Développement professionnel en PHP, quels frameworks s'offrent aux entreprises?

Travail de diplôme réalisé en vue de l'obtention du diplôme HES

par :

Jérôme Ehrbar

Conseiller au travail de diplôme :

Rolf Hauri

Genève, le 18 octobre 08

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de gestion

Déclaration

Ce travail de diplôme est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention d'un bachelor en Informatique de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de diplôme, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de diplôme, du juré et de la HEG.

« J'atteste avoir réalisé seul(e) le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève le 20 octobre 08

Jérôme Ehrbar

Sommaire

Ce document a pour thème « Développement professionnel en PHP, quels frameworks s'offrent aux entreprises? ». Il existe sur le marché différent outils aidant aux développements. Les frameworks en font partis. Le but de ce travail était de sélectionner différents frameworks PHP existant sur le marché. Après avoir fait une sélection de frameworks (en fonction de leur intérêt, de leur popularité,…), une analyse globale a été réalisée. Pour pouvoir réaliser cette analyse, il était important de définir des critères de comparaisons. Une fois ces derniers définis, chaque framework de la sélection a été évalué avec chaque critère. De cette analyse ressort une nouvelle sélection plus limitée de frameworks. Le but étant ensuite de réaliser une application exemple avec les frameworks de cette sélection plus exclusive. L'application n'est pas d'une extrême complexité. Il s'agit d'une liste de meubles. Le but étant de pouvoir consulter la liste des meubles, ajouter, modifier ou supprimer un meuble, soit les opérations standard d'une base de données. Dans ce travail, trois frameworks ont été utilisés pour créer l'application exemple. Pour chaque framework, on retrouve les différentes étapes nécessaires pour réaliser l'application. On trouve ensuite une analyse de la structure de l'application (celle-ci change d'un framework à un autre). Cela amène ensuite à une conclusion plus spécifique à chaque framework. Le but étant de définir si son utilisation serait un avantage ou non pour une entreprise et les circonstances.

Une conclusion générale termine ce travail. Elle reprend le thème principal (quels frameworks s'offrent aux entreprise) et met une nouvelle question en avant : « est-ce intéressant d'utiliser un framework pour une entreprise ? ». En réponse, il y a certains éléments à prendre en compte. Un framework PHP n'est pas forcément un outil simple à utiliser (du moins ceux testés). La première opération, l'installation, est spécifique à chaque framework. Elle consiste généralement à configurer le serveur, à installer des modules PHP, et à définir une bonne arborescence des répertoires et des fichiers. Cependant, la moindre erreur fait que le framework ne fonctionne pas. Ensuite, l'utilisation d'un framework ne se fait pas en trente secondes. Cela est dû au faite que chaque framework utilise un « langage » propre à lui-même. Le langage de base est bien sûr du PHP. Il s'agit plutôt de l'utilisation des différents outils intégrés au framework, du format de

certains fichiers (et donc du langage) qui varient d'un framework à un autre. Cela est dû au fait que les frameworks ne sont pas basés sur un seul standard. Il faut donc chaque fois du temps pour s'habituer à leur utilisation.

Le choix d'un framework à la place d'un autre n'est pas le réel problème. Comme on le voit dans ce document, le résultat d'un framework à un autre est assez identique. On peut donc en déduire qu'une entreprise utilisera un framework si elle réalise une importante application ou si elle réalise plusieurs applications. Pourquoi ? A cause du temps nécessaire pour comprendre et pouvoir utiliser un framework. De plus, il est plutôt conseillé d'en choisir un et de ne pas changer en cours de développement. Cela aurait pour conséquence de redemander du temps pour se familiariser avec le nouveau framework. Cependant, après avoir compris l'utilisation d'un framework, l'apprentissage pour en maîtriser un autre est un peu plus rapide, mais prend quand même un peu de temps.

Selon mon avis, en ayant utilisé certains frameworks, ce qu'il leur manquait, c'était une interface graphique complète. J'entends par là que l'on trouve une barre d'outils permettant directement de créer par exemple un formulaire avec ses paramètres (action, champs,…). Car pour le moment, tout ou presque se code en programmant manuellement. Un framework proposant ceci deviendrait un outil simple et rapide à utiliser, ce qui encouragerait de plus en plus d'entreprises à utiliser un framework et faciliterait la prise en main. Pour conclure, dans l'avenir, je pense que l'utilisation de tels outils se fera de plus en plus.

Table des matières

Tableau d’analyse des frameworks

Liste des Figures

Introduction

Ce document est une analyse des frameworks. Il consiste à déterminer quels frameworks s'offrent aux entreprises. Le but des frameworks est de faciliter le développement d'application. Nous nous intéresserons uniquement aux frameworks PHP.

Le chapitre 1 de ce document commence par une explication et répond à la question « qu'es-ce qu'un framework » ? Il permet ainsi au lecteur à comprendre le but et l'intérêt des frameworks.

Le chapitre 2 présente une sélection des frameworks potentiellement intéressant pouvant répondre au titre de ce document. Cette sélection permet déjà d'identifier certains frameworks que l'on trouve sur le marché.

Le chapitre 3 contient la définition des critères d'analyse permettant par la suite d'analyser la sélection des frameworks du chapitre 2.

Le chapitre 4 présente un tableau d'analyse. Ce tableau contient la liste des frameworks du chapitre 2 analysés selon les critères définis au chapitre 3. C'est sur la base de ce tableau que sera définie une sélection plus limitée de frameworks au chapitre 5.

Le chapitre 5 contient une nouvelle sélection de frameworks. Il s'agit de ceux s'averrant potentiellement intéressant se trouvant dans le tableau d'analyse du point 4.

Le chapitre 6 présente une application standard. L'idée est de réaliser cette application avec les frameworks sélectionnés au chapitre 5 dans le but de voir les différences entre eux et de pouvoir mieux juger de leur intérêt.

Le chapitre 7 illustre la partie pratique de ce travail. Elle représente en détails chaque fois la même application, mais avec un framework différent. Le but étant d'avoir une analyse pratique plutôt que théorique. Elle met en avant les différences entre les différents frameworks.

Le chapitre 8 de ce document est la conclusion finale sur l'ensemble de ce travail. Elle essaie de montrer l'intérêt d'utiliser un framework pour une entreprise et ce que pourrait être un framework dans quelques années.

Le chapitre 9 contient la liste des différents ouvrages utilisés dans ce document. On y trouve aussi la liste des ouvrages m'ayant aidé à réaliser l'application exemple avec chaque framework. Cela peut être utile pour les personnes intéressées par le sujet et désirantes de savoir comment utiliser un framework.

1. Qu’est-ce qu’un framework ?

Les frameworks sont de plus en plus utilisés par les concepteurs d’application. Nous allons nous intéresser aux frameworks PHP. Le grand avantage des frameworks est de proposer une structure déjà implantée. Cela permet d’économiser du temps dans la phase de conception d’une application PHP.

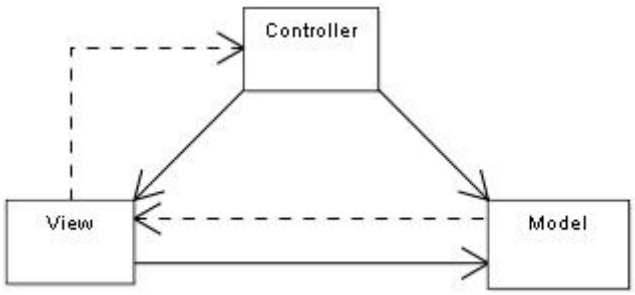
On peut définir un framework comme une aide à la réalisation d’application (PHP dans le cadre de ce document). Il propose un mécanisme de fonctionnement basé sur la réutilisabilité. En plus de cela, il propose une palette d’outil facilitant la réalisation d’application. A l’heure actuelle, on trouve une multitude de frameworks prêts à être utilisés. Un framework est généralement divisé en deux parties :

- Structure d’application (architecture logiciel)
- Bibliothèque de composants (boîte à outils)

Ce qui fait que les frameworks sont de plus en plus utilisés, c’est aussi la facilité de pouvoir les adapter au besoin d’un projet. La plupart de ces frameworks sont en plus Open Source. Ce qui fait qu’il n’y a pas de licence à devoir payer pour les utiliser. On peut les représenter comme un ensemble de briques logicielles que l’on peut utiliser comme on le désire, comme des legos.

On peut qualifier les frameworks de progiciels. Les progiciels sont ce qu’on peut appeler des produits « clé en main », c’est à dire prêt à l’emploi. Le mot vient de la fusion des mots « produit » et « logiciel ».Une bonne partie de ces frameworks PHP ont une architecture de type MVC (Model View Controller). On appelle architecture la manière qu’à un framework de fonctionner. Voici comment se compose cette architecture :

FIGURE 1
MODÈLE MVC



Source: http://en.wikipedia.org/wiki/Image:ModelViewControllerDiagram.svg)

Dans ce modèle, les flèches représentent la communication entre les différents éléments. On voit par exemple que la View qui s'occupe de l'affiche va communiquer avec le Model pour lui demander la structure de la page qui sera afficher.

Le **model** (modèle) représente le comportement d'une application. C'est là que sont stockés l'ensemble des données d'une application.

La **view** (vue) représente l'interface avec laquelle l'utilisateur va interagir. Il s'agit surtout de tout ce qui concerne l'affichage.

Le **controller** (contrôleur) est celui qui effectue le « dialogue » entre le *model* et la *view*. Par exemple, lorsque l'utilisateur modifie des données via l'interface, c'est le *controller* qui va demander au *model* d'appliquer ces changements.

Nous allons maintenant passés au chapitre suivant concernant une sélection des frameworks PHP se trouvant sur le marché.

2. Panorama des frameworks du marché

Il existe actuellement un grand nombre de frameworks. Cependant, il existe différents types de frameworks. Certains sont ce qu'on appelle des CMS[1]. Il existe différents CMS, certains utilisent comme langage de programmation Java et d'autres PHP.

Un CMS permet par exemple de réaliser un blog. On a une structure prédéfinie que l'on peut personnaliser en déterminant par exemple l'apparence du blog.

L'inconvénient d'un CMS est qu'il fonctionne sans base de données. Ce qui fait que nous n'allons pas choisir dans la liste de nos frameworks des CMS. Nous allons choisir des frameworks complets permettant la réalisation d'une application.

Ce document contient donc une sélection limitée de frameworks. Le critère de sélection était aussi de choisir des frameworks les plus répandus .Car certains ne sont utilisés que par un nombre limité de personnes. Il y en a aussi qui ne sont pas encore complet.

Comme on le voit vers la fin de ce chapitre, beaucoup de frameworks présentent des caractéristiques générales identiques. C'est pourquoi il n'y a qu'un certain nombre de frameworks. Le but a été aussi d'essayer d'en prendre certains présentant des caractéristiques différentes afin d'avoir des critères de comparaison.

Voici une sélection de différents frameworks PHP 5 existant sur le marché. La plupart sont téléchargeables depuis le site de la société qui les a créés :

2.1 Zend Framework

[1] Un CMS signifie en français : système de gestion de contenu Open Source

Ce framework a été réalisé par la société Zend en partenariat avec IBM. Cette société propose différents produits, tous ayant pour but de faciliter la gestion de projet en PHP (création, gestion, protection code source, production…).

Il s'agit d'un outil complet et très sérieux. Il ne fonctionne qu'avec PHP 5. Il s'adapte facilement à chaque projet. Il intègre même des outils utiles pour la conception d'application de type Web 2.0.

Version : 1.5.3

Taille : 5.17Mo (zippé)

Licence : Open Source

Société : Zend

Site Officiel : http://www.zend.com/fr/

2.2 Copix

Créé par la société Copix Team, ce framework a été créé en 1999. Il existe plusieurs versions. La dernière version de Copix est en deux versions : allégée et avec des modules complémentaires. Voici une liste des grandes caractéristiques de ce framework :

Un ensemble d' API (Application programming interface soit en français : interface de programmation applicative)
Structure modulaire
Dispatcheur d'URL
Système de plugins pour intégrer des fonctionnalités optionnelles

Système d'authentification et de gestion des droits des utilisateurs

Système d'installation automatique des modules développés

Des DAO automatiques facilitant l'interaction avec la base

Gestion de plusieurs langues

Un système de cache accélérant le temps de réponse

Des librairies de génération HTML

Personnalisation graphique de vos applications

Des tests unitaires automatisés (rapport de code)

Des modules prêts à l'emploi (wiki, moteur de recherche,…)

(http://www.copix.org/index.php/wiki/Presentation)

Version : 3.0.3

Taille : 4.79Mo (zippé)

Licence : Open Source

Société : Copix Team

Site Officiel : http://www.copix.org

2.3 CakePHP

Créé par la société Cake Software, ce framework a été créé en 2005. Ce framework est en train de devenir de plus en plus populaire. Il est à noter que, comme bien souvent, la principale communauté de ce framework est anglophone. Voici une liste des grandes caractéristiques de ce produit :

Un ensemble d' API (Application programming interface soit en français : interface de programmation applicative)

Structure modulaire

Intégration de CRUD (Recherche des données dans la BDD simple)

Dispatcheur d'URL

Différents templates PHP déjà préfabriqués

Helpeurs de vue permettant l'utilisation de plusieurs langages (Ajax, JavaScript, HTML,…)

Formulaires préconçus

Contrôle des données

Système d'authentification et de gestion des droits des utilisateurs

(http://www.adullact.org/documents/fiche_cakephp.pdf)

Version : 1.2.0.7296-rc2

Taille : 1.11Mo (zippé)

Licence : Open Source

Société : Cake Software Foundation

Site Officiel : http://www.cakephp.org/

2.4 Jaz

Ce framework n'est pas un dès plus connu. Je dirais même que peu de personnes doivent le connaître. L'idée était justement de choisir un framework inconnu pour voir s'il valait la peine et s'il y avait des différences notoires avec les autres.

Ce framework a été créé pour un travail de recherche. Son auteur considère encore que Jaz est en version Beta. Pourquoi ? Car Jaz doit être testé encore plus finement par les développeurs et les utilisateurs afin de garantir que ce framework fonctionne à 100%. Cependant, il propose déjà une structure intéressante et la communauté ne cesse de l'améliorer. Pour les non-

anglophones, Jaz est un framework entièrement en français. Ce qu'on peut dire aussi, c'est qu'il a une taille très petite comparé aux frameworks que nous avons déjà vu, ce qui n'exclu en rien les différentes possibilités de ce framework.

Version : phpJaz.20050110.tar

Taille : 38Ko (zippé)

Licence : Open Source

Société : -

Site Officiel : http://phpjaz.velay.greta.fr/

2.5 Jelix

Créé par la société du même nom, Jelix propose une structure de code basé sur celle de Copix que nous avons vu précédemment. Malgré une base identique, ces deux frameworks sont un peu différents entre eux. Il est aussi considéré comme un framework permettant de gérer des sites à fortes charges. Par exemple le site « www.over-blog.com » l'utilise. Il faut quand même noter que ce site héberge plus de 500'000 blogs et que 5 millions de pages sont vues chaque jour. On peut donc le considérer comme un outil puissant.

Pour entrer un peu plus dans les détails, voici une liste générale des caractéristiques qu'il propose :

- Un ensemble d' API
- Structure modulaire
- Structure arborescente des fichiers du projet
- Un cadre et des normes de développement
- La structure MVC permet un découpage en couche du projet

Version : 1.0.5

Taille : 893Ko (zippé)

Licence : Open Source

Société : Jelix

Site Officiel : http://jelix.org/

2.6 Prado

Prado a été créé en 2005. Ce dernier a remporté le concours des développeurs de frameworks 2005 organisé par Zend. Ce framework, bien que complet, a une taille plus conséquente que l'ensemble des frameworks que ce document présente. Attention, la documentation n'est qu'en anglais. Voici une liste générale des différentes caractéristiques que propose ce framework :

- Structure modulaire
- Différents templates PHP déjà préfabriqué
- Dispatcheur d'URL
- Des tests unitaires automatisés (rapport de code)
- Structure arborescente des fichiers du projet

Version : 1.04

Taille : 16.9Mo (zippé)

Licence : Open Source

Société : Xisc

Site Officiel : http://www.xisc.com/

2.7 ***Symfony***

Symfony est le résultat d'un projet français. Il est né en 2005. Un de ces grands avantages est qu'il est accompagné d'une documentation complète. Il ne fonctionne qu'avec PHP 5. Ces caractéristiques sont les mêmes que la plupart des frameworks que nous avons vus :

- Structure modulaire
- Différents templates PHP déjà préfabriqué
- Dispatcheur d'URL
- Architecture MVC
- Structure arborescente des fichiers du projet

Version : 1.1

Taille : 2.8Mo (zippé)

Licence : Open Source

Société : Symfony

Site Officiel : http://www.symfony-project.org/

2.8 ***Hoa***

Hoa est un framework créé en 1999. Il n'y avait qu'un développeur. Ce framework n'est qu'en PHP 5. Ces caractéristiques sont passablement identiques à l'ensemble des frameworks présentés :

- Structure modulaire
- Différents templates PHP déjà préfabriqué
- Dispatcheur d'URL
- Gestion de plusieurs langues
- Structure arborescente des fichiers du projet
- Gestion de messageries

Version : 0.3.7b

Taille : 574Ko (zippé)

Licence : Open Source

Société : Hoa

Site Officiel : http://hoa-project.net/

3. Définition et justification des critères de comparaison

Nous allons maintenant comparer les différents frameworks présentés au chapitre précédent de ce document.

Avant de définir des critères de comparaison, nous allons voir s'il existe déjà des méthodes de comparaison reconnues.

Comme méthode, il existe QSOS[2]. Cette méthode a été mise au point par la société Atos Origin. Cette méthode est une licence de documentation libre GPL[3] . Cette licence définit les conditions légales de la distribution des logiciels libres.

QSOS est un processus itératif[4] divisé en quatre étapes. Voici ces différentes étapes :

> *« **Définir** les données de référentiel (types de licences, types de communautés, grilles de couverture fonctionnelle par domaine...) ;*
> ***Évaluer** les logiciels selon trois axes principaux : couverture fonctionnelle, risques du point de vue de l'entreprise utilisatrice, risques du point de vue du fournisseur de services (expertise, formation, support). Chaque axe est constitué d'un certain nombre de critères. Par exemple, l'axe des risques entreprise comprend : la pérennité intrinsèque, l'intégration, l'adaptabilité technique, le niveau d'industrialisation et la stratégie du projet. Ces critères étant eux-mêmes composés de sous-critères ;*
> ***Qualifier** le contexte spécifique d'une entreprise (ou d'un utilisateur) en effectuant une pondération des critères précédents ;*

[2] QSOS signifie Qualification et Sélection de logiciels Open Source

[3] GPL signifie en anglais General Public License (Licence Publique Générale)

[4] Itératif signifie que la méthode est basé sur une boucle. Cette dernière est parcourue autant de fois que nécessaire et chaque étape est répétée plusieurs fois

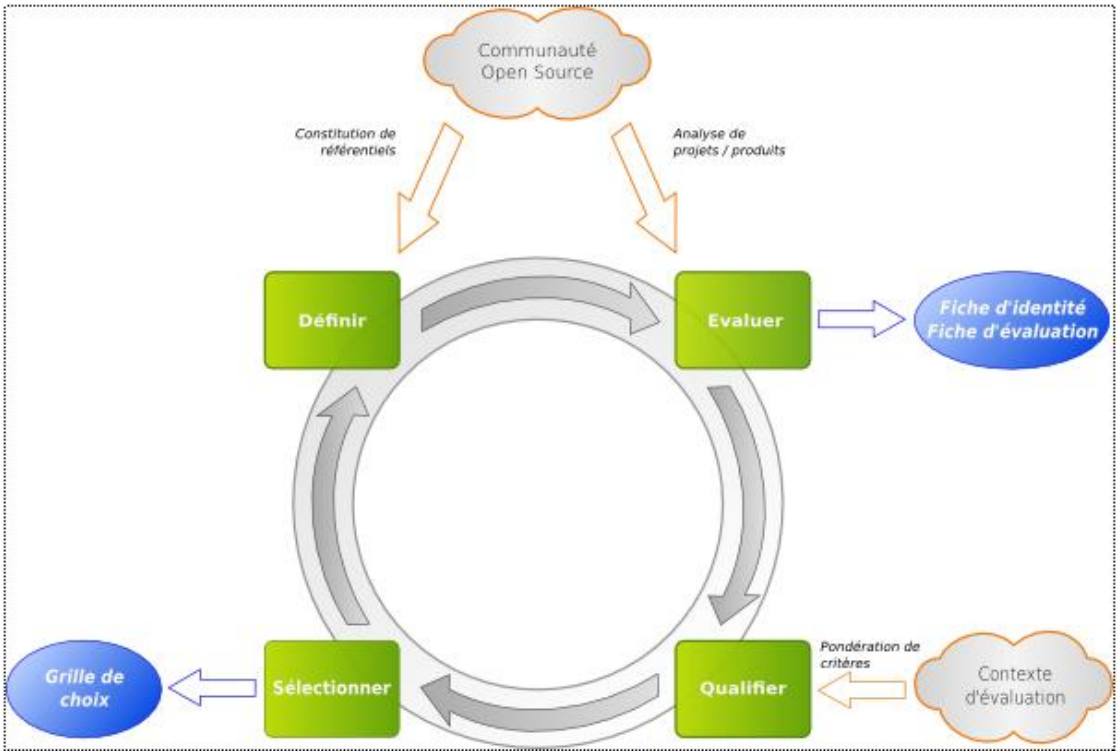
Sélectionner *et comparer les logiciels répondant aux besoins. »*

(QSOS. Wikipedia. http://fr.wikipedia.org/wiki/QSOS)

Pour mieux comprendre ce processus, voici une figure illustrant QSOS :

Figure 2

Processus QSOS



source de l'image : http://fr.wikipedia.org/wiki/Image:QSOS-processus.png

Le nuage représente n'importe quel logiciel Open Source. On commence par définir les mesures permettant d'évaluer le logiciel. Grâce à ces mesures on va pouvoir l'évaluer. On peut ensuite le comparer avec d'autres logiciels Open Source de même nature. Ensuite on peut déterminer lequel intégrera la grille de choix selon le résultat de la comparaison.

Cette méthode n'est donc pas spécifique aux frameworks PHP, mais à l'ensemble des logiciels libres. Dans le cadre de ce travail, elle n'est donc pas assez précise.

Il existe une autre méthode de comparaison. Celle-ci concerne seulement les frameworks PHP. Cette méthode a été créée par la société SMILE. Cette société a créé un ouvrage parlant des frameworks. Elle a ensuite utilisé des critères de comparaisons pour les évaluer :

Version de PHP

Modèle d'architecture

ORM

Templates

Cache

URL conviviales

Validation Formulaires

Ajax

Extensions

Génération de code

Internationalisation I18n

Recommandation

(Livre Blanc Frameworks PHP . Smile Motoristes Internet p 76)

Certains de ces critères sont intéressants comme « Version de PHP », « Modèle d'architecture »,…Nous allons les garder. Mais certains de ces critères ne sont peut être pas parlant comme « Internationalisation I18n », à moins d'être spécialisé dessus. Cependant, cette méthode est plus proche des frameworks que la première.

Par rapport à ce travail, la deuxième méthode est donc plus intéressante que la première puisqu'elle est appliquée aux frameworks. Cependant, il faut noter que cette dernière n'est pas reconnue officiellement comme un standard contrairement à la première. Nous allons donc garder certains éléments de ces méthodes.

Ce qui est important, c'est bien entendu la facilité d'utilisation de chaque framework, ainsi que les outils qu'il propose. Plus un framework propose d'outils aidant au développement, plus la phase réalisation d'une application sera rapide. Si l'on prend par exemple le critère « Tests unitaires automatisés », le framework proposant cet outil évite que le développeur d'application doive passer du temps à tester l'ensemble de son application. Alors que si un rapport est généré automatiquement, il sera plus rapide de savoir où son les erreurs de l'application .Il nous faut aussi un critère concernant l'apprentissage du framework.

Voici donc un choix de critères pouvant différencier un bon d'un moins bon framework :

- **Version de PHP supporté :** ce critère détermine avec quelle version de PHP un framework est compatible. Cela permet à une entreprise de déterminer si l'utilisation d'un framework est compatible avec la version de PHP qu'elle utilise ou qu'elle veut utiliser.
- **Documentation** : ce critère est simple, mais important. Il s'agit de déterminer si le framework propose une documentation claire est complète. Une bonne documentation diminuera le temps nécessaire pour maîtriser et utiliser un framework. Nous allons définir une échelle de trois valeurs :
    - **1** : inexistante
    - **2** : standard
    - **3** : complète
- **Communauté** : ce critère sert à savoir si derrière un framework, une importante communauté d'utilisateurs existe. L'avantage d'une grande communauté est qu'en cas de problèmes ou de questions, il sera plus facile de trouver une solution adéquate par le biais de forums, de blogs que si la communauté est faible ou inexistante. Ce critère aura donc trois niveaux :
    - **1** : petite
    - **2** : moyenne
    - **3** : grande
- **Tutorial** : il s'agit de savoir s'il existe un ou des tutoriaux pour un framework. Un tutorial permet de facilité la prise en main d'un framework. Nous allons déterminer trois valeurs pour ce critère :
    - **1** : inexistant
    - **2** : standard
    - **3** : complet
- **Tests unitaires automatisés** : ce critère détermine si des tests automatiques sont intégrés à un framework. La grande force de ce critère est de pouvoir tester facilement avec un rapport à la clé une application développée par une entreprise. Cela lui évite de devoir passer du temps supplémentaire pendant la phase de développement afin qu'elle s'assure que tout fonctionne correctement.
- **Structure arborescente des fichiers du projet** : ce critère détermine si un framework propose d'ordonner les fichiers d'un projet selon une structure définie. Une telle structure permet ainsi de retrouver plus facilement où est quoi dans un projet PHP.
- **Gestion des droits** : ce critère détermine s'il est possible de gérer différents « types » d'utilisateurs, c'est à dire de pouvoir définir des droits différents (lecture, modification, suppression…) selon les utilisateurs utilisant une application

- **Modèle d'architecture** : ce critère détermine quel type d'architecture un framework utilise. Il en existe plusieurs types différents : MVC, Event… L'architecture peut influencer l'utilisation d'un framework
- **ORM utilisés :** ce critère détermine quel ORM[5] un framework utilise. Un ORM est une base de données relationnelle sous la forme d'une base de données orientée objet. L'avantage est qu'il n'est pas nécessaire de connaître les commandes SQL pour créer une requête. Comme ORM, il y a DAO[6], Pear, AR,…
- **Templates utilisés :** un template est une structure graphique qui permet de séparer la forme du fond. Ce critère est plus un détail, mais il permet de voir quel format de template le framework utilise. Il y en a des différents tels que Flexy, PHP, Prado,…
- **Taille de la bibliothèque d'outils :** ce critère détermine si un framework propose un nombre conséquent d'outils. Ce critère est intéressant en fonction de la complexité du projet. Plus ce dernier aura des spécificités particulière, plus la probabilité de trouver l'outil adéquat sera importante. Comme pour certains critères, nous déterminerons 3 valeurs :
    - **1** : inexistant
    - **2** : standard
    - **3** : complet

[5] ORM signifie Object-Relationnel Mapping

[6] DAO siginifie Data Access Object

4. Analyse des frameworks selon les critères définis

Nous allons maintenant analyser les différents frameworks présentés au chapitre 2 de ce document. Il est à noter que les informations recueillis grâce à l'analyse sont principalement statistiques. En effet, en raison du temps nécessaire qu'il aurait fallut pour tester chaque framework séparément, suivant le type de critère, cela aurait été long. Le but de ce tableau est de pouvoir définir une sélection des frameworks les plus intéressants avec lesquels nous allons réaliser une application exemple pour pouvoir vraiment voir les différences au niveau pratique.

Voici le tableau de comparaison :

TABLEAU 1
COMPARATIF DES FRAMEWORKS CHOISIS AU CHAPITRE 2

Nom du framework	Zend Framework	Copix	CakePHP	Jaz	Jelix	Prado	Symfony	HOA
Version PHP	5	4 et 5	4 et 5	4 et 5	4 et 5	5	5	5
Documentation 1.inexistant, 2.standard et 3.complet	3	3	3	2	3	2	2	2
Communauté (1 petite, 2 moyenne et 3 grande)	3	2	2	1	2	2	2	1
Tutoriaux 1 inexistant, 2 standard et 3 complet	3	3	3	1	3	3	2	2
Tests unitaires automatis	Oui	Oui	Oui	Non	Oui	Non	Oui	Non
Gestion des droits	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Non
Modèle d'architecture	MVC	MVC	MVC	MVC	MVC	Event	MVC	MVC
ORM	-	DAO	AR	-	DAO	-	DAO	-
Templates	PHP	Smarty ou PHP	PHP	PHP	PHP	Prado	PHP	PHP
Taille de la bibliothèque d'outils (1 petite, 2 moyenne et 3 grande)	3	2	3	1	2	2	3	2

Ce tableau permet d'avoir une première analyse des frameworks sélectionnés au chapitre 2.

On constate que la plupart des frameworks ont des résultats assez identiques. Nous allons donc choisir ceux répondant le mieux à des critères importants comme les tutoriaux, la communauté. Le but étant de garantir le développement de l'application exemple décrite au chapitre 6. Le choix des frameworks intéressants se fait au chapitre suivant.

5. Liste des frameworks recommandés

Après avoir analysé les différents frameworks de ce document, nous allons déterminer lesquels sont les plus intéressants.

5.1 ZendFramework

D'après les critères de comparaison, ce framework est un des meilleurs. Il remplit l'ensemble des critères. La documentation est très détaillée. La communauté est importante. Il existe plusieurs tutoriaux. Il propose des tests unitaires et une gestion des droits.

Il faut noter que l'absence d'ORM n'est pas un problème. ZendFramework utilise grâce à la programmation objet des objets permettant d'interpréter les commandes entre la base de données et l'application.

La seule chose qui pourrait faire qu'une entreprise ne choisisse pas ce framework est peut être le faite qu'il ne fonctionne pas avec PHP 4. Mais, PHP 5 va le remplacer d'ici quelques années.

5.2 Copix

Ce framework fait partie de la liste de ceux que je recommande. En plus d'exister en PHP 4 et 5, il remplit l'ensemble des critères de comparaison.

Le seul bémol est la communauté active derrière ce framework. En effet, elle est dans l'ensemble de moyenne taille. Ce qui peut poser des difficultés si un problème se produit durant le développement et qu'aucune solution ne peut être trouvée. Mais je pense que dans l'ensemble, ce cas est assez extrême.

5.3 Jelix

Ce framework remplit aussi l'ensemble des critères. Ces résultats sont assez identiques à ceux de Copix. Il paraît donc intéressant de le sélectionner. La seule différence se situe au niveau des templates utilisés, leur format est de type jTpl.

Même remarque que pour Copix, la communauté de ce framework est aussi moyenne.

Il aurait été aussi intéressant de choisir d'autres frameworks. Le choix de 3 est un peu limité, mais cela permettra déjà d'avoir un aperçu des différences possibles. J'aurais encore choisi CakePHP, mais le manque temps ne m'a permis de le tester suffisamment pour le faire paraître dans ce travail. Je tiens quand même à signaler qu'il a aussi un potentiel intéressant.

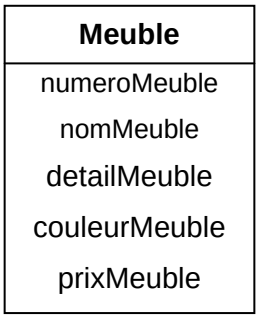
6. Présentation de l'application prototype

Maintenant que nous avons comparé différents frameworks PHP de façon théorique, nous allons maintenant les comparer par le biais d'une application. Cette application sera réalisée avec chaque framework recommandé (chapitre 5). Nous allons définir les spécifications de l'application. Ce qui sera intéressant, ce ne sera pas l'application elle-même, mais les différences entre chaque framework. Voici donc le cahier des charges de notre application standard :

L'entreprise Meublyx est dans la vente de meubles. Il s'agit d'une petite entreprise familiale. Elle désire pouvoir mettre en place sur son Intranet une application permettant de mettre à jour son catalogue de meubles. Chaque collaborateur doit pouvoir ajouter/modifier/supprimer un meuble au catalogue. L'application sera réalisée avec chaque framework PHP choisi au chapitre précédent. La base de données sera identique dans tous les cas. A l'avenir, le catalogue sera accessible aux clients via Internet.

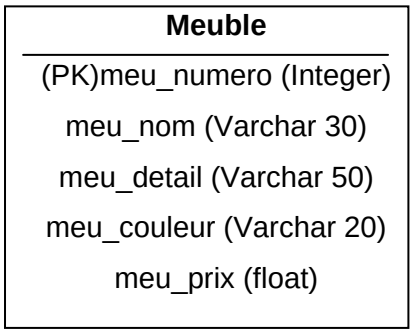
Nous allons commencer par modéliser la base de données. Le modèle ne sera pas trop compliqué. Voici le MCD (Modèle Conceptuel de Données) de cette base de données :

MCD



Voici maintenant le MLD de cette base de données :

MLD



On peut voir ainsi qu'un meuble a un nom. On trouve le détail du meuble, sa couleur et son prix. Il est vrai que pour les habitués des bases de données, cette table est simple et le modèle général est vide. Mais comme annoncée en entrée de ce chapitre, ce qui compte n'est pas la complexité de l'application, mais les différences entre les frameworks.

7. Analyse des frameworks utilisés

Après avoir réalisé la même application avec différents frameworks, il est temps d'analyser les résultats obtenus. Il est à noter que le développement à pris plus ou moins de temps selon le framework. Certains étant particulièrement sensibles à la configuration du serveur ou encore aux extensions PHP installés.

Chaque application a été réalisée sur le même ordinateur. Chaque application a été testée en local avec l'application Wamp Server[7].

7.1 ZendFramework

7.1.1 Avant de commencer le développement de l'application

L'installation de ZendFramework est simple, elle ne nécessite que de dézipper[8] le fichier représentant le framework.

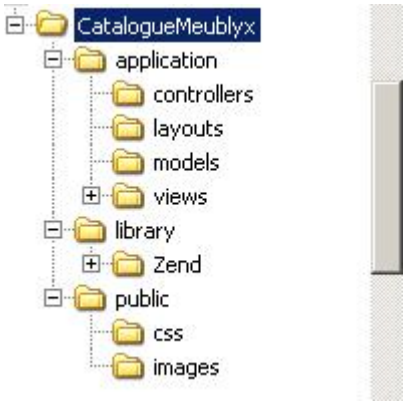
La prise en main de ce framework n'est pas forcément facile. Il est important de vérifier la bonne configuration du serveur (Apache dans mon cas). En effet, le fichier httpd.conf qui contient la configuration du serveur doit avoir par exemple les commande « AllowOverride » à la valeur « All » et non « None » comme c'est le cas par défaut. De plus, votre installation d'Apache doit avoir le module « mod_rewrite » ou « rewrite_module » installé et configuré.

Voilà pour les problèmes liés au serveur. La création d'une application peut commencer. Un point important est la structure des répertoires. Pour que notre application fonctionne, il est important de respecter la structure suivante :

[7] WAMP est une application utilisé pour développer des applications de type PHP utilisant une interface client-serveur.

[8] Dézipper est tiré du logiciel Winzip. Il est utilisé pour compresser et décompresser des données.

FIGURE 3
STRUCTURE DES RÉPERTOIRES D'UN PROJET AVEC ZENDFRAMEWORK



La structure des Répertoires est expliquée dans la suite de ce document

Pour faciliter la création de cette structure, il existe un outil que propose ZendFramework. Il s'agit de « ZendFramework QuickStart », téléchargeable sous http://www.framework.zend.com/wiki/display/ZFDEV/Official+ZF+QuickStart. Cet outil permet de générer directement cette structure sans devoir la créer manuellement. L'avantage réside aussi dans le fait que des fichiers nécessaires pour le bon fonctionnement de notre application sont déjà créés. Bien sûr, il s'agit de fichiers par défaut, il faudra les modifier afin de pouvoir réaliser une application.

7.1.2 Les fichiers qui composent l'application

Nous allons maintenant voir les différents fichiers se trouvant dans notre application. Nous allons les analyser par répertoire. Nous allons donc remonter l'arborescence des répertoires en partant du répertoire de base. Si vous avez de la peine à suivre, référencez-vous à la figure 3 (en haut de page).

7.1.2.1 CatalogueMeublyx

Ce répertoire représente notre projet. Il porte du reste logiquement le nom du projet. Il contient trois répertoires : application, library et public. Il contient deux fichiers importants :

- **Index.php** effectue différentes étapes nécessaires pour lancer correctement l'application. Il va commencer par charger l'ensemble des classes de ZendFramework. Il va ensuite charger la configuration de l'application via un fichier « Config.ini »(ce dernier se trouve dans le répertoire « application »). Il va ensuite se connecter à la base de

données. Et enfin, il va charger le *controller* de notre application responsable d'afficher la première page de notre application.

- **.htaccess** est un fichier qui permet de communiquer au serveur de charger le fichier *Index.php* que nous avons vu précédemment.

7.1.2.2 Application

Ce répertoire contient l'ensemble des fichiers représentant l'architecture de notre application. C'est dans ce répertoire que se trouve les fichiers représentant le cœur même de votre application.

Ce répertoire contient un fichier :

- **Config.ini** est un fichier de configuration. Il contient les différents éléments permettant de se connecter à la base de données. Ce fichier peut contenir d'autres informations liées à la configuration de notre projet.

7.1.2.3 Controllers

Ce répertoire contient un fichier. Ce fichier est le plus important de tous pour assurer le bon fonctionnement d'une application. Il est peu le cœur de notre application :

- **IndexController.php** est un fichier qui représente le *controller* de notre application, c'est à dire qu'il va être utilisé chaque fois qu'une action est exécutée par l'utilisateur (afficher la liste de base, afficher la page d'ajout, de modification ou de suppression). Il contient donc l'ensemble des méthodes permettant d'effectuer les différentes actions voulues.

7.1.2.4 Layouts

Ce répertoire contient un fichier :

- **Layout.phtml** est un fichier de type mise en page. Le but de ce fichier est d'utiliser une seule mise en page de type HTML qui sera utilisée par chaque *view* (voir les fichiers du répertoire views plus bas) que nous aurons. Cela permet de standardiser l'ensemble des pages de l'application sans devoir utiliser plusieurs fois les mêmes commandes HTML. Il est bien sûr possible d'avoir plusieurs fichiers dans ce répertoire afin d'avoir plusieurs styles de mise en page différents. Dans notre exemple, nous n'en aurons qu'un.

7.1.2.5 Models

Ce répertoire contient deux fichiers. C'est dans ce répertoire que l'on trouve les éléments qui composent une page PHP avec ce framework. Voici ces fichiers:

- **FormulaireMeubles.php** est un fichier contenant la structure d'un formulaire de saisie. Il sera utilisé lors de l'ajout ou la modification d'un meuble. Il contient l'ensemble des champs de saisies (nom, détail, couleur,…) et les boutons permettant l'enregistrement ou l'annulation. Pour chaque élément de ce formulaire, il est possible de déterminer certaines règles : nom, obligatoire, type (texte, boutons,…), non-vide,…
- **ListeMeubles.php** est un fichier permettant de définir les tables de la base de données. Dans notre cas, nous n'avons besoin que de la table « Meuble ».

7.1.2.6 Views

Ce répertoire contient deux sous-répertoires que nous allons analyser. Il représente la partie affichage de l'application.

7.1.2.7 Helpers

Ce répertoire contient un fichier appelé « helper ». Le but d'un helper est d'assister la *view* pour afficher une page :

- **BaseUrl.php** est un donc un helper. Le but étant de récupérer une instance du *controller*. Il va retourner une valeur qui va permettre de définir quelle *view* utilisée. Par exemple si le *controller* renvoie la valeur « Ajouter », l'helper va charger la *view* portant le nom « Ajouter ».

7.1.2.8 Script

Ce répertoire contient un sous-répertoire : **index**. Ce dernier contient les *views* de l'application. La *view* sert à afficher une page, soit le code HTML. Cependant, certaines de ces *views* ne font que charger un *model* existant (voir les fichier du répertoire « Models » un peu plus haut dans ce document). Nous allons voir chacune de ces *views* :

- **Ajouter.phmtl** est un fichier qui est utilisé pour afficher le formulaire d'ajout d'un meuble. Le formulaire est celui dont la structure est définie dans le fichier *FormulaireMeubles.php*
- **Modifier.phmtl** est un fichier qui est utilisé pour afficher le formulaire de modification d'un meuble. Comme pour *Ajouter.phtml*, le formulaire est celui défini dans le fichier *FormulaireMeubles.php.*
- **Supprimer.php** est un fichier utilisé pour afficher la page de suppression d'un meuble. A noter que cette *view* est la seule à avoir du code HTML. En effet, pour l'ajout et la modification, l'opération consistait juste à charger le formulaire de saisie avec (modification) ou sans information (ajout). Le formulaire est défini de façon à savoir comment

se comporter. Comme la suppression ne se fait pas via le formulaire de saisie, la page est entièrement définie dans ce fichier.

7.1.2.9 Library

Ce répertoire doit contenir un répertoire nommé « Zend ». Il contient l'ensemble des librairies de ZendFramework. Il suffit simplement de copier le contenu du répertoire « Zend » se trouvant dans le répertoire « library » de ZendFramework. Vous n'avez ensuite plus besoin de modifier quoi que ce soit dans ce répertoire. Une modification d'un des fichiers ou nom des répertoires risque fort d'empêcher l'application de fonctionner.

7.1.2.10 Public

Ce répertoire contient l'ensemble des fichiers qui ne rentre pas dans l'architecture MVC. Il s'agit d'images, de fichiers CSS (contient les informations statiques des pages) et des fichiers de type Java Script. Il s'agit donc d'éléments secondaires au fonctionnement d'une application utilisant ZendFramework.

7.1.3 Application

Après avoir analysé ce framework, il est temps de voir le résultat. Voici donc les différentes pages de notre catalogue de meubles.

7.1.3.1 Liste des meubles

FIGURE 4
PAGE DE LA LISTE DES MEUBLES AVEC ZENDFRAMEWORK

Liste des meubles

Ajouter un nouveau meuble

Nom	Detail	Couleur	Prix(FRS)	Modifier	Supprimer
Chaise longue	Chaise en plastique	blanc	50	Modifier	Supprimer
Fontaine a eau	Fontaine a eau electrique en marbre	rose	275	Modifier	Supprimer
Fauteuil en cuire	Fauteuil en veritable cuire de buffle d'Afrique	cuire	550	Modifier	Supprimer
Canape lit	Canape avec un lit pliable a l'interieur	noir	750	Modifier	Supprimer
Table de cuisine	Table en bois clair		150	Modifier	Supprimer
Tabouret de cuisine	Tabouret en plastique	blanc	25.7	Modifier	Supprimer
Cabine de douche	Cabine en plastique	gris	155	Modifier	Supprimer
Lavabo de salle de bain	Lavabo en pierre des alpages	gris	1120	Modifier	Supprimer
Lit 2 places	Lit 2 places en bois	brun	570	Modifier	Supprimer
Table de nuit	Table en aluminium avec 3 tiroires	gris	75	Modifier	Supprimer
Lit pour enfant	lit pour enfant de moins de 3 ans en plastique	noir	150	Modifier	Supprimer
Lampe de bureau	Lampe halogène de bureau	gris	30	Modifier	Supprimer
Etagere de bureau	Etagere a 6 rayons 2x0.5x05 mètre	rose	75.5	Modifier	Supprimer

Voici le résultat visuel de la liste des meubles avec ZendFramework

Le résultat est simple. On retrouve la liste de nos meubles avec leurs informations (détail, couleur, prix,..). Pour ajouter un meuble, il suffit de cliquer sur le lien en dessus du tableau « Ajouter un nouveau meuble ».

Pour modifier ou supprimer un article, il suffit de cliquer sur « Modifier » ou « Supprimer » situé sur la même ligne que le meuble voulu.

7.1.3.2 Ajouter/Modifier un meuble

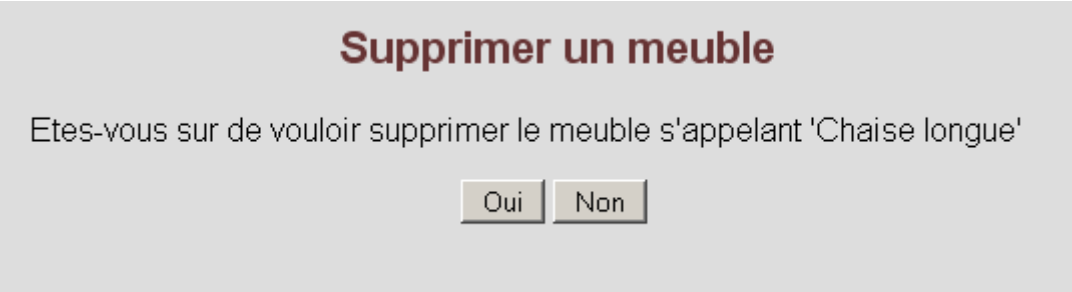
FIGURE 5
PAGE D'AJOUT/MODIFICATION D'UN MEUBLE AVEC ZENDFRAMEWORK

Voici le résultat visuel de la page d'ajout/modification d'un meuble avec ZendFramework

La page d'ajout ou de modification est identique. Bien sûr, lorsque l'on veut ajouter un meuble les champs de saisies sont vides. Il suffit ensuite de cliquer sur le bouton « Enregistrer » pour valider l'opération ou « Annuler » pour revenir à la liste.

7.1.3.3 Supprimer un meuble

FIGURE 6
PAGE DE SUPPRESSION D'UN MEUBLE AVEC ZENDFRAMEWORK



Voici le résultat visuel de la page de suppression d'un meuble avec ZendFramework

Cette page affiche simplement une phrase avec le nom de l'article concerné par la suppression entre ' '. Ensuite l'utilisateur choisit « Oui » pour valider la suppression ou « Non » pour annuler.

7.1.4 Résultat du framework

ZendFramework est un framework intéressant. Sa bibliothèque d'outils est grande et variée. Cependant, lorsqu'on commence à programmer, il faut être très attentif à ce que l'on fait. En effet, la structure de ce framework est exigeante. Aucun écart n'est permis. De plus, le nommage des fichiers ne se fait pas n'importe comment. Il est là encore important de respecter les standards. Au bout d'un moment, on commence à comprendre comment appeler telle librairie et comment l'utiliser. Cependant il est clair qu'il faut avoir travaillé longtemps avec ce framework pour pouvoir se passer de la documentation fournie par Zend. De plus, la documentation officielle explique le contenu de chaque classe et son fonctionnement. Si vous n'avez jamais utilisé de framework, il est difficile de pouvoir réaliser quelque chose simplement avec la documentation officielle. Le conseil que je donne est d'utiliser un tutorial pour pouvoir saisir le fonctionnement de ce framework.

La structure d'un formulaire est aussi très différente comparé aux autres frameworks. En effet, comme nous le verrons en continuons cette analyse, les formulaires des autres frameworks sont simplement de type HTML. Si l'on prend l'exemple d'un simple champ texte, la syntaxe HTML est de type « <input type="text" name="meu_nom" value=""> », alors qu'avec ZendFramework, ce type de champ est déjà défini dans une des classes de ce framework. Il suffit de créer un objet et d'entrer les paramètres que l'on désire. Voici comment définir un champ de type texte:

```
$meu_nom = new Zend_Form_Element_Text('meu_nom');

        $meu_nom->setLabel('Nom')

        ->setRequired(true)

        ->addFilter('StripTags')

        ->addFilter('StringTrim')

        ->addValidator('NotEmpty');
```

Si l'on traduit la commande, on créé un objet héritant de la classe « Zend_Form_Element_Text ». On définit ensuite les paramètres que l'on désire (« setReqired(true) » signifie par exemple que le champ doit être rempli).

ZendFramework n'utilise pas comme ORM DAO. Il a dans sa librairies des classes spécifiques. Il y a par exemple « Zend_Db_Table ». Tout objet héritant de cette classe est considéré comme une table. Du moment que l'application et la BDD peuvent communiquer entre elles, toutes les opérations faites sur un objet de ce type (rechercher, ajout, modification,…) vont se faire grâce aux méthodes déjà définis dans la classe de ZendFramework.

Pour conclure ZendFramework est très intéressant. Une fois qu'on a un serveur correctement paramétré, les bons modules PHP installés, que l'on respecte l'arborescence et le nommage des fichiers. Il demande néanmoins un peu de temps pour se familiariser avec sa syntaxe. Nous retiendrons donc que ce framework est un outil complet, même si sa prise en main peut prendre un peu de temps. Il n'est donc pas conseillé aux entreprises désirant faire un petit projet en peu de temps, du faite de son apprentissage, mais aux entreprises désirant réaliser un projet important.

En résumé, voilà les points positifs et négatifs de ce framework :

TABLEAU 2
TABLEAU DES POINTS POSITIFS ET NÉGATIFS DE ZENDFRAMEWORK

+	-
Librairies	Temps nécessaire pour une bonne utilisation
Tutoriaux	Configuration du serveur
Communauté	Prise en main
Documentation complète	
Architecture MVC	
Orienté objet (programmation)	
Personnalisable	

Ce tableau résume l'analyse de ZendFramework

7.2 Copix

7.2.1 Avant de commencer le développement de l'application

L'installation de Copix est simple, elle ne nécessite que de dézipper le fichier représentant le framework dans un répertoire. Pour que ce framework fonctionne, il y a peu de chose à configurer. Il suffit juste d'avoir installé l'extension PHP « PDO » sur le serveur.

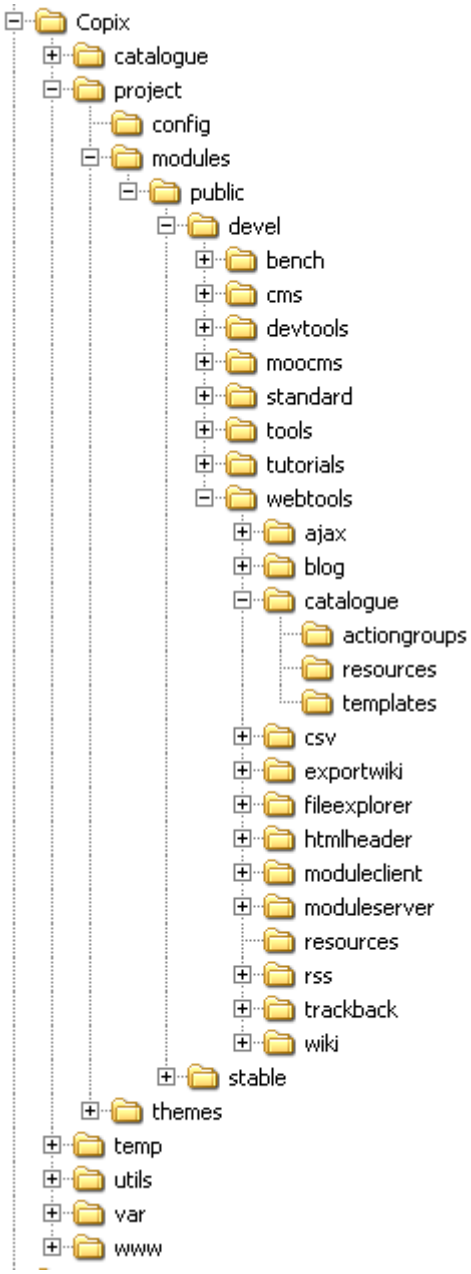
Une fois cette opération effectuée, vous pouvez lancer Copix par le biais de votre serveur. Pour ma part, comme il s'agit d'un serveur local, voici l'adresse de départ : « http://localhost/Copix/www/index.php ».

L'avantage par rapport à ZendFramework, c'est que des menus sont déjà implantés. Différentes options vous sont proposées (Présentation, Documentation,…). Pour bien commencer, la première chose à faire consiste à configurer le lien entre le framework et la base de données MySql. Pour cela, il suffit de cliquer sur « Configurer une base de données supplémentaire ». Remplissez ensuite les différentes informations (nom, driver, nom d'utilisateur,…). Il vous suffit ensuite de tester le lien. Si le lien fonctionne, vous n'avez plus qu'à cliquer sur le bouton « Enregistrer ». Faites ensuite très attention. Lors de la première utilisation, Copix vous affichera un nom d'utilisateur (par défaut admin) ainsi qu'un mot de passe que vous devrez utiliser chaque fois que vous voulez accéder à l'interface d'administration du framework. **Ne l'oubliez pas !** Sinon il est difficile de pouvoir le récupérer. Effacer les fichiers du framework et le dézipper une nouvelle fois ne sert à rien. Le mot de passe est mémorisé dans une table de MySql. L'ensemble de ces tables est créé et stockés lorsque vous créé le 1er lien entre le framework et MySql.

7.2.2 La structure des répertoires

L'arborescence des répertoires se fait automatiquement lors du dézippage du fichier représentant le framework. Voici la structure des répertoires :

FIGURE 7
STRUCTURE DES RÉPERTOIRES D'UN PROJET AVEC COPIX



La structure des Répertoires est expliquée dans la suite de ce document

On voit qu'il y a beaucoup de répertoires. On peut déjà voir une grande différence entre ZendFramework et Copix. Nous n'allons pas tous les analyser en détails, nous allons nous intéresser aux répertoires représentant l'application.

Pour créer un projet, il suffit de créer un répertoire dans l'un des deux répertoires suivant (en partant depuis le répertoire de base de Copix) :

« Copix/project/modules/public/stable/standard » ou
« Copix/project/modules/publis/devel/webtools ».

Comme on peut le voir sur l'image de la structure, c'est la deuxième possibilité que j'ai choisie.

Même si les termes ne sont pas *Models*, *Views* ou encore *Controllers*, on n'en est pas loin. Le répertoire « actiongroups » représente les *Controllers*. Le répertoire « templates » représente les *Views* et le répertoire « *Resources* » représente les Models.

7.2.3 Les fichiers qui composent l'application

Nous allons maintenant voir les différents fichiers se trouvant dans notre application. Nous allons les analyser par répertoire. Nous allons donc remonter l'arborescence des répertoires en partant du répertoire de base de notre projet.

7.2.3.1 Catalogue

Comme nous venons de le voir, ce répertoire est celui représentant notre projet. On trouve un seul fichier à l'intérieur :

- **module.xml** est un fichier de type description. C'est grâce à ce fichier que Copix identifie le projet. Il contient le nom et la description du projet

7.2.3.2 actionGroups

C'est dans ce répertoire que se trouveront tous les *controllers*. Le rôle des *controllers* est déjà expliqué lors de l'analyse de ZendFramework. On trouve un fichier :

- **meubles.actiongroup.php** est donc le *controller* de notre application. La 1ère chose importante est le nom du fichier. Il reflète le fonctionnement d'un *controller* sous Copix. On définit d'abord le nom que l'on veut (meubles). Il représente le nom d'un module de notre application. Rappelons qu'un projet peut être divisé en plusieurs modules. La 2ème partie du nom du fichier est « actiongroup ». C'est grâce à ce terme que Copix va comprendre que ce fichier est un *controller* et comprendre comment appeler ses différentes méthodes. Lorsque par exemple, l'utilisateur clique sur le bouton « ajouter », la méthode choisie va être exécutée. La bonne *view* va être chargée. On retrouve dans ce fichier l'ensemble des méthodes permettant l'affichage, l'ajout, la modification ou la suppression. De plus, pour

chaque méthode la *view* nécessaire sera chargée et les opérations sur la base de données seront exécutées.

7.2.3.3 resources

Ce répertoire représente les différents models d'une application. Dans notre exemple, je n'ai pas utilisé de models. Les fichiers que l'ont retrouve on comme extension « .properties ». Il s'agit des différents paramètres telles que la police de caractère, les couleurs de la page, bref toutes les structures d'affichage que l'on désire utilisé par défaut. Dans le cas de ZendFramework, on utilisait Models pour définir un formulaire par défaut (utilisé pour la modification et l'ajout). Avec Copix, il serait tout à fait possible d'en définir un. Dans la réalisation actuelle, chaque formulaire est défini au niveau de la *view*. Ce qui veut dire qu'en cas de changement de mise en page, il faudra le répercuter sur chaque *view*. On voit donc l'intérêt d'utiliser un *model* par défaut.

7.2.3.4 templates

Ce répertoire contient donc la partie « affichage » de notre application. Pour chaque action, nous aurons donc un fichier dans ce répertoire. Voici ces différents fichiers :

- **meubles.ajout.tpl** est un fichier qui sert à afficher le formulaire permettant l'ajout d'un meuble. De nouveau, analysons le nom de ce fichier. Il commence par « meubles ». On retrouve, comme pour le *controller*, le nom du module dans lequel ce template sera utilisé. La seconde partie du nom du fichier est « ajout ». Là, aucune contrainte. Cette seconde partie du nom est libre. Ce fichier sert donc à afficher un formulaire permettant l'ajout d'un meuble. Le code est celui d'un formulaire normal. La seule différence est dans l'action du formulaire. Au lieu de lui indiquer une page PHP où on enverra les informations saisies par l'utilisateur, on va lui indiquer le *controller*. La syntaxe est sous cette forme« catalogue|meubles|enregistrer ». On voit bien la structure qu'utilise ce framework. Lorsque l'on valide le formulaire d'ajout, on sera redirigé dans le projet « catalogue », puis dans le module « meubles ». Ensuite Copix va appeler le *controller* par défaut (celui se trouvant dans actiongroups et portant le nom du module). Il va ensuite appeler la méthode « enregistrer » se trouvant dans le controller. Cette dernière va enregistrer le nouvel enregistrement et appeler la méthode affichant la liste des meubles.
- **meubles.fiche.tpl** est un fichier qui sert à afficher l'ensemble des informations d'un meuble. L'utilisateur peut ensuite les modifier et les valider ou non. De nouveau, le nom du fichier est toujours en deux parties « meubles » pour indiquer le module » et « fiche » pour indiquer le template. Il est à noter que l'on aurait pu avoir un seul *model* pour représenter ce formulaire et celui d'ajout, ce qui n'est pas le cas ici. Comme pour le formulaire d'ajout, lorsque l'utilisateur veut valider ces enregistrements, le formulaire va appeler le *controller*. La méthode

utilisée est aussi « enregistrer ». Cette dernière analyse si une clé primaire a été définie au meuble. Si c'est le cas, la modification est enregistrée.

- **meubles.liste.tpl** est un fichier utilisé pour afficher la liste des meubles de notre catalogue. De nouveau son nom est en deux parties : module (meubles) et liste (nom du template). Depuis ce formulaire, il est donc possible d'ajouter, modifier ou supprimer un meuble selon les besoins de l'utilisateur
- **meubles.suppression.tpl** est un fichier affichant le nom du meuble ainsi que deux boutons (pour supprimer le meuble ou annuler l'action). Lorsqu'une suppression est demandée, c'est le *controller* qui se charge de l'exécution.

7.2.4 Application

Voici ce que donne visuellement l'application du catalogue Meublyx.

7.2.4.1 Liste des meubles

FIGURE 8
PAGE DE LA LISTE DES MEUBLES AVEC COPIX

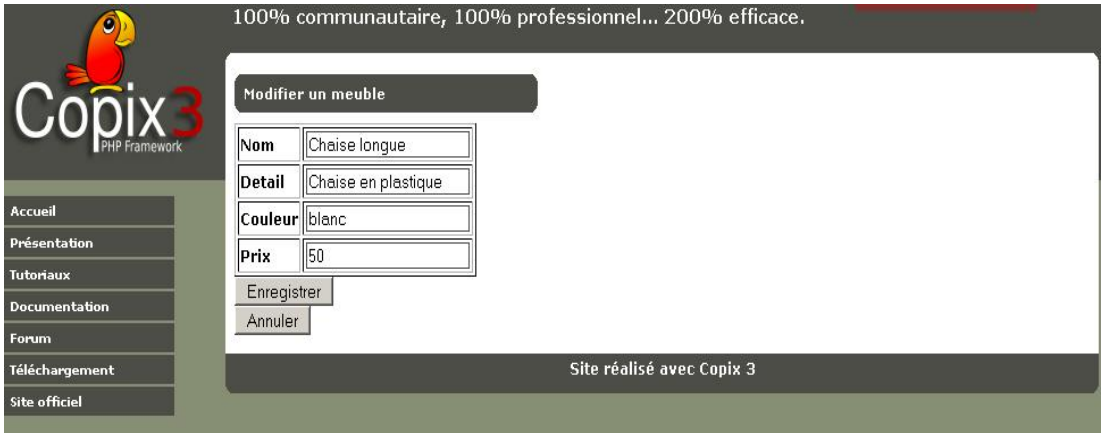
Voici le résultat visuel de la liste des meubles avec Copix

La première chose que l'on remarque, c'est l'environnement graphique que propose Copix, chose complètement absente avec ZendFramework. Même si cela aide pour la prise en main de l'outil, on se retrouve avec une page avec les logos de Copix. Enfin on retrouve quand même la liste de nos meubles. La différence (à part l'environnement graphique de Copix) que l'on voit est que l'on a des boutons pour les actions (ajout,

modification et suppression) à la place des liens HTML comme dans ZendFramework. Cette différence est minime, car avec Copix, on aurait pu avoir des liens à la place des boutons.

7.2.4.2 Ajouter/Modifier un meuble

FIGURE 9
PAGE D'AJOUT/MODIFICATION D'UN MEUBLE AVEC COPIX

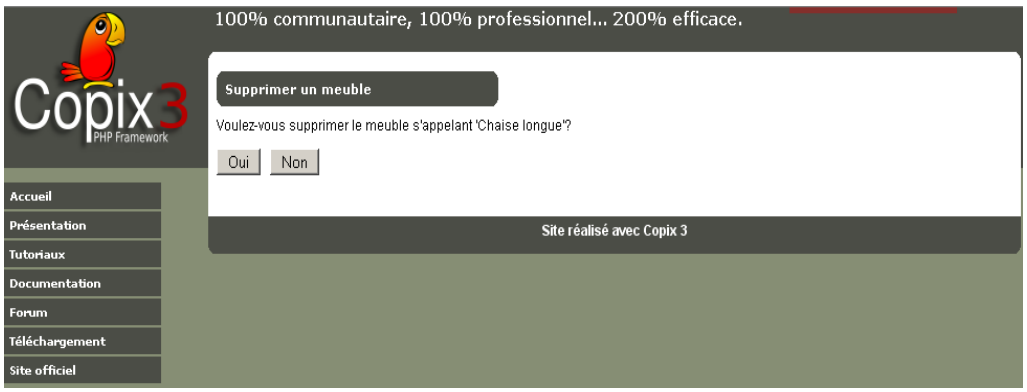


Voici le résultat visuel de la page d'ajout/modification d'un meuble avec Copix

Aucune grande différence avec ZendFramework, si ce n'est toujours la présence des logos Copix et de son environnement graphique. Le formulaire est le même pour l'ajout.

7.2.4.3 Supprimer un meuble

FIGURE 10
PAGE DE SUPPRESSION D'UN MEUBLE AVEC ZENDFRAMEWORK



Voici le résultat visuel de la page de suppression d'un meuble avec Copix

Comme pour l'ensemble des pages, aucune différence flagrante avec ZendFramework, sauf ce qui a été dit précédemment (logo + environnement graphique)

7.2.5 Résultat du framework

Copix propose une prise en main plus rapide que ZendFramework. Le faite d'arriver sur une interface graphique lorsqu'on lance ce framework par défaut permet rapidement de se familiariser. Suivant ce qu'on désire faire, ce menu est très intéressant :

FIGURE 11
MENU DE COPIX

Voici l'ensemble des éléments du menu de Copix

Il permet d'avoir un lien direct avec le site et le forum officiel, ainsi que différents éléments permettant une prise en main direct.

Les tutoriaux proposés sont simples et facilement compréhensible. La documentation fournie est intéressante. Quelque soit l'opération que l'on désire trouver, la recherche est facile. De plus, le forum de Copix contient une grande quantité d'informations utiles à la conception. Si vous rencontrez un problème, et que la documentation ne vous fournie pas la solution, le forum peut être un outil intéressant.

Durant la réalisation de l'application d'exemple, je n'ai cependant pas rencontré de problème bloquant le développement. Il s'agissait plus d'erreurs de syntaxe que des erreurs empêchant le fonctionnement de l'application.

Cependant, il faut faire toujours attention aux nommages des fichiers. Devant chaque fichier, on doit d'abord trouver le nom du module et ensuite le nom que l'on désire (module.exemple).

La grande différence entre Copix et ZendFramework, se situe par exemple dans un formulaire. Avec Copix, on est à la vieille école. C'est à dire qu'il faut créer un formulaire de type HTML avec les champs désirés. ZendFramework est là tout de suite beaucoup plus orienté objet, puisque comme nous l'avons vu dans l'analyse de ZendFramework; créer un formulaire ou un autre champ de type HTML (exemple textbox) se fait en créant un objet. Cet objet est ensuite de type formulaire, champ texte,...Il suffit ensuite de passer en paramètre les valeurs voulues et le formulaire et ses champs sont générés. Copix est moins orienté PHP 5. Et l'on comprend ainsi pourquoi il est aussi compatible avec PHP 4.

L'utilisation des DAO facilite le dialogue entre l'application et la base de données. Effectuer une opération se fait facilement.

Ce que l'on peut reprocher à Copix, c'est le contenu de sa librairie. Bien sûr, elle n'est pas vide, on peut réaliser des choses intéressantes. On peut par exemple définir des groupes d'utilisateurs, générer des fichiers de « log » indiquant ce qui s'est passé sur le fonctionnement de l'application,… Il propose l'ensemble de ces outils sous la forme de modules. Ces modules sont accessible et activables via la console d'administration. Mais, les outils proposés de base ne sont pas aussi important comparée à ZendFramework par exemple. On peut bien sûr charger de nouveaux modules proposant des outils supplémentaires, mais avec ZendFramework aussi.

De plus, même si au début, avoir une interface graphique permettant de se familiariser rapidement à Copix est intéressante, elle l'est moins lorsque l'on veut réaliser un produit professionnel sérieux. Je doute fort qu'une entreprise développant une importante application PHP apprécie de voir sur toutes les pages de son application la petite phrase « site réalisé avec Copix 3 » ainsi que les logos de Copix. Je tiens à souligner que ce dernier argument est peu être infondé. D'après la documentation, l'environnement graphique peut être changé. En tout cas, ce n'est pas aussi simple qu'avec ZendFramework ou Jelix.

Pour conclure, je dirais que Copix a une prise en main facilité par l'accès directe à la documentation et aux tutoriaux. Il offre aussi une compatibilité avec PHP 4, ce qui peut être un atout pour certaines entreprises l'utilisant encore. Mais une entreprise travaillant avec PHP 5 sera plus intéressée par un framework utilisant plus la programmation orientée objet. Si j'étais sûr que l'environnement graphique et les logos

de Copix ne figurent pas forcément sur un projet, je dirais que Copix est un framework intéressant. Mais, comme je n'ai pas cette certitude, je ne peux que dire que Copix est intéressant pour un projet en interne à une entreprise. Pour un projet externe, il faudrait avoir la certitude que cette interface graphique est changeable.

En résumé, voilà les points positifs et négatifs :

TABLEAU 3
TABLEAU DES POINTS POSITIFS ET NÉGATIFS DE COPIX

+	-
Prise en main	Librairies
Menu graphique	Présence des logos + style graphique
Documentation	Structure de Copix
Tutoriaux	Convention de nommage des fichiers
Grande communauté	Pas très orienté objet (programmation)
Architecture MVC	Formulaire HTML
DAO (facilite les requêtes)	

Ce tableau résume l'analyse de Copix

7.3 Jelix

7.3.1 Avant de commencer le développement de l'application

L'installation de base du framework est aussi simple que pour les précédents. Il suffit de dézipper le fichier représentant le framework dans le répertoire de base de votre serveur (www en local).

Une fois dézippé, il faut exécuter différentes opérations afin de créer le projet et l'arborescence de l'application. Pour y parvenir, il faut utiliser des lignes de commande sous MS-DOS (Windows) ou sous Linux. A noter que pour ma part je n'ai testé ce framework que sous Windows. Il faut donc créer le projet. Pour cela il faut exécuter un fichier nommé « jelix.php » qui se trouve dans le répertoire *lib/ jelix-scripts* du framework. La commande prend alors la forme :

jelix.php --nomProjet createapp

J’ai pour ma part rencontré un petit problème. Ce script ne peut pas s’exécuter comme cela en mode ligne de commande. Le fichier permettant d’interpréter le fichier « jelix.php » en entrant les paramètres suivant : nomProjet (le nom du projet) createapp (commande) est le fichier *php.exe* (dans l’exemple le fichier php.exe se trouve sous *c:\wamp\bin\php\php5.2.6\php.exe)*.

Une fois donc dans le répertoire *jelix-scripts* (Exemple « *cd c:\wamp\www\jelix\lib\jelix-script* ») en mode ligne de commande, l’instruction complète pour pouvoir générer le projet est :

c:\wamp\bin\php\php5.2.6\php.exe jelix.php --nomProjet createapp

A noter que notre exemple *nomProjet* est remplacé par *Catalogue*. Le projet est donc créé.

Comme dans l’ensemble des frameworks que nous avons analysés, un projet est constitué de un ou plusieurs module(s). Mais dans notre cas il n’y en aura qu’un. L’instruction pour pouvoir créer un module est :

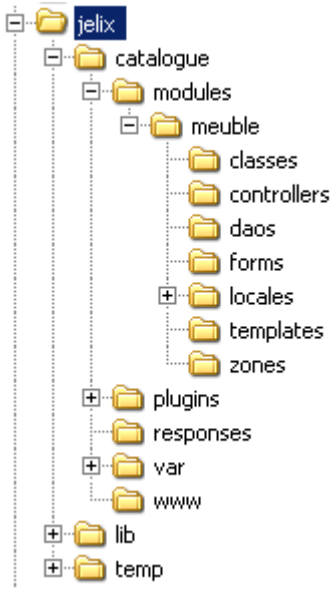
c:\wamp\bin\php\php5.2.6\php.exe jelix.php --nomProjet createmodule nomModule

Dans notre exemple *nomProjet* sera remplacé par *Catalogue* et *nomModule* par *Meubles.*

7.3.2 La structure des répertoires

Une fois que les différentes commandes exécutées au point précédent sont faites, voici la structure arborescente de notre projet :

FIGURE 12
STRUCTURE DES RÉPERTOIRES D'UN PROJET AVEC JELIX



La structure des Répertoires est expliquée dans la suite de ce document

La structure est ainsi faite. Le répertoire racine est « jelix ».Il s'agit du dossier dans lequel l le framework a été dézippé. Dans ce dossier on trouve le répertoire « lib ». Ce dernier contient les librairies de ce framework. Lorsque les commandes du point précédent ont été exécutées, le répertoire « catalogue » et le répertoire « meuble » ont été créé ainsi que l'ensemble de leurs sous-répertoires. Nous allons nous intéresser au contenu du répertoire « meuble » (le module de notre projet).

Comme avec les frameworks précédemment utilisés pour développer notre application, on retrouve la structure MVC : Models = Forms, Views = templates et Controllers = controllers.

7.3.3 Les fichiers qui composent l'application

Nous allons maintenant voir les différents fichiers se trouvant au dans notre application (le répertoire « meuble » dans « modules »). Nous allons les analyser par répertoire. Nous allons donc remonter l'arborescence des répertoires en partant du répertoire de base de notre projet.

7.3.3.1 catalogue

Ce répertoire représente notre projet, il porte du reste le nom du projet. Il contient cinq répertoires : modules, plugins, responses, var et www. Ils sont générés automatiquement. Par la suite nous ne verrons pas en détails ces répertoires, mais en gros nous verrons leur utilité générale. Il contient trois fichiers importants:

- **application.init.php** est un fichier utilisé pour charger les différents éléments nécessaires à l'exécution de notre application
- **htaccess** est un fichier créé automatiquement. Il peut être configuré afin de lancer l'application lorsque l'on exécute le projet.
- **module.xml** est un fichier de type description. Il décrit l'identité du module. Ce fichier est généré automatiquement. Il est bien sûr possible de l'éditer.

7.3.3.2 modules

Ce répertoire sert à stocker les différents modules d'un projet. Dans notre cas il n'y aura qu'un module (meuble)

7.3.3.3 plugins

Ce répertoire sert à stocker les différents plugins[9] que vous pouvez utiliser pour votre application. Dans notre cas ce répertoire est vide.

7.3.3.4 responses

Ce répertoire est utilisé par le framework. Il est généré automatiquement. A noter que le fichier s'y trouvant *myHtmlResponse.class.php* est aussi généré automatiquement à la création du projet.

7.3.3.5 var

Ce répertoire contient quatre sous-répertoires :

- **config** : contient les fichiers de configuration de l'application
- **log** : contient les éventuels fichiers de journalisations (si des tests automatiques sont générés par exemple)
- **themes** : contient les différents thèmes possibles dans votre application

[9] Un plugin sert à ajouter des fonctionnalités à un programme (au framework dans notre cas)

- **overload** : contiendra les différents fichiers redéfinis (modififiés) par le concepteur, issus de modules tiers.

7.3.3.6 meuble

Comme nous venons de le voir, ce répertoire représente un module. Dans notre cas, c'est donc le seul module. On trouve un seul fichier à l'intérieur :

- **module.xml** est un fichier de type description. Il décrit l'identité du module. Ce fichier est généré automatiquement. Il est bien sûr possible de l'éditer.

7.3.3.7 classes

C'est dans ce répertoire que l'on peut enregistrer les classes supplémentaires que l'on désire intégrer. Dans notre exemple ce répertoire est vide.

7.3.3.8 controllers

Dans ce répertoire, comme son nom l'indique, on retrouve notre *controller* qui va gérer l'ensemble des actions de notre application. On trouve donc un fichier :

- **default.classic.php** est donc notre *controller*. On retrouve un nommage à deux niveaux. « default » est le nom de notre controller, il est bien entendu possible de changer ce nom. « classic » permet à Jelix de comprendre qu'il s'agit là de notre *controller*. Comme nous l'avons vu dans les autres frameworks, l'ensemble des méthodes d'affichage, ajout, modification et suppression se trouve dans ce fichier. Sa tâche sera donc d'exécuter la méthode voulue et d'afficher le template (view) adéquat.

7.3.3.9 dao

Jelix propose l'utilisation d'un design pattern appelé DAO. Il servira pour l'exécution des différentes requêtes nécessaires à l'affichage, ajout, modification ou suppression d'informations dans la base de données. Ce design pattern se base sur deux types d'objets : record (représente un enregistrement) et factory (liste d'enregistrements (liste de records).) Voici le fichier de ce répertoire :

- **meuble.dao.xml** est donc un fichier, comme expliqué servant pour exécuter des requêtes SQL sans devoir les écrire « en dur ». Jelix propose de générer automatiquement ce fichier. Il suffit d'utiliser d'être en mode ligne de commande et de taper (dans notre exemple) :

  c:\wamp\bin\php\php5.2.6\php.exe jelix.php createdao meuble meuble meuble

Si l'on analyse un peu cette commande, le premier *meuble* signifie le nom du module, le second *meuble* représente le nom du fichier dao généré et le dernier *meuble* représente le nom de la table dans la base de donnés *catalogueMeublyx.* Une fois la commande exécuté le fichier est généré. Il n'est nullement nécessaire de le modifier

7.3.3.10 forms

Ce répertoire représente, comme expliqué précédemment notre *models*. Cependant, dans notre exemple, comme avec Copix, il n'y a aucun m*odel* défini.

7.3.3.11 locals

Ce répertoire est utilisé si l'application permet de choisir une langue parmi plusieurs (exemple Français, Anglais). Dans notre exemple, ce répertoire sera vide.

7.3.3.12 templates

Ce répertoire contient donc la partie affichage de notre application. Pour chaque action, nous aurons donc un fichier dans ce répertoire. Voici ces différents fichiers :

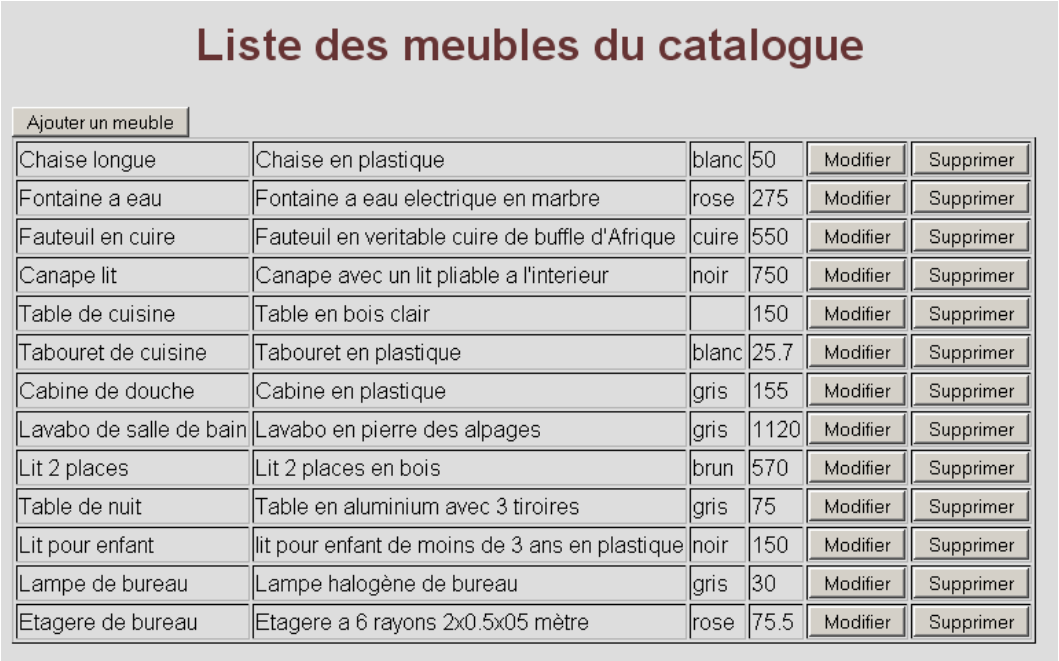
- **ajoutmeuble.tpl** est un fichier qui sert à afficher le formulaire permettant l'ajout d'un meuble. Son rôle et sa structure sont identiques à ceux de Copix. La seule différence est dans l'action du formulaire. La syntaxe est un peu différente : « {formurl 'meuble~default:createsave'}». On voit bien la structure qu'utilise ce framework. Lorsque l'on valide le formulaire d'ajout, on sera redirigé dans le module « meuble » et le controller « default » sera appelé. Ensuite, on appelle la méthode « createsave » de ce formulaire. On peut noter une ressemblance entre un template Copix et un de Jelix.
- **fichemeuble.tpl** est un fichier qui sert à afficher l'ensemble des informations d'un meuble. On retrouve des mêmes similitudes entre le template *meubles.fiche.tpl* de Copix et ce dernier. La nuance est comme pour *ajoutmeuble.tpl*, au niveau de l'action du formulaire
- **listemeuble.tpl** est un fichier utilisé pour afficher la liste des meubles de notre catalogue. Mêmes remarques que pour le fichier *fichemeuble.tpl*
- **suppressionmeuble.tpl** est un fichier affichant le nom du meuble ainsi que deux boutons (pour supprimer le meuble ou annuler l'action). Lorsqu'une suppression est demandée, c'est le *controller* qui se charge de l'exécution. Mêmes remarques que fichemeuble.tpl

7.3.4 Application

Voici ce que donne visuellement l'application du catalogue Meublyx.

7.3.4.1 *Liste des meubles*

FIGURE 13
PAGE DE LA LISTE DES MEUBLES AVEC JELIX



Chaise longue	Chaise en plastique	blanc	50	Modifier	Supprimer
Fontaine a eau	Fontaine a eau electrique en marbre	rose	275	Modifier	Supprimer
Fauteuil en cuire	Fauteuil en veritable cuire de buffle d'Afrique	cuire	550	Modifier	Supprimer
Canape lit	Canape avec un lit pliable a l'interieur	noir	750	Modifier	Supprimer
Table de cuisine	Table en bois clair		150	Modifier	Supprimer
Tabouret de cuisine	Tabouret en plastique	blanc	25.7	Modifier	Supprimer
Cabine de douche	Cabine en plastique	gris	155	Modifier	Supprimer
Lavabo de salle de bain	Lavabo en pierre des alpages	gris	1120	Modifier	Supprimer
Lit 2 places	Lit 2 places en bois	brun	570	Modifier	Supprimer
Table de nuit	Table en aluminium avec 3 tiroires	gris	75	Modifier	Supprimer
Lit pour enfant	lit pour enfant de moins de 3 ans en plastique	noir	150	Modifier	Supprimer
Lampe de bureau	Lampe halogène de bureau	gris	30	Modifier	Supprimer
Etagere de bureau	Etagere a 6 rayons 2x0.5x05 mètre	rose	75.5	Modifier	Supprimer

Voici le résultat visuel de la liste des meubles avec Jelix

Cette page est presque identique à celle de ZendFramework . La seule différence est que comme pour Copix, les liens HTML ont été remplacés par des boutons (ajouter, modifier et supprimer), mais il s'agit là d'un détail. Sinon le rendu est identique à ZendFramework. On retrouve l'ensemble des meubles du catalogue Meublyx.

7.3.4.2 *Ajouter/Modifier un meuble*

FIGURE 14
PAGE D'AJOUT/MODIFICATION D'UN MEUBLE AVEC JELIX

Voici le résultat visuel de la page d'ajout/suppression d'un meuble avec Jelix

On constate de nouveau une grande similitude entre cette page et celle équivalente dans ZendFramework. Dans l'ensemble, le résultat est assez identique. D'un point de vue programmation, nous sommes dans le même cas que Copix. Le formulaire a été codé en dur en HTML. Il n'y a pas de notion d'objet dans ce framework.

7.3.4.3 Supprimer un meuble

FIGURE 15
PAGE DE SUPPRESSION D'UN MEUBLE AVEC JELIX

Voici le résultat visuel de la page de suppression d'un meuble avec Jelix

La page de suppression est 100% similaire à celle de ZendFramework. On peut donc définir les mêmes remarques (On retrouve le nom de l'article et le choix de valider la suppression (bouton « Oui » ou « Non »)).

7.3.5 Résultat du framework

Jelix est un framework intéressant. Sa prise en main n'est pas trop compliquée. Elle prend cependant un petit peu plus de temps que celle de Copix. De nouveau, les tutoriaux proposés permettent de comprendre le fonctionnement de ce framework. Les principaux problèmes que j'ai rencontrés étaient principalement dus à la création du projet et des fichiers nécessaire.

Comme nous l'avons vu, entre Copix et Jelix, il y a quelques similitudes. On constate que les templates sont identiques. La seule différence se situe lorsque l'on définit l'action d'un formulaire (La page que l'on va afficher selon le bouton sur lequel l'utilisateur clique).

L’autre similitude se situe au niveau de la programmation. Comme Copix, Jelix est compatible avec PHP 4. Ce qui fait que de nouveau, nos formulaires se retrouvent codés en HTML. Aucune notion d’objet formulaire défini dans une classe n’apparaît. Comme pour Copix, cela peut être un avantage. Mais dans le contexte actuelle ou le PHP 5 est en train de s’imposer comme standard, il est donc dommage de n’avoir pas de notion d’héritage de classe.

Cependant, Jelix ne propose pas comme Copix des logos un peu partout, ce qui permet de réaliser une application plus personnalisée.

Dans les points négatifs, comme Copix, la librairie ne possède pas beaucoup d’outils. On retrouve du reste une collection de modules, et non des classes prédéfinis.

L’installation peut aussi être laborieuse si vous n’êtes pas à l’aise avec les lignes de commandes. Il aurait été bien plus simple d’avoir un petit programme demandant le nom du projet et le nom de son (ses) module(s) et qui génère automatiquement la structure des répertoires.

La communauté de ce framework est restreinte. Celle de Copix est plus importante. Pour chercher une solution à un problème, il est parfois difficile de trouver l’information voulue.

En conclusion, l’installation demande de bien suivre les étapes pour garantir la bonne création de la structure d’une application. Le reste de cette conclusion sur l’analyse de ce framework est un peu la même que celle de Copix, à part la prise en main du framework qui est moins rapide. Mais selon la nature de l’application que l’on désire réaliser, les outils à disposition sont peut être un peu faibles au regard de framework comme ZendFramework. Ce framework est donc intéressant pour une entreprise. Cependant, si le projet a besoin de certains modules, il vaut mieux vérifier qu’il existe bien avant de commencer à développer avec ce framework.

En résumé, voilà les points positifs et négatifs :

TABLEAU 4
TABLEAU DES POINTS POSITIFS ET NÉGATIFS DE JELIX

+	-
Prise en main	Configurer un nouveau projet
Documentation	Librairies
Tutoriaux	Petite communauté
DAO (facilite les requêtes)	Convention de nommage des fichiers
Architecture MVC	Pas très orienté objet (programmation)
Personnalisable	Formulaire HTML
	Installation

Ce tableau résume l'analyse de Jelix

8. Conclusion

C'est ainsi que nous arrivons à la conclusion générale de ce travail. Il est dommage de n'avoir qu'une analyse plus détaillée qu'avec les trois frameworks sélectionnés pour réaliser l'application exemple. Pour avoir une analyse plus poussée, il aurait été nécessaire de réaliser l'application exemple avec chaque framework listé au début. Le problème est bien sûr le temps nécessaire pour pouvoir utiliser ces frameworks et créer l'application. Lors de la réalisation de l'application exemple avec chaque framework, il m'a fallut un certain temps. Entre la configuration du serveur, les modules PHP qui doivent être installés ainsi que la structure arborescente des répertoires ; tous cela sont des contraintes qui varient d'un framework à l'autre. C'est du reste dommage que les frameworks ne soit pas tous nés d'un même standard. On constate plus que ces frameworks ont été développé par des sociétés qui ont faites chacune « leur cuisine » dans leur coin. On ne s'étonne pas alors que le point commun de tous ces frameworks ne soit que le langage de programmations de base (PHP), les bibliothèques d'outils (malgré quelques différences suivant la compatibilité avec la version de PHP) et l'architecture générale MVC (sauf dans très peu de cas). Bien sûr, comme nous l'avons vu, le résultat final est assez équivalent visuellement, mais pas du point de vue de la programmation.

Ce qu'on peut en analyser est que la prise en main est le point faible des frameworks. Un temps d'apprentissage est nécessaire avec chaque framework pour comprendre son fonctionnement et pouvoir l'utiliser à sa guise. Pourtant, une fois le temps d'acclimatation fini, on se sent à l'aise, on comprend les mécanismes et on arrive à réaliser quelques choses.

Pour revenir à la grande question de ce document « **quels frameworks s'offrent aux entreprises? »,** nous avons vu que le résultat visuel est assez identique d'un framework à l'autre. Avant de choisir un framework, l'entreprise doit tenir compte de différents paramètres dans un projet. Le choix du framework doit se faire selon la nature de l'application. C'est ce qui modère l'utilisation d'un framework : sa complexité à l'utiliser. En effet, aucune barre d'outils ne permet en quelques clicks de souris de créer une application (du moins avec les frameworks testés). Chaque framework a sa façon de fonctionner. Chacun utilise un peu son « langage » qu'il faut apprendre à

maîtriser. Oui, le « langage » de base est bien du PHP, mais chaque framework a des commandes propre à lui-même.

Il en ressort aussi que lorsque l'on sait en utiliser un, il est un peu plus facile d'apprendre à en utiliser d'autres.

Si une entreprise est intéressée par utiliser un framework pour développer une ou plusieurs application(s), je conseille pour ma part de choisir un seul et unique framework et de ne plus changer en cours de route. Sinon, en cas de changement, il faudra de nouveau compter un laps de temps nécessaire pour pouvoir se familiariser avec le nouveau framework. Il est impossible de pouvoir utiliser un framework sans avoir suivi au préalable un tutorial ou alors en connaissant bien le sujet. Ce qui fait que si une entreprise désire réaliser une petite application, elle mettra moins de temps pour la réaliser sans utiliser un framework qu'avec.

Pour ma part, je pense que les entreprise les plus intéressées par les frameworks sont celles qui produisent une grande ou plusieurs applications PHP. La prise en main passé, un framework permet de gagner du temps.

Ce qui manque aux frameworks utilisés pour réaliser l'application exemple, c'est une interface graphique permettant de générer des formulaires, ou d'autres objets de manières interactives. Ainsi, il ne serait pas nécessaire de devoir écrire manuellement le code, mais il serait généré en partie automatiquement. Après, à l'utilisateur de l'éditer s'il désire effectuer une modification. Si l'on prend un formulaire, en cliquant sur le bouton adéquat, je devrais pouvoir déterminer l'action du formulaire, les champs qui le composent ainsi que les boutons et la partie affichage et mise en page du formulaire.

Avec un système pareil, la prise en main d'un framework serait beaucoup plus intuitive et dynamique, ce qui en ferait tout de suite un outil beaucoup plus intéressant pour n'importe quelle entreprise désirant réaliser une application en PHP.

En conclusion final, le choix d'un framework ou d'un autre pour réaliser une application est un peu égal. Ce qui compte c'est la présence de documentation. Le tutorial est aussi important, puisqu'il va vous permettre de prendre en main le framework. Plus ce dernier est bien fait, plus l'apprentissage nécessaire pou pouvoir utiliser un framework

sera rapide et optimale. Il est clair que l'on peut se passer des tutoriaux si l'on connaît déjà le framework, mais alors à ce moment là le choix ne se fait plus.

9. Bibliographie

9.1 Ouvrages cités dans ce document

Copix Team. Copix, Framework PHP. 05.04.08. http://www.copix.org/index.php/wiki/Presentation (consulté le 20.07.08).

ADULTLAG Association des Développeurs et des Utilisateurs de Logiciels Libres pour les Administrations et les Collectivités Territoriales. Fiche technique : CakePHP. http://www.adullact.org/documents/fiche_cakephp.pdf (21.07.08)

QSOS. Wikipedia l'encyclopédie libre. 13.03.2007. http://fr.wikipedia.org/wiki/QSOS (consulté le 27.07.08).

Emmanuelle Gouleau, Olivier Mansour, Tristan Rivoallan, Vincent Lemaire, Xavier Lacot. Livre Blanc Frameworks PHP pour l'entreprise : Définition, critères de choix et analyses. 37 boulevard des Capucines 75002 Paris : Clever Age, 14.05.08. 38 p. http://www.clever-age.com/veille/publications/developpement-specifique/livre-blanc-frameworks-php-pour-l-entreprise.html (consulté le 28.07.08)

Nicolas Richeton. Livre Blanc Frameworks PHP . Smile Motoristes Internet, 19.09.08. 78 p. http://www.smile.fr/publications/livres-blancs/frameworks-php (consulté le 31.07.08)

9.2 Ouvrages utiles pour apprendre à utiliser un framework

Rob Allen, Guillaume Rossolini. Tutoriel principal Jelix 1.0 Developpez.com club des développeurs, 24.07.08. 22 p. http://g-rossolini.developpez.com/tutoriels/php/zend-framework/debuter/ (consulté le 10.08.08)

Contributers au wiki de jelix.org. Débuter avec Zend Framework 1.5 (approche MVC). Developpez.com club des développeurs, 16.05.08. 23 p. http://g-rossolini.developpez.com/tutoriels/php/zend-framework/debuter/ (consulté le 17.08.08)