1. Standards XHTML : respect du balisage

La qualité du respect des standards XHTML par un framework se mesure selon trois critères :

- Conformité des pages XHTML par défaut : tout framework fournit un ensemble de pages web prédéfinies, comme celles destinées à l'affichage d'erreurs (404, 500, etc).

- Conformité du code généré par les helpers : on utilise généralement un framework pour sa capacité à réaliser à la place du développeur les tâches fastidieuses et répétitives propres au développement d'applications Web. Cette capacité est mise en œuvre grâce à la génération automatique de code PHP et HTML.

- Conformité du langage de gabarit : on utilise souvent un langage dédié pour l'implémentation des vues. Si c'est le cas (certains framework se contentent d'utiliser PHP), il convient de s'assurer que ce langage permet de réaliser du code conforme au standard XHTML.

2. Standards de développement, utilisation de librairies reconnues

Les développeurs à l'origine d'un logiciel, et a fortiori d'un framework, ont beau être extrêmement doués, documentés et disponibles, la qualité du résultat dépend toujours très largement des processus mis en place pour assurer la qualité et l'homogénéité des développements. Disposer de standards de codage clairement explicités, de règles syntaxiques, de méthodes de documentation des classes constituant le framework, tout cela contribue à améliorer l'homogénéité du produit.

Ceci dit, il est des fonctionnalités dont l'implémentation dépasse très largement le cadre d'un framework Web ; c'est par exemple le cas des problématiques liées à l'Object-Relation Mapping, ou encore de la gestion des appels Ajax. Dans ces cas, s'appuyer sur des briques logiciels externes est une bonne solution pour les concepteurs des frameworks Web, dans la mesure où cela leur permet de se concentrer sur des problématiques qui n'ont pas été déjà traitées correctement. Reste à évaluer les librairies techniques choisies, effectuer des choix judicieux, et à les intégrer convenablement !

3. Implémentation correcte des RFC : l'exemple de HTTP

La « guerre des navigateurs Web », qui a été largement médiatisée ces dernières années avec la sortie du navigateur libre Mozilla Firefox, a mis en lumière la question du support XHTML au sein des applications Web. Cependant, il existe de nombreux autres standards au moins aussi importants, mais souvent plus négligés car ils ont reçu un éclairage moins important au cours des dernières années.

Le respect du standard HTTP, par exemple, est un bon critère d'évaluation du respect des standards : cette norme est complexe, étendue, et sa maîtrise est essentielle dans le cadre de la recherche de bonnes performances. De nombreuses facettes de HTTP sont souvent mal connues, et donc rarement implémentées dans les frameworks Web. C'est, par exemple, le cas de Etags (entête HTTP renvoyée par le serveur, à partir de la version 1.1 du protocole, permettant d'informer le client d'un changement du contenu, afin d'économiser des téléchargements de contenus inutiles), des cache headers (mécanismes permettant au client de savoir s'il doit mettre en cache les données), du support de HTTP/1.1 (introduisant, notamment, une meilleure gestion du cache et de nombreuses améliorations par rapport à l'ancienne version du protocole[7]), de la gestion du pipelining (combinaison de plusieurs requêtes HTTP dans une seule connexion TCP, sans attendre les réponses correspondant à ces requêtes[8]), de l'idempotence (même effet d'une opération si elle est appliquée une ou plusieurs fois).

7 http://tools.ietf.org/html/rfc2616

9. Ajax

Ajax, et plus largement les technologies du « Web 2.0 », est une des modes du développement Web depuis 2005/2006, et est très prisé des développeurs et des utilisateurs. Tout bon framework doit donc proposer des moyens aisés pour le mettre en œuvre.

Cependant, la question n'est pas de savoir si le framework le permet, mais surtout *comment* il le permet : il n'est réellement intéressant d'utiliser Ajax que si on peut le faire de manière non intrusive[9], sans remettre en question l'accessibilité de l'application et son utilisation dans des environnements moins riches (téléphone mobile, etc.).

10. Extensibilité

Outre la possibilité d'emploi de briques externes, un bon framework doit être aisément extensible, ce qui est sensiblement différent. En résumé, l'extensibilité d'un framework est la capacité qu'a un développeur à étendre certains comportements (voire à les remplacer), sans pour autant devoir modifier le code d'origine du framework. Cette extensibilité peut se manifester sous plusieurs formes :

- héritage de classes,
- emploi de plugins,
- simplicité d'écriture des plugins,
- possibilité de choisir un ORM donné (ORM = *Object-relational mapping* – un outil permettant de traiter les données issues ou à destination de base de données relationnelle en base de données orientée objet),
- possibilité de choisir le moteur de rendu des vues,
- présence d'un système de hooks ou de mixins,
- etc.

11. Authentification et permissions

L'authentification permet de contrôler l'identité d'un utilisateur, tandis que les permissions sont les droits d'accès donnés à l'utilisateur, en fonction de son authentification et de son statut.

Sauf dans quelques rares cas, la plupart des applications Web développées aujourd'hui incluent la notion de compte utilisateur et, parfois, la sauvegarde de préférences utilisateur. Sans framework, la réalisation d'une application Web gérant des profils - et éventuellement des droits d'accès - peut rapidement tourner au cauchemar. Alors que, bien pensée, la gestion des droits d'accès aux différentes parties d'une application Web peut être enfantine à mettre en place. Avant de choisir un framework pour vos développements, vous devriez donc examiner avec attention les fonctionnalités

8 http://tools.ietf.org/html/rfc2616#section-14.9

9 http://www.thefutureoftheweb.com/talks/2006-10-ajax-experience/slides/

supportées en matière de gestion des utilisateurs et de leurs droits d'accès.

12.Sécurité

La question de la sécurité est directement liée à celle de l'authentification et des permissions, même si elle recouvre en plus d'autres problématiques : gestion des attaques XSS ou des injections SQL, possibilité de déni de service, etc. Tous les frameworks ne le proposent pas, mais le filtrage des données entrées par l'utilisateur devrait être une mesure de sécurité systématique, de même que l'échappement des données renvoyées au navigateur.

Évidemment, la question de la sécurité est une question difficile à évaluer, car elle dépend de nombreux autres éléments que la conception intrinsèque du framework : soin apporté par le développeur à la conception de son application, environnement de déploiement. Ceci dit, un framework qui prend en charge des éléments basiques de sécurité sera toujours d'une aide plus précieuse qu'un framework qui élude gentiment la question. Pour résumer, la sécurité d'un framework est un point qui doit être examiné indépendamment des compétences des développeurs qui l'emploieront, et de l'environnement au sein duquel il sera déployé. La conception d'un framework doit donc prendre en compte les différents aspects de la sécurité et de la sûreté de fonctionnement. Par exemple, la validation et filtrage des données en entrée et le contrôle des informations renvoyées permettent de se libérer des attaques de type XSS ou injections SQL.

13.Déploiement

Le développement d'une application n'est pas tout, il faut aussi pouvoir la mettre en recette, production, etc. Le framework évalué propose-t-il des fonctionnalités relatives à ce problème ? Autre point important, la gestion des montées en version du produit par rapport aux applications existantes est-il possible ?

Les évolutions des versions du framework entraînent bien souvent des modifications à apporter au niveau du code, que ce soit dans sa structure ou ses principes de développement : est-il prévu de pouvoir travailler sur ces différentes versions ? De même, est-il possible de maintenir en production, sur un même environnement, des applications utilisant différentes versions du framework ?

14.Tests unitaires et fonctionnels

Nous ne parlons pas ici de savoir si le framework est testé unitairement et fonctionnellement (élément essentiel mais propre au critère « Risques pour l'utilisateur »), mais de la possibilité de mettre en œuvre des tests pour les développements spécifiques réalisés. Cette possibilité soulève généralement d'autres questions, directement liées :

- La solution de tests employée est-elle un standard ?
- Richesse de la solution de tests ? (mock objects (*bouchon*, alternative temporaire permettant de remplacer un code non fonctionnel ou non développé)), fixtures (contrôle effectué sur les données), etc.)
- Intégration avec des outils existants (Selenium [SEL 2008], CruiseControl [CRU 2008], etc.)

15.Courbe d'apprentissage

Même s'il est censé améliorer la productivité des équipes de développement en leur facilitant le travail, un framework Web nécessite toujours une phase d'apprentissage à ne pas négliger, ne serait-ce que pour maîtriser l'outillage, ou connaître l'ensemble des fonctionnalités disponibles : la maîtrise d'un framework PHP n'induit pas forcément la connaissance et la compréhension des frameworks concurrents. Par ailleurs, les frameworks concernés par cette étude font pour la plupart l'objet de cycles de développement très dynamiques. Si c'est un bon point en terme d'amélioration du produit, cela signifie cependant que les équipes de développement doivent être préalablement formées à l'emploi du framework, et capables d'organiser une veille technologique autour du produit, de façon à rester au courant des nouvelles possibilités.

La qualité de la documentation et la mise en place d'une communauté dynamique sont primordiales pour permettre un apprentissage aisé du framework. Certains produits ont choisi de mettre en place des incubateurs de projets, d'autres mettent l'accent sur les tutoriels, la documentation, la promotion par la communauté. Une bonne idée, malheureusement trop rarement mise en pratique, consiste à mettre en place une implémentation de référence, qui met en lumière toutes les qualités du projet, et montre comment effectuer diverses opérations (validation de formulaire, appel AJAX, etc.). Ces implémentations de référence sont en général d'une grande aide au développeur, car elles illustrent de manière pratique les mécanismes du framework pour répondre à un problème donné.

16.Aspects légaux

Même s'il elle ne participe pas à la qualité de son code, la licence d'un framework est un élément à vérifier. En pratique, la plupart des frameworks PHP du marché sont placés sous des licences très permissives, comme les licences MIT[10] ou new BSD[11]. Cependant, il existe quelques exceptions plus restrictives, au risque que les contributions se fassent de façon moins spontanée : en général, cela nuit au dynamisme du développement du framework.

Pour plus d'informations sur les licences logicielles habituellement employées, vous pouvez consulter [GNU 2008].

10 http://www.opensource.org/licenses/mit-license.php

11 http://www.opensource.org/licenses/bsd-license.php

5. CakePHP

1. Origine et motivation du framework

CakePHP est apparu en 2005, suite à l'initiative de développeur polonais Michal Tatarynowicz. CakePHP a rapidement été publié sous licence MIT et repris par une communauté de développeurs sous le nom de CakePHP. Le projet CakePHP est assez actif; il attire de nombreuses contributions et est utilisé dans une grande quantité de projets professionnels, dont certains sont hébergés sur CakeForge, la plateforme de développement de projets en lien avec CakePHP.

CakePHP se définit comme un outil dont l'objectif est d'aider les développeurs PHP à créer plus aisément et rapidement des applications extensibles. Inspiré de Ruby On Rails, CakePHP implémente MVC *(Modèle Vue Contrôleur)*.

2. Points forts du framework

Le framework CakePHP a le point fort de ne pas nécessiter de configuration particulière. D'autre part, il fonctionne avec le duo PHP4/PHP5, et sa prise en main est rapide.

3. Points faibles du framework

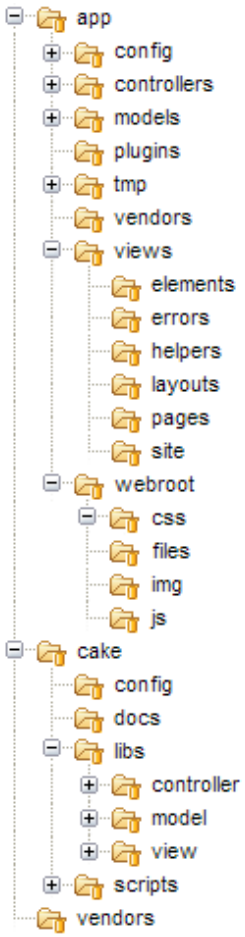
Corollaires de ses points forts, les points faibles de CakePHP concernent principalement le manque d'ampleur du projet, qui souffre en plus, depuis quelques temps, d'un dynamisme moins marqué que ses concurrents. Certes, CakePHP est plus facile d'accès que Symfony ou que le Zend Framework, mais cela se paye au prix de manques fonctionnels marqués. Par exemple, l'outillage de CakePHP est assez restreint, l'outil en ligne de commande étant peu efficace et, surtout, trop complexe d'utilisation.

On regrette également la difficulté d'intégration des débogueurs PHP ou l'imposition de conventions de nommage pour la schématisation de base de données, rendant l'adaptation d'un modèle existant assez complexe.

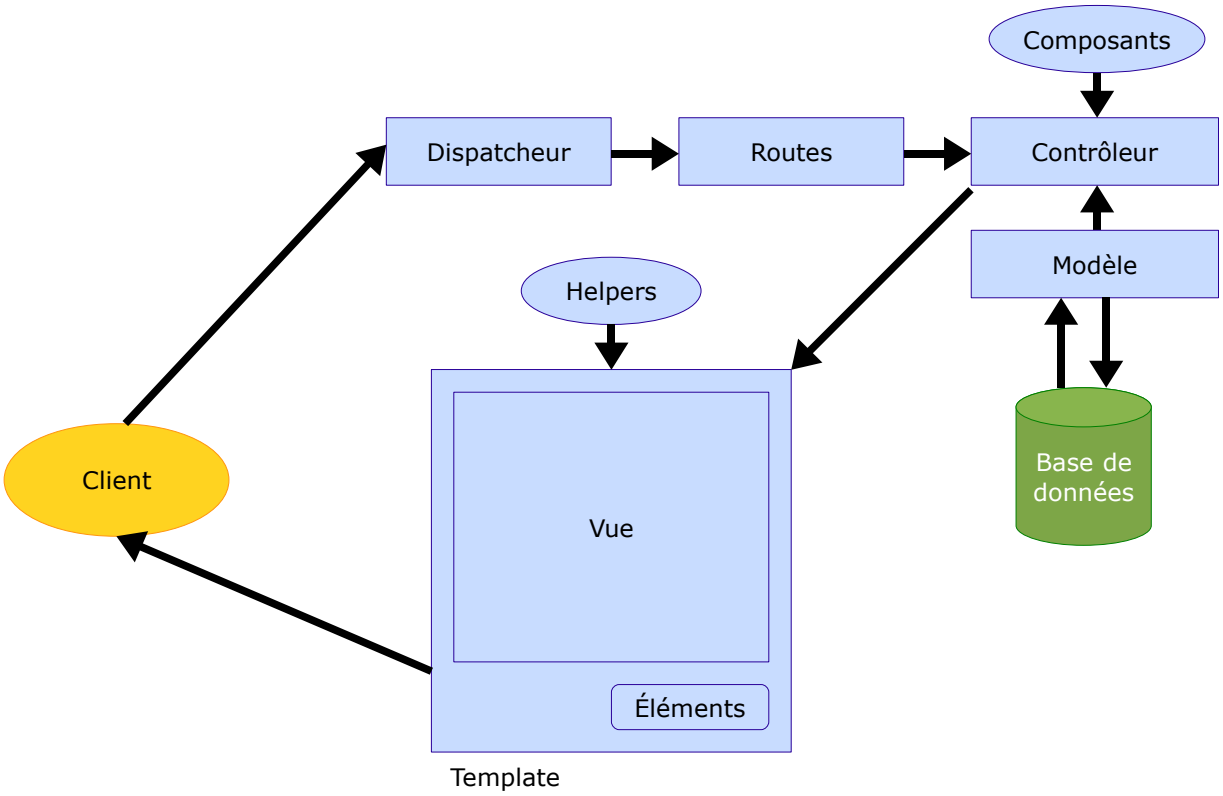
Enfin, sa double compatibilité PHP4/PHP5 contraint CakePHP à souffrir d'importantes lenteurs dans son évolution.

4. Organisation des fichiers

CakePHP propose, comme tous les frameworks de ce Livre Blanc, une implémentation de MVC. Cependant, contrairement à d'autres, la notion de module n'apparaît pas dans l'arborescence des fichiers du framework. Au sein d'un projet CakePHP, tous les contrôleurs sont stockés dans le dossier « controllers », tous les modèles dans le dossier « models », toutes les vues dans le dossier « views ». Point négatif, cela peut nuire à la productivité et à la lisibilité des dossiers de développement.



5. Traitement d'une requête



6. Processus d'internationalisation

La gestion de l'internationalisation d'un projet est très récente sous CakePHP. En effet, cette fonctionnalité, bien qu'essentielle pour un framework, n'a été apportée que depuis la pré-version 1.2, non finalisée à ce jour.

7. Processus de contribution du framework

CakePHP met à disposition un Trac permettant de participer au développement et à l'amélioration du framework. Chaque modification est suivie par un responsable du projet par l'intermédiaire du système de tickets intégré à Trac. De même, la soumission d'erreurs peut se faire par n'importe quel utilisateur et le suivi est régulier.

CakePHP a également mis en place un outil baptisé « CakeForge », qui offre un accès aux développeurs à différents outils comme un système de gestion de bugs, de gestion des versions, etc. L'inscription est obligatoire mais gratuite, et permet de contribuer efficacement à n'importe quel projet, qu'il soit existant ou non. L'outil est également proposé dans une version payante destinée aux sociétés, et fournit un espace complet de développement privé et sécurisé pour les projets.

8. Exemple d'extension

CakePHP propose une bibliothèque d'environ 150 extensions comme un forum, un gestionnaire d'utilisateurs ou un système de boutique en ligne avec gestion de caddie.

Il peut également être complété par des plugins issus de PEAR ou d'eZComponents.

9. Courbe d'activité du framework

L'activité autour de CakePHP est assez importante : bon nombre d'extensions sont proposées par des développeurs sur CakeForge. Cependant, un manque d'activité ces derniers mois peut laisser planer des doutes sur la maintenabilité du projet dans le temps.

10. Quelques références

- Mozilla Addons
- NoseRub
- Yale Daily News

11. L'avis de Clever Age

Bien que sa prise en main soit rapide, CakePHP souffre d'un nombre important de points faibles, comme le manque de fonctionnalités de base (l'internationalisation, par exemple), ce qui réduit l'intérêt final du framework.

CakePHP est adapté aux projets de taille raisonnable (100 à 150 jours homme) dans un contexte où l'internationalisation n'est pas nécessaire. Au delà de ces besoins, il est préférable d'employer Symfony ou le Zend Framework.

6. Code Igniter

1. Origine et motivations du framework

CodeIgniter est apparu en 2006 sous l'impulsion du développeur Rick Ellis. Il est activement développé depuis. L'objectif est de produire un framework rapide et peu gourmand en mémoire, pouvant être facilement et rapidement déployé, proposant une courbe d'apprentissage rapide et restant simple d'utilisation.

Le développement du framework est largement orienté vers la vitesse d'exécution de ce dernier. A ce titre, de nombreux choix ont été faits, rejetant certaines propositions d'enrichissement de fonctionnalités afin de concentrer les efforts de développement sur le cœur du framework. Le choix a également été fait de ne pas imposer de conventions contraignantes aux développeurs et de les laisser librement appliquer les leurs.

De ce point de vue, CodeIgniter est une vrai réussite, respectant totalement ses motivations de départ.

2. Points forts du framework

Les points forts du framework sont de manière évidente sa versatilité et son faible impact sur les infrastructures. CodeIgniter est vraiment rapide, s'installe rapidement sur toutes les plateformes (PHP4 et PHP5) et propose une prise en main excessivement aisée. Un programmeur PHP habitué à coder lui-même ses contrôleurs et ses vues augmentera ainsi son rendement. Il trouvera beaucoup de fonctionnalités pratiques couvrant les besoins les plus basiques mais aussi les plus récurrents (logs, cache, cookies, etc.).

La documentation a fait l'objet de gros efforts de la part de la communauté. Elle est relativement simple à appréhender et suffit largement à un démarrage efficace.

La compatibilité PHP4 peut être un avantage si vos plateformes ne supportent pas PHP5, bien qu'il soit à l'heure actuelle vivement recommandé d'effectuer la migration, PHP4 n'étant plus maintenu ou en passe de ne plus l'être.

3. Points faibles du framework

CodeIgniter est un framework « léger ». Par conception, il laisse au développeur beaucoup de choses à faire à la main comme l'escaping des données ou l'appel des vues. Le système MVC présente de graves lacunes :

- la couche modèle est récente et reste facultative. Il faut pratiquement l'écrire à la main, ce qui devrait décourager la plupart des programmeurs,
- les contrôleurs sont monolithiques, il est difficile de réutiliser le code des contrôleurs et si leur logique devient complexe, ils sont lourds à maintenir,
- le système de vue est minimal et ne propose pas non plus de mécanismes pratiques de réutilisation. Les helpers sont assez pauvres (pas d'intégration d'Ajax

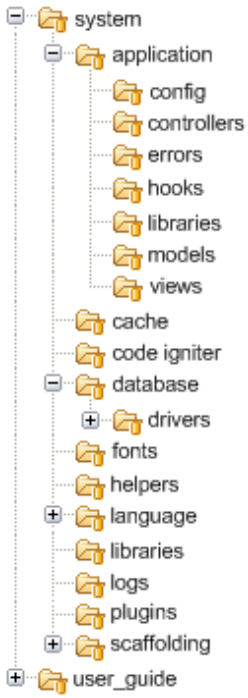
notamment).

Le système de scaffolding permet uniquement d'alimenter les données de l'application CodeIgniter et ne peut pas être utilisé pour construire des back-offices. CodeIgniter ne propose rien ou pas grand chose pour gérer ses applications (environnements, outils de déboguage, ...).

Le souhait de conserver la compatibilité avec PHP4 peut être considéré comme un point faible : le parti pris de CodeIgniter ne lui permettra pas de se hisser au niveau technique de ses concurrents.

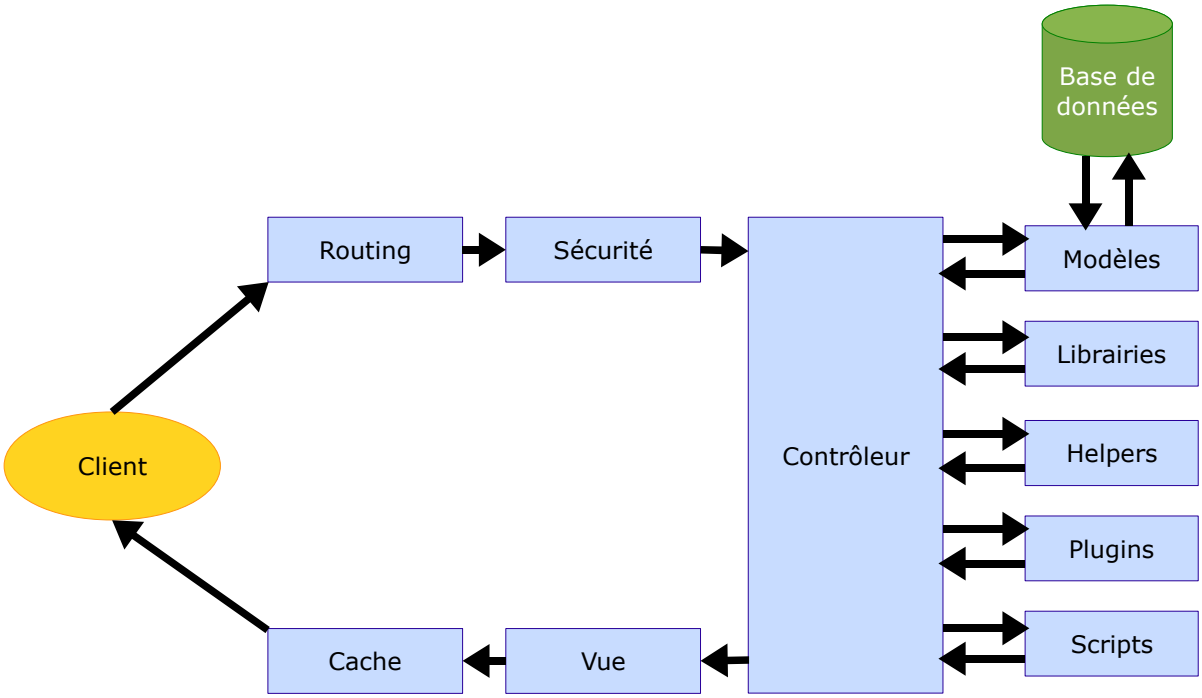
4. Organisation des fichiers

L'organisation par défaut est peu sécurisante, obligeant les répertoires contenant les librairies a être publiés sous la racine du serveur web. Il est toutefois possible de changer cela en modifiant le paramétrage de base. On retrouve une organisation assez logique à l'intérieur du répertoire « application ». De manière générale, l'organisation des fichiers est classique et efficace.



Par défaut également, tous les vues et contrôleurs sont regroupées sous les répertoires « views » et « controllers ». Encore une fois, c'est aux développeurs de décider s'ils veulent l'organiser différemment (en créant des sous répertoires et en les appelant spécifiquement dans leur code). L'organisation des fichiers est finalement assez souple, mais doit être fixée dés le début du projet.

5. Traitement d'une requête



6. Processus d'internationalisation

CodeIgniter ne supporte pas le comportement i18n standard et implémente son propre mécanisme d'internationalisation.

Le processus d'internationalisation est géré dans autant de fichier PHP que de langues utilisées par l'application.

```
$lang['language_key'] = "The actual message to be shown";
```

Il faut ensuite utiliser la classe de langue de CodeIgniter pour accéder au message :

```
$this->lang->line('language_key');
```

Ce système est pauvre, mais reste dans la logique de l'application, car il est facilement compréhensible par tous les développeurs.

7. Processus de contribution au framework

La contribution au framework est très contrôlée par la société EllisLabs. Les principales contributions sont la remontée de bugs dans un bug tracker spécifique au projet CodeIgniter.

Le wiki permet toutefois assez librement de publier des contributions (librairies, hooks, etc.) afin d'étendre le framework. De nombreuses contributions sont faites de manière un peu désordonnée via le forum.

8. Exemple d'entension

CodeIgniter propose deux mécanismes pour étendre la solution : des hooks permettant de modifier les classes de base, et des plugins.

Les hooks PHP4 permettent d'étendre et de modifier le cœur du framework. Ce sont des points d'entrées prédéfinis auxquels il est possible d'attacher ses propres fonctions. Huit points d'entrées sont ainsi définis. Ils permettent, par exemple, d'améliorer les fonctionnalités de *profiling* et de cache.

Les plugins sont en réalité des collections de fonctions contribuées par la communauté CodeIgniter et destinées à être appelées depuis les contrôleurs ou les vues. Par exemple, un contributeur propose un plugin pour réaliser des formulaires d'upload de documents en Ajax.

Les extensions sont majoritairement des extensions techniques du framework, on trouve peu de modules prêts à l'emploi permettant directement de bénéficier nativement de nouvelles fonctionnalités.

9. Courbe d'activité du framework

La courbe d'activité du framework est en augmentation régulière depuis la première version. Toutefois, peu d'acteurs professionnels semblent impliqués dans la communauté, qui est surtout active sur forum, endroit privilégié pour obtenir de l'aide autour du framework.

CodeIgniter séduit un grand nombre de développeurs contraints par PHP4 et/ou peu au fait des apports de PHP5. De ce fait, CodeIgniter reste le framework le plus simple à prendre en main de cette étude, même si cette facilité s'explique par le peu de fonctionnalités offertes nativement par le framework.

Le nombre de contributions est vraiment très important, il est parfois difficile de faire un choix parmi les alternatives s'offrant au développeur, d'autant plus que la qualité de ces contributions externes est assez variable et difficile à estimer.

10. Quelques références

Plusieurs références sont disponibles sur http://codeigniter.com/projects/, mais aucun gros acteur du Web ne semble employer Code Igniter comme la base de ses développements.

11. L'avis de Clever Age

CodeIgniter est un framework qui respecte ses promesses : simplicité, rapidité et souplesse. Il a, en quelque sorte, les défauts de ses qualités.

CodeIgniter est recommandé pour de petits projets (moins de 100 jours homme) qui sont contraints par la présence de PHP4. Si vous pouvez bénéficier de PHP5, il y a peu de raisons d'utiliser CodeIgniter.

7. Symfony

1. Origine et motivations du framework

Symfony est un framework développé par l'entreprise française Sensio. A ses origines se trouve le framework Mojavi et, comme la plupart des frameworks de la nouvelle génération, Symfony s'inspire très largement de certains concepts de Ruby On Rails.

Symfony a été créé à partir d'un constat bien simple : il n'existait pas, auparavant, de solution PHP suffisamment vaste et bien réalisée pour assurer la pérennité et la ré-utilisabilité du code. Même PEAR, effort lancé en 2001, ne parvient pas à fournir une réponse satisfaisante aux problématiques récurrentes des projets PHP : sécurité, respect d'un modèle (MVC, par exemple), nommage, portabilité, etc.

2. Points forts du framework

Même si cela conduit parfois à qualifier Symfony de "mastodonte", il faut bien reconnaître que Symfony est, de tous les frameworks, celui qui propose le plus de fonctionnalités. L'extensibilité du framework est assurée par le biais des plugins, qui permettent de profiter très rapidement de nouvelles fonctionnalités développées par d'autres contributeurs.

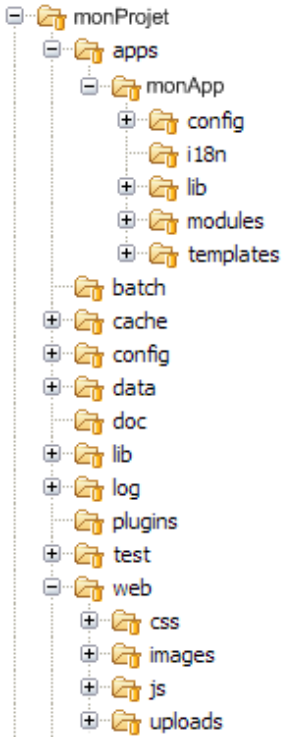
Fait remarquable, Symfony dispose sans doute d'une des meilleures documentations : l'API est quasiment intégralement documentée (quelques points restent vierges de toute explication), et les auteurs du framework ont publié un livre en début d'année 2007, "The definitive Guide to Symfony", qui explique, pas à pas, l'emploi de Symfony. C'est ce livre qui, depuis, sert de documentation officielle à Symfony via sa mise à disposition gratuite sur le site officiel du framework.

Un des derniers gros avantages de Symfony en tant que framework de développement est, justement, qu'il est sans doute un de ceux qui assure le mieux son rôle de framework, c'est-à-dire de *cadre de développement*. Le formalisme de développement et les conventions de codage sont bien définis, de sorte que chaque partie du code d'un projet trouve une place logique dans l'arborescence.

3. Points faibles du framework

Différentes études font état d'une relative lenteur de Symfony, mais il faut être méfiant avec des comparaisons aussi abruptes. Tout d'abord, le propre d'un projet Web digne de ce nom est de ne pas se limiter à un simple echo "Hello World". Ensuite, différents exemples montrent que Symfony peut être employé dans des environnements très demandeurs en terme de performances (voir les références).

4. Organisation des fichiers



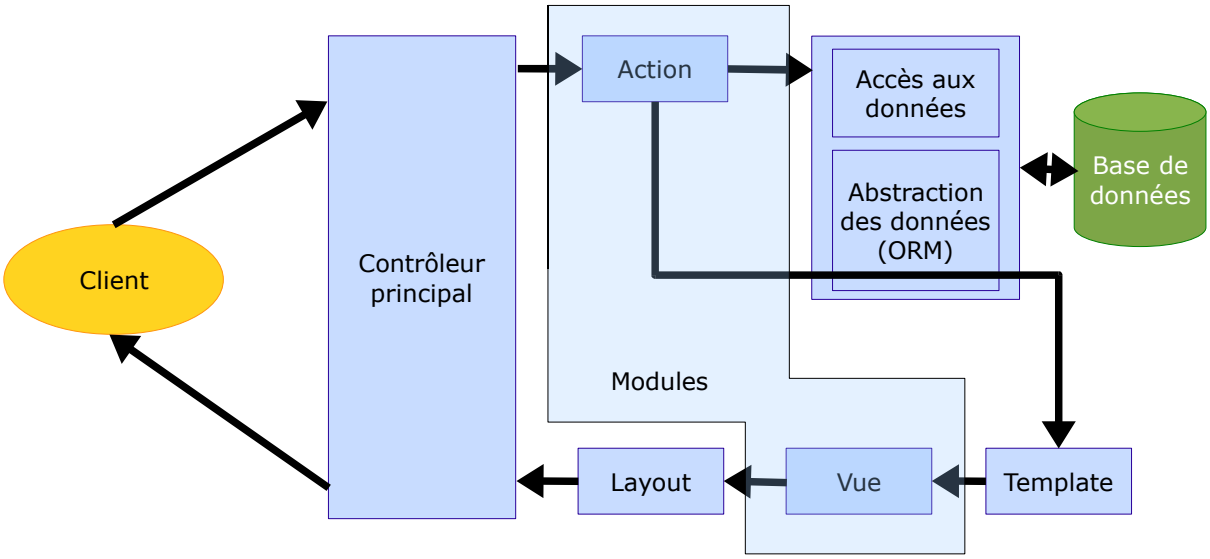
Un projet Symfony propose une structure hiérarchique d'organisation du code, sous forme d'applications elles-même divisées en différents modules. Au sein de chaque module, on trouve différents dossiers :

- /actions
- /config
- /templates
- /validation

Symfony emploie la syntaxe YAML pour ses fichiers de configuration. Cette syntaxe, même si elle ne répond pas au formalisme de XML, a cependant l'avantage de la simplicité. L'extrait ci-dessous donne un exemple de fichier de configuration de Symfony.

```
all:
  version:         1.5
  .general:
    tax:           19.6
  default_user:
    name:          John Doe
  mail:
    webmaster:     webmaster@example.com
    contact:       contact@example.com
dev:
  mail:
    webmaster:     dummy@example.com
    contact:       dummy@example.com
```

5. Traitement d'une requête



6. Processus d'internationalisation

Symfony est sans doute le premier framework à proposer une internationalisation (i18n), reprise du framework Prado[12], et une localisation (l10n) intégrales, à base de fichiers XLIFF. C'est l'un des gros points forts du framework puisque le principe est fort bien explicité dans la documentation et relativement simple à mettre en œuvre.

7. Processus de contribution au framework

Le projet Symfony s'appuie sur un atelier de développement composé de Trac et de Subversion, et dispose d'une communauté vraiment vivante. Après obtention d'un compte d'accès au Trac, chaque utilisateur du framework est libre de créer un ticket, de proposer un patch, etc. Le wiki de Symfony est le lieu approprié pour discuter des évolutions majeures à apporter au framework.

Les plugins, déjà très nombreux, peuvent être contribués par le biais d'une procédure plus simple : il suffit simplement d'attacher le plugin en pièce jointe à une page du wiki pour que celui-ci soit disponible à l'installation par le biais de l'outil en ligne de commande, pour tous les utilisateurs du framework.

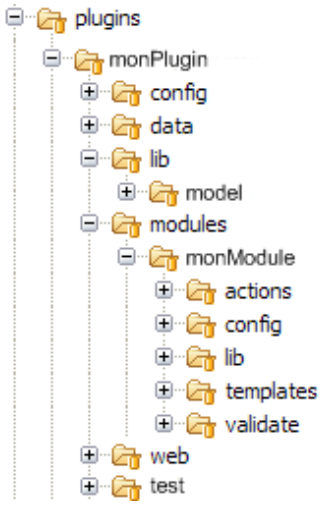
Outre le forum officiel, très actif, et les mailing-lists développeur et utilisateur, toutes deux anglophones, des listes de diffusion locales sont disponibles. La liste de diffusion symfony-fr, par exemple, permet des discussions en français au sujet du framework.

8. Gestion des extensions

Un plugin Symfony ajoute des fonctionnalités à la base fonctionnelle initiale. Ces fonctionnalités peuvent être de différentes sortes : modification du comportement par défaut de l'ORM, gestion des droits utilisateurs, librairies Javascript, etc. L'installation d'un plugin Symfony est largement simplifiée par l'emploi de l'interface en ligne de commande :

12 http://www.pradosoft.com

```
$ ./symfony plugin-install http://plugins.symfony-project.com/NomDuPlugin-1.0.0.tar.gz
```



Structurellement, un plugin Symfony se présente sous la forme d'un dossier placé dans le dossier « plugins/ », à la racine du projet. La structure arborescente de celui-ci est décrite dans le schéma ci-dessous, chaque dossier étant évidemment optionnel :

Au moment de l'écriture de ce document, Symfony propose plus de 150 plugins, qui vont de la simple inclusion de librairies externes (Swift, jQuery, etc.) à l'application complète (CMS, forum, blog, etc.).

9. Courbe d'activité autour du framework

La première version publique de Symfony, bien qu'alors encore en béta, est apparue en Octobre 2005. Symfony a finalement été publié en version 1.0 stable en janvier 2007. Pour l'avenir, les prochaines mises à jour majeures seront Symfony 1.1, disponible en mai ou juin 2008, puis Symfony 2.0, dont la date de sortie n'est, à l'heure actuelle, non définie.

En ce qui concerne le support, les mailing-lists (googlegroups) concernant Symfony sont très réactives (plus de 16000 messages depuis la création de ces groupes).

10. Implémentation de référence

Une implémentation de référence de Symfony a été réalisée peu après le lancement de Symfony, en Décembre 2005 : il s'agit d'Askeet, le Digg-like du pauvre. Remise à jour de manière irrégulière, cette implémentation ne reflète malheureusement plus toute la qualité et la richesse de Symfony, et il est fréquent que les utilisateurs débutants aient du mal à faire fonctionner l'application, même en suivant parfaitement le guide. Cependant, Askeet reste d'une grande aide pour les utilisateurs confirmés, et permet de bien intégrer les grands concepts architecturaux du framework.

De plus, Symfony met à disposition un certain nombre de screencasts ou d'exemples de fonctionnement d'applications, permettant une prise en main pas à pas du framework à travers des cas divers et variés.

11. Quelques références

- Delicious - http://del.icio.us/
- Yahoo! Answers - http://answers.yahoo.com/[13]

13 Voir la présentation de Dustin Whittle, http://assets.en.oreilly.com/1/event/3/A%20Symfony%20Answer%20Presentation.pdf

- Symfonians - http://symfonians.net/
- Richelieu Finance - http://www.richelieufinance.fr/

12.L'avis de Clever Age

Symfony est à l'heure actuelle LE framework qui a le vent en poupe. En plus d'être de plus en plus utilisé, une vrai dynamique s'est créée autour de la documentation du framework dans de nouvelles langues, des plugins (ou les mises à jour et évolutions sont fréquentes), etc. L'abondance de documentation et d'exemples d'applications fonctionnelles distribuées en open source rendent son apprentissage aisé.

Clever Age recommande Symfony pour la plupart des projets PHP, notamment ceux où une démarche d'industrialisation est engagée. Symfony est particulièrement adapté aux projets volumineux (au delà de 1000 j/h), mais il sera également très profitable aux projets plus restreints (à partir de 50 j/h).

8. Zend Framework

1. Origines et motivations du framework

Ce framework a été créé en 2006 par la société Zend, qui est depuis de nombreuses années la façade commerciale de PHP. L'objectif, pour Zend, est de parvenir à créer LE framework PHP5 de référence.

Pour ce faire, Zend a mis en place une infrastructure de gestion de projet libre complète (et complexe), mettant en œuvre des mécanismes semblables à ceux que l'on peut trouver dans les communautés Java / Apache (RFC, incubateurs, etc). Une fois les formalités administratives effectuées, Zend donne accès aux développeurs à un système de remontée d'erreurs, de proposition d'améliorations et de développement.

La qualité des composants du framework est élevée, puisque le processus de développement est très fortement contrôlé par la société Zend, notamment via la signature d'une Licence de Contribution, si l'on souhaite participer à son amélioration.

2. Points forts du framework

Zend, l'acteur majeur du monde PHP, est à l'origine du Zend Framework : c'est là un de ses principaux points fort, d'autant que le framework est maintenu par des salariés spécifiquement affectés à son développement. Pour ces raisons, la pérennité du framework semble plutôt bien assurée.

De plus, Zend mise beaucoup sur le respect de conventions d'écriture et de motifs de conception. L'abondance de documentation du framework ainsi que le grand nombre de tests unitaires réalisés sur des portions du code sont un gage de sérieux.

La participation à l'amélioration du framework étant très stricte, la moindre modification effectuée suit un mécanisme de surveillance très strict, garantissant la qualité du framework.

3. Points faibles du framework

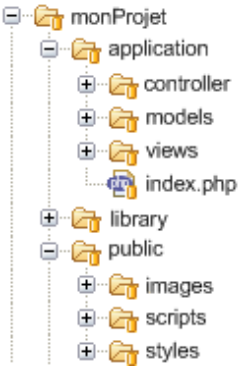
Un de ses point fort en fait aussi sa faiblesse : le mode drastique de contribution au Zend Framework rend son évolution ralentie. En effet, les communautés sont moins actives que sur d'autres frameworks.

Lors de la première installation, le Zend Framework peut dérouter. En effet, il faut tout créer soi-même, y compris le « bootstrap » qui servira à initialiser l'application (avec l'accès aux librairies, l'accès au contrôleur, etc.).

L'intégration avec des composants d'autres frameworks semble, à l'heure actuelle, assez complexe voire impossible. La qualité de certains composants est discutable comme, par exemple, le composant Zend_Rest_Server, dans lequel le respect du MVC est discutable.

Enfin, la gestion des routes ou encore la gestion des formulaires avec des retours d'erreurs sont des éléments manquants, même s'ils nous semblent pourtant vitaux pour

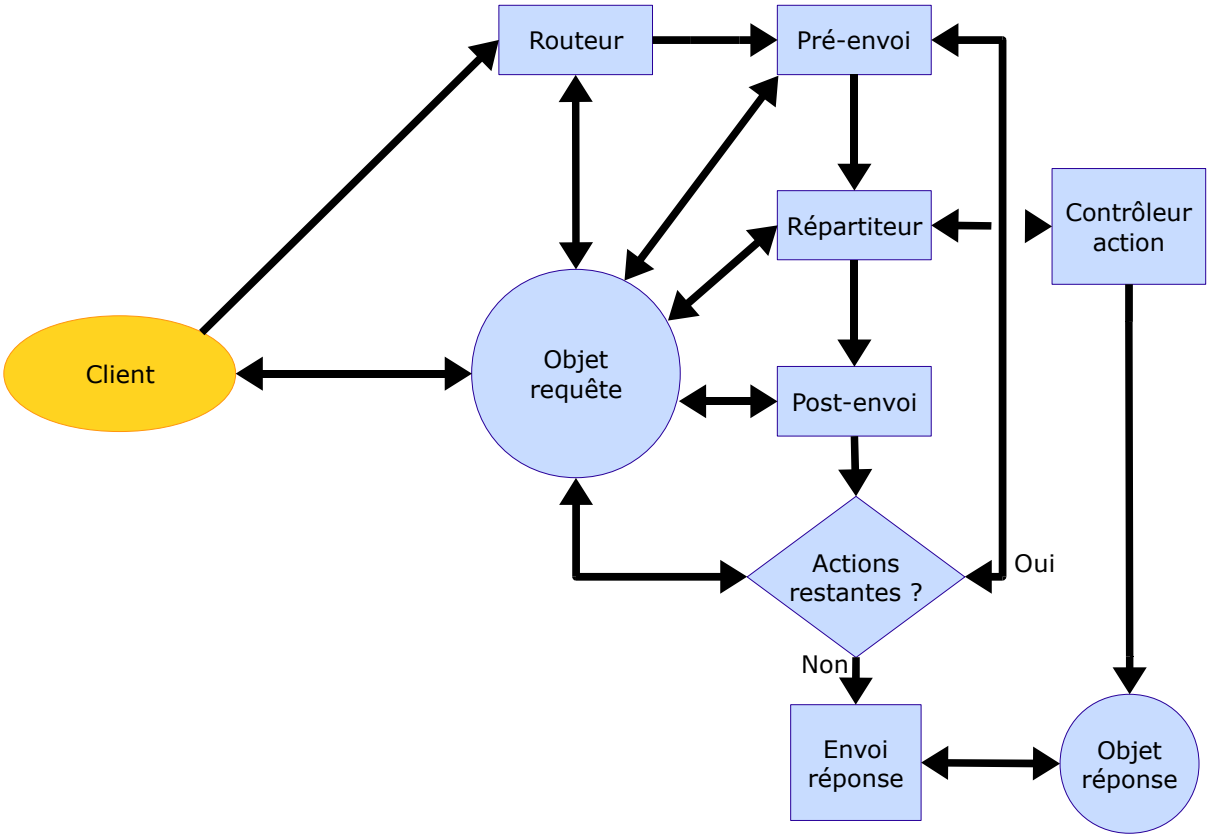
un framework.



4. Organisation des fichiers

Le Zend Framework laisse au développeur le choix de l'organisation des fichiers. Le manuel propose cependant une structure de base, que chaque projet peut librement adapter.

5. Traitement d'une requête

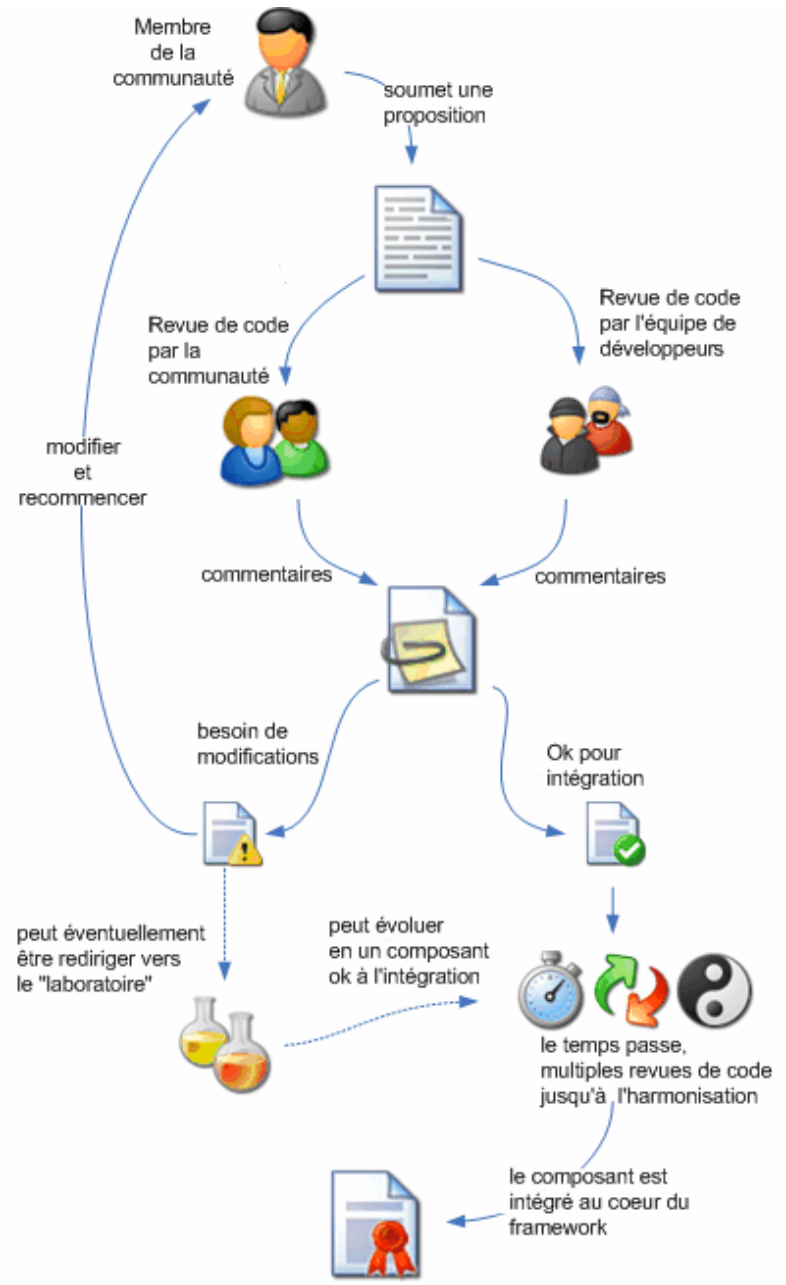


6. Processus d'internationalisation

Nativement, Zend Framework inclut divers composants comme Zend_Translate, Zend_Locale, Zend_Date et Zend_Mesure pour gérer l'internationalisation et la localisation des applications. De nombreux formats sont supportés, dont les formats « industriels », comme par exemple XLIFF.

7. Processus de contribution au framework

Comme nous l'évoquions en présentation, le processus de contribution au Zend Framework est relativement complexe. Bien que bon nombre de développeurs soient freinés dans leurs propositions d'améliorations, cette structure permet à Zend de garantir la qualité du code source de son framework.



Source : Zend

8. Gestion des extensions

Le Zend Framework est entièrement extensible via l'utilisation des nombreux composants natifs qu'il propose. Cependant, il n'existe pas de notion de « plugin » comme on peut en trouver dans les autres frameworks de ce livre blanc. De plus, l'emploi d'un composant nécessite l'inclusion de tous les composants, ce qui est plutôt contraignant.

D'autres frameworks ont éét développés à partir du Zend Framework, comme par exemple le Zend Framework Extended[14], qui apporte un certain nombre de fonctionnalités supplémentaires à l'outil. Nous pouvons également citer

14 http://www.axel-etcheverry.com/wiki/index.php/Zend_framework_extended

Zendex[15], qui est un système de gestion d'extensions pour le Zend Framework.

9. Courbe d'activité autour du framework

La courbe d'activité autour du Zend Framework montre que son processus de développement est assez rapide. La première version bêta est sortie en début d'année 2007, et la première version stable (1.0) a été publiée à la fin juin 2007. La dernière version stable à ce jour (1.5) date de la mi-mars 2008.

10. Quelques références

- IBM
- Nokia
- Intranet M6

11. L'avis de Clever Age

Le Zend Framework a le grand avantage de bénéficier de toute l'expérience de Zend, acteur majeur dans le domaine de la professionnalisation et l'industrialisation de PHP : son sérieux ne peut que rassurer face à l'avenir du framework. Cependant, la structure modulaire du framework, bien qu'elle puisse être un avantage selon les projets, peut rendre le travail en équipe assez douloureux, si celle-ci n'est pas correctement équipée en outils documentaires.

Le Zend Framework est donc à réserver aux projets volumineux (plus de 500 j/h) présentant des contraintes techniques fortes. Par ailleurs, l'utilisation du Zend Framework nécessite un effort documentaire tout particulier, afin de documenter toutes les conventions adoptées lors de la phase de développement.

15 http://code.google.com/p/zendex/

9. Évaluation par la méthode QSOS

1. Tableau récapitulatif

La méthode QSOS *(Qualification et de Sélection de logiciels Open Source)* a été conçue pour évaluer les logiciels libres et OpenSource de manière objective.

Voici la grille d'évaluation, disponible sur le site de QSOS[16], axée autour des critères permettant d'estimer les risques encourus par un utilisateur.

Pour chaque fonctionnalité, la règle de notation est la suivante:

- 0 si la fonctionnalité n'est pas couverte,
- 1 si la fonctionnalité est partiellement couverte,
- 2 si la fonctionnalité est totalement couverte.

Pérénité intrinsèque		Cake PHP	Symfony	Zend Framework	Code Igniter
Maturité					
	Âge	1 date de sortie: 2005	1 date de sortie: décembre 2005	1 date de sortie: 2007	1 date de sortie: 2006
	Stabilité	2 version stable 1.19	2 version stable 1.0	2 version stable 1.0	2 version stable 1.6
	Historique, problèmes connus	2	2	2	2
		un historique des bugs est disponible sur le site			
	Probabilité ou provenance d'un FORK	2	1 Symfony est un FORK de Mojavi2	2	0 Kohana en est un FORK
Adoption					
	Popularité	2	2	2	1
	Références	2 de nombreuses références dont Mozilla Addons	2 nombreiuses références (Yahoo!, etc.). cf. http://symfonians.net/applications/	2 nombreuses références (IBM, Nokia, etc.). cf. http://framework.zend.com/community/applications/	1 quelques références dont Emmaus.uk
	Communauté des contributeurs	2 la communauté est active sur les forums dédiés, blogs et Google Groups	2 la communauté est importante et active sur les forums dédiés, blogs et Google Groups	2 la communauté est importante et active sur les forums dédiés et blogs	2 la communauté est active sur les forums dédiés
	Publications	1	1	1	0

16 http://www.qsos.org/?lp_lang_pref=fr&page_id=3

		parution d'un livre en anglais « CakePHP Recipes »	parution d'un livre en anglais « The Definitive Guide to Symfony »	parution de livres en anglais « Architect's Guide to Programming with Zend Framework » et « Zend Framework in action »	aucun livre paru
Direction des développements					
	Equipe dirigeante	1 Larry E.Masters et 4 autres « core developpers »	1 Fabien Potencier de Sensio Labs	2 équipe de direction de Zend Technologies	2 équipe de direction de EllisLab
	Mode de direction	1 une personne	1 une personne	2 groupe de personnes	1 une personne
Activité					
	Nombre de développeurs, identification, turnover	2 une douzaine de personnes	2 une cinquantaine de personnes	2 une cinquantaine de personnes	0 pas d'information à ce sujet
	Activité autour des bugs	2 un suivi des bugs est disponible via un trac wiki	2 forum réactif et suivi des bugs disponibles via un trac wiki	2 suivi des bugs disponibles au travers d'un wiki dédié	2 suivi des bugs accessible via la page BugTracker
	Activités autour des fonctionnalités	1 l'évolution du produit se fait sur la démarche de l'équipe de développeurs	2 une section du forum est consacrée pour la demande de nouvelles fonctionnalités	2 une page IssueTracker est consacrée pour la demande de nouvelles fonctionnalités	2 une section du forum est consacrée pour la demande de nouvelles fonctionnalités
	Activités sur les releases	2 release en cours: 1.2	2 release en cours: 1.1	2 release en cours: 1.5	2 release récente: 1.6
Indépendance des développements					
	Indépendance des développements	1	1	1	1

SOLUTION INDUSTRIALISEE	Cake PHP	Symfony	Zend Framework	Code Igniter
Services				
	Formation	0 aucune offre de formation n'est disponible pour ce framework	2 Anaska, Sqli ou encore Clever Age sont par exemple des prestataires proposant une formation Symfony	2 Zend et Anaska sont par exemple des prestataires proposant une formation au Zend Framework	0 aucune offre de formation n'est disponible pour ce framework
	Support	0 aucune offre de support hormis forums, mailing-lists et Google Groups	2 plusieurs offres existantes (Clever Age, Sensio, etc), forums, mailing-lists et Google Groups	1 Zend technologies offre un support pour ce framework via son réseau	0 aucune offre de support hormis les forums et mailing-lists
	Conseil	Non Applicable	Non Applicable	Non Applicable	Non Applicable
Documentation					
	Documentation	2	2	2	2

		documentation présentée sous la forme d'un manuel utilisateur accessible uniquement en ligne	documentation importante présentée sous la forme d'un manuel utilisateur disponible en ligne et en livre	documentation détaillée présentée sous la forme d'un guide de référence disponible et téléchargeable en ligne	documentation présentée sous la forme d'un guide de l'utilisateur accessible en ligne et téléchargeable avec le framework
Méthode qualité					
	Assurance qualité	1	2	2	1
	Outillage	1	2	2	1
Packaging					
	Source	1	1	1	1
	Debian	2	2	2	1
		Paquets Debian disponibles			Aucun paquet Debian disponible
	FreeBSD	1	1	1	1
	HP-UX	1	1	1	1
	Mac OS X	1	1	1	1
	Mandriva	1	1	1	1
	NetBSD	1	1	1	1
	OpenBSD	1	1	1	1
	RedHat/Fedora	1	1	1	1
	Solaris	1	1	1	1
	SuSE	1	1	1	1
	Windows	1	1	1	1
Exploitabilité					
	Facilité d'utilisation, ergonomie	Non Applicable	Non Applicable	Non Applicable	Non Applicable
	Administration / Supervision	Non Applicable	Non Applicable	Non Applicable	Non Applicable

ADAPTABILITE TECHNIQUE		Cake PHP	Symfony	Zend Framework	Code Igniter
Modularité					
	Modularité	1	2 plugins	2 extensions	1
Travaux dérivés					
	Facilité technique de modification du code existant	2	2	2	2
	Facilité d'extension du code	2 possibilité d'utiliser des composants, modèles et plugins additionnels	2 possibilité d'utiliser des plugins additionnels	2 possibilité d'utiliser des extensions	2 possibilité d'ajouter des plugins

STRATEGIE		Cake PHP	Symfony	Zend Framework	Code Igniter
Licence					
	Permissivité	2 Licence MIT	2 Licence MIT	2 nouvelle Licence BSD	2 Licence MIT
	Protection contre les forks commerciaux	0 pas de protection par définition	0 pas de protection par définition	0 pas de protection par définition	0 pas de protection par définition

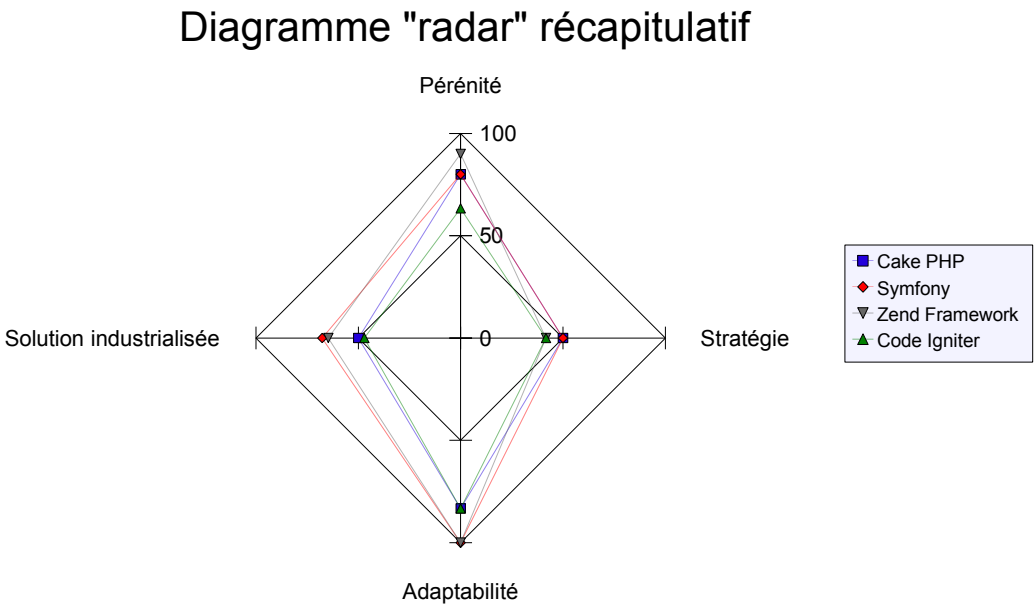
		de la licence MIT	de la licence MIT	de la licence BSD	de la licence MIT
Détenteurs des droits					
	Détenteurs des droits	1	1	1	1
Roadmap					
	Roadmap	1 existence d'une roadmap sans planning prévisionnel	1 existence d'une roadmap sans planning prévisionnel	1 existence d'une roadmap sans planning prévisionnel	0 aucune roadmap publiée
Sponsor					
	Sponsor	1 Sponsor unique: Cake Development corp.	1 Sponsor unique: Sensio Labs	1 Sponsor unique: Zend technologies	1 Sponsor unique: EllisLabs
Indépendance stratégique					
	Indépendance stratégique	1	1	0	1

2. Notes globales

Exprimée en pourcentage du score maximal atteignable, voici la répartition des notes de ces 4 frameworks suivant les grandes catégories analysées :

	Cake PHP	Symfony	Zend Framework	Code Igniter
Pérénité	80,00%	80,00%	90,00%	63,33%
Solution industrialisée	50,00%	67,65%	64,71%	47,06%
Adaptabilité	83,33%	100,00%	100,00%	83,33%
Stratégie	50,00%	50,00%	41,67%	41,67%
Moyenne	**65,83%**	**74,41%**	**74,10%**	**58,85%**

Une autre présentation, sous la forme d'un graphique radar :



10. Conclusion

Pour une entreprise, choisir un framework de développement Web n'est pas une tâche facile, dans la mesure où ce choix implique des problématiques assez variées : conception interne du framework, facilité de prise en main, qualités techniques, évolution future, etc.

Mais, au delà des simples qualités des frameworks comparés dans ce livre blanc, plusieurs constantes sont à prendre en compte avant de commencer une industrialisation des développements sur la base d'un de ces frameworks : le mode de fonctionnement des équipes de développement, d'infrastructure et de maintenance, les besoins fonctionnels actuels et futurs, les contraintes de la plateforme d'hébergement.

Dans tous les cas, le choix d'un framework de développement ne peut qu'améliorer la qualité d'un existant qui, sans procédures, conventions ou normes, est rarement bien conçu. Reste encore à déterminer quel est le framework qui conviendra le mieux à vos besoins et à votre contexte.

11. Bibliographie

- [CRU 2008] CruiseControl - http://cruisecontrol.sourceforge.net/
- [GNU 2008] Various Licenses and Comments about Them - http://www.gnu.org/philosophy/license-list.html
- [SEL 2008] Slenium - http://www.openqa.org/selenium/