h e g
Haute école de gestion de Genève
Geneva School of Business Administration

Les frameworks au coeur des applications web

Mémoire de bachelor réalisé par :

Arielle Moro

Directeur de mémoire :

Peter Daehne, Professeur HES

Genève, le vendredi 27 août 2010,

Haute Ecole de Gestion de Genève (HEG),

Informatique de Gestion

1. Déclaration

Ce mémoire de bachelor est réalisé dans le cadre de l'examen final de la Haute Ecole de Gestion de Genève, en vue de l'obtention du titre de 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 bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de bachelor, du juré et de la HEG.

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

Fait à Genève, le 27 août 2010,

Arielle Moro

2. Remerciements

Pour débuter, je tiens à remercier les personnes qui m'ont soutenue durant ces deux mois et qui ont participé de près ou de loin à ce travail.

En premier lieu, je remercie vivement Monsieur Peter Daehne pour son soutien et ses conseils précieux tout au long du travail. En second lieu, il est indispensable de remercier mes proches et amis qui m'ont soutenue et permis d'effectuer ce mémoire dans de très bonnes conditions.

En ce qui concerne l'élaboration de certains chapitres, je remercie beaucoup les entreprises et étudiants qui ont répondu positivement à la participation à mes questionnaires.

Et pour terminer, les derniers remerciements seront pour Matias Louro et Julien Racordon avec qui j'ai pu discuter d'aspects techniques et débattre dans le but d'enrichir mon mémoire.

3. Sommaire

Depuis quelques années, Internet est vraiment entré dans les mœurs : tant dans les entreprises qu'au sein de chaque foyer. En effet, Internet permet de communiquer à travers le monde en quelques secondes, de vendre toute sorte de produits en déployant des solutions e-commerce facilement et bien d'autres choses. Internet est donc un véritable vecteur de communication, de commerce et à présent, avec le Web 2.0, un vrai berceau d'informations (tant des informations personnelles que des informations d'entreprises) grandement facilité avec le cloud computing.

Dans le but de déployer au mieux des solutions de e-commerce, pour créer des applications internes aux entreprises, ou alors pour simplement mettre en place un blog ou un album photo, on a recours à des solutions (logiciels) aidant à ces créations. Pour développer des applications web plus facilement et gagner du temps, c'est là qu'intervient un framework. Il s'agit d'un outil orienté développeur et qui ne peut pas être utilisé sans avoir de solides connaissances dans le domaine informatique ainsi que dans le langage de programmation choisi.

Au sein de ce travail, cinq parties cruciales seront développées. Tout d'abord, nous débuterons par la description des notions de base essentielles à connaître avant d'aborder les frameworks.

Puis nous décrirons précisément ce qu'est un framework ainsi que sa structure tout en passant par un aperçu précis de ce que signifie un ORM (Object Relational Mapping) et son lien important avec les frameworks.

Suite à ces descriptions, nous analyserons trois frameworks Php et Java très populaires en vous donnant les clés pour démarrer des projets. Il s'agira de Symfony™, Zend™ framework et Struts™ avec l'utilisation d'ORM tels que Doctrine™ et Hibernate™. Une analyse clôturera le tout en évaluant ces différents frameworks.

Ensuite nous aborderons le substitut le plus connu du framework : le CMS (Content Management System).

L'avant dernière partie sera basée sur une analyse effectuée auprès d'entreprises et d'étudiants dans le but d'obtenir des informations concernant leurs choix vis-à-vis des frameworks. Ce chapitre se terminera sur un questionnaire aidant à choisir son propre framework en fonction du projet qu'on souhaite réaliser.

Pour terminer ce travail de bachelor sur les frameworks au cœur des applications web, nous finirons sur les frameworks et leurs évolutions.

4. Introduction

4.1 Préambule

A l'heure où l'informatique est au cœur d'un nombre chaque jour grandissant d'entreprises et occupe une place importante, le nombre d'applications augmente de façon exponentielle et ces dernières deviennent de plus en plus complexes. Pour tout projet, les départements informatiques des entreprises ou toute personne réalisant un projet seront confrontés à la problématique suivante : vais-je avoir besoin d'un framework pour réaliser mon application ?

Nombre de questions surviennent s'agissant de cette problématique, au premier rang desquelles celle de déterminer si un framework est réellement utile pour un projet. Ce qui implique de savoir à quoi sert concrètement un framework et quelles en sont ses caractéristiques.

Ayant déjà été confrontée à ce problème, je souhaite, par le biais de ce travail de diplôme, décrire les grandes lignes d'un framework, vous donner des conseils clés qui vous permettront de faire un choix pertinent pour votre futur projet. A l'aide de démonstrations, nous comparerons les frameworks les plus populaires et vous fournirons les bases nécessaires pour débuter un projet (Symfony™, Zend™ framework pour le **Php** ou encore Struts™ pour **Java**).

4.2 Guide de lecture

Pour lire avec le plus d'aisance possible ce mémoire de bachelor, je vous invite à prendre note des indications suivantes :

- les mots en **bleu** (comme ci-dessus : « **Php** » ou encore « **Java** ») sont définis dans le glossaire en fin de document ;
- les mots en **vert** font référence à des dossiers ;
- les mots en **violet** représentent des fichiers ;
- les mots en **gras** symbolisent des classes, des attributs, des méthodes ou encore les mots marquants du chapitre en question ;
- les phrases ayant cette police : `exemple` (Comic Sans MS) représentent du code ou des lignes de commande ;
- et des fiches récapitulatives à la fin de chaque grand chapitre (fiches que vous retrouverez également en annexe à la fin du mémoire).

5. Table des matières

6. Liste des tableaux

7. Liste des figures

8. Notions préliminaires

8.1 Programmation orientée objet

La programmation orientée objet a été inventée dans les années soixante-dix par l'informaticien américain Alan Kay [Kay68].

Par rapport à la **programmation procédurale**, la programmation orientée objet est un paradigme de programmation impliquant une toute autre manière d'organiser le code de l'application à réaliser. Plus précisément ce paradigme fait interagir une multitude d'objets entre eux. Ces derniers ont chacun des données ainsi qu'un comportement qui leur est associé.

Par exemple, nous pouvons avoir une classe **Client** avec son nom et prénom comme données ainsi que **passerCommande()** et **majConnexion()** comme méthodes qui vont définir son comportement.

Un objet est tout simplement créé lors de l'instanciation d'une classe. Et bien évidemment, une classe peut être instanciée autant de fois que nécessaire lors de l'exécution du programme.

Pour organiser correctement le code, il faut débuter par une **modélisation UML** pertinente en élaborant un **diagramme de cas d'utilisation** ainsi qu'en effectuant une description détaillée de chaque **cas d'utilisation** (flot de base + flots alternatifs). Suite à cela, nous pourrons concevoir les **diagrammes d'analyse** ainsi que les **diagrammes de séquence** décrivant chacun des cas d'utilisation. Pour terminer, il sera aisé de définir le **diagramme de classes** de l'application en regroupant tous les diagrammes d'analyse des différents cas d'utilisation.

8.1.1 Notion de classe et d'interface

8.1.1.1 Classe

Une classe permet de décrire un type regroupant des données (attributs) et des méthodes qui permettent de manipuler ces données ; les méthodes définissent le comportement de la classe.

Un constructeur est une méthode spécifique qui permet d'initialiser un contexte à la création d'un objet.

Les accesseurs (*getters* et *setters*) sont des méthodes particulières qui permettent d'obtenir ou de modifier les données de façon simple. La signature de la méthode détermine la fonctionnalité de celle-ci.

Considérons par exemple une classe **Client** qui aurait comme attributs **nom**, **prenom** et **dernièreConnexion** avec des accesseurs ainsi qu'une méthode **majConnexion()**.

Le chapitre 8.1.3 définit la signification les termes **private** et **public** préfixant des identificateurs d'attributs ou de méthodes.

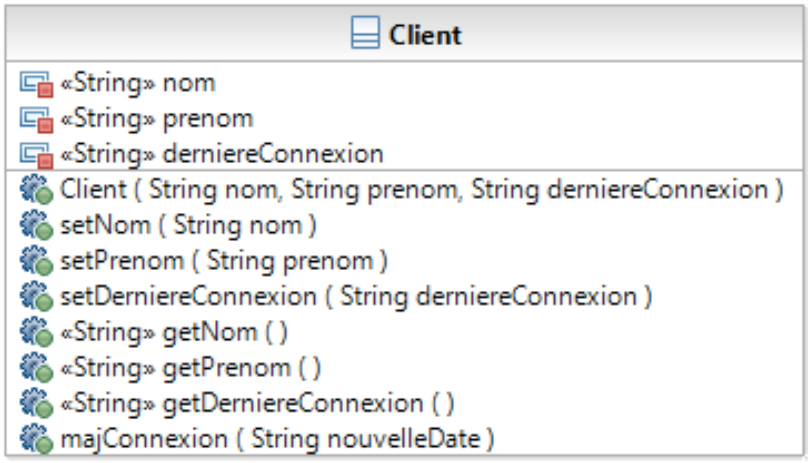


Figure 1 : Schéma UML de la classe Client

Code en Php :

```
class Client {

        private $nom;

        private $prenom;

        private $derniereConnexion;

        public function __construct($nom, $prenom, $derniereConnexion){
                $this->nom = $nom;
                $this->prenom = $prenom;
                $this->derniereConnexion = $derniereConnexion;
        }

        public function majCompteClient($nouvelleDate){
                $derniereConnexion = $nouvelleDate;
        }

        public function setNom($nom){
                $this->nom = $nom;
        }

        public function setPrenom($prenom){
                $this->prenom = $prenom;
        }

        public function setDerniereConnexion($derniereConnexion){
                $this->derniereConnexion = $derniereConnexion;
        }

        public function getNom(){
                return $nom;
        }

        public function getPrenom(){
                return $prenom;
        }

        public function getDerniereConnexion(){
                return $derniereConnexion;
        }
        public function majConnexion($nouvelleDate){
                setDerniereConnexion($nouvelleDate);
        }

}
```

Code en Java :

```
class Client {

        private String nom;

        private String prenom;

        private String derniereConnexion;

        public Client(String nom, String prenom, String derniereConnexion){
                this.nom = nom;
                this.prenom = prenom;
                this.derniereConnexion = derniereConnexion;
        }

        public void setNom(String nom){
                this.nom = nom;
        }

        public void setPrenom(String prenom){
                this.prenom = prenom;
        }

        public void setDerniereConnexion(String derniereConnexion){
                this.derniereConnexion = derniereConnexion;
        }

        public String getNom(){
                return nom;
        }

        public String getPrenom(){
                return prenom;
        }

        public String getDerniereConnexion(){
                return derniereConnexion;
        }

        public void majConnexion(String nouvelleDate){
                setDerniereConnexion(nouvelleDate);
        }

}
```

La classe présentée ci-dessus est concrète (toutes les méthodes sont concrètes). Cependant, si nous souhaitons définir une classe abstraite composée de méthodes abstraites et de méthodes concrètes, nous pouvons employer une classe abstraite.

Une telle classe ne pourra pas être instanciée ; il faudra donc la concrétiser en créant un héritage[1], la classe abstraite deviendra la classe mère.

Une classe abstraite se définit ainsi :

(les termes en italique signifient le caractère abstrait de la classe, de l'attribut ou de la méthode)

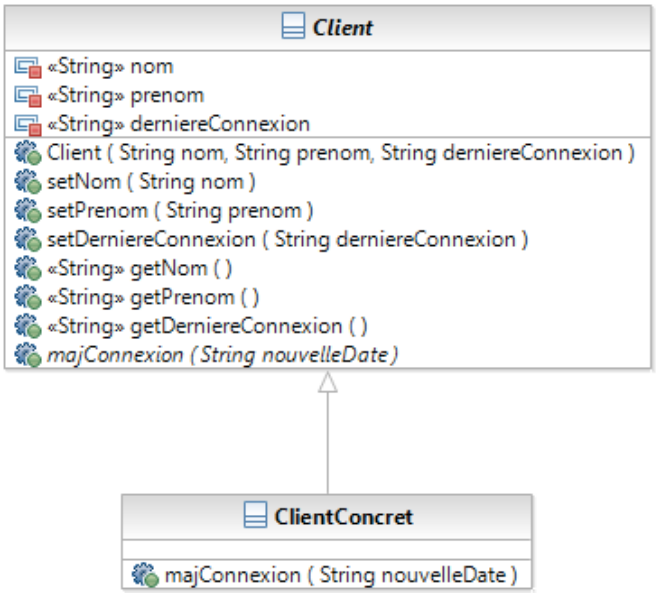


Figure 2 : Schéma UML des classes

Code en Php :

```
class Client {

        private $nom;

        private $prenom;

        private $dernièreConnexion;

        public function __construct($nom, $prenom, $dernièreConnexion){
                $this->nom = $nom;
                $this->prenom = $prenom;
                $this->dernièreConnexion = $dernièreConnexion;
        }
```

[1] Voir le sous-chapitre 8.1.2

```
        public function majCompteClient($nouvelleDate){
                $derniereConnexion = $nouvelleDate;
        }

        public function setNom($nom){
                $this->nom = $nom;
        }

        public function setPrenom($prenom){
                $this->prenom = $prenom;
        }

        public function setDerniereConnexion($derniereConnexion){
                $this->derniereConnexion = $derniereConnexion;
        }

        public function getNom(){
                return $nom;
        }

        public function getPrenom(){
                return $prenom;
        }

        public function getDerniereConnexion(){
                return $derniereConnexion;
        }

        abstract public function majConnexion($nouvelleDate);

}

class ClientConcrete extends Client {

        public function majConnexion($nouvelleDate){
                setDerniereConnexion($nouvelleDate);
        }

}
```

Code en Java :

```
abstract class Client {

        private String nom;

        private String prenom;

        private String derniereConnexion;

        public Client(String nom, String prenom, String derniereConnexion){
                this.nom = nom;
                this.prenom = prenom;
```

```
            this.derniereConnexion = derniereConnexion;
        }

        public void setNom(String nom){
            this.nom = nom;
        }

        public void setPrenom(String prenom){
            this.prenom = prenom;
        }

        public void setDerniereConnexion(String derniereConnexion){
            this.derniereConnexion = derniereConnexion;
        }

        public String getNom(){
            return nom;
        }

        public String getPrenom(){
            return prenom;
        }

        public String getDerniereConnexion(){
            return derniereConnexion;
        }

        abstract public void majConnexion(String nouvelleDate);

}

class ClientConcrete extends Client {

        public void majConnexion(String nouvelleDate){
            setDerniereConnexion(nouvelleDate);
        }

}
```

- Attribut et méthode *static* :

Un attribut *static* est global à une classe et peut être atteint sans même avoir instancié la classe. Il est unique et peut être atteint par toutes les instances de la classe.

Une méthode *static* peut être atteinte sans que la classe ait été instanciée au préalable. Elle ne peut utiliser que des attributs et des méthodes statiques. Ce type de méthode est utilisé au sein du *design pattern* **Singleton**[2].

[2] Voir le sous-chapitre 8.3.1

- Attribut, classe et méthode *final* :

Un attribut *final* possède une valeur qui ne pourra pas être modifiée au cours de l'exécution du programme. Une méthode *final* est une méthode qui ne pourra jamais être redéfinie par une autre classe dans l'arbre de dérivation (héritage). Pour terminer, une classe *final* ne pourra être étendue et sera par conséquent une feuille de l'arbre de dérivation.

- Liaison dynamique :

La liaison dynamique est un mécanisme étroitement lié à l'héritage. Nous considérons **C** comme classe mère avec pour méthode **m()** et **C'** comme classe fille. Si ensuite nous créons un objet **o** de type statique **C** (type connu à la compilation) et de type dynamique **C'** (type mis en évidence lors de l'exécution du programme) et que nous appelons la méthode **m()** ainsi **o.m()** il s'agira d'une liaison dynamique. Plus concrètement, lorsque cette méthode sera appelée, elle sera cherchée localement dans un premier temps (au niveau de la classe fille) puis si elle ne la trouve pas, elle la cherchera au niveau de la classe mère.

- Surcharge d'une méthode :

Le principe de surcharge d'une méthode permet de définir plusieurs méthodes ayant le même nom (ce qui est naturel si ce nom définit une fonctionnalité). Cependant ces méthodes ne posséderont pas les mêmes signatures (nom + paramètres). Lors de son emploi dans du code, c'est la signature qui détermine univoquement de quelle méthode il s'agit.

Code en Php :

En Php, il n'est pas possible de surcharger les méthodes. Cependant, il existe un moyen de contourner ce problème avec la méthode **public function __call($method, $args){…}**. Cette méthode est déclenchée automatiquement si une méthode n'est pas trouvée (alors qu'elle est appelée dans le code) lors de l'exécution du programme. A l'intérieur de cette méthode **__call** au sein de la classe, nous pouvons faire le traitement souhaité en fonction de la méthode et de(s) paramètre(s) qui auront été interceptés lors de l'erreur (qui n'en devient plus une avec ce contournement).

Code en Java :

```
majConnexion(String nouvelleDate){…}
majConnexion(String nouvelleDate, String heure){…}
```

8.1.1.2 Interface

Une interface permet de définir le comportement global (une ou plusieurs méthodes) d'un ensemble de classes qui vont l'implémenter. Par exemple, si vous déterminez qu'il serait plus adéquat de définir un comportement précis pour gérer une connexion de client sur un site Internet, l'interface est la bonne solution. De plus, elle est un véritable élément réutilisable. Il suffit de définir la signature des méthodes qui modélisent ce comportement.

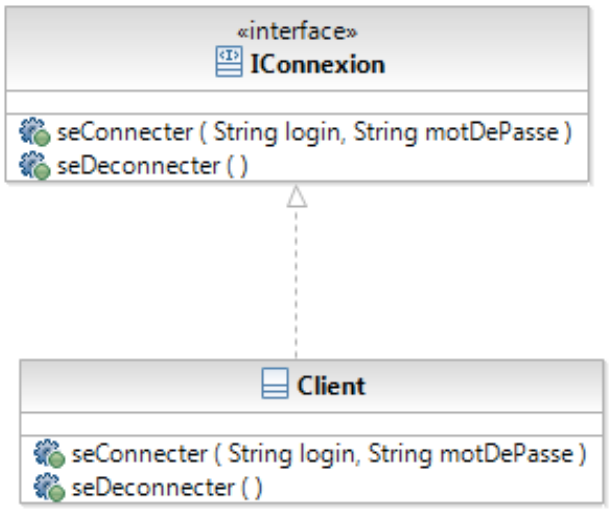


Figure 3 : Schéma UML des classes

Code en Php :

```
interface IConnexion{
        public function seConnecter($login, $motDePasse);
        public function seDeconnecter();
}
class Client implements IConnexion{
        …
        public function seConnecter($login, $motDePasse){…};
        public function seDeconnecter(){…};
}
```

Code en Java :

```
interface IConnexion{
        public void seConnecter(String login, String motDePasse);
        public void seDeconnecter();
}
class Client implements IConnexion{
        …
        public void seConnecter(String login, String motDePasse){…};
        public void seDeconnecter(){…};
}
```

8.1.2 Concept d'héritage

Le concept d'héritage est, avec la liaison dynamique, l'un des deux atouts majeurs de l'objet. Pour expliquer simplement ce concept, il faut considérer le contexte suivant : nous avons différents types de clients à modéliser sous forme de classes (client Internet, client magasin). Nous remarquons que ces deux types de client ont des attributs en commun tels que leurs noms et prénoms, cependant des attributs vont aussi les distinguer : un login et un mot de passe seront définis uniquement pour le client Internet. Leurs comportements vont également être différents : le client Internet disposera de la méthode **seConnecter()** et le client magasin aura uniquement la méthode **verifierAchatsCaisseRapide()**. Tous les deux auront comme méthode commune **payerAchats()**.

Nous allons donc, pour utiliser pleinement l'héritage, créer une classe mère **Client** avec les attributs communs aux deux types de client ainsi que la méthode qu'ils possèdent tous les deux. Puis nous allons faire dériver, de cette dernière, deux classes filles **ClientInternet** et **ClientMagasin** avec leurs éventuels attributs supplémentaires et leurs méthodes respectives.

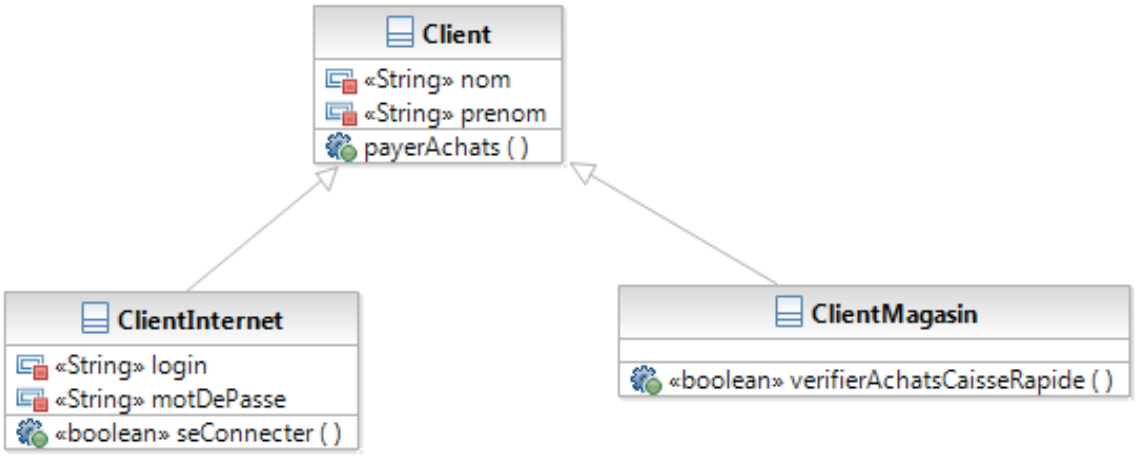


Figure 4 : Schéma UML des classes

Ces différentes classes s'instancient ainsi :

```
Client c = new ClientInternet();

Client cc = new ClientMagasin();
```

Client est le type statique de **c** ou **cc** et **ClientInternet** ainsi que **ClientMagasin** sont des types dynamiques. Le type statique est identifié lors de la compilation, tandis que le type dynamique est identifié uniquement lors de l'exécution du programme.

Ensuite, en Java, pour désigner les attributs ou une méthode de la classe mère **Client**, il suffit d'écrire **super.payerAchats() ;** ou encore **super.nom**. En Php, nous écrirons **parent::payerAchats()** et **parent::nom** respectivement.

Exemple de la définition de cette hiérarchie de classe :

Code en Php :

```
class Client {
        $nom;
        $prenom;
        function payerAchats(){…}
}
class ClientInternet extends Client {
        $login;
        $motDePasse;
        function seConnecter(){…}
}
class ClientMagasin extends Client {
        function verifierAchatsCaisseRapide(){…}
}
```

Code en Java :

```
class Client {
        String nom;
        String prenom;
        public void payerAchats(){…}
}
class ClientInternet extends Client {
        String login;
        String motDePasse;
        public boolean seConnecter(){…}
}
class ClientMagasin extends Client {
        public boolean verifierAchatsCaisseRapide(){…}
}
```

8.1.3 Notion de visibilité

Dans les sous-chapitres précédents, vous aurez pu remarquer que, devant certaines méthodes ou attributs figuraient des mots tels que **public** ou encore **private**. Ce sont des éléments très importants qui vont vous permettre de rendre visible ou non des attributs, des méthodes ou encore des classes. Par défaut bien évidemment chaque élément est de visibilité publique (**public**).

En Php :

- **public** : un attribut ou une méthode sera visible par tous ;
- **private** : un attribut ou une méthode sera visible uniquement au sein de sa classe ;
- **protected** : un attribut ou une méthode sera visible uniquement au sein de sa classe et de ses sous-classes.

En Java :

- **public** : un attribut ou une méthode sera visible par tous ;
- **private** : un attribut ou une méthode sera visible uniquement au sein de sa classe ;
- **protected** : un attribut ou une méthode sera visible uniquement au sein de sa classe, de ses sous-classes et du package de définition ;
- **« rien » :** un attribut ou une méthode sera visible uniquement au sein d'un package (un package permet de rassembler un ensemble de classes et d'interfaces qui sont en relation pour fournir une fonctionnalité de l'application par exemple).

8.1.4 Principaux diagrammes UML

A chaque étape d'un développement informatique, nous avons des diagrammes pertinents à réaliser pour créer une documentation qui sera efficace et utilisée sur le long terme (principalement pour la maintenance).

Le meilleur exemple, pour illustrer tous ces diagrammes, est le modèle de **Kruchten** ou encore le **modèle 4 + 1**. Le voici ci-dessous :

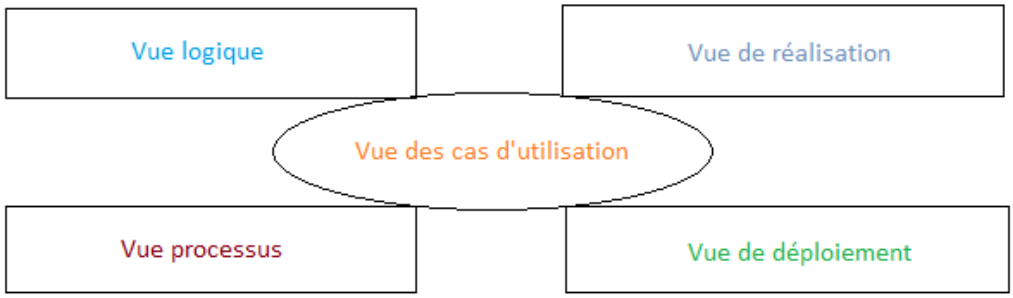


Figure 5 : Modèle 4 + 1 de Kruchten

Au sein de la vue logique, nous réalisons des diagrammes de classes, **d'objets**. De plus, pour montrer les différentes relations de ces objets entre eux et leurs comportements, nous pouvons définir des diagrammes de séquence, d'activités, **de communication** ou encore **d'états transitions**.

Ce qui caractérise la vue de réalisation est le(s) composant(s) que nous allons trouver dans l'application et son (leur) organisation. Par conséquent, nous réalisons, à cette étape, un **diagramme de composants**.

La vue processus est très proche de la vue logique, tout simplement car nous réalisons des diagrammes dynamiques similaires (diagrammes de séquence, d'activités, de communication et d'états transitions). Cependant ils sont tous axés processus et non objets.

L'avant-dernière vue est la vue de déploiement, elle permet de mettre en évidence l'application dans son environnement final. Il faut prendre en compte les caractéristiques et les contraintes techniques (serveur, bande passante, etc). Pour modéliser le tout, nous ferons un **diagramme de déploiement**.

Pour terminer, la vue des cas d'utilisation est centrale et importante car elle prend en compte les fonctionnalités de l'application. Elle va confronter les acteurs de l'application ainsi que les cas d'utilisation pour mettre en évidence leurs interactions. Nous réaliserons donc des diagrammes de cas d'utilisation, des diagrammes de séquence, d'activités, de communication et d'états transitions.

8.2 Base de données

8.2.1 Architecture

Une base de données sert à stocker les données qui sont utiles à l'application ou alors qui sont amenées à changer au cours de l'exécution de l'application (des enregistrements relatifs à des clients ou des commandes par exemple). A cette fin, le système de base de données (SGBD) saura interagir avec ces enregistrements et réaliser des opérations de recherche, d'ajout, de modification ou encore de suppression (CRUD).

Un SGBD est constitué de nombreux éléments (ou objets de base de données) tels que :

- des tables composées de champs (table Client ayant pour attributs nom et prénom par exemple) ;
- des procédures composées d'opérations qui permettent de manipuler les données ;
- des triggers qui sont des traitements qui se déclenchent lorsqu'il se produit un événement particulier (ajout d'un enregistrement dans une table par exemple) ;
- des vues qui permettent de considérer les données résultant d'une ou plusieurs requête(s) imbriquées ;
- et bien d'autres objets selon la solution que vous utilisez.

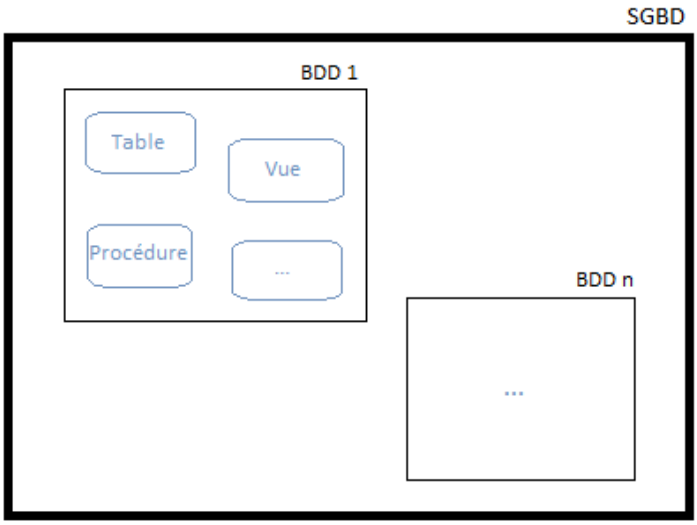


Figure 6 : Schéma de l'architecture d'un SGBD

8.2.2 Types de données, clés et contraintes d'intégrité

Au sein d'une table, nous pouvons attribuer un certain type à un champ donné (tout comme en Java). Nous pouvons distinguer différents types de données ou encore de champs qui sont intimement liés à la solution que nous choisissons.

Par exemple, Oracle™ et MySQL™ proposent tous deux de gérer les chaînes de caractères à l'aide du type VARCHAR. Cependant pour les nombres, Oracle™ proposera les types NUMBER, FLOAT, etc tandis que MySQL™ offrira les types INT, FLOAT etc.

Pour gérer au mieux les tables et les relations qu'elles ont entre elles, on emploie des clés :

Clé primaire : Cette clé va permettre d'identifier chaque enregistrement ; il suffit simplement de créer un champ au sein de la table et d'indiquer qu'il s'agit bien d'une clé primaire :

- MySQL™ : option primary key,
- Oracle™ : PRIMARY KEY (*nomDeLaColonne*).

Clé étrangère : Cette clé va mettre en évidence qu'il existe une relation entre deux tables. Il s'agit d'une contrainte d'intégrité.

Pour MySQL™ et Oracle™ : FOREIGN KEY (*nomDeLaColonne*) REFERENCES NOM_TABLE_ETRANGERE(*nomDeLaColonneTableEtrangere*).

 Ces clés sont très importantes car elles devront être prises en compte lors de la génération des classes (passage du relationnel à l'objet) effectué par un framework spécifique.

Les contraintes d'intégrité permettent, entre autres, de définir qu'un champ est unique, d'indiquer la présence d'une clé étrangère ou d'une clé primaire ainsi que de mettre en évidence qu'un champ est « non null » (null = vide).

8.2.3 Bien concevoir sa base de données

Pour concevoir la base de données, plusieurs choix s'offrent à vous. Celui dont nous allons parler est l'un de ceux qui sont les plus utilisés.

Premièrement, il s'agit d'analyser, avec votre mandant, quelles seront les données qu'il souhaite conserver ou alors les données qui vont croître au fil du temps au sein de l'application. De plus il faudra définir les données dites « paramètres » de l'application qui vont contribuer à la structure de l'application.

Deuxièmement, vous allez réaliser un **modèle conceptuel de données (MCD)**, composé d'entités et de relations. Il s'agira de mettre en évidence les différentes cardinalités entre les entités pour pouvoir au mieux transformer votre modèle pour l'étape suivante.

Troisièmement, vous devez transformer le modèle conceptuel de données en **modèle logique de données (MLD)**. Cette transformation est une étape cruciale pour la conception de la base de données, car vous allez analyser chaque association entre entités (cardinalités comprises) dans le but de faire ressortir des clés étrangères[3] qui seront pertinentes pour l'établissement du script de création de la base de données de votre application. Dans certains cas, en fonction de cardinalités spécifiques (relation plusieurs à plusieurs par exemple), vous ne pourrez pas simplement rajouter une clé étrangère, vous devrez créer une table d'association qui fait la liaison entre les deux tables initiales.

Pour terminer, vous devrez rajouter des types spécifiques pour chacun des champs de chaque table de la base de données. Il faut bien évidemment que ces types correspondent à votre choix de solution pour contenir votre base de données[4]. Il vous suffira ensuite de créer le script en l'exportant depuis le logiciel qui vous a permis de construire vos modèles. Tous ces modèles correspondent à la **méthode Merise**.

Dans le cas où vous préférez directement construire votre base de données dans un logiciel prévu à cet effet, vous pouvez ensuite facilement générer un **modèle relationnel** (MySQL Workbench™ vous aide à réaliser le modèle).

Les différents logiciels qui peuvent vous aider à réaliser ces opérations, et que nous avons pu tester en réalisant des projets, sont RSA™ ou encore Win'Design™.

[3] Voir le sous-chapitre 8.2.2
[4] Voir le sous-chapitre 8.2.4

Schéma récapitulatif :

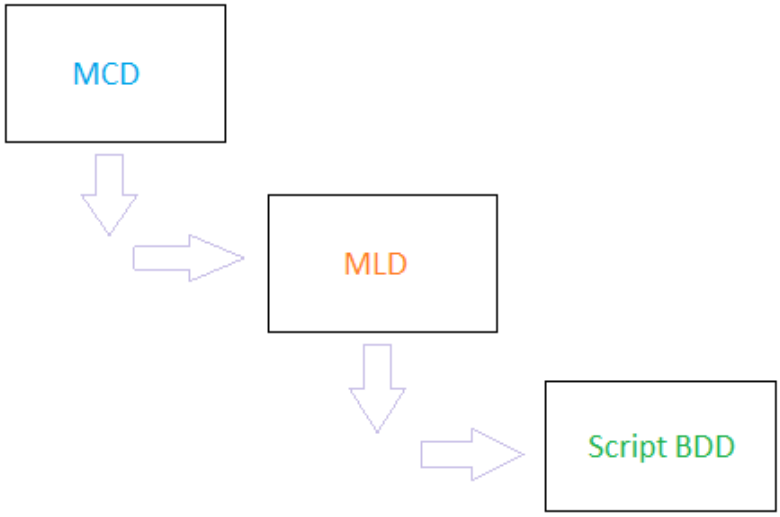


Figure 7 : Du MCD au Script BDD

⚠ Faites bien attention de mettre correctement à jour vos différents modèles lors de l'avancement de votre projet. Cette étape est primordiale, qui plus est si vous travaillez à plusieurs. Le mieux serait de définir un responsable base de données au sein du groupe de projet.

8.2.4 Solutions actuelles

Les solutions actuelles majeures sont Oracle™ et MySQL™ pour une application web. Au premier abord, Oracle™ apparaît comme étant d'une grande complexité comparé à MySQL™. Le choix de l'une ou l'autre solution est déterminé par les besoins du mandant ou par les contraintes de réalisation de l'application. Ils supportent tous les deux le langage **PL/SQL**.

- Oracle™ : Il s'agit d'un système de gestion de base de données payant mis en place en 1977. Outil relativement portable, il est supporté par un grand nombre de systèmes d'exploitation. Nous pouvons bien évidemment utiliser des logiciels gratuits qui permettent d'interagir plus facilement avec Oracle™. Il supporte le langage PL/SQL. Si vous vous rendez sur le site **http://www.oracle.com/us/products/index.html** vous pourrez constater le nombre important de produits et de services payants qu'Oracle™ propose et pourrez faire votre choix de solution.

- MySQL™ : Tant pour les débutants que pour les plus avancés, MySQL™ sera une excellente solution alternative à Oracle™. Cette solution est gratuite et entièrement supportée par la plupart des frameworks du marché, voire même intégrée[5]. MySQL™ est également un système de gestion de base de données qui est très bien supporté par la majeure partie des systèmes d'exploitation. Pour l'exploiter au mieux, vous pouvez télécharger **EasyPhp™**, **Wamp™** ou encore **Mamp™** (selon votre système d'exploitation) pour gérer facilement votre base de données et la mettre en relation avec votre application.

[5] Si vous choisissez le framework Zend™, vous pourrez le trouver en téléchargement sur le site Zend Server qui propose MySQL™ avec une interface graphique de gestion de ce dernier : **http://www.zend.com/fr/downloads/**.

- Comparaison succincte d'Oracle™ et de MySQL™:

Critères	Oracle™	MySQL™
Compatibilité avec PHP, Java ou tout autre langage de programmation web	X	X
Multi-utilisateurs	X	X
Multi-systèmes d'exploitation	X	X
Open Source	-	X
Options SQL disponibles	X	peu
Gratuit	-	X

Table 1 : Tableau comparatif Oracle™ et MySQL™

8.3 Design Patterns

8.3.1 Définition

- Qu'est-ce qu'un Design Pattern ?

Un Design Pattern est un élément d'architecture logicielle réutilisable qui va fournir une solution à un problème récurrent.

Considérons par exemple le problème suivant : comment faire pour être certain qu'une classe n'a qu'une et une seule instance d'elle-même au sein du programme en cours d'exécution ?

Nous allons réduire la visibilité du constructeur de cette classe (en lui mettant une visibilité private) et nous allons créer une méthode statique (une méthode de création de classe qui sera appelée par une autre classe qui dépendra d'elle) qui vérifiera si une instance n'a pas été créée au préalable. S'il en existe une, la méthode renverra l'instance en cours, sinon une instance de classe sera créée.

En étudiant cet exemple, vous venez de découvrir ou de redécouvrir le *Design Pattern* **Singleton**. Il permet de résoudre le type de problème évoqué ci-dessus et est énormément utilisé au sein des frameworks sans même que nous nous en rendions compte. En effet, Zend™ framework l'utilise pour n'avoir qu'une seule application par script et tout cela est géré avec le contrôleur Zend_Controller_Front.

Code en Php :

```
class DPSingleton {

        private static $instance = null;

        private function __construct(){
        }

        public static function getInstance(){
                if(is_null(self::$instance)){
                        self::$instance = new self;
                }
                return self::$instance;
        }

}
```

Code en Java :

```
class DPSingleton {

        private static DPSingleton instance = null;

        private DPSingleton(){
        }

        public static DPSinglton getInstance(){
                if(this.instance == null){
                        this.instance = new DPSingleton();
                }
                return this.instance;
        }

}
```

En définitive, les *Design Patterns* vont vous faire gagner beaucoup de temps, tant au niveau du développement, qu'au niveau de la maintenance de votre application. L'objectif étant de définir un maximum de *Design Patterns* que vous pourrez réutiliser pour concevoir vos applications, être plus performant et gagner un temps considérable pour vos projets.

8.3.2 Principaux Design Patterns

Premièrement, les *Design Patterns* jouent un rôle crucial au sein des frameworks. Deuxièmement, ils sont liés à la politique de développement informatique de l'entreprise ou du service informatique. Soit quels sont les composants que vous allez créer et conserver. Plus vous faites de projets, plus vous allez créer de composants et étoffer votre librairie.

Vingt-trois Design Patterns ont été recensés comme les plus connus [GOF] et un seul compose le cœur d'un framework : le Design Pattern Modèle-Vue-Contrôleur (MVC)[6].

De plus, il existe trois grands types de Design Patterns : de création, de structure et de comportement.

Pour en savoir plus, voici le lien wikipédia : **http://fr.wikipedia.org/wiki/Patron_de_conception**.

[6] Voir le sous-chapitre 9.2

8.4 Tests d'application

8.4.1 Concept de test

Le test est une phase majeure du développement informatique. Dans la mise en œuvre de la **méthode de développement RUP**, il est présent à chaque itération. De plus, dans des projets **Agile** il est important d'avoir un suivi régulier et journalier (voire hebdomadaire) des tests.

Les tests peuvent être réalisés à différents niveaux[7] et par différentes personnes (impliqués de près ou de loin dans le projet) ou programmes de test.

En ce qui concerne les langages de programmation, Php et Java intègrent tous deux, respectivement, **PHPUnit** ou **JUnit** qui permettent d'effectuer des tests unitaires ou fonctionnels à l'aide d'assertions pour vérifier le bon comportement d'éléments du code de l'application. Les frameworks intègrent d'ailleurs les solutions de test citées ci-dessus.

Bien évidemment, si vous comptez effectuer une phase de test digne de ce nom, il faut l'inclure à la phase de développement (construction de l'application) car cela implique de rajouter du code de « test » (avec les « **assert** »).

Au niveau des programmes de test, il existe des programmes open-sources tels que OpenSTA™ ou encore Apache JMeter™. Ou alors des outils IBM™ pouvant être intégré à **RSA™** comme IBM Rational Performance Tester™ pour les tests de charge ou encore IBM Rational Functional Tester™ pour les tests fonctionnels.

[7] Voir le sous-chapitre 8.4.2

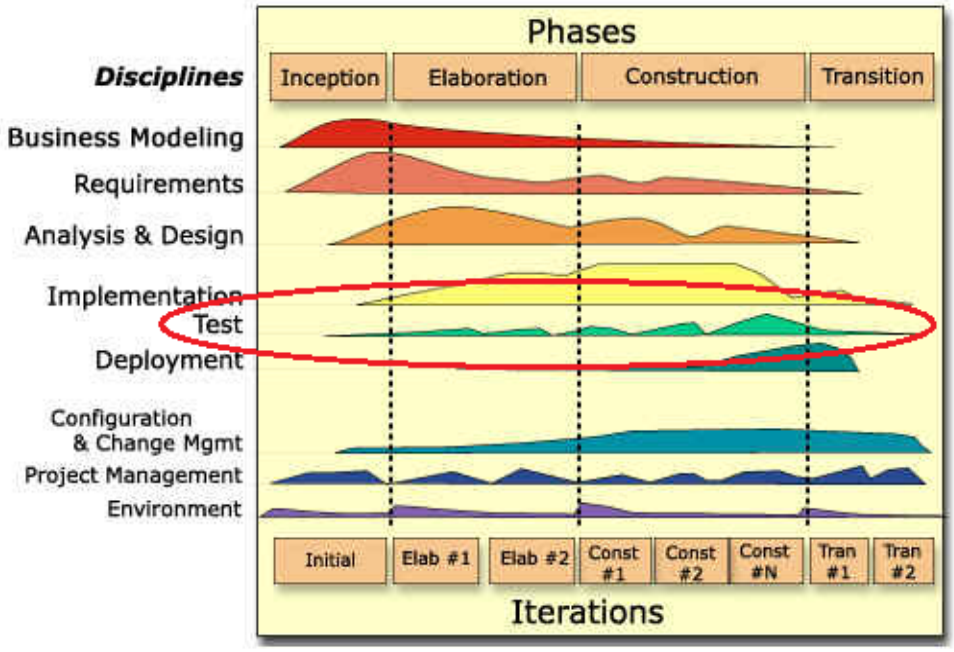


Figure 8 : Schéma développement RUP (Copyright IBM Rational)

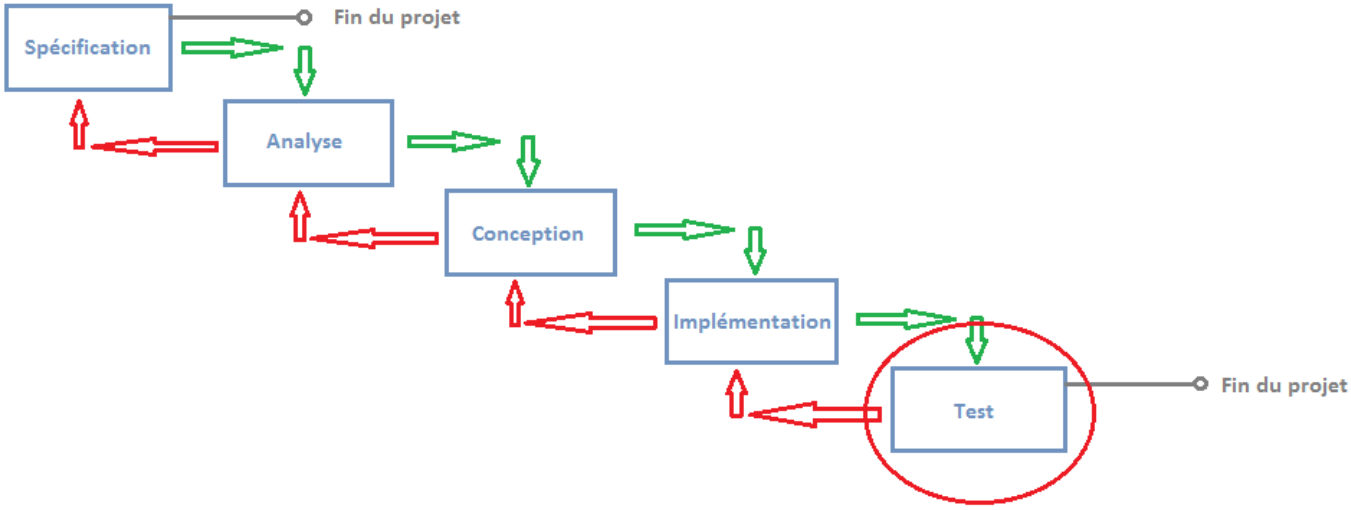


Figure 9 : Schéma de développement en cascade (Voir le glossaire en fin de document)

8.4.2 Principaux tests

- **Tests unitaires :** Ils servent à vérifier un composant ou alors une simple méthode de calcul par exemple.
- **Tests fonctionnels :** Ils sont axés principalement sur la fonctionnalité et le bon déroulement d'un scénario métier (scénario d'une commande au paiement par exemple).
- **Tests de charge :** Le but principal d'un test de charge est de voir quelles sont les limites de l'application (par exemple, tester le nombre maximal de connexions que nous pouvons avoir en même temps, très utile dans le cas d'une billetterie en ligne).
- **Tests d'intégration :** Ils permettent de vérifier qu'une nouvelle fonctionnalité s'intègre correctement à l'application (proche des tests unitaires et fonctionnels).
- **Tests de non régression :** Ils vérifient qu'une mise à niveau ne perturbe pas le comportement global de l'application.
- **Tests de performance :** Ils permettent de mettre en évidence si les souhaits de performance du mandant ou autre ont bien été atteints.
- **Tests de robustesse :** Il s'agit de simulation de problème hardware, software ou autre qui permet de voir comment réagit l'application développée lorsque ces problèmes surviennent.
- **Tests d'acceptation :** Avec ce test (qui s'effectue généralement à la fin de chaque livrable dans RUP) il est possible de déterminer si le livrable ou le logiciel correspond réellement aux attentes du mandant. Le plus souvent, le test est organisé en plusieurs phases de validation avec les personnes de l'entreprise mandataire qui devront utiliser le logiciel le plus couramment.

Evidemment, il n'est pas obligatoire de mettre en place l'intégralité des tests mentionnés ci-dessus. Il s'agit de planifier une stratégie performante de test et de choisir les tests les plus pertinents pour que l'application développée soit conforme à ses spécifications, que le client soit satisfait de l'application et qu'elle lui corresponde entièrement.

8.5 Fiche n°1 : Notions préliminaires

1. Programmation orientée objet

Paradigme objet : Héritage, liaison dynamique, masquage, encapsulation.

Classe : attribut(s) + méthode(s).

Objet : instance d'une classe.

Classe abstraite : attribut(s) + méthode(s) concrète(s) et abstraite(s) (doit être étendue par une autre classe).

Interface : signature(s) de méthode(s) (doit être implémentée par une autre classe).

2. Base de données

Base de données : composée d'objet(s) de schéma tels que une ou plusieurs table(s), vue(s), procédure(s)...

Table : composée d'un ou plusieurs champ(s) (corps de la table) dont au moins une clé primaire et qui contient des enregistrements.

Clé primaire : champ d'identification d'une enregistrement d'une table (unique).

Clé étrangère : champ qui permet de faire des liaisons entre deux ou plusieurs tables (correspond généralement à la clé primaire d'une autre table).

Modèles principaux pour réaliser une bonne base de données : modèle conceptuel de données (MCD), modèle logique de données (MLD) et modèle relationnel de données.

3. Design Patterns

Définition : Brique d'architecture logicielle réutilisable qui répond à un problème récurrent.

4. Tests d'application

Principaux tests d'application : Tests unitaires et tests fonctionnels.

9. Qu'est-ce qu'un framework ?

9.1 Définition

Précédemment, lorsque que nous développions sans framework, nous utilisions uniquement le langage de programmation pour produire des scripts (de plus, nous choisissions également l'organisation de nos dossiers). Selon le langage de programmation choisi, il existait déjà des **IDE** (exemple : Netbeans™) qui nous permettaient de réaliser des applications plus facilement (dans un environnement plus agréable qu'un simple bloc-notes).

Le framework va venir se greffer à l'IDE, tout comme le langage de programmation choisi, tel une grande librairie qui va se trouver dans un dossier spécifique au sein de l'application.

Il s'agit donc un outil de haut niveau structuré selon une architecture de fichiers qui lui est propre et qu'il faudra respecter tout au long du développement.

Le framework va permettre de mettre en place une application en effectuant des séries d'opérations de manière simplifiée en utilisant des méthodes qui auront été conçues au préalable au sein des librairies du framework.

Nous pouvons trouver des frameworks gratuits ou payants, **open-source** ou non. Les communautés qui les entourent sont généralement très présentes et réactives. Ce qui permet de fournir des frameworks gratuits d'excellente qualité.

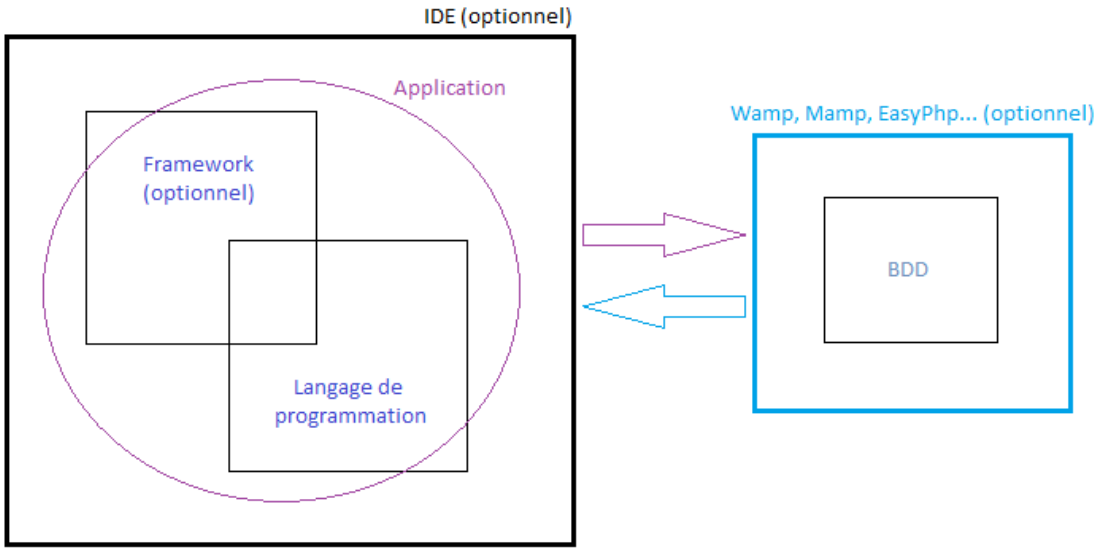


Figure 10 : IDE, framework et langage de programmation

Pourquoi utiliser un framework ?

- **Gagner du temps :**

  L'utilisation d'un framework vous fera gagner du temps. Par exemple, le fait de générer automatiquement toutes les classes correspondant à votre base de données vous fera gagner un temps considérable (si vous n'avez que trois tables, ce gain n'est pas énorme, mais si vous en avez une centaine, c'est à ce moment là que se fera sentir la puissance et la valeur ajoutée de l'utilisation d'un framework).

- **Améliorer l'organisation de votre application :**

  Comme nous l'avons évoqué précédemment, tout framework possède une architecture de fichiers qui lui est propre. Comme le *Design Pattern* **Modèle-Vue-Contrôleur** dirige la plupart des frameworks, ces derniers suivent également la répartition des fichiers selon ce même *Design Pattern*. En séparant bien distinctement les scripts qui concernent l'affichage (**vues**) de ceux qui sont relatifs aux classes (**modèles**) et pour terminer ceux qui associent les deux précédents (**contrôleurs**).

- **Améliorer la maintenance de votre application :**

  Mieux votre code sera organisé, plus les éléments composant votre application pourront être retrouvés facilement. En définitive, la maintenance (qui consiste à corriger d'éventuels problèmes qui pourraient survenir) sera améliorée si l'organisation l'est au préalable.

- **Disposer de normes de développement :**

  La plupart des frameworks utilisent des normes de développement qui facilitent, par exemple, l'interaction avec la base de données. Lors de la définition de nouveaux composants, on peut avoir à élaborer des méthodes de base qui permettront de gérer le composant à travers le framework. Tout cela permettra également de se repérer plus facilement au sein du code et de développer de manière plus efficace en équipe.

- **Répartir plus facilement le travail au sein de l'équipe de projet :**

  En utilisant un framework basé sur le *Design Pattern* **Modèle-Vue-Contrôleur**, vous pourrez séparer les interactions avec la base de données, les différentes vues qui vont composer un design et les scripts Php qui relient les deux. Au final, il est plus simple de répartir le travail en fonction des compétences de chacun des membres de l'équipe de projet.

- **Personnaliser facilement des composants existants :**

  Les frameworks possèdent toujours des composants évolués tels que des composants de gestion d'adobe PDF ou encore des composants de gestion de Google Maps (en installant des extensions). Et nous pouvons facilement les personnaliser en utilisant le principe d'héritage (de classe dérivée ou étendue).

9.2 Architecture Modèle-Vue-Contrôleur

L'architecture **Modèle-Vue-Contrôleur** est l'application propre du *Design Pattern* **Modèle-Vue-Contrôleur** (**Model-View-Controller**).

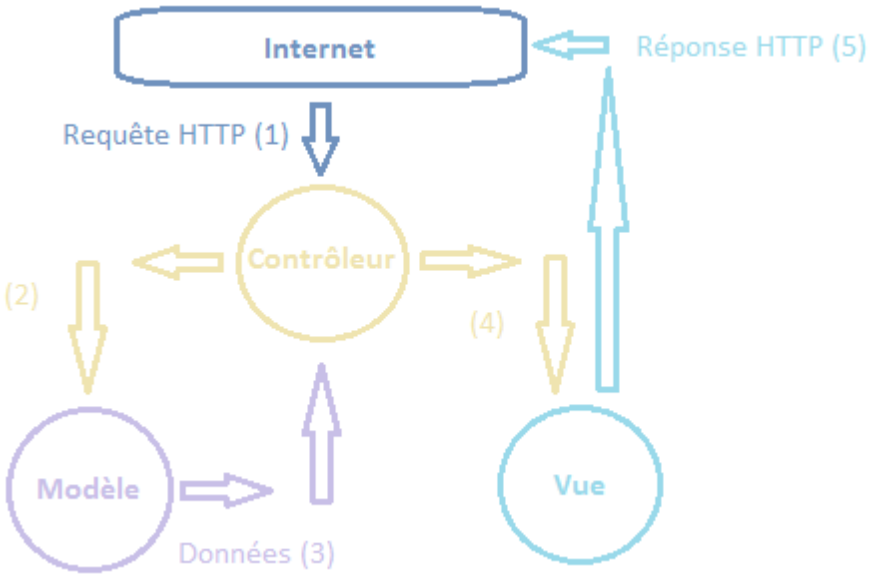


Figure 11 : Schéma global Modèle-Vue-Contrôleur

Ce design pattern est composé de trois couches :

La couche « modèle ou model » permet de structurer les données et interagit directement avec la base de données. En définitive elle se rapporte à la couche métier.

La seconde couche « vue ou view » est présente pour disposer les éléments à afficher et mettre en forme les pages. Elle définit donc la couche présentation.

Pour terminer, la troisième couche « contrôleur ou controller » est l'élément qui dirige les requêtes vers le modèle, la vue et le serveur http. Le contrôleur a une fonction de dispatcheur ou encore d'organisateur.

Architecture Client-Serveur :

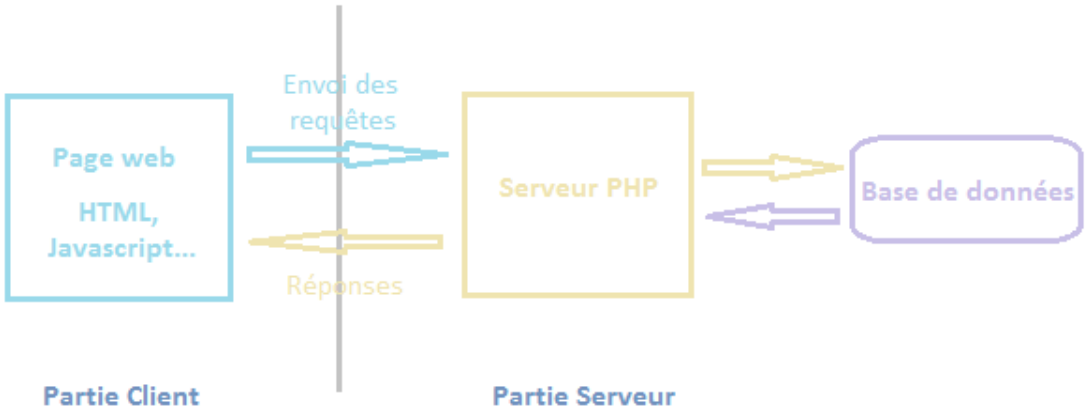


Figure 12 : Architecture Client-Serveur

Cette architecture Client-Serveur met en évidence les différents niveaux des applications web ainsi que les communications entre ceux-ci.

En quelques mots, la partie Client se trouve être la page web de votre navigateur (ou browser). Lorsqu'un client (ou utilisateur) remplit un formulaire de login par exemple et le valide, une requête est envoyée au serveur qui va l'analyser et rechercher des informations au sein de la base de données. Ensuite, dès que le serveur reçoit la réponse de la base de données, il va renvoyer une réponse à la partie Client et votre navigateur va afficher votre page personnelle suite au succès de votre authentification.

Après cette brève explication, nous voyons rapidement quelles superpositions nous pouvons établir entre l'architecture Modèle-Vue-Contrôleur et l'architecture Client-Serveur.

En effet, la partie Client peut s'apparenter aux différentes vues d'une application web (présence d'HTLM ou Javascript). La partie Serveur est tout naturellement étroitement liée aux contrôleurs de l'application (fichiers Php). Et au final, les modèles seront aussi dans la partie Serveur (fichiers Php).

Il est très important d'avoir fait cette superposition dans le but de mieux comprendre les architectures des applications web ou sites Internet.

Comment se répartir le travail au sein de l'équipe de projet ?

- couche « modèle ou model » : développeurs spécifiques et spécialistes base de données ;
- couche « vue ou view » : designers ;
- couche « contrôleur ou controller » : développeurs spécifiques.

Comment cette architecture intervient-elle au sein d'un framework ?

Pour illustrer l'intervention de ce Design Pattern au sein d'un framework, cette architecture sera présentée avec les trois frameworks suivants : Symfony™ et Zend™ framework pour débuter, qui possèdent une architecture commune car ils utilisent le même langage de programmation Php. Et, à titre comparatif, nous présenterons le framework Struts™ qui est basé sur le langage de programmation Java.

- **Symfony™ et Zend™ framework :**
  - Couche « modèle » :
    - Interface d'accès à la base de données (**API**)
    - Accès aux données
  - Couche « vue » :
    - Vue
    - Template
    - Layout
  - Couche « contrôleur » :
    - Contrôleur frontal
    - Action

D'un point de vue global, la structure MVC de Symfony™ et de Zend™ framework sont relativement proches. Elles diffèrent quelque peu si l'on considère certains éléments tels que :

- la structure interne du contrôleur frontal (contrôleur père de l'application) ;
- la couche « modèle » qui ne se crée pas de la même manière en fonction des frameworks.

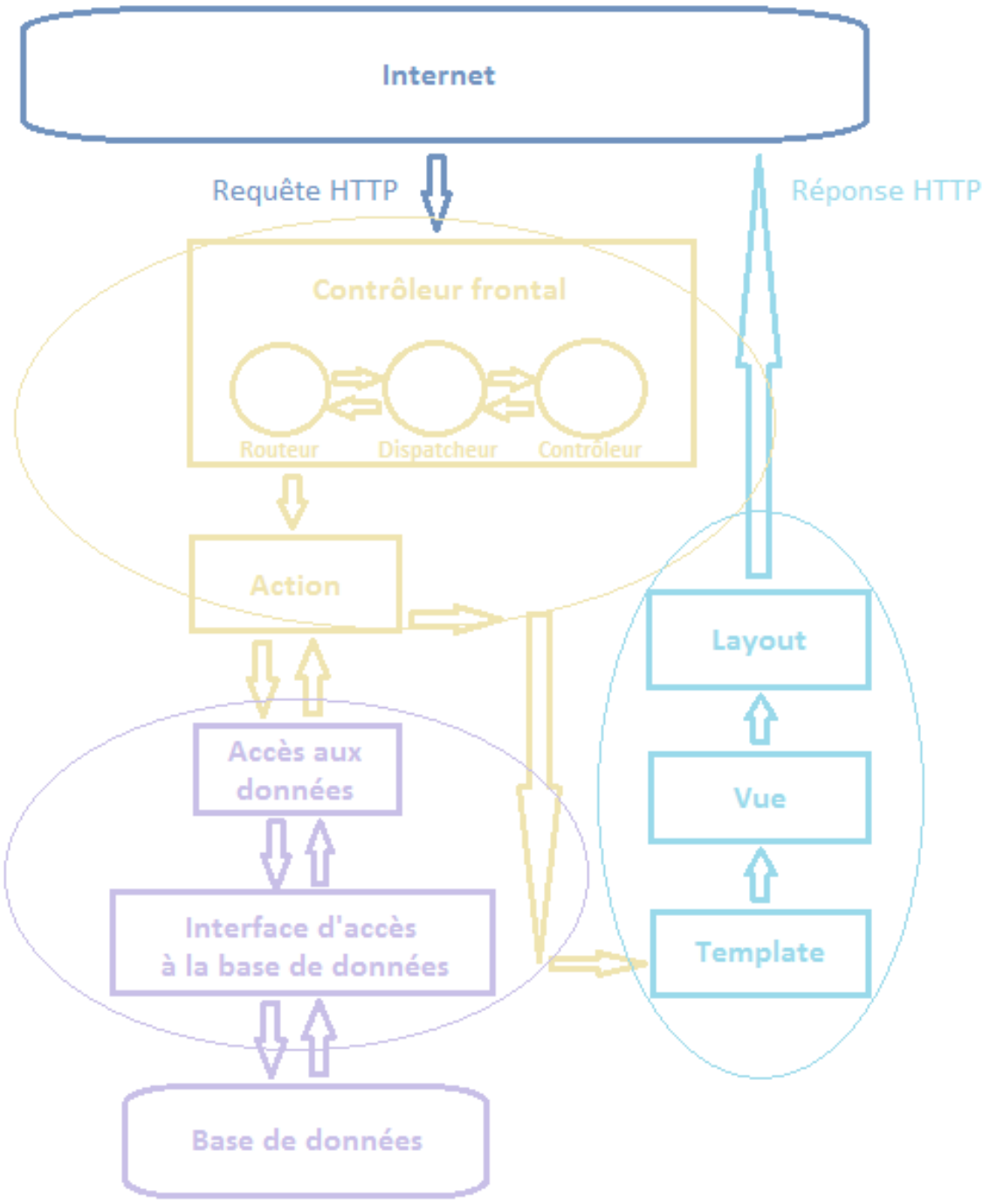


Figure 13 : Schéma global MVC pour Symfony™ et Zend™ framework

- **Struts™ :**

  - Couche « modèle » :
    - **Modèle Java Beans**
    - Accès aux données
  - Couche « vue » :
    - **Fichier JSP**
  - Couche « contrôleur » :
    - **Contrôleur servlet**
    - Action

En comparant les deux approches MVC du schéma pour Symfony™ et Zend™ framework et celui pour Struts™, on peut en conclure que les principales différences sont liées à la différence de langage de programmation. En effet, pour la composition de Struts™, nous avons la présence d'un contrôleur servlet, de fichiers JSP et de modèles qui respectent les conventions de type JavaBeans.

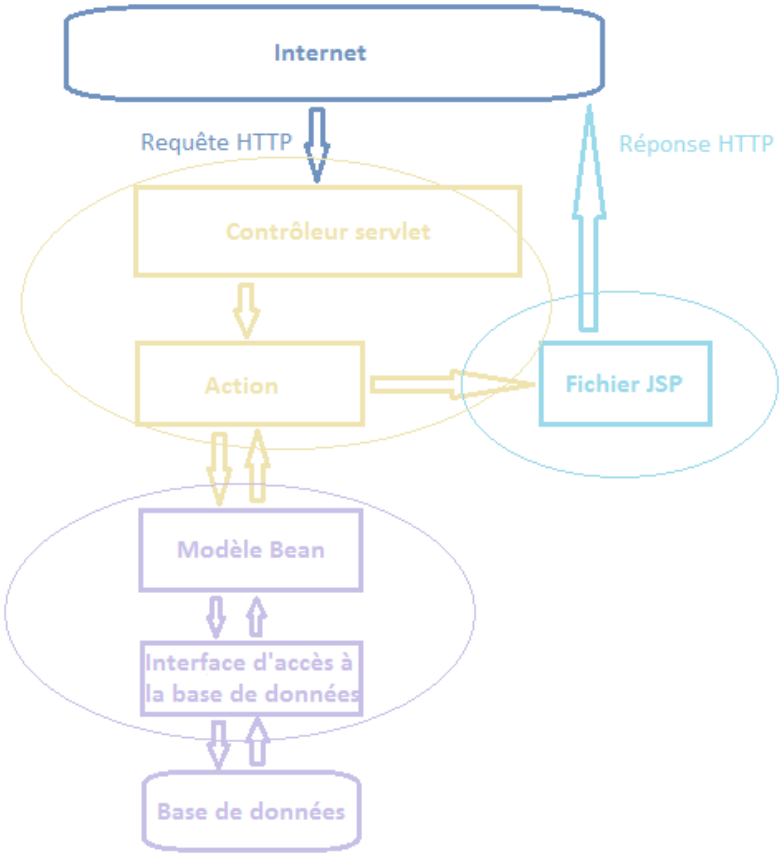


Figure 14 : Schéma MVC pour Struts™

9.3 Mapping relationnel-objet

Pour savoir au mieux comment le mapping relationnel-objet se met en place, il faut comprendre pourquoi nous en avons tant besoin. Les frameworks permettent de créer une couche d'abstraction qui encapsule la base de données relationnelle en générant des classes spécifiques pour donner l'illusion d'une base orientée objet au programmeur. Pour ce dernier, le travail sera plus aisé car il utilisera des méthodes qui simplifient énormément les interactions avec la base de données relationnelle et aura l'impression de manipuler des objets.

Par conséquent, une table de la base de données relationnelle correspondra à une classe (le framework aura généré automatiquement du code supplémentaire, lors de la génération des classes, qui permettra de faire facilement du CRUD à travers un simple formulaire). De plus, chaque enregistrement au sein de la base de données relationnelle sera perçu comme l'instance d'une classe (ou un objet).

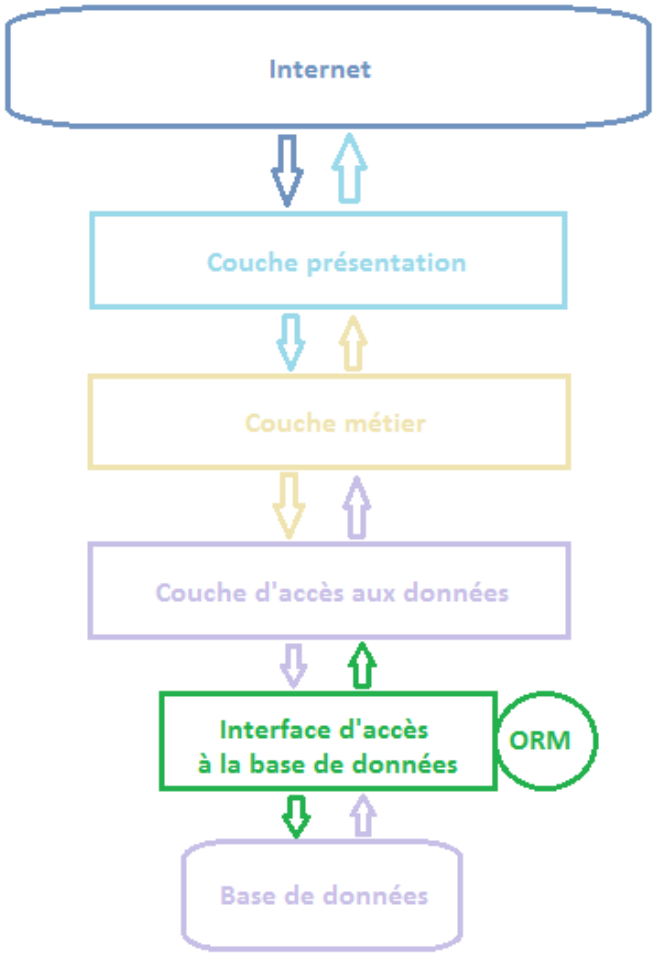


Figure 15 : Situation du mapping relationnel-objet au sein d'un modèle en couche

Outre le fait que le mapping relationnel-objet puisse être intégré par défaut au sein d'un framework choisi, il existe des ORMs qui permettent de passer du modèle relationnel directement aux classes de l'application. Il s'agit aussi de frameworks orientés mapping relationnel-objet. En Php, il existe le framework Doctrine™, Propel™ et en Java Hibernate™.

9.4 Avantages

Un framework a beaucoup de points positifs[8]. Toutefois s'il permet de simplifier de nombreux points, comme notamment l'interaction avec la base de données, le cœur de l'application reste à construire en code pur autour de ce que propose déjà le framework.

Les frameworks permettent donc de mettre en place plus facilement un projet en langage de programmation objet. Pour ceux qui débutent la programmation, les frameworks imposent une certaine architecture, une organisation globale d'application, des normes de développement et un cadre de travail. Tout cela permet de discipliner la personne qui développe. Et pour les plus chevronnés, un gain considérable de temps est obtenu. On peut aller au cœur du sujet plus rapidement, soit élaborer et personnaliser l'application plus facilement.

C'est un outil qui convient à toute personne élaborant des applications. La question est de savoir si on a réellement besoin d'un framework pour construire l'application pour laquelle nous sommes mandaté et si un CMS[9] n'est pas plus adapté à nos besoins.

Pour terminer, ce sont des outils open-source qui s'étoffent rapidement grâce à une grande communauté de développeurs. Par conséquent les développeurs détectent rapidement des anomalies de performance, de sécurité ou autre, ce qui garantit l'obtention d'une application de qualité si le code réalisé par la suite l'est également.

[8] Voir le sous-chapitre 9.1
[9] Voir le chapitre 11

9.5 Inconvénients

Premièrement un framework nous oriente dans la manière de développer notre application, nous impose des normes de développement. Entre deux frameworks, il peut y avoir des différences très importantes. Malheureusement, plus le framework propose des fonctionnalités ou composants de haut niveau, moins nous pouvons aller modifier ou personnaliser la structure de base de l'application. Effectivement si nous souhaitons plus de liberté, il faudrait créer un framework ayant les fonctionnalités souhaitées. Mais en contrepartie, cela nous oblige reprendre l'ensemble de la structure d'un framework.

Deuxièmement, il ne faut pas négliger le poids d'un framework. En effet, même s'il n'est composé que de simples fichiers, le framework occupe une place non négligeable sur le serveur et peut facilement atteindre un poids important si on ne fait pas le tri des fichiers obsolètes ou encore des classes générées par le framework et qui ne servent plus suite à une modification de la base de données.

Troisièmement, les mises à jour de framework ne sont pas toujours faciles à réaliser mais peuvent apporter beaucoup, notamment en termes de sécurité.

9.6 Analyse SWOT

Dans le but de synthétiser les forces et faiblesses des frameworks, nous allons faire une analyse (matrice) SWOT (respectivement Strengths ou Forces, Weaknesses ou Faiblesses, Opportunities ou Opportunités et Threats ou Menaces). Elle permettra de mettre en évidence une analyse interne (forces et faiblesses) et externe (opportunités et menaces) des frameworks.

	Positif	Négatif
Origine interne	- Gain de temps - Structure imposée - Maintenance facilitée - Normes de travail à respecter - Répartition du travail améliorée - Personnalisation des composants - Communautés réactives - Solutions non payantes	- Apprentissage parfois difficile - Mises à jour non évidentes - Poids du framework important - Structure imposée - La sécurité des frameworks
Origine externe	- Internet est au cœur d'une majorité d'entreprises - Les entreprises ont besoin de développeurs pour leurs projets	- Les attaques de site Internet (injection SQL…)

Table 2 : Analyse SWOT des frameworks

9.7 Fiche n°2 : Qu'est-ce qu'un framework ?

1. Définition

Un Framework est un outil de haut niveau qui est composé d'une multitude de composants qui aide à l'élaboration d'une application le plus souvent (mais il existe aussi des frameworks qui ont d'autres finalités tel que le mapping relationnel-objet). Le framework possède une architecture qui lui est propre et qu'il faudra respecter tout le long du développement.

2. Architecture Modèle-Vue-Contrôleur

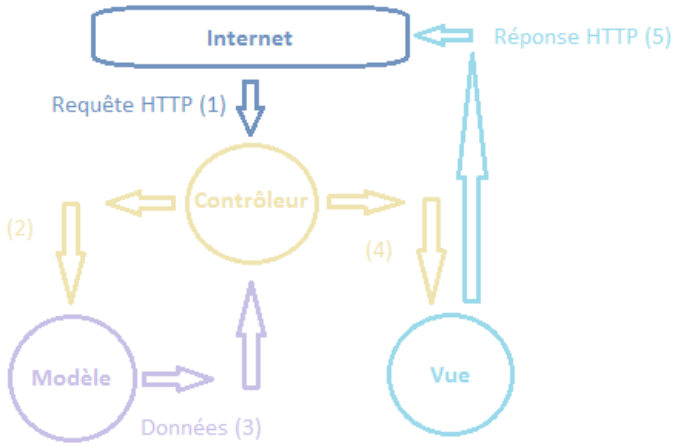


Figure 16 : Architecture Modèle-Vue-Contrôleur (fiche n°1)

3. Mapping relationnel-objet

Le mapping relationnel-objet permet de relier une base de données relationnelle avec les notions objets liées au langage de programmation objet choisi. Il intervient lors de la génération de classes par le framework.

4. Points positifs et négatifs

Points positifs : Gagner du temps, améliorer la maintenance, l'organisation de l'application, avoir des normes de développement, mieux répartir le travail.

Points négatifs : Temps d'apprentissage non négligeable, poids important, mise à jour plus ou moins difficile.

10. Analyse de frameworks Php et Java

Pour débuter nos démonstrations et l'analyse des différents frameworks, nous allons réaliser une application d'exemple relativement simple au niveau de la base de données et des fonctionnalités. Cette analyse a pour but de comparer Symfony™ et Zend™ framework ainsi que de vous familiariser avec les applications web que l'on peut réaliser en Java avec le framework Struts™.

Nous allons directement débuter cette analyse en présentant la structure du projet avec les différents modèles et le script de base de données MySQL™ (fichiers que vous pourrez également télécharger directement sur le site Internet relatif au travail de diplôme : **http://www.ariellemoro.com/TDFrameworks**).

Il est important de préciser que les applications réalisées sont axées « fonctionnalités » et non « design ». Par conséquent, il n'y aura aucune feuille de style CSS. Le design de celles-ci ne sera donc pas élaboré.

10.1 Structure du projet

Pour débuter le projet en bonne et due forme, voici différents modèles qui vous permettront de comprendre le contexte de l'application. Celle-ci permettra de se connecter à une plateforme, grâce à une interface de connexion et de lister les commandes de tous les clients (en mode administrateur) ou du client connecté (en mode utilisateur).

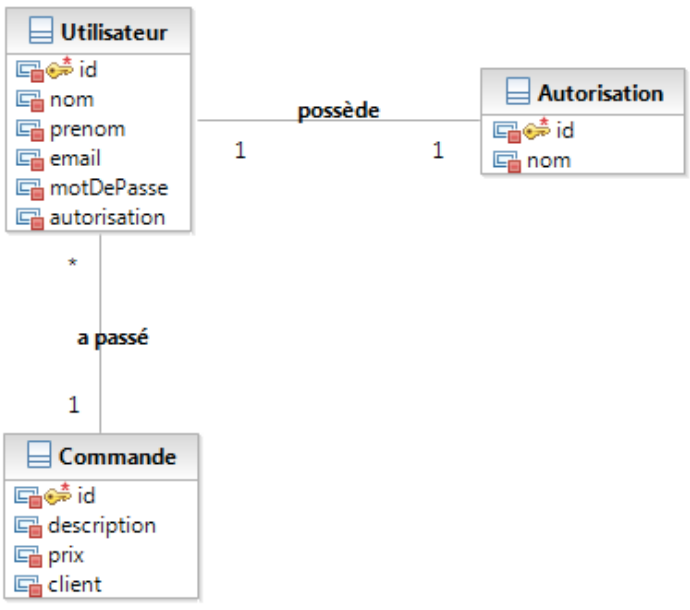


Figure 17 : Modèle conceptuel de données (MCD) réalisé avec Rational Software Architect™ d'IBM™

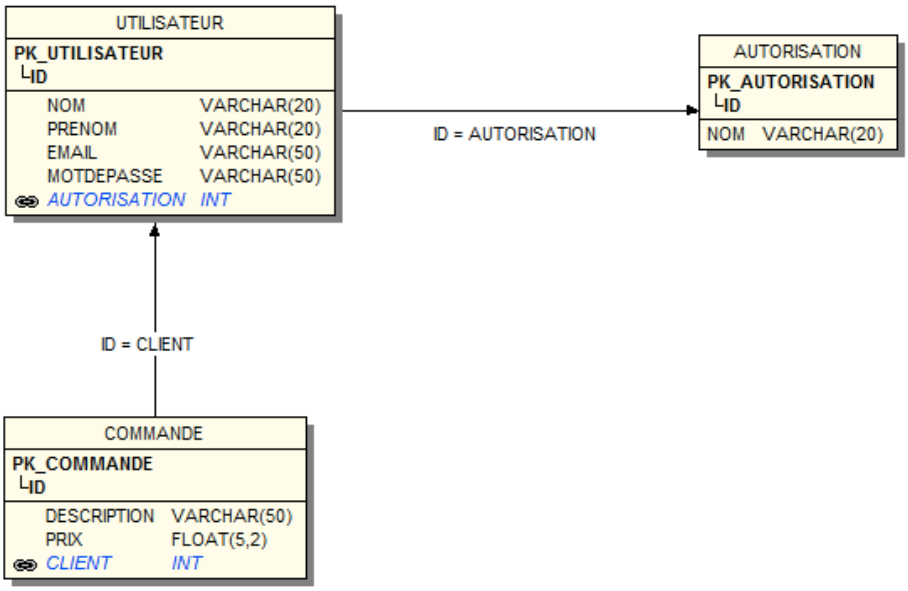


Figure 18 : Modèle logique de données (MLD) réalisé avec Win'Design™

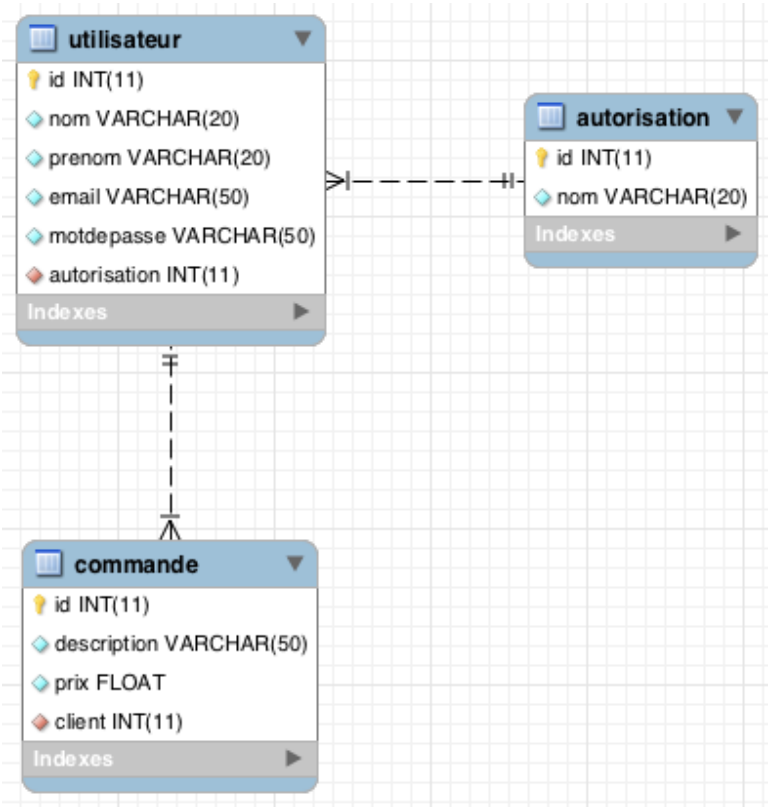


Figure 19 : Modèle relationnel de données réalisé avec MySQL™ Workbench

Script de base de données et jeu de test pour MySQL™ :

La base de données se nomme **gestClients** et possède trois tables : **autorisation**, **commande** et **utilisateur**.

Petites indications utiles :

- n'oubliez pas de choisir le mode **utf8_unicode_ci** pour la base de données ainsi que pour les champs composés de caractères pour éviter tout problème avec ces derniers ;
- pour que les contraintes de clés étrangères soient correctement prises en compte par MySQL et par le framework, vous devez précisez un moteur spécifique pour chaque table : **InnoDB** ;
- pour terminer, pour éviter tout problème lors de la génération de classes à partir de la base de données, je vous conseille d'écrire le nom des tables et des attributs en **minuscule**.

Installation :

- connectez-vous sur **PhpMyAdmin** et ne créez pas de nouvelle base de données ;
- cliquez sur l'onglet **SQL** puis faites un copier-coller de votre script sql et exécutez-le ;
- si tout s'est bien déroulé, vous découvrirez votre nouvelle base de données **gestClients** dans le menu de gauche.

```
-- PhpMyAdmin SQL Dump
-- version 3.2.5
-- http://www.Phpmyadmin.net
--
-- Serveur: localhost
-- Généré le : Mar 27 Juillet 2010 à 20:40
-- Version du serveur: 5.1.44
-- Version de PHP: 5.2.13

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Base de données: `gestClients`
--
CREATE DATABASE `gestClients` DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
USE `gestClients`;
```

```
-- --------------------------------------------------------

--
-- Structure de la table `autorisation`
--

DROP TABLE IF EXISTS `autorisation`;
CREATE TABLE IF NOT EXISTS `autorisation` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nom` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=3 ;

--
-- Contenu de la table `autorisation`
--

INSERT INTO `autorisation` VALUES(1, 'Utilisateur');
INSERT INTO `autorisation` VALUES(2, 'Administrateur');

-- --------------------------------------------------------

--
-- Structure de la table `utilisateur`
--

DROP TABLE IF EXISTS `utilisateur`;
CREATE TABLE IF NOT EXISTS `utilisateur` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nom` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
 `prenom` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
 `email` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
 `motdepasse` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
 `autorisation` int(11) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `utilisateur_ibfk_1` (`autorisation`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=4 ;

--
-- Contenu de la table `utilisateur`
--

INSERT INTO `utilisateur` VALUES(1, 'Moro', 'Arielle', 'a.m@swisscom.ch',
'f85938cd09cb1defec291aa2fb87a8337b9e60e8', 2);
INSERT INTO `utilisateur` VALUES(2, 'Dupont', 'Pierre', 'dupont.p@swisscom.ch',
'08af0b58293949d29462815ff61735c527642808', 1);
```

```
INSERT INTO `utilisateur` VALUES(3, 'Paul', 'Jean', 'p.jean@orange.ch', '144320d3b3418778de67e09b9428bf412290fc96', 1);


-- --------------------------------------------------------

--
-- Structure de la table `commande`
--

DROP TABLE IF EXISTS `commande`;
CREATE TABLE IF NOT EXISTS `commande` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `description` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
 `prix` float NOT NULL,
 `client` int(11) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `commande_ibfk_1` (`client`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=4 ;

--
-- Contenu de la table `commande`
--

INSERT INTO `commande` VALUES(1, 'Apple G5 + Apple Display 24"', 4000, 2);
INSERT INTO `commande` VALUES(2, 'iPhone 4G', 700, 2);
INSERT INTO `commande` VALUES(3, 'Apple MacBook Pro 17"', 2900, 3);

--
-- Contraintes pour les tables exportées
--

--
-- Contraintes pour la table `utilisateur`
--
ALTER TABLE `utilisateur`
 ADD CONSTRAINT `utilisateur_ibfk_1` FOREIGN KEY (`autorisation`) REFERENCES `autorisation` (`id`);

--
-- Contraintes pour la table `commande`
--
ALTER TABLE `commande`
 ADD CONSTRAINT `commande_ibfk_1` FOREIGN KEY (`client`) REFERENCES `utilisateur` (`id`);
```

10.2 Frameworks Php

10.2.1 Pré-requis

Avant toute chose, pour installer au mieux le projet, nous avons besoin de quelques logiciels qui vont nous être très utiles lors du développement. Voici notre boîte à outils :

- un logiciel qui possède un serveur **Apache™**, MySQL™ et PHP5 (WAMP™ sur pc et MAMP™ sur mac) ;
- un éditeur de scripts (EditPlus™ sur pc et Smultron™ sur mac) ou un IDE qui vous permettra de réaliser vos scripts avec beaucoup plus d'aisance (NetBeans™ sur pc et mac) ;
- un framework Php à placer dans le dossier de vos sites Internet (vous pouvez normalement modifier cet emplacement dans les paramètres de MAMP™ ou WAMP™).

Après avoir téléchargé le tout, commencez par installer MAMP™ ou WAMP™ et personnalisez les paramètres souhaités (dossier de sites, ports). Puis installez directement votre base de données dans PhpMyAdmin™ (script SQL se trouvant dans le sous-chapitre 10.1). Ensuite finalisez le tout en installant votre éditeur de scripts ou IDE.

A présent, nous allons nous concentrer sur la création de votre projet en fonction du framework choisi (création qui se déroulera bien évidemment dans le dossier de sites que vous aurez choisi).

Attention, si vous utilisez un IDE, lorsque vous aurez créé votre projet, importez votre dossier dans votre IDE NetBeans™ en sélectionnant « Php Application with Existing Sources ».

10.2.2 Symfony™

10.2.2.1 Création du projet

Premièrement, téléchargez la dernière version de Symfony™ (**1.4.6**) sur le site **http://www.symfony-project.org/installation**.

Commencez par créer un dossier qui aura le nom de votre projet au sein du dossier qui contient vos sites et qui a été indiqué dans WAMP™ ou MAMP™ (pour la démonstration, le nom du dossier du projet sera **GestClientsSF** et le nom du dossier comportant les sites s'intitule **htdocs**).

Créez ensuite un dossier au sein de **GestClientsSF** nommé **lib** et un autre, à l'intérieur de ce dernier, **vendor** (dossier qui comportera l'entièreté de la librairie du framework Symfony™).

Puis décompressez l'archive de Symfony™ au sein de ce dossier **vendor** et renommez cette archive (dans la démonstration, il s'agira du nom **symfony**).

A présent, testez si tout fonctionne correctement au sein d'un navigateur web : **http://localhost:8888/GestClientsSF/lib/vendor/symfony/data/bin/check_configuration.Php**

Si tout fonctionne correctement, vous devriez obtenir ceci :

```
********************************
*                              *
*  symfony requirements check  *
*                              *
********************************

php.ini used by PHP: /Applications/MAMP/conf/php5.3/php.ini


** Mandatory requirements **

  OK        PHP version is at least 5.2.4 (5.3.2)

** Optional checks **

  OK        PDO is installed
  OK        PDO has some drivers installed: sqlite, sqlite2, pgsql, mysql
  OK        PHP-XML module is installed
  OK        XSL module is installed
  OK        The token_get_all() function is available
  OK        The mb_strlen() function is available
  OK        The iconv() function is available
  OK        The utf8_decode() is available
  OK        The posix_isatty() is available
  OK        A PHP accelerator is installed
  OK        php.ini has short_open_tag set to off
  OK        php.ini has magic_quotes_gpc set to off
  OK        php.ini has register_globals set to off
  OK        php.ini has session.auto_start set to off
  OK        PHP version is not 5.2.9
```

Pour créer concrètement le projet, nous allons ouvrir une **fenêtre DOS** ou un **terminal** et, à la racine de **GestClientsSF**, nous allons exécuter les lignes de commande suivantes :

```
cd /Applications/MAMP/htdocs/GestClientsSF
```

```
php lib/vendor/symfony/data/bin/symfony generate:project GestClientsSF
```

Après la création du projet, vous allez constater que de nombreux dossiers ont été créés :

- **apps** : les applications relatives au projet
- **cache** : les fichiers du cache
- **config** : les fichiers de configuration du projet
- **data** : les fichiers de données importantes du projet (script SQL par exemple)
- **lib** : les librairies du projet
- **log** : les fichiers de log du framework
- **plugins** : les plugins installés pour le projet
- **test** : les tests unitaires et fonctionnels du projet
- **web** : le répertoire web du projet

Nous allons ensuite créer, en ligne de commande, la structure de dossiers qui nous permettra de créer notre application :

```
php symfony generate:app --escaping-strategy=on --csrf-secret=UniqueSecret frontend
```

Suite à cette création, vous pourrez tout de suite constater que dans **/GestClientsSF/apps/frontend** vous avez les nouveaux dossiers suivants :

- **config** : les fichiers de configuration de l'application
- **i18n** : les fichiers de langues de l'application
- **lib** : les librairies de l'application
- **modules** : le code de l'application
- **templates** : les templates de l'application

Pour terminer la création de ce projet et pour en effectuer la vérification, nous allons modifier le fichier **httpd.conf** du serveur Apache™ en rajoutant les lignes suivantes à la fin :

```
### Section 3: Virtual Hosts
# …
# NameVirtualHost *

# Soyez sûr d'avoir seulement cette ligne une fois dans votre configuration
NameVirtualHost 127.0.0.1:8888

# C'est la configuration pour votre projet
Listen 127.0.0.1:8888

<VirtualHost 127.0.0.1:8888>
  DocumentRoot "/Applications/MAMP/htdocs/GestClientsSF/web"
  DirectoryIndex index.Php
  <Directory "/Applications/MAMP/htdocs/GestClientsSF/web">

 AllowOverride All
    Allow from All
  </Directory>

  Alias                                                                    /sf
/Applications/MAMP/htdocs/GestClientsSF/lib/vendor/symfony/data/web/sf
  <Directory
"/Applications/MAMP/htdocs/GestClientsSF/lib/vendor/symfony/data/web/sf">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

Redémarrez WAMP™ ou MAMP™ et testez votre projet avec l'**URL** suivante : **http://127.0.0.1:8888** ou encore **http://127.0.0.1:8888/frontend_dev.Php** (interface de développement).

10.2.2.2 Génération automatique des classes avec Doctrine™

Symfony™ dispose de deux ORMs Doctrine™ et Propel™ pré-installés. Lors de cette démonstration, nous utiliserons l'ORM Doctrine™.

Tout d'abord, nous allons configurer la base de données au sein de notre projet avec ces lignes de commande :

```
php symfony configure:database --name=doctrine --class=sfDoctrineDatabase "mysql:host=localhost;dbname=gestClients" root root
```

Ensuite, il suffit de vérifier que Doctrine™ est correctement activé pour Symfony™ en consultant le fichier **/config/ProjectConfiguration.class.Php** (le plugin **sfDoctrinePlugin** doit être *enable*).

Puis, par sécurité, videz le cache avec cette ligne de commande :

```
php symfony cache:clear
```

Créez maintenant le fichier en format **YAML** de la base de données à l'aide de cette ligne de commande (à la suite de l'exécution de cette ligne de commande, le fichier **/config/doctrine/schema.yml** sera mis à jour):

```
php symfony doctrine:build-schema
```

 Attention ! Si vous rencontrez un problème lié à la socket mysql, il suffit d'éditer le fichier **/etc/Php.ini.default** (et de le renommer ensuite en **Php.ini**) et de rajouter ou compléter la ligne suivante :

```
pdo_mysql.default_socket = /Applications/MAMP/tmp/mysql/mysql.sock
```

A cette étape nous allons générer les classes Php relatives à la base de données qui correspondent au schéma YAML créé précédemment avec la ligne de commande suivante (attention cette manipulation va effacer les données de test, il faudra les réintroduire ensuite dans la base de données) :

```
php symfony doctrine:build-model
```

Toutes les classes ont maintenant été générées et se trouvent au sein du dossier **/lib/model/doctrine**

Ensuite, pour avoir les formulaires de base qui vont nous permettre de faire le CRUD pour chacune des tables, nous allons exécuter les lignes de commande suivantes dans le but de créer des modules (module avec actions complètes de CRUD pour un modèle donné) fonctionnels :

```
php symfony doctrine:generate-module --with-show --non-verbose-templates frontend autorisation Autorisation
php symfony doctrine:generate-module --with-show --non-verbose-templates frontend commande Commande
```

```
php symfony doctrine:generate-module --with-show --non-verbose-templates frontend utilisateur Utilisateur
```

Au sein de chaque module, vous pouvez maintenant remarquer deux dossiers :

- **actions** : les différentes actions d'un module
- **templates** : les différents gabarits d'un module

Pour chaque module, le CRUD de base a déjà été généré.

10.2.2.3 Mise en place de la structure du projet

Tout va débuter par la création d'un nouveau module **login** au sein de notre application **frontend** :
php symfony generate:module frontend login

Après la création de ce module, nous allons élaborer un formulaire de login **/lib/form/LoginForm.class.php** :

```
<?php

class LoginForm extends sfForm
{
  public function configure()
  {
    $this->setWidgets(array(
      'login'    => new sfWidgetFormInput(),
      'motDePasse'   => new sfWidgetFormInput(),
    ));
  }
}

?>
```

Si vous souhaitez personnaliser vos champs, n'hésitez pas à consulter la documentation sur les APIs de Symfony™ sur le site Internet suivant : **http://www.symfony-project.org/api/1_4/**.

Puis nous allons le relier à notre module **login** en éditant le fichier **/apps/frontend/modules/login/actions/actions.class.php** :

```
class loginActions extends sfActions
{

  public function executeIndex()
  {
    $this->form = new LoginForm();
  }

}
```

Et pour terminer, il ne nous reste plus qu'à afficher le formulaire en personnalisant le template relié à notre module login **/apps/frontend/modules/login/templates/indexSuccess.php** (vous pourrez ensuite tester le tout avec l'URL suivante **http://127.0.0.1:8888/frontend_dev.php/login**) :

```
<form action="<?php echo url_for('login/submit') ?>" method="POST">
 <h1>Connexion à GestClientsSF</h1>
 <table>
  <?php echo $form ?>
  <tr>
   <td colspan="2">
    <input type="submit" />
   </td>
  </tr>
 </table>
</form>
```

Ensuite, pour que le submit soit correctement effecuté il vous suffit de créer une méthode **executeSubmit()** au sein du **/apps/frontend/modules/login/actions/actions.class.php** ou de modifier avec soin la méthode **executeIndex()** (méthode choisie pour la version finale de la démonstration).

Dans le but d'étoffer l'application, il conviendra de compléter les différents scripts dans le but de définir les fonctionnalités que doit implémenter cette application de démonstration.

Pour la suite du développement de ce premier projet avec Symfony™, nous vous suggérons d'examiner de plus près les scripts de l'application de démonstration réalisée et fournie avec ce travail de diplôme au sein du CDRom ou alors directement sur le site Internet relatif au mémoire : **http://www.ariellemoro.com/TDFrameworks**.

10.2.3 Zend™ framework

10.2.3.1 Création du projet

Pour débuter notre projet Zend™, nous allons tout d'abord télécharger le framework avec le lien suivant : **http://framework.zend.com/download/latest** (la version, avec laquelle nous allons faire la démonstration, est la suivante : **ZendFramework-1.10.7 (full)**).

Puis décompressez l'archive téléchargée correspondant au framework, placez-la dans le dossier (**htdocs** pour la démonstration) qui contient tous vos sites et renommez-le comme vous le souhaitez (pour la démonstration, nous le renommerons **ZendFramework**) .

Création formelle de notre premier projet **GestClientsZF**:

Ouvrez maintenant une fenêtre DOS ou un terminal (car nous allons créer le projet directement à l'aide de quelques lignes de commande) :

```
alias zf="/Applications/MAMP/htdocs/ZendFramework/bin/zf.sh"
cd /Applications/MAMP/htdocs
zf create project ./GestClientsZF GestClientsZF
```

Cette création de projet est rendue possible en ligne de commande grâce à l'outil **Zend_Tool** (particularité de Zend™). Il est donc important de créer un alias avant d'utiliser la commande **zf**.

A présent, vous devez obtenir l'architecture de dossiers et fichiers suivante au sein de votre projet :

- **application**
  - **Bootstrap.php**
  - **configs**
    - **application.ini**
  - **controllers**
    - **ErrorController.php**
    - **IndexController.php**
  - **models**

- views
  - helpers
  - scripts
    - error
      - error.phtml
    - index
      - index.phtml
- docs
- library
- public
  - index.php
- tests
  - application
    - bootstrap.php
  - library
    - bootstrap.php
  - phpunit.xml

Ensuite copiez le dossier **Zend** (qui se trouve au chemin suivant **/htdocs/ZendFramework/library**) et le coller dans le dossier du projet **/GestClientsZF/library**.

Pour vérifier que le projet fonctionne correctement, nous allons ouvrir un navigateur web et taper l'URL qui vous permet de tester les sites (dans notre cas c'est MAMP™ qui interprète toutes nos pages Php car il possède un serveur Apache™) : **http://localhost:8888/GestClientsZF/public/index.Php**.

Voici le résultat obtenu :

10.2.3.2 Génération automatique des classes avec Doctrine™

Après avoir créé notre projet, nous arrivons à une seconde étape importante : la génération des classes depuis le framework. Cependant Zend™ framework ne possède aucun ORM embarqué. Par conséquent il faut donc télécharger un ORM d'une part (la démonstration se fera avec Doctrine™) et paramétrer ce dernier dans Zend™ framework pour pouvoir l'utiliser d'autre part.

Vous pouvez télécharger Doctrine™ directement sur ce site **http://www.doctrine-project.org/projects/orm/download** (la démonstration se fera avec la version **1.2.2**).

- **Première étape :**

Nous allons créer deux fichiers qui vont nous servir à utiliser Doctrine™ pour générer les modèles (ou models) liés à notre base de données MySQL™.

- **doctrine** (Fichier exécutable Unix) contenant :

```
#!/usr/bin/env php
<?php
chdir(dirname(__FILE__));
include('doctrine.php');
```

- **doctrine.php** avec le code suivant :

```
<?php

// Define path to application directory
defined('APPLICATION_PATH')
   || define('APPLICATION_PATH', realpath(dirname(__FILE__) . '/..'));

// Define application environment
defined('APPLICATION_ENV')
   ||        define('APPLICATION_ENV',        (getenv('APPLICATION_ENV')        ?
getenv('APPLICATION_ENV') : 'production'));

// Ensure library/ is on include_path
set_include_path(implode(PATH_SEPARATOR, array(
   realpath(APPLICATION_PATH . '/../library'),
   get_include_path(),
)));

/** Zend_Application */
require_once 'Zend/Application.php';
```

```
// Create application, bootstrap, and run
$application = new Zend_Application(
   APPLICATION_ENV,
   APPLICATION_PATH . '/configs/application.ini'
);

$application->getBootstrap()->bootstrap('doctrine');

$config = $application->getOption('doctrine');

$cli = new Doctrine_Cli($config);
$cli->run($_SERVER['argv']);
```

- **Deuxième étape :**

Après avoir créé ces deux fichiers, nous allons rajouter des dossiers et fichiers au sein de notre projet et plus particulièrement dans le dossier **/GestClientsZF/application** pour qu'il ait la structure ci-dessous :

- application
  - Bootstrap.php
  - configs
    - application.ini
    - data
      - fixtures
      - sql
    - migrations
    - schema.yml
  - controllers
    - ErrorController.php
    - IndexController.php
  - models
  - scripts
    - doctrine (Fichier exécutable Unix)
    - doctrine.php

- views
  - helpers
  - scripts
    - error
      - error.phtml
    - index
      - index.phtml

Comme vous pouvez le constater, au sein du dossier **scripts**, nous plaçons les fichiers **doctrine (Fichier exécutable Unix)** et **doctrine.php** qui ont été élaborés lors de la première étape.

- **Troisième étape :**

A présent, reprenons l'archive de Doctrine™, que vous avez téléchargée précédemment, et copiez les dossiers **Doctrine** et **vendor** et le fichier **Doctrine.php** contenus à cet endroit précis **/Doctrine-1.2.2/lib**. Puis collez-les au sein du dossier **/GestClientsZF/library** de votre framework.

Ces éléments vont nous permettre d'avoir la librairie nécessaire à la bonne utilisation de Doctrine™ dans le projet Zend™.

- **Quatrième étape :**

Pour cette avant-dernière étape, nous allons procéder à la modification de deux fichiers comme ci-dessous :

- **application.ini** (ce fichier se trouve dans **/GestClientsZF/application/configs** et il va nous permettre d'indiquer des paramètres qui nous seront utiles pour connecter Doctrine™ avec MySQL™) :

```
[production]
phpSettings.display_startup_errors = 0
phpSettings.display_errors = 0
phpSettings.date.timezone = "Europe/Geneva"
includePaths.library = APPLICATION_PATH "/../library"
bootstrap.path = APPLICATION_PATH "/Bootstrap.php"
bootstrap.class = "Bootstrap"
appnamespace = "Application"
```

```
resources.frontController.controllerDirectory = APPLICATION_PATH
"/controllers"
resources.frontController.params.displayExceptions = 0
autoloaderNamespaces[] = "Doctrine"
resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts/"
resources.view[] =

; ---
; Database
; ---

doctrine.dsn = "mysql://root:root@localhost/gestClients"
doctrine.data_fixtures_path = APPLICATION_PATH "/configs/data/fixtures"
doctrine.sql_path = APPLICATION_PATH
"/configs/data/sql"
doctrine.migrations_path = APPLICATION_PATH "/configs/migrations"
doctrine.yaml_schema_path = APPLICATION_PATH "/configs/schema.yml"
doctrine.models_path = APPLICATION_PATH "/models"

[staging : production]

[testing : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1

[development : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1
resources.frontController.params.displayExceptions = 1
```

- **Bootstrap.php** (ce fichier se trouve dans **/GestClientsZF/application** et la fonction créée dans ce fichier servira à créer l'objet client pour utiliser Doctrine™) :

```
<?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{

    protected function _initDoctype()
  {
     $this->bootstrap('view');
     $view = $this->getResource('view');
     $view->doctype('XHTML1_STRICT');
  }
```

```
    protected function _initDoctrine()
    {

            $this->getApplication()->getAutoloader()
                    ->pushAutoloader(array('Doctrine', 'autoload'));
            spl_autoload_register(array('Doctrine', 'modelsAutoload'));

            $manager = Doctrine_Manager::getInstance();
            $manager-
>setAttribute(Doctrine::ATTR_AUTO_ACCESSOR_OVERRIDE, true);
            $manager->setAttribute(
                    Doctrine::ATTR_MODEL_LOADING,
                    Doctrine::MODEL_LOADING_CONSERVATIVE
            );
            $manager-
>setAttribute(Doctrine::ATTR_AUTOLOAD_TABLE_CLASSES, true);

            $doctrineConfig = $this->getOption('doctrine');

            Doctrine::loadModels($doctrineConfig['models_path']);

            $conn       =       Doctrine_Manager::connection($doctrineConfig['dsn'],
'doctrine');
            $conn->setAttribute(Doctrine::ATTR_USE_NATIVE_ENUM, true);

            $conn->setCharset('utf8');
            $conn->setCollate('utf8_general_ci');

            return $conn;

    }

}
```

- **Cinquième étape :**

Pour finaliser cette génération de classes, nous allons ouvrir la fenêtre DOS ou le terminal et exécuter ces lignes de commande :

cd /Applications/MAMP/htdocs/GestClientsZF/application/scripts

./doctrine (pour voir les différentes lignes de commande disponibles)

./doctrine generate-yaml-models ou ./doctrine build-all-reload (si on dispose déjà du schema.yml complet)

10.2.3.3 Mise en place de la structure du projet

Pour débuter le projet, il faut mettre en place sa structure globale à l'aide de formulaires (forms), de contrôleurs (controllers) et de vues (views).

Nous allons tout d'abord activer les layouts (qui vont nous permettre de mettre en place le corps global des pages de l'application) avec cette ligne de commande (vous allez pouvoir constater qu'un nouveau dossier nommé **layouts** se trouve à présent dans le dossier **GestClientsZF/application**) :

```
zf enable layout
```

Après cette étape, il suffit de créer un formulaire de login ainsi qu'un contrôleur associé à ce formulaire avec les lignes de commande suivantes (un dossier **forms** (avec le fichier **Login.php**) sera créé dans le dossier **GestClientsZF/application** et un contrôleur **LoginController.php** ainsi qu'une vue (créée automatiquement avec le contrôleur) **GestClientsZF/application/views/scripts/login/index.phtml** seront créés) :

```
zf create form Login
```

```
zf create controller Login (l'action index sera créée de base)
```

Il faudra ensuite compléter le code de ces fichiers pour qu'ils soient appelés au bon moment (à la validation (**submit**) du formulaire de connexion).

A cette étape, votre login sera réalisé, cependant les listes de commandes relatives au mode administrateur ou au mode utilisateur ne le sont pas encore. Nous allons donc créer un contrôleur **Commande** et deux actions correspondantes **listcommandesadm** et **listcommandesuti** :

```
zf create controller Commande
```

```
zf create action listcommandesadm Commande
```

```
zf create action listcommandesuti Commande
```

Pour finaliser le tout, il suffit de compléter de la même façon les fichiers créés. Et de tester l'URL suivante : **http://localhost:8888/GestClientsZF/public/index.php/login**

Pour de plus amples informations concernant le code interne aux pages cruciales de cette application de test, je vous invite à télécharger les sources de l'application sur le site Internet suivant **http://www.ariellemoro.com/TDFrameworks** ou alors sur le CDRom relatifs à mon travail de diplôme.

10.2.4 Analyse comparative

10.2.4.1 Démarche

Les différents points de comparaison qui apparaissent dans l'analyse ont été sélectionnés lors de l'installation et la manipulation des deux frameworks, dans les sous-chapitres précédents, pour réaliser les applications de démonstration.

Tout d'abord, nous allons détailler chaque point l'un après l'autre, puis nous terminerons par un tableau récapitulatif.

10.2.4.2 Analyse détaillée

- **Prise en main du framework :**

L'architecture des dossiers est très différente entre Symfony™ et Zend™ framework mais on peut se familiariser rapidement avec l'un et l'autre. Comme nous pouvons le constater très vite, Zend™ framework regroupe tous les éléments d'une application dans le dossier **application** (y compris les fichiers de configuration) tandis que Symfony™ installe la majeure partie des fichiers cruciaux pour le projet directement à la racine de ce dernier.

- **Installation :**

Symfony™ et Zend™ framework sont tous deux relativement facile à appréhender. Si nous suivons scrupuleusement les étapes d'un bon tutoriel, nous arrivons à mettre en place rapidement la structure de notre premier projet. La différence majeure que l'on constate entre les deux frameworks est que Symfony™ possède un ORM embarqué (Doctrine™) alors que Zend™ framework n'en propose malheureusement pas. De plus, pour Zend™ framework, l'installation de Doctrine™ est complexe.

- **Routing :**

Le routing permet de définir des routes, des chemins particuliers au sein de l'application. Par conséquent, en fonction d'une action particulière, l'application dirigera automatiquement l'utilisateur vers la bonne page (définie dans le fichier de routing). Le fichier **routing.yml** permet de définir les routes dans Symfony™ (et notamment la page principale). Tandis qu'au sein de Zend™ framework, il faudra définir un contrôleur qui fera office de routeur. De manière générale, les fichiers de routing comportent des balises comme un fichier XML.

- **Mapping relationnel-objet (ORM) :**

Comme nous l'avons mis en évidence ci-dessus, Doctrine™ est déjà installé dans Symfony™. Par conséquent, si l'on souhaite utiliser Doctrine™ avec Zend™ framework, il faudra procéder à une installation minutieuse. Ce qui donne, en définitive, un avantage majeur à Symfony™.

- **Mise en place de modules :**

Au sein de Symfony™, Doctrine™ et la création de modules vont de pair. Car, en créant un module lié à un modèle correspondant à une table de base de données (généré par Doctrine™), cela permet d'obtenir les **actions** et **templates** (soit contrôleur et vues) nécessaire au CRUD de base. A contrario, Zend™ framework ne le propose pas (ou alors propose un simple formulaire pour introduire des données) et il faudra développer le CRUD nous-mêmes.

Pour terminer, il est tout à fait possible de créer des modules au sein de Symfony™ et de Zend™ framework, ce qui permet de hiérarchiser avec précision l'application.

- **Formulaires :**

Les formulaires peuvent être réalisés avec Symfony™ et Zend™ framework. Nous pouvons facilement les personnaliser avec des composants utiles (champs e-mail, captcha…). De plus, nous pouvons ajouter des validateurs et des erreurs personnalisées.

- **Frontend et backend :**

Symfony™ et Zend™ framework vont avoir des manières tout à fait différentes de gérer un frontend ou un backend. Dans Symfony™, nous allons créer des applications spécifiques frontend et backend qui vont avoir des options particulières (de générer des CRUD différents en fonction du frontend ou du backend : le backend aura des formulaires plus élaborés par exemple). Au sein de Zend™ framework, nous devrons mettre en place des modules frontend et backend.

- **Sessions :**

Les deux frameworks proposent des solutions pour simplifier les sessions. Cependant nous pouvons créer des utilisateurs qui sont spécifiés en dur dans un fichier.

- **Organisation du code :**

L'organisation du code est correcte dans chacun des frameworks. Ce qui structure le tout est le Design Pattern MVC qui permet de laisser les fonctions méthodes ou les tests spécifiques (avec leurs actions) au sein des contrôleurs. Et les vues restent de la présentation pure.

- **Interactions avec la base de données :**

Dans le cas où nous ne choisissons pas un ORM avec Zend™ framework, les interactions avec la base de données peuvent se faire très facilement à l'aide de la classe **Zend_Db**. Concernant Symfony™, nous allons utiliser par défaut Doctrine™ ou Propel™ à l'aide de la classe **Doctrine_Query** (nous utilisons d'ailleurs la même si on intègre Doctrine™ à Zend™ framework).

- **Documentation concernant les APIs :**

La qualité de la documentation concernant les APIs est primordiale. Elle décrit normalement les différents attributs ou méthodes (comportement(s), paramètre(s) éventuel(s)) se trouvant déjà dans le framework (au sein des librairies). Celles de Symfony™ et de Zend™ framework sont de qualité et il y a parfois quelques bons exemples d'usage.

- **Outils de développement :**

Symfony™ possède, contrairement à Zend™ framework, une interface de développement liée au frontend ou backend. Cette interface permet de voir le temps réel nécessaire à l'exécution d'une requête ainsi que sa description. Ces indicateurs sont pertinents si non souhaitons avoir une application performante et de qualité.

- **Feuilles de style (CSS) :**

Les feuilles de style permettent de présenter de façon agréable l'application et de concevoir un design qui pourra être commun à toute l'application. Symfony™ permet de définir une feuille de style globale et d'autres par module, ce qui permet de garder une présentation intègre sur toutes les pages de l'application mais d'en personnaliser quelques-unes qui comportent plus d'éléments à afficher. Dans Zend™ framework, nous pouvons définir une feuille de style globale et nous pouvons associer des feuilles de style aux vues.

- **Présence de composants élaborés :**

Avec Zend™ framework, il existe de nombreux composants, que l'on peut intégrer, tels que des APIs Google ou encore Adobe PDF. Du côté de Symfony™ on peut trouver des plugins similaires qui sont également à incorporer au sein du projet.

10.2.4.3 Tableau récapitulatif

Le tableau ci-dessous présente une synthèse de l'analyse détaillée. Chaque framework est noté sur 5 pour chacun des critères mentionnés et obtient une note finale (également sur 5).

Critères / Frameworks	Symfony™	Zend™
Prise en main du framework	5/5	4/5
Installation	5/5	2/5
Routing	5/5	5/5
Mapping relationnel-objet	5/5	3/5
Mise en place de modules	5/5	4/5
Formulaires	5/5	5/5
Frontend et backend	5/5	4/5
Sessions	3/5	3/5
Organisation du code	5/5	5/5
Interactions avec la base de données	5/5	5/5
Documentation sur les APIs	4/5	4/5
Outils de développement	5/5	3/5
Feuilles de style CSS	5/5	5/5
Composants élaborés	5/5	5/5
Note finale	**4.8/5**	**4.1/5**

Table 3 : Tableau récapitulatif Symfony™ et Zend™

Comme vous pouvez le constater, Symfony™ arrive en tête mais Zend™ framework n'est pas très loin derrière. Ils sont tout deux de riches et bons frameworks. Le tout est de bien choisir votre framework tout en prenant en compte certains critères.

10.3 Framework Java

10.3.1 Pré-requis

Pour mettre en place le framework Java Struts™, notre boîte à outil sera composée des éléments suivants :

- un logiciel comportant un serveur Apache™, PHP5 et MySQL™ (MAMP™ ou WAMP™ ou encore EasyPhp™ selon votre système d'exploitation),
- un IDE (Netbeans™ 6.8 sera utilisé pour la démonstration) ;
- le serveur Tomcat 6.0 qui servira à interpréter les fichiers JSP (téléchargez les fichiers sur le site suivant : **http://tomcat.apache.org/**) ;
- les plugins Struts 2 pour Netbeans™ (téléchargez les deux fichiers sur le site suivant : **https://nbstruts2support.dev.java.net/servlets/ProjectDocumentList?folderID=9422&expandFolder=9422&folderID=11970**).

10.3.2 Struts™

10.3.2.1 Création du projet

- **Première étape :**

Tout d'abord, procédez à l'installation de Netbeans™. Ensuite démarrez-le dans le but de mettre en place le serveur Tomcat 6.0 ainsi que les plugins Struts 2 au sein de l'IDE.

Serveur Tomcat 6.0 :

Cliquez dans le menu sur **Outils** → **Serveurs**. Dans la fenêtre **Serveurs**, ajouter un nouveau serveur en n'oubliant pas de spécifier son type (soit Tomcat 6.0) et le dossier où se trouve l'archive que vous aurez téléchargé et décompressé au sein du répertoire racine de Netbeans™ (**/Applications/NetBeans/sges-v3/apache-tomcat-6.0.29**).

A la fin, vous obtiendrez une configuration de ce type :

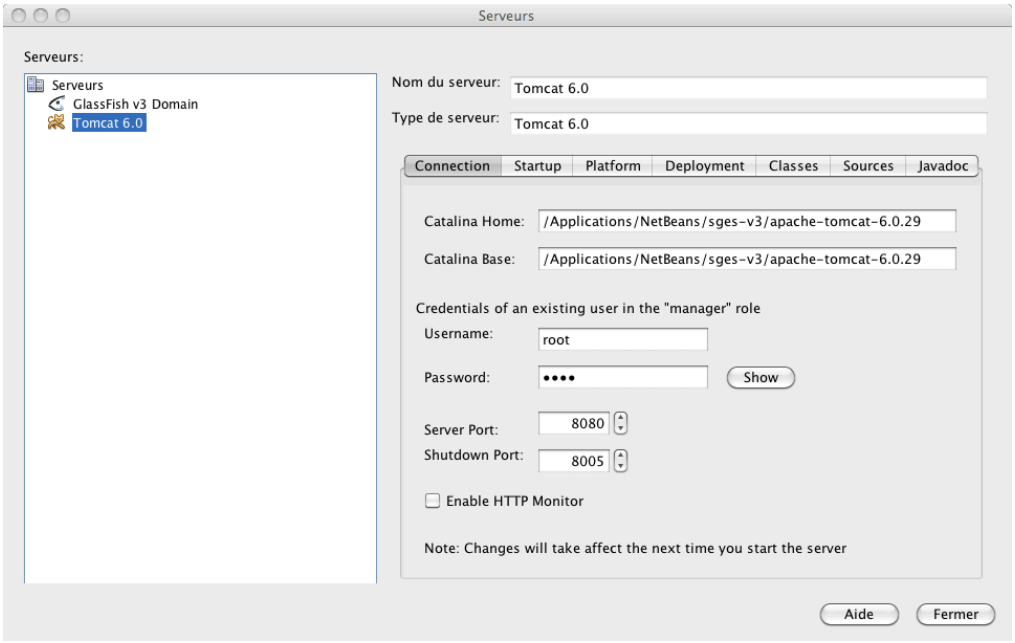


Figure 20 : Configuration de Tomcat 6.0 dans Netbeans™

Plugins Struts 2 :

Cliquez dans le menu sur **Outils → Plug-ins**. Puis ajoutez les plugins que vous avez téléchargés par le biais de l'onglet **Téléchargés**.

Pour terminer, voici ce que vous devriez obtenir dans l'onglet **Installés** :

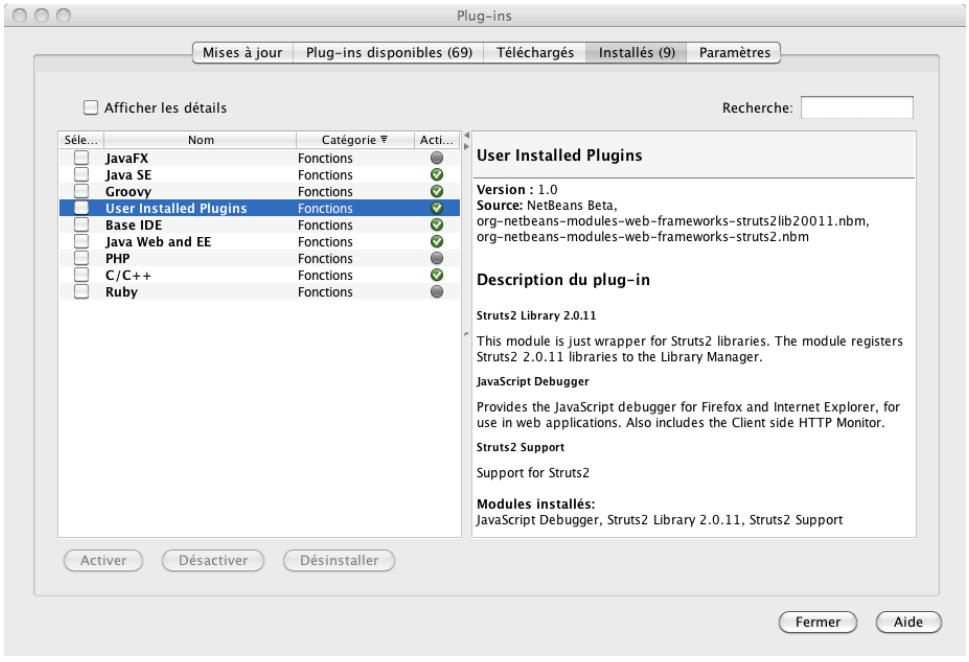


Figure 21 : Plugins installés dans Netbeans™

- **Deuxième étape :**

A présent, passons à l'étape de création du projet Netbeans™ (projet que vous retrouverez dans le dossier prévu à cet effet **/Users/ariellemoro/NetBeansProjects**).

Cliquez dans le menu sur **Fichier → Nouveau projet…** Ensuite, sélectionnez un projet de catégorie **Java Web** et de type de projet **Java Application**. Précisez le nom du projet et son emplacement. Puis sélectionnez le serveur Tomcat 6.0 que vous avez créé précédemment ainsi que les frameworks suivants : Struts 2 (qui correspond au plugin installé à l'étape précédente) et Hibernate 3.2.5 (sans oublier de créer une nouvelle **Database Connection** de type mysql).

Si votre projet a été correctement mis en place, en l'exécutant vous devriez obtenir une page d'exemple de ce type dans votre navigateur web (si cela n'était pas le cas, consultez les logs au sein des fenêtres de sortie) :

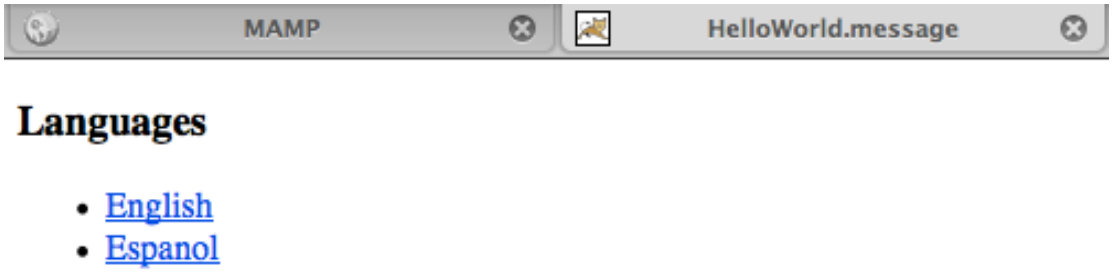


Figure 22 : Page de démarrage du projet avec Struts 2

Voici comment se déroulent les requêtes avec un serveur Tomcat™ :

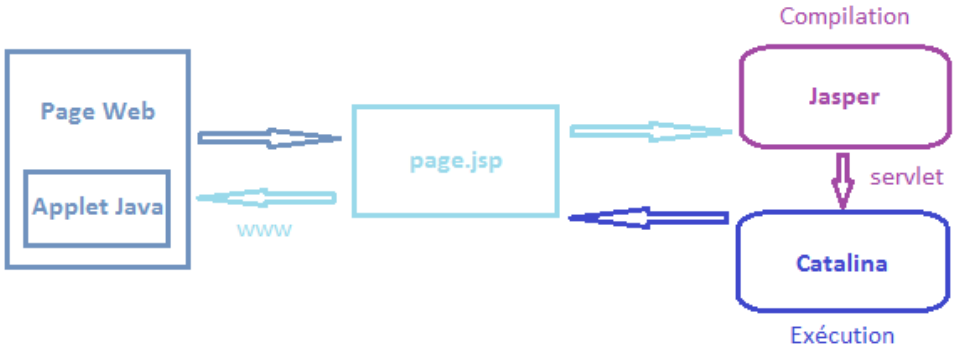


Figure 23 : Les requêtes avec Tomcat™

- **Troisième étape :**

Rajoutons deux packages au sein du dossier **Source Packages** : **controller** et **model**. En ce qui concerne les vues ou les fameuses JSP, elles se trouveront dans le dossier **Web Pages**.

10.3.2.2 Génération automatique des classes avec Hibernate™

- **Première étape :**

Nous allons en premier lieu rajouter dans le fichier **/Sources Packages/hibernate.cfg.xml** une information concernant le password pour accéder à la base de données MySQL™ ainsi que la classe qui définit le contexte pour les sessions Hibernate™ :

```
<property name="hibernate.connection.password">root</property>
<property name="hibernate.current_session_context_class">thread</property>
```

En second lieu, nous procéderons à la création d'un fichier **/Sources Packages/hibernate.reveng.xml** en effectuant un clique droit sur le dossier **Sources Packages** et en sélectionnant un nouveau fichier de type **Hibernate Reverse Engineering Wizard…** Il vous suffira de sélectionner les bonnes tables de votre base de données lors de la création de ce fichier.

- **Deuxième étape :**

Nous allons à présent créer les classes Java ainsi que les fichiers de mapping associés. En cliquant droit sur le package **model**, créez un nouveau « fichier » **Hibernate Mapping Files and POJOs from Database…** (cela va créer plusieurs fichiers en fonction du nombre de tables que vous avez indiqué dans le fichier **/Sources Packages/hibernate.reveng.xml**).

N'oubliez pas de cocher la case « **JDK 5 Language Features** » comme ci-dessous :

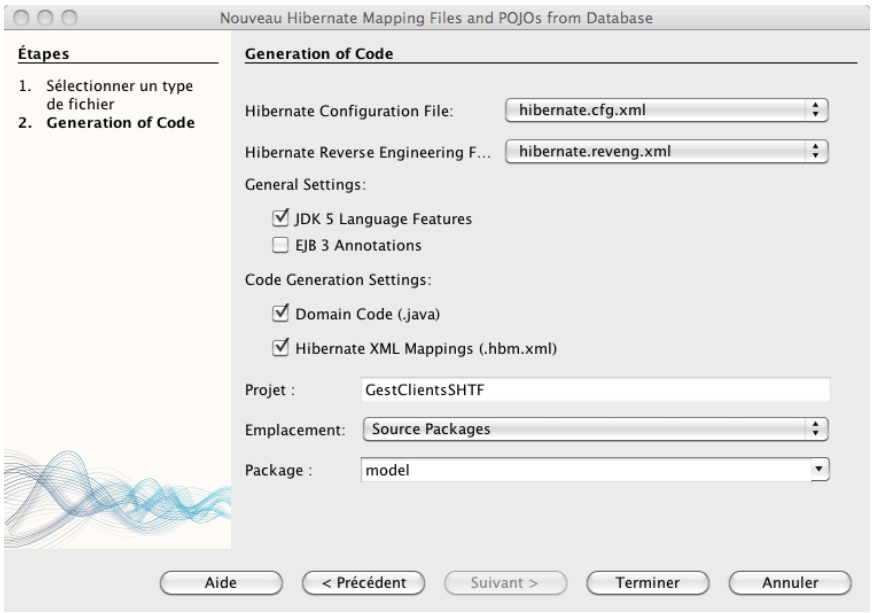


Figure 24 : Génération des classes avec Hibernate™

10.3.2.3 Mise en place de la structure du projet

A cette étape, nous allons mettre en place l'architecture de fichiers de l'application. Suite à l'étape précédente, nous avons déjà généré nos différents modèles (plus particulièrement nos .java et nos .hmb.xml au sein du package **model**).

Nous allons à présent ajouter un fichier à notre package **model**, un fichier qui permettra de créer une session Hibernate™. Faites un clique droit sur le package, puis créez un nouveau fichier **HibernateUtil.java**. Après sa création, renommez la classe **AnnotationConfiguration** en **Configuration**.

A présent, mettons en place la structure de notre login. Premièrement vous constaterez qu'un dossier **example** se trouve au sein du dossier **Web Pages** et qu'un package **example** se trouve au sein du dossier **Source Packages** (ils ont été générés lors de la création du projet).

Commencez par ouvrir le fichier **Web Pages/example/HelloWorld.jsp** et créez un lien comme ci-dessous :

```
<a href="LoginForm.jsp">Authentification</a>
```

Dans un second temps, ouvrez votre fichier **struts.xml** et créez votre action spécifique et vos résulats en fonction de cette dernière :

```
<action name="identificationUtilisateur" class="controller.utilisateursController" method="identifierUtilisateur">
        <result name="SUCCESSADM">/commandes.jsp</result>
        <result name="SUCCESSUTI">/commandesUti.jsp</result>
        <result name="ERROR">/LoginForm.jsp</result>
</action>
```

Puis ensuite vous allez créer un **LoginForm.jsp** dans votre dossier **Web Pages** ainsi qu'un **utilisateursController.java** dans votre package **/Source Packages/controller** (avec la méthode **identifierUtilisateur()** ainsi que les méthodes d'accès pour communiquer entre le .jsp et le contrôleur).

Vous pouvez dès à présent accéder à votre login avec cette URL : **http://localhost:8080/GestClientsSHTF/LoginForm.jsp**

A la fin de la création de cette structure, il vous suffira de compléter vos différents fichiers pour que votre application fonctionne complètement.

Au moindre problème, nous vous conseillons de consulter les sources de l'application de test sur le CDRom ou sur le site Internet suivant (relatif au travail de bachelor) : **http://www.ariellemoro.com/TDFrameworks**.


Au début de chacun de vos fichiers .jsp, n'oubliez pas de mettre la ligne suivante : <%@ taglib prefix="s" uri="/struts-tags" %> pour que les balises Struts™ soient correctement prises en compte.

10.3.3 Analyse de Struts™

10.3.3.1 Démarche

Dans le but d'évaluer Struts™, certains critères (qui ont la particularité d'être commun à Symfony™ et Zend™ framework également) ont été sélectionnés. Chaque critère est noté sur 5, tout comme la notre finale du framework.

10.3.3.2 Analyse détaillée

Puisque chaque critère a déjà été détaillé dans le sous-chapitre 10.2.4.2, nous n'allons aborder que certains points, notamment ceux qui ont une note inférieure à cinq.

La prise en main du framework est accessible mais reste cependant moins facile que celle des frameworks Php car il faut avoir des connaissances particulières en Java. L'installation de ce framework est très particulière car des dossiers importants (pour contenir des contrôleurs et des modèles par exemple) ne sont pas créés automatiquement. Concernant les formulaires, ils ne peuvent pas être générés à partir des classes java créées avec Hibernate™. L'organisation du code peut vite devenir compliquée si on ne crée pas les dossiers importants (voir ci-dessus) au bon endroit pour chaque application. La documentation sur les APIs de Struts™ est vraiment minime et il faut parfois chercher sur Internet pour trouver de bons exemples pour obtenir d'autres informations. Et pour terminer, il y a peu de composants très élaborés, il en existe moins que pour les frameworks Php.

10.3.3.3 Tableau récapitulatif

Voici le tableau d'évaluation du framework Struts™ :

Critères / Frameworks	Struts™
Prise en main du framework	4/5
Installation	4/5
Routing	5/5
Mapping relationnel-objet	5/5
Formulaires	4/5
Sessions	5/5
Organisation du code	4/5
Interactions avec la base de données	5/5
Documentation sur les APIs	4/5
Outils de développement	5/5
Feuilles de style CSS	5/5
Composants élaborés	3/5
Internationalisation	5/5
Note finale	**4.4/5**

Table 4 : Tableau d'évaluation de Struts™

10.4 Fiche n°3 : Analyse de frameworks Php et Java

Comparaison des frameworks

Critères / Frameworks	Symfony™	Zend™	Struts™
Prise en main du framework	5/5	4/5	4/5
Installation	5/5	2/5	4/5
Routing	5/5	5/5	5/5
Mapping relationnel-objet	5/5	3/5	5/5
Formulaires	5/5	5/5	4/5
Sessions	3/5	3/5	5/5
Organisation du code	5/5	5/5	4/5
Interactions avec la base de données	5/5	5/5	5/5
Documentation sur les APIs	4/5	4/5	4/5
Outils de développement	5/5	3/5	5/5
Feuilles de style CSS	5/5	5/5	5/5
Composants élaborés	5/5	5/5	3/5
Internationalisation	5/5	5/5	5/5
Note finale	**4.8/5**	**4.2/5**	**4.4/5**

Table 5 : Comparaison des frameworks Symfony™, Zend™ et Struts™ (fiche n°3)

Ce tableau met en évidence que ces trois solutions sont relativement appropriées pour réaliser une bonne application web. Les différences majeures sont le langage de programmation et évidemment le poids du framework car, en choisissant une application Java, celui-ci est souvent plus important. C'est pour cela que l'on utilise un framework Java, le plus souvent, pour des applications web très spécifiques.

11. Une solution alternative au framework : le CMS

11.1 Qu'est-ce qu'un CMS ?

Un CMS (Content Management System ou Système de gestion de contenu) est un assemblage de plusieurs logiciels qui permettent de réaliser des sites Internet dynamiques. Les utilisateurs peuvent mettre en forme du contenu et le personnaliser. Les différents niveaux d'accès à l'application sont également facilités.

Généralement tout est conçu pour que la majeure partie de l'application soit mise en place directement par le biais d'une interface web en local où tout a déjà été paramétré lors de l'installation (base de données, **serveur FTP** et autres paramètres).

Le CMS offre donc la possibilité de créer des applications Internet tout en occultant le passage difficile de la mise en place d'une structure de site avec du code pur. Il est d'avantage destiné à des personnes qui préfèrent passer par une interface pour mettre à jour un site Internet plutôt que de coder directement dans un logiciel prévu à cet effet (IDE par exemple). Il devient extrêmement intéressant en entreprise lorsque des personnes, qui n'ont pas forcément fait des études d'informatique, doivent mettre à jour une page de news par exemple. Car il est très facile de prendre en main un CMS.

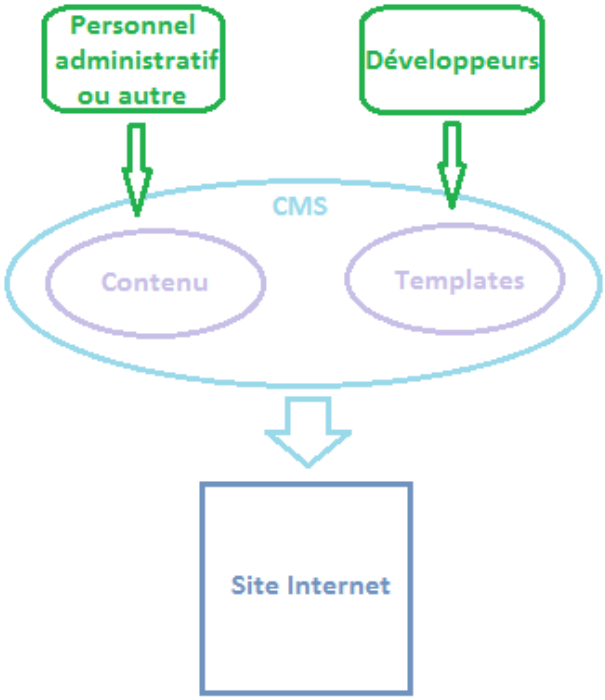


Figure 25 : Structure d'un CMS

11.2 Solutions actuelles

Il existe bon nombre de CMS sur le marché, certains gratuits, d'autres payants. Mais sachez que ça n'est pas parce qu'un CMS est gratuit qu'il est moins performant qu'un logiciel payant, bien au contraire. Car les communautés sont souvent, tout comme pour les frameworks, très présentes et contribuent activement aux améliorations et à la correction des éventuels problèmes relevés.

11.2.1 Joomla™

Le premier CMS dont nous allons parler est Joomla™. Il s'agit d'un CMS open-source, gratuit et très populaire. Vous pouvez le télécharger directement sur le site suivant : **http://aide.joomla.fr/telechargements/Joomla-1.5.x-packages-dinstallation-et-patches/index.php**. Cet CMS est très facile à installer (vous le placez, comme tout bon framework au sein de votre dossier qui contient vos sites web : **htdocs** par exemple dans MAMP™) et vous démarrez simplement son installation à l'aide de l'URL suivante : **http://localhost:8888/Joomla**.

Attention, lors de l'installation, Joomla™ va rajouter une multitude de tables au sein de votre base de données MySQL™. Si vous utilisez votre base de données pour plusieurs applications, cela peut être problématique.

Après cette installation et la création de votre site Internet, vous pouvez très facilement modifier une multitude de paramètres à l'aide d'une interface administrateur protégée par login et mot de passe (**http://localhost:8888/Joomla/administrator/index.php**). Vous pourrez créer des utilisateurs, personnaliser le contenu du site et vous pourrez également ajouter des extensions très utiles (par exemple une extension va vous permettre d'avoir la possibilité d'adapter votre site pour que les visiteurs puissent le consulter depuis un téléphone mobile).

Au final, on peut obtenir un site fonctionnel en quelques minutes et charger une personne, connaissant peu l'informatique, de créer des articles chaque jour pour faire vivre le site Internet de l'entreprise. Il suffit de bien analyser, au début du projet, quels sont les réels besoins du mandant et les enjeux de l'application ou du site Internet.

Figure 26 : Interface d'administration de Joomla™

11.2.2 Drupal™

Drupal™ est un CMS également open-source et gratuit. Il se présente pratiquement de la même manière que Joomla™. Vous pourrez le télécharger sur ce site : **http://drupalfr.org/**. En le plaçant dans votre dossier de sites web, vous procéderez à son installation par le biais de cette URL : **http://localhost:8888/Drupal/**.

A première vue, l'interface d'administration ne change pas beaucoup de Joomla™. Nous retrouverons la plupart des fonctionnalités dont nous disposions avec Joomla™ (gestion des articles, des utilisateurs, ajout de modules et bien d'autres paramètres). Nous voyons directement, sur la page principale, les dernières mises à jour importantes à installer.

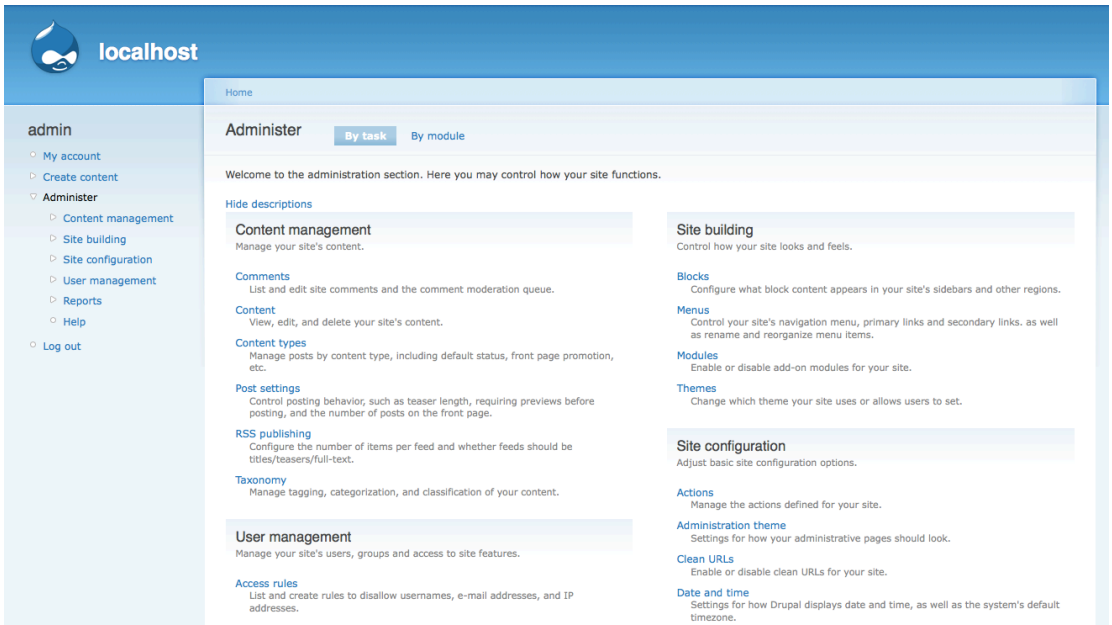


Figure 27 : Interface d'administration de Drupal™

11.2.3 Comparaison succincte

Contrairement à Joomla™, qui propose une structure et une gestion de contenu relativement simple à prendre en main (un article est rattaché à une catégorie, elle-même rattachée à une section, le tout pouvant être personnalisé avec des composants, plugins et autres). Drupal™ permet de réaliser une gestion de contenu basée sur des nœuds (types de contenu auquel on va attribuer un rôle), blocs (éléments de contenu) et modules (pour ajouter une fonctionnalité particulière).

Concernant la performance, Drupal™ est plus rapide que Joomla™ si on utilise le cache. La tendance s'inverse lorsqu'on le désactive : Joomla™ semble alors meilleur que Drupal™.

En définitive ces deux CMS sont différents au niveau de leur architecture de gestion de contenu et de leurs performances (avec ou sans cache). Drupal™ paraît, au premier abord, plus professionnel et plus complexe tandis que Joomla™ possède une interface plus conviviale et sera destiné à un public plus large.

Pour conclure par rapport à l'utilisation du CMS comme solution alternative au framework, le choix se fait concrètement au tout début du projet. Il ne faut pas utiliser un framework si le projet peut être réalisé tout simplement à l'aide d'un CMS. Dans le chapitre suivant, nous aborderons le choix d'un framework pour l'entreprise ainsi que pour les étudiants.

Figure 29 : Logo du CMS Joomla™

Figure 28 : Logo du CMS Drupal™

11.3 Fiche n°4 : Une solution alternative au framework : le CMS

1. Définition

Le CMS est un ensemble de logiciels servant à gérer du contenu de façon simplifié, par le biais d'interfaces. Avec ceci nous pouvons facilement réaliser et mettre à jour des sites Internet dynamiques. Ils permettent de travailler en équipe avec des gens qui n'ont pas forcément besoin d'avoir des notions très pointues en informatique, s'agissant de mettre à jour le contenu d'un site. Cependant un informaticien sera indispensable pour personnaliser le CMS, rajouter des plugins ou des modules.

2. Architecture

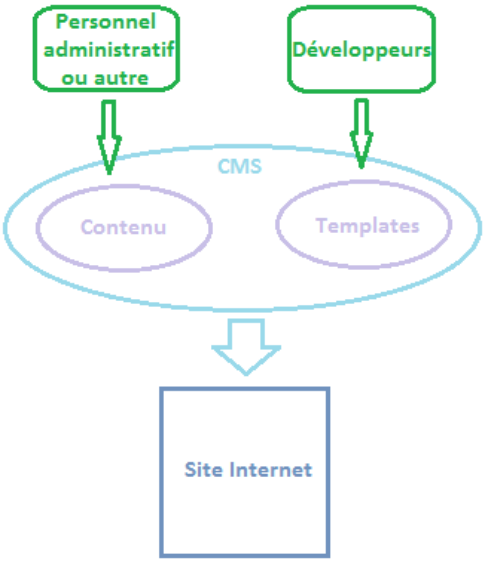


Figure 30 : Structure d'un CMS (fiche n°4)

3. Points positifs et négatifs

Points positifs : Création d'un site Internet en quelques minutes, maintenance facile et pouvant être effectuée par des non informaticiens (informaticiennes) (excepté les mises à jour très techniques ou la personnalisation du CMS).

Points négatifs : Structure imposée des articles et du contenu ainsi que difficulté parfois à ajouter du code pur au sein du CMS.

12. Choix d'un framework

12.1 Pour l'entreprise

12.1.1 Démarche

Ce chapitre a été réalisé à la suite d'un sondage effectué durant ce travail de bachelor auprès de développeurs travaillant en entreprise qui ont été confrontés au choix d'un framework.

Des entreprises ont été sélectionnées et contactées par téléphone dans la région de Genève. Le principal critère de sélection est qu'elles effectuent toutes du développement informatique. Sur quinze entreprises appelées, cinq ont répondu au questionnaire en ligne (questionnaire que vous trouverez en annexe à la fin du document).

Les deux diagrammes circulaires ci-dessous montrent les types des entreprises interrogées ainsi que leur spécialisation :

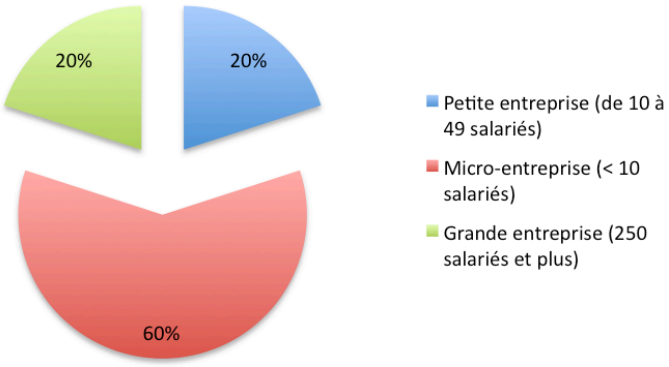


Figure 31 : Types d'entreprises interrogées

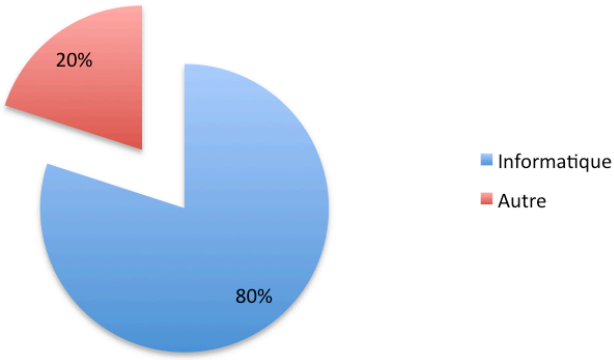


Figure 32 : Pourcentage d'entreprises spécialisées en informatique

12.1.2 Le framework et l'entreprise

12.1.2.1 Utilisation d'un framework

Comme indiqué dans le chapitre précédent, chacune des entreprises ayant répondu au questionnaire, développe des applications web ou des sites Internet. Cependant elles n'utilisent pas toutes un framework. Leur choix est intimement lié aux projets (au cas par cas). Le client peut influer sur le choix du framework ; en effet s'il travaille avec tel ou tel logiciel, il peut vouloir souhaiter que l'on développe son application ou site web avec le même logiciel pour simplifier la maintenance (dans le cas où il « outsource » uniquement le développement).

Ce qui est clairement ressorti de cette analyse est que le substitut au framework est le CMS. En effet, si le projet consiste en de la pure gestion de contenu, le choix se porte généralement sur le CMS. Cependant, celui-ci ne permet pas de construire des applications spécifiques. Dans ce cas particulier, les frameworks prennent le relais.

Les principaux choix qui motivent l'utilisation d'un framework sont les suivants : la gratuité de ces derniers, leur organisation et leur structure, la possibilité de mettre en place des tests facilement, le gain de temps ainsi que les standards imposés (et souvent retrouvés en entreprise).

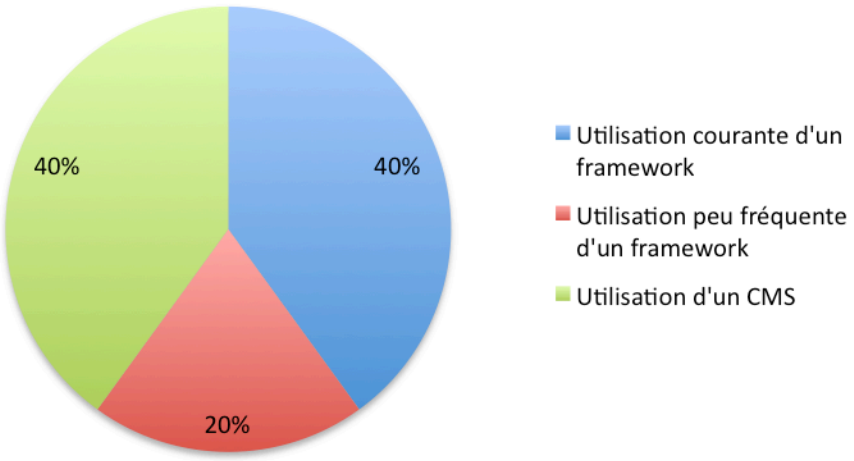


Figure 33 : Pourcentage d'utilisation des frameworks en entreprise

12.1.2.2 Création de son propre framework

Entre le CMS et le framework il n'y a qu'un pas, tant et si bien que certaines entreprises ont choisi de n'en faire qu'un seul logiciel en créant une sorte de framework élaboré pour l'adapter au mieux à leurs clients et gagner ainsi du temps lors du développement.

Dans d'autres cas, la création d'un framework est venue naturellement en rajoutant des fonctionnalités (ou composants) et en personnalisant le framework à l'image de l'entreprise.

Si l'entreprise ne crée pas de framework, elle va le personnaliser pour répondre au mieux aux besoins de sa clientèle. Et la personnalisation la plus courante est d'augmenter la sécurité du framework.

Finalement, lorsque le choix a été fait de créer un framework, il fait l'objet d'un réel projet en interne pour l'entreprise. Et pour le réaliser au mieux, il convient de le planifier en dehors des autres projets que l'entreprise doit effectuer.

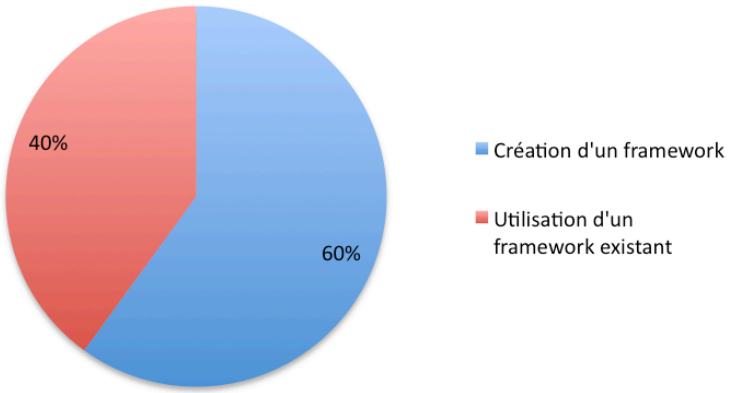


Figure 34 : Pourcentage d'entreprises ayant créé leur framework

12.1.2.3 Classement

Nous retrouvons, sans surprise, en tête du classement des frameworks pour l'entreprise Symfony™ et Zend™ framework. Les frameworks qui les suivent de près sont Struts™ (les applications web en java sont parfois plus rares) et CakePHP™.

Symfony™ reste néanmoins le « préféré » pour les avantages qu'il possède tant au niveau de sa structure qu'au niveau de son alliance avec Doctrine™ (qui permet de générer des formulaires de CRUD automatiquement et de faciliter d'autres opérations).

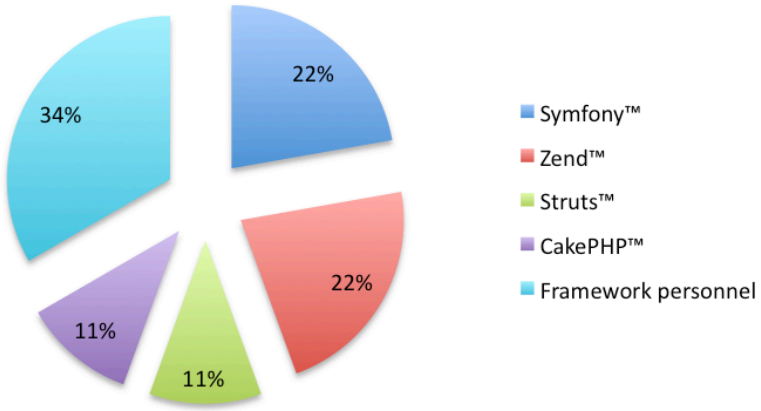


Figure 35 : Frameworks utilisés en entreprise

12.1.2.4 Les critères de choix

Les critères de choix d'un framework restent des indicateurs très importants. Ils sont conditionnés par les tendances du web actuel et certains aspects relatifs au développement (sécurité, internationalisation, gain de temps lors du développement…).

Lors de cette étude, nous avons relevé que le critère le plus important pour le choix d'un framework est la **sécurité**. Et ce même en considérant que de nombreux développeurs la renforcent dès le début du projet.

Le critère arrivant en deuxième position se trouve être la **maintenance**. En effet, plus le framework impose une structure, des normes, plus il est facile de maintenir les applications web par la suite.

Le troisième critère est la **présence d'outils aidant au développement** (logs pertinents ou interface de développement). Par exemple, les interfaces de développement intégrées à Symfony™ permettent de tracer les différentes étapes d'une requête entre l'application web et sa base de données ainsi que leur temps d'exécution.

Le classement final que nous avons obtenu est le suivant (du critère le plus au moins important) :

1. Sécurité
2. Maintenance
3. Présence d'outils aidant au développement
4. Présence de système d'authentification
5. Facilité de déploiement
6. Tests unitaires et fonctionnels facilités
7. Ajout d'extensions
8. Routage
9. Internationalisation

12.2 Pour les étudiants

12.2.1 Démarche

Pour obtenir des informations pertinentes pour ce chapitre, une étude a été réalisée auprès d'étudiants en informatique ayant déjà une petite expérience en matière de développement.

Sur une dizaine d'étudiants, sept ont répondu au questionnaire (questionnaire que vous trouverez en annexe à la fin du document).

L'objectif était de voir si un framework avait été choisi et sur quel framework leur choix s'était porté lors de leurs premiers développements. Ci-dessous, vous trouverez un graphique qui démontre que les frameworks sont choisis par un grand nombre d'étudiants pour leurs premiers projets.

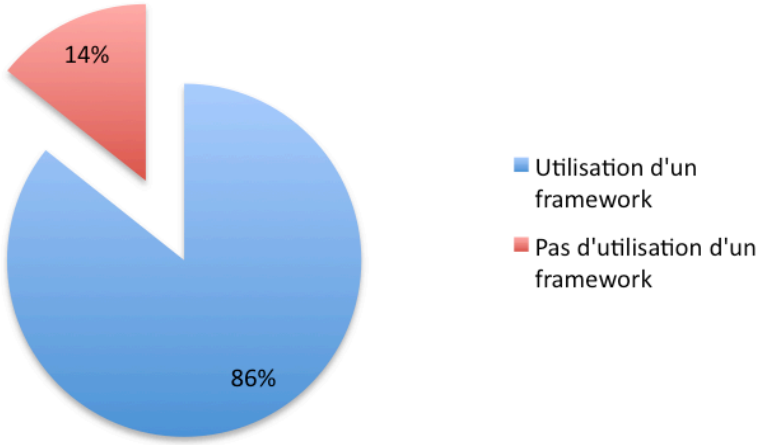


Figure 36 : Pourcentage d'étudiants ayant choisi d'utiliser un framework pour leurs premiers projets

12.2.2 Problématique

Lors d'un premier projet il est toujours difficile d'évaluer si nous avons réellement besoin d'un framework ou s'il va être pertinent d'en utiliser un sachant qu'on ne le connaît pas en profondeur et qu'on n'utilisera pas la plupart des capacités de ce dernier (ceci étant dû notamment au manque d'expérience en la matière).

Par conséquent, la question suivante se pose : dans le cas où le projet le permet (en ayant de grandes libertés lors du développement de l'application web) ne serait-ce pas mieux, en termes d'apprentissage, de débuter avec un framework plus flexible et plus malléable dans le but de comprendre concrètement comment fonctionne un framework ?

Lors de la consultation, les étudiants ont répondu en masse qu'ils préféraient, si leur projet le permettait, débuter sur un framework plus simple à aborder dans le but de comprendre réellement son fonctionnement.

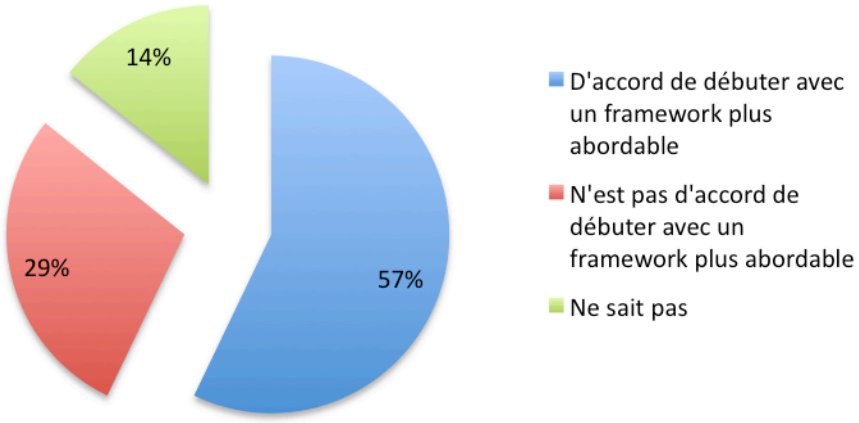


Figure 37 : Pourcentage d'étudiants qui seraient ouvert à l'idée de débuter avec un framework plus abordable pour comprendre son fonctionnement

12.2.3 Classement

Concernant les frameworks les plus utilisés pour les premiers projets, le classement est pratiquement similaire à celui des entreprises. Symfony™ arrive encore une fois en tête avec Zend™. Ce qui confirme leur notoriété pour les développements d'applications web. Mais attention, les CMS sont aussi énormément sélectionnés par les étudiants pour certains projets (car tout projet ne requiert pas forcément l'utilisation d'un framework). Et en ce qui concerne les CMS, Joomla™ est sélectionné en priorité (la facilité de prise en main est son plus grand atout). Ce qui prouve encore une fois que l'analyse d'un projet est déterminante pour le choix des outils par la suite qui vont aider au développement de celui-ci.

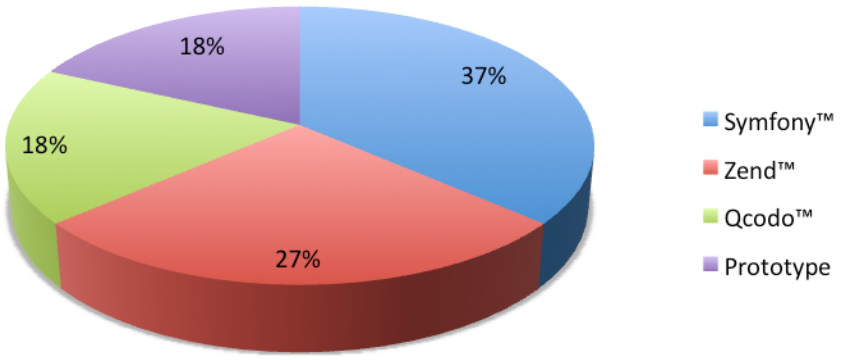


Figure 38 : Frameworks utilisés par les étudiants

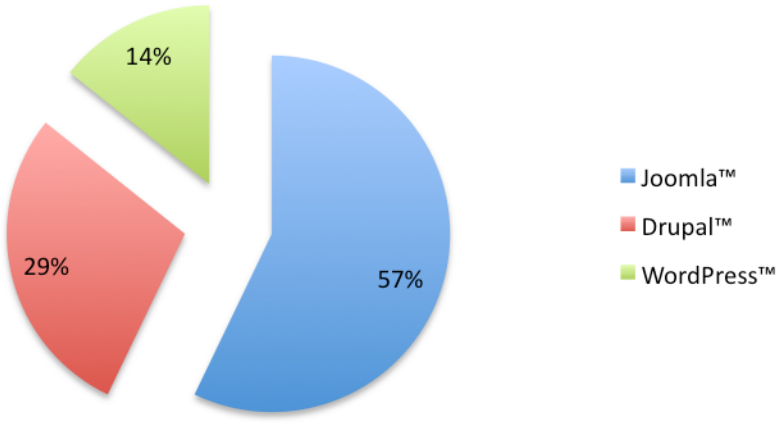


Figure 39 : CMSs utilisés par les étudiants

12.2.4 Exemple de framework de projet d'étude : QCodo™

QCodo™ est un framework open-source qui supporte PHP5.

Il a été sélectionné lors d'un projet d'étude. Il n'est peut-être pas aussi populaire que d'autres frameworks mais possède de nombreuses qualités qui nous ont permis de comprendre le fonctionnement concret d'un framework et ses possibilités en termes de structure, de personnalisation (créations de nouveaux composants par exemple).

Même si la documentation de ses APIs n'est pas complète, il bénéficie d'une communauté assez réactive qui permet de ne pas rester bloqué sur certains problèmes que nous pourrions rencontrer lors du développement.

Un des principaux avantages de QCodo™ est qu'il contient son propre ORM et qu'il génère des formulaires de CRUD très efficaces pour chaque table de la base de données.

Comme pour l'utilisation de tout premier framework, la prise en main revêt quelques difficultés au départ mais très rapidement, le fonctionnement de QCodo™ et la création de nouveaux composants deviennent un jeu d'enfant.

Organisation des fichiers de QCodo™ (selon la dernière version du framework : 0.4.14) :

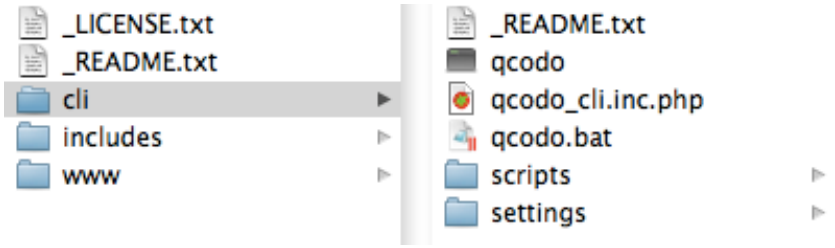


Figure 40 : Structure de fichiers de QCodo™ 1

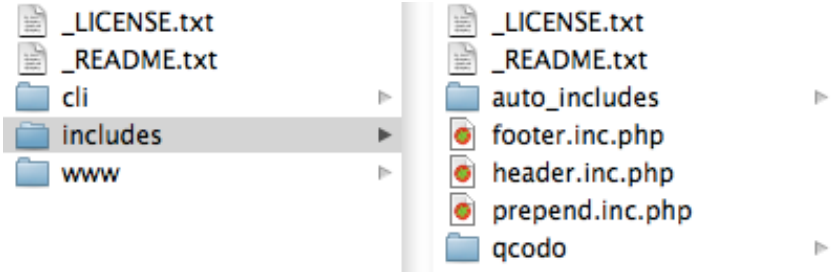


Figure 41 : Structure de fichiers de QCodo™ 2

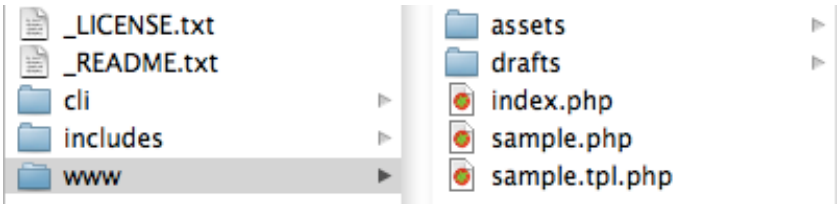


Figure 42 : Structure de fichiers de QCodo™ 3

Le dossier **cli** contient les principaux fichiers de configuration du framework (connection à la base de données…). Le dossier **includes** est le dossier conteneur de l'intégralité des librairies de QCodo™. Et pour terminer, le dossier **www** est le dossier qui va servir de démarrage du projet (on y retrouve le fameux **index.php**).

Il faudra absolument rajouter un dossier **web** (par exemple) qui contiendra vos vues, contrôleurs et nouveaux composants créés).

Vous trouverez de nombreuses informations complémentaires sur le site officiel de QCodo™ : **http://www.qcodo.com/**.

Points forts de QCodo™ :

- présence d'un ORM ;
- génération de formulaires de CRUD ;
- intégration d'Ajax de façon très simple ;
- communauté réactive (forum) ;
- tutoriels bien détaillés avec code source.

Points faibles de QCodo™ :

- une documentation sur les API peu détaillée mais qui liste néanmoins toutes les méthodes et attributs des divers composants ;
- mise à jour du framework peu évidente ;
- poids du framework.

Composants de QCodo™ :

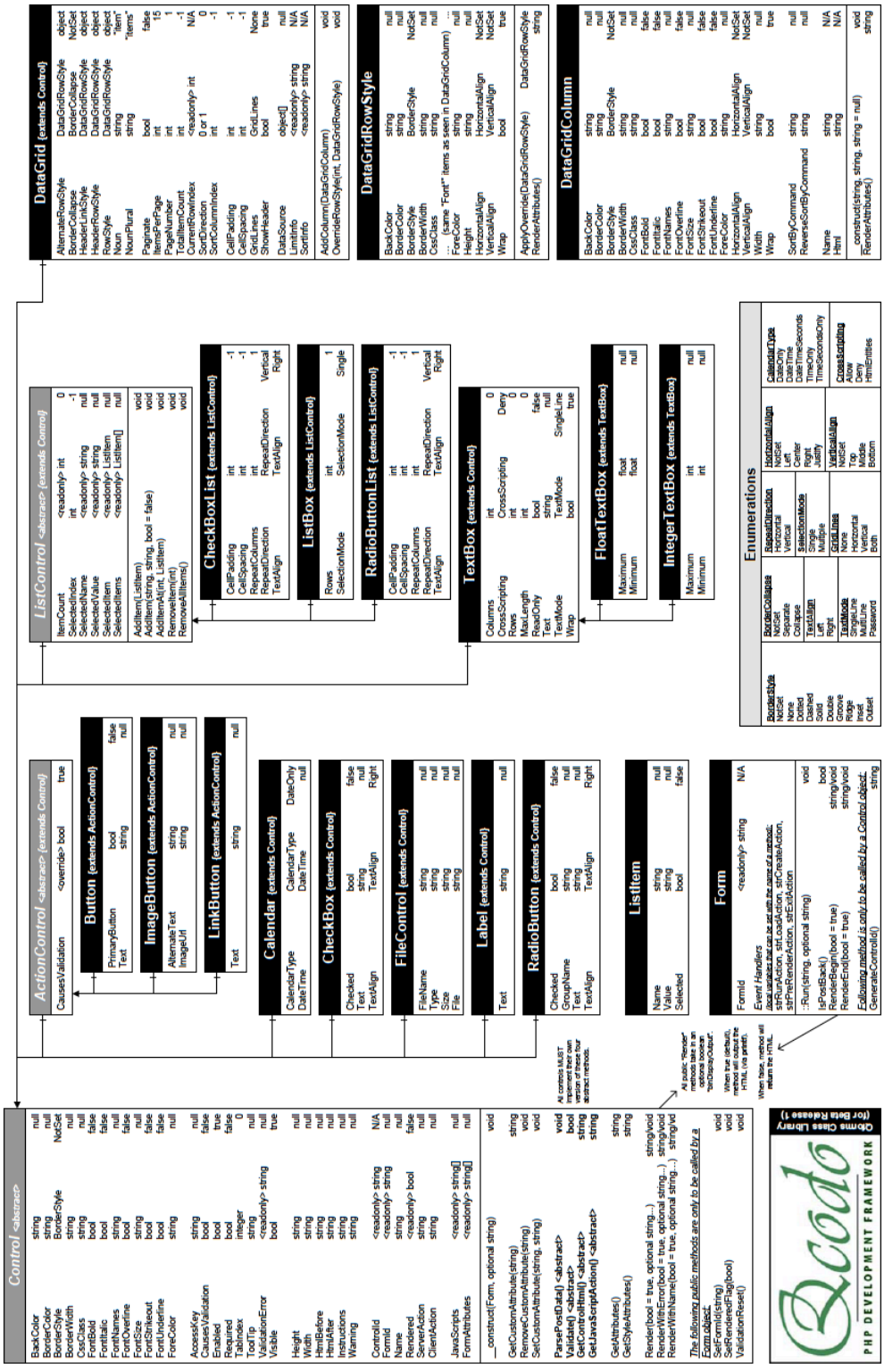


Figure 43 : Composants du framework QCodo™

12.3 Checklist d'aide à la décision

Cette checklist d'aide à la décision vous permettra de faire un choix tout en tenant compte de certains critères essentiels. N'oubliez donc pas de prendre en compte ce qui va suivre :

- le type de l'entreprise qui souhaite le projet (petite, moyenne ou grande entreprise) ;
- le type de projet souhaité (application complexe, site d'entreprise avec gestion des utilisateurs, site vitrine de l'entreprise ou encore site personnel) ;
- la présence d'un service informatique dans l'entreprise (pour la maintenance de l'application) ;
- les logiciels existants et utilisés par le service informatique ;
- les connaissances en informatique des personnes souhaitant le projet ;
- le budget destiné au projet ;
- les potentielles évolutions futures de l'application ou du site Internet.

12.4 Fiche n°5 : Choix d'un framework

1. Pour l'entreprise

Critères de sélection d'un framework (du plus au moins important) :

1. Sécurité
2. Maintenance
3. Présence d'outils aidant au développement
4. Présence de système d'authentification
5. Facilité de déploiement
6. Tests unitaires et fonctionnels facilités
7. Ajout d'extensions
8. Routage
9. Internationalisation

Classement des frameworks :

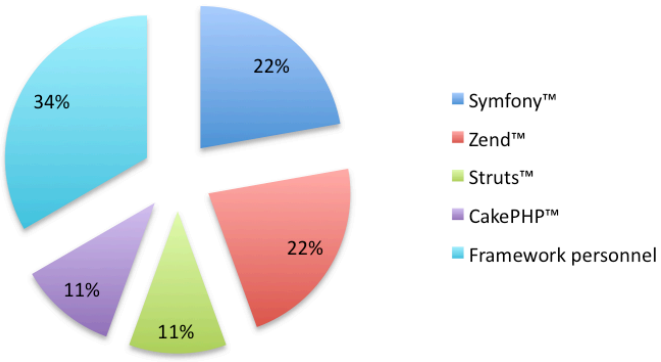


Figure 44 : Frameworks utilisés par les entreprises (fiche n°5)

2. Pour les étudiants

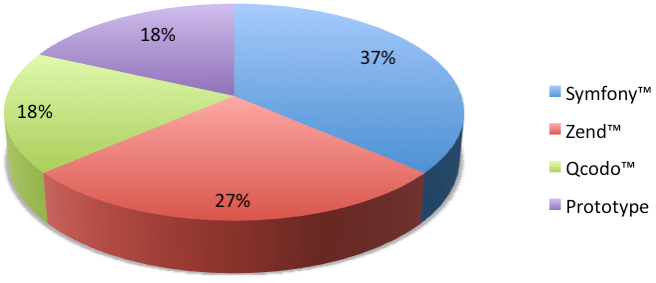


Figure 45 : Frameworks utilisés par les étudiants (fiche n°5)

13. Conclusion : Les frameworks et leurs évolutions

A la vitesse à laquelle évolue le web, les développeurs, qui mettent à jour les frameworks et qui créent des extensions, sont obligés de tenir compte des nouveautés.

L'exemple le plus flagrant se trouve être la possibilité d'intégrer une carte Google™ et d'interagir avec cette dernière pour obtenir les coordonnées d'un lieu (latitude et longitude). Suite à cela, des plugins ou extensions ont été développés pour permettre ces interactions de façon simplifiée.

Les évolutions futures se feront donc avec ces extensions (sans compter les importantes mises à jour structurelles : versions 1.0, 2.0…). Suite au boom de Facebook™, peut-être que nous trouverons dans quelque temps des API Facebook™ ou de tout autre réseau social.

En ce qui concerne les frameworks Php, Symfony™ et Zend™ framework sont incontestablement les leaders. Cependant, au lieu de décider de s'affronter, il serait possible que des évolutions communes soient envisagées. En effet, le plus important challenge actuel est de créer une véritable interopérabilité entre les deux frameworks. Pour ce faire, une charte concernant l'architecture des composants Php a été mise en place. Ayant déjà une forte vision commune du développement et des architectures très proches, il semblerait que les deux parties aient décidé de travailler ensemble pour mieux appréhender les évolutions liées au web en général et au développement.

En ce qui concerne les ORM Php, Doctrine™ semble devenir le leader le développement, Symfony™ l'a déjà intégré et Zend™ framework va l'incorporer dans sa future version 2.0.

Quoi qu'il en soit, en fin d'année 2010, les deux frameworks Symfony™ et Zend™ framework vont tous deux sortir leur version 2.0. La tendance est inévitablement à la simplification de l'utilisation des frameworks. Prenons l'exemple de Symfony™, nous aurons la possibilité de suivre une véritable visite guidée qui nous aidera à créer notre premier projet. Par conséquent, le framework sera donc encore plus accessible. Au niveau de la rapidité, elle sera augmentée considérablement par rapport à la version 1.4…

En ce qui concerne les frameworks Java et Struts™ plus précisément, la grande évolution avait été entre Struts 1 et Struts 2 qui était une fusion de Struts™ et Webwork. Les classes d'action ont beaucoup évolué et leur cycle de vie lors de l'exécution d'un programme également.

Pour conclure sur ce sujet, je dirais qu'avec l'évolution du langage Php nous avons encore beaucoup de nouveautés à découvrir en matière d'aide au développement. Le but est vraiment d'avancer ensemble et de regrouper les communautés de Symfony™ et de Zend™ framework.

14. Bilan personnel

Ce travail de diplôme a vraiment marqué la fin de mon bachelor en informatique de gestion. Il s'agit d'un travail relativement complet dans le sens où il est teinté de recherche d'informations, d'élaboration d'applications de démonstration ainsi que d'études que j'ai pu réaliser auprès d'entreprises et d'étudiants.

Il m'a permis d'aborder un sujet qui n'était pas traité à proprement parler dans ma formation : celui qui concerne les frameworks. C'est d'ailleurs en réalisant un projet au cours de celle-ci que j'ai eu l'idée de ce sujet de mémoire. Je trouvais et je trouve important d'avoir une réponse concrète à ce qui a pu nous poser problème lors de ce projet soit, le choix du framework.

Ce qui a été le plus difficile durant ces deux mois reste le fait que nous sommes confronté au début à une page blanche. Le plus dur, en effet, est de débuter l'écrit. Après tout s'enchaîne et devient plus fluide. Le deuxième élément qui n'est pas évident dans ce travail est qu'il ne s'agit pas d'un travail de groupe où on confronte avec plaisir nos différents points de vue et nos idées en comparaison avec les différents travaux de groupe réalisés durant ma formation. Mais fort heureusement les discussions avec le professeur HES qui m'a encadrée, soit Monsieur Peter Daehne, ont été très importantes.

Au final, ce que je retiens de ces deux mois est l'importance de la rigueur du travail. Chaque jour nous apportons un regard différent sur notre travail et nous le faisons évoluer tout en respectant une trame qui a été limitée dès le début pour ne pas s'éloigner du sujet. C'est donc une expérience très positive qui marque la fin d'une première partie d'étude et le début d'une autre.

« Ecrire, c'est de toute façon tenter une manière de lecture d'un texte encore caché, mais qui existe préalablement et qu'il s'agit d'amener au jour. »

Alice Rivaz

15. Glossaire

A

Apache™ : Il s'agit du serveur http le plus souvent utilisé. Il peut être installé sur divers systèmes d'exploitation et est très souvent embarqué dans des logiciels aidant au développement d'applications tels que WAMP™, MAMP™ ou EasyPhp™.

API (Application Programming Interface) : Une API est une interface qui permet d'interagir facilement avec un ou plusieurs composants (eux-mêmes contenus dans une librairie de frameworks par exemple).

Assert (ou assertion) : Une assert est utilisée principalement au sein des tests en programmation réalisés avec JUnit ou PhpUnit. Une assertion permet de détecter toute anomalie au sein d'un code en énonçant une condition qui est censé être toujours vraie.

C

Cas d'utilisation : Il représente, dans le langage de modélisation UML, une fonctionnalité de l'application telle que « Passer une commande » pour un acteur de l'application qui serait un client par exemple.

Contrôleur Servlet : Contrôleur qui permet de distribuer les différentes requêtes entre modèles et vues pour une servlet qui est une application Java (application qui permet d'interagir facilement avec un serveur http, soit de générer des données ou pages qui sont comprises par un serveur http ou navigateur web).

CRUD (Create-Read-Update-Delete) : Cet acronyme informatique représente en lui-même les quatre opérations suivantes : créer, lire, mettre à jour et supprimer un élément.

D

Diagramme d'analyse (ou de robustesse) : Diagramme défini pour déterminer quels seront les objets (interface, contrôleur ou encore conteneur d'informations) au sein de l'application. Très facile à obtenir après l'étude de chacun des cas d'utilisation et après avoir fait les cartes CRC qui s'y rapportent.

Diagramme de cas d'utilisation : Diagramme qui va permettre de lister les différents cas d'utilisation de l'application ainsi que leurs interactions avec les différents acteurs de l'application.

Diagramme de classes : Diagramme qui va mettre en évidence toutes les classes qui interagissent entre elles au sein de l'application créée.

Diagramme de communication : Diagramme qui mettra en avant les interactions qui présentera une version plus simple du diagramme de séquence représentant uniquement qu'il y a des échanges de messages entre acteurs, composants ou objets.

Diagramme de composants : Diagramme mis en place pour définir quels sont tous les composants du programme ainsi que leurs éventuelles dépendances.

Diagramme de déploiement : Diagramme mettant en évidence les différents composants ainsi que l'infrastructure physique du système dont il est question.

Diagramme d'états transitions : Diagramme permettant de définir les différents états d'un programme et de montrer quelles sont les différentes transitions possibles entre ces états.

Diagramme d'objets : Diagramme servant à mettre en avant tous les objets qui seront instanciés durant l'exécution du programme. Il vient compléter le diagramme de classes.

Diagramme de séquence : Diagramme mis en place pour montrer les différentes interactions entre objets, composants ou encore acteurs. Il sera composé de messages asynchrones ou synchrones.

F

Fenêtre DOS : Une fenêtre DOS est une interface, sous Windows, qui permet d'interagir avec l'ordinateur par le biais de lignes de commande (lignes qui permettent d'accéder à un répertoire particulier, d'exécuter un programme…).

Fichiers JSP (JSP : Jasper) : Fichier composé de données (HTML, directives, balises…) qui pourront être interprétées par un navigateur web. Les fichiers JSP sont compilés par un compilateur JSP pour obtenir des servlets Java.

I

IDE (Integrated Development Environment) : Programme (constitué de plusieurs outils) qui permet d'écrire des scripts en Java ou en Php de façon plus aisée qu'avec un simple bloc-notes.

J

Java : Langage de programmation orienté objet qui a été créé par deux employés de Sun Microsystems, James Gosling et Patrick Naughton, en 1995.

M

MCD (Modèle conceptuel de données) : Modèle permettant de mettre en évidence des entités ainsi que les associations qu'elles ont entre elles et leurs cardinalités associées.

Méthode de développement Agile : Méthode de développement pour des projets de courte durée. Elle est axée *pair programming* (deux programmeurs travaillent toujours ensemble).

Méthode de développement Cascade : Méthode de développement établie pour passer d'une phase à l'autre du projet en ayant la possibilité de modifier la phase antérieure mais en ne pouvant jamais revenir à la première phase du projet. Elle protège le groupe de projet mais ne garantit pas le résultat espéré par le client.

Méthode de développement RUP : Méthode de développement de projet répartie en quatre étapes composées elles-mêmes de workflows spécifiques. Au sein de chaque phase nous pouvons faire plusieurs itérations. Généralement chaque itération correspond à un cas d'utilisation de l'application et il faut impérativement les mettre par ordre d'importance (du plus au moins important).

Méthode Merise : Méthode de réalisation de systèmes d'information en passant par une étape d'analyse et de conception. Méthode pouvant servir également à la réalisation de base de données relationnelle à la suite d'un MCD et d'un MLD.

MLD (Modèle logique de données) : Modèle permettant de mettre en évidence les tables (entités transformées du MCD) et les relations qu'elles ont à l'aide de clés étrangères ou de tables d'association (selon la modification subie en fonction de l'association d'origine du MCD).

Modèle Bean (ou Beans) : Modèles qui sont en réalité des classes qui respectent un certain nombre de règles tels que l'obligation de posséder un constructeur, d'avoir des propriétés dont les méthodes d'accès possèdent un nom et une signature normalisés.

Modèle relationnel de données : Modèle qui met en évidence des tables ainsi que leurs relations entre elles. Essentiellement basé sur le concept des bases de données relationnelles.

Modélisation : Dans le but d'amoindrir la complexité d'un futur système et pour comprendre avec exactitude les besoins de notre mandant ou le tenant et l'aboutissant de notre projet, nous utilisons la modélisation. Etape importante où, par le biais de modèles qui mettent en évidence les différentes interactions entre les acteurs et le logiciel final ou alors les interactions purement internes du système, nous arrivons à définir précisément ce dernier avant même de passer à l'étape de conception.

O

Open-source : Ce terme désigne un logiciel dont la licence englobe les critères précis de l'open-source soit une licence libre. Ce qui signifie le plus souvent que nous avons accès au code source d'un logiciel. Mais attention, cette désignation ne signifie en aucun cas que le logiciel est gratuit.

ORM (Object-Relational Mapping) : Technique qui sert à relier une base de données relationnelle avec des propriétés objets au sein d'un code pour avoir l'impression de manipuler une base de données orientée objet. Un ORM est en général implanté sous la forme d'un framework tel que Doctrine™ en Php ou Hibernate™ pour Java.

P

Php (Hypertext Preprocessor) : Langage de programmation orienté objet non typé créé en 1994 par Rasmus Lerdorf.

Programmation procédurale : Programmation antérieure à la programmation orientée objet qui est basée sur un enchainement d'appels de procédures ou de fonctions durant toute l'exécution du programme.

R

Rational Software Architect™ (IBM™) : Suite de logiciels qui permettent de modéliser un système informatique (modélisation basée sur UML) et de produire du code Java ou du C++.

S

Serveur FTP (File Transfer Protocol) : Un serveur FTP met en place un protocole de transfert de fichiers dans le but de copier des fichiers d'un ordinateur à un autre ordinateur ou serveur du réseau. On s'en sert le plus souvent pour alimenter un site web.

PL/SQL (Procedural Language / Structured Query Language) : Langage de programmation SQL créé par Oracle™ et permettant d'interagir avec sa base de données ou encore MySQL™. Il permet, entre autres, d'effectuer des procédures, de réaliser des requêtes, des vues et encore une d'autres opérations selon la solution choisie.

T

Terminal : Un Terminal est un utilitaire Mac qui supporte un shell qui n'est autre qu'un interpréteur de lignes de commande (tout comme une fenêtre DOS sous Windows).

U

UML (Unified Modeling Language) : Langage de modélisation unifié étroitement lié à la programmation orientée objet. Il est le résultat d'une fusion de plusieurs anciens langages de modélisation objet tels que Booch, OMT et OOSE.

URL (Uniform Ressource Locator) : URL est un synonyme d'adresse web qui va être se trouver sous la forme suivante : http://www.exemple.com. Elle permet donc d'accéder à un site web.

Y

YAML (format) : Format de fichier très proche du XML (Extensible Markup Language) qui permet de réaliser des fichiers structurés à l'aide de balises (fichier plat qui pourrait remplacer aisément une base de données très simple au niveau de son architecture). Il est, lui aussi, composé de balises et sert à stocker des données de type texte.

16. Bibliographie

- **Introduction :**

[Kay68] KAY, Alan FLEX, A flexible extensible language, M. S. thesis, University of Utah, May 1968.

- **Notions préliminaires :**

MULLER, Pierre-Alain (ENSISA). *Représentation des vues d'architecture avec UML*. Présentation power-point (PDF).

LANG, Frédéric (Ecole Normale Supérieure de Lyon). *Le langage Java (I)*. Présentation power-point (PDF).

GRIN, Richard (Université de Nice Sophia-Antipolis). *Correspondance Objet-Relationnel – Version 1.0.1*. Présentation power-point (PDF). 15 septembre 2007.

[GOF] GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, John, *Design Pattern Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN : 0-201-63361-2.

- **Description des frameworks :**

PAULI, Julien, PONÇON, Guillaume. *Zend Framework – Bien développer en PHP*. Paris : Eyrolles, 2009. 446 p. (Les cahiers du programmeur).

POTENCIER, Fabien, HAMON, Hugo. *Symfony – Mieux développer en PHP avec Symfony 1.2 et Doctrine*. Paris : Eyrolles, 2009. 486 p. (Les cahiers du programmeur).

- **Analyse des frameworks :**

Wielenga, GeertJan, BRABANT, Vincent (Traducteur). *Développer une application Struts à l'aide de NetBeans.* Présentation power-point (PDF). 28 novembre 2005 (date de publication) et 8 janvier 2006 (dernière mise à jour).

- **Frameworks pour l'entreprise :**

GOULEAU, Emmanuelle, MANSOUR, Olivier, RIVOALLAN, Tristan, LEMAIRE, Vincent, LACOT, Xavier. *Livre blanc – Framework PHP pour l'entreprise – Définition, critères de choix et analyse.* Document PDF. 14 mai 2008.

17. Webographie

- **Notions préliminaires :**

Wikipédia – Définition de la programmation orientée objet. Consulté le 12 juillet 2010. http://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet

Wikipédia – Définition de la programmation procédurale. Consulté le 12 juillet 2010. http://fr.wikipedia.org/wiki/Programmation_proc%C3%A9durale

Wikipédia – Définition des Design Patterns. Consulté le 12 juillet 2010. http://fr.wikipedia.org/wiki/Patron_de_conception

Image concernant la méthode de gestion de projet Rational Unified Process (IBM). Consulté le 12 juillet 2010. http://lgl.isnetne.ch/isnet43/phase4/images/rup_concepts.jpg

Image concernant la méthode de gestion de projet Cascade (Waterfall). Consulté le 12 juillet 2010. http://lagace.developpez.com/extreme-programming/images/XP-Cascade02-thumb.gif

Wikipédia – Architecture logicielle et modélisation. Consulté le 12 juillet 2010. http://fr.wikipedia.org/wiki/Architecture_logicielle#La_vue_des_cas_d.27utilisation

Définitions Java (attributs). Consulté le 12 juillet 2010. http://www.vulgarisation-informatique.com/java-attributs.php

Définitions Java (méthodes). Consulté le 12 juillet 2010. http://www.vulgarisation-informatique.com/java-methodes.php

- **Définition des frameworks :**

Définition des frameworks. Consulté le 12 juillet 2010. http://www.techno-science.net/?onglet=glossaire&definition=1471

MySQL Workbench. Consulté le 12 juillet 2010**.** http://www.mysql.fr/products/workbench/

Architecture client-serveur. Consulté le 12 juillet 2010**.** http://fr.wikipedia.org/wiki/Client-serveur

- **Symfony™ :**

Installation de Symfony. Consulté le 12 juillet 2010**.** http://www.symfony-project.org/book/1_0/02-Exploring-Symfony-s-Code

Création de projet avec Symfony. Consulté le 12 juillet 2010**.** http://www.symfony-project.org/jobeet/1_4/Doctrine/fr/01

Les bases de Symfony. Consulté le 12 juillet 2010**.** http://bydorian.com/faire-un-site-avec-symfony-les-bases/

Démarrer un projet avec Symfony. Consulté le 12 juillet 2010**.** http://doc.ubuntu-fr.org/tutoriel/demarrer_un_projet_web_avec_symfony

La création de formulaires avec Symfony. Consulté le 12 juillet 2010**.** http://www.symfony-project.org/forms/1_1/fr/01-Form-Creation

Snow Leopard et Symfony. Consulté le 12 juillet 2010**.** http://phpmn.blogspot.com/2009/09/snow-leopard-and-symfony.html

- **Zend™ framework et Doctrine™ :**

Création d'un premier projet avec Zend. Consulté le 12 juillet 2010**.** http://www.alexandre-julien.com/zend-framework-tutoriaux-complets/zend-framework-tutoriel-partie-1-installation-du-framework-et-creation-dun-premier-projet/

Description de l'outil Zend_Tool. Consulté le 12 juillet 2010**.** http://blog.lyrixx.info/zend/initialisation-dun-projet-zend/

Créer une application avec Zend framework. Consulté le 12 juillet 2010**.** http://www.dator.fr/tutorial-creer-une-application-avec-le-zend-framework-1-preparation-de-lenvironnement-de-watchmydesk/

Exemples avec Zend framework. Consulté le 12 juillet 2010**.** http://github.com/abtris/ZF-Examples-My-Quickstart

Zend Framework et Doctrine. Consulté le 12 juillet 2010**.** http://www.itanea.com/blog/2008/11/27/zend-framework-utiliser-zend-framework-et-doctrine-auteur-ruben-vermeersch-traduction-fred-blanc/

Throrïn's Studio - Introduction au ZF - Partie 5 : Doctrine. Consulté le 12 juillet 2010**.** http://www.throrinstudio.com/blog/index/article/idarticle/51

Introducing Doctrine 1.2 Integration | free Zend Framework screencasts - Zendcasts. Consulté le 12 juillet 2010**.** http://www.zendcasts.com/introducing-doctrine-1-2-integration/2009/11/

Zend Framework: Documentation: Create A Form - Zend Framework Manual. Consulté le 12 juillet 2010**.** http://framework.zend.com/manual/en/learning.quickstart.create-form.html

Zend Framework / Contrôleur CRUD abstrait. Consulté le 12 juillet 2010**.** http://www.z-f.fr/forum/viewtopic.php?pid=4750#p4750

Comment utiliser l'outil Zend_Tool en ligne de commande | LyRiX Blog. Consulté le 12 juillet 2010**.** http://blog.lyrixx.info/zend/comment-bien-demarrer-un-projet-zend-framework-grace-aux-zend_tool/

Zend Framework / [_forward ou _redirect] la différence. Consulté le 12 juillet 2010**.** http://www.z-f.fr/forum/viewtopic.php?id=1637

- **Struts™ et Hibernate™:**

Netbeans, Tomcat, Struts et Hibernate (partie 1). Consulté le 12 juillet 2010**.** http://www.bourzeix.com/weblog/post/2005/05/21/121-netbeans-tomcat-struts-et-hibernate

Netbeans, Tomcat, Struts et Hibernate (partie 2). Consulté le 12 juillet 2010**.** http://www.bourzeix.com/weblog/post/2005/06/09/122-netbeans-une-application-web-hibernate

Netbeans, Tomcat, Struts et Hibernate (partie 3). Consulté le 12 juillet 2010**.** http://www.bourzeix.com/weblog/post/2005/07/10/139-netbeans-une-application-web-hibernate-et-servlet-partie-iii

YouTube – NetBeans 6.7.1 Struts 2 Hibernate 3. Consulté le 12 juillet 2010. http://www.youtube.com/watch?v=z4ys0dXST94

Struts 2 Plugin for NetBeans IDE. Consulté le 12 juillet 2010. http://beans.seartipy.com/2008/08/04/struts-2-plugin-for-netbeans-ide-nbstruts2support/

Documents & files : Release Modules. Consulté le 12 juillet 2010. https://nbstruts2support.dev.java.net/servlets/ProjectDocumentList?folderID=9422&expandFolder=9422&folderID=11970

Apache Tomcat. Consulté le 12 juillet 2010. http://tomcat.apache.org/

NetBeans IDE for Development. Consulté le 12 juillet 2010. http://dlc.sun.com/osol/docs/content/OSDEV/gentextid-605.html

Using Hibernate in a Web Application - NetBeans IDE Tutorial. Consulté le 12 juillet 2010. http://netbeans.org/kb/docs/web/hibernate-webapp.html

NetBeans Struts Login. Consulté le 12 juillet 2010. http://qa.netbeans.org/modules/webapps/promo-h/struts-user-scenario.html

Introduction à Java EE. Consulté le 12 juillet 2010. http://cherkaoui.developpez.com/tutoriels/j2ee/introduction-a-java-ee/?page=6-struts

Image concernant le fonctionnement d'un framework (Struts). Consulté le 12 juillet 2010. http://www.ibm.com/developerworks/websphere/techjournal/0302_fung/images/image1.jpg

Developing a project with Struts 2 and Hibernate 3 in Netbeans 6.5 - Rahul Arya. Consulté le 24 août 2010. http://www.rahularya.com/1/post/2010/03/developing-a-project-with-struts-2-and-hibernate-3-in-netbeans-65.html

Langage de requêtage d'Hibernate. Consulté le 24 août 2010. http://docs.jboss.org/hibernate/core/3.3/reference/fr/html/queryhql.html

Using Hibernate in a Web Application - NetBeans IDE Tutorial. Consulté le 24 août 2010. http://netbeans.org/kb/docs/web/hibernate-webapp.html

Struts - Iterator. Consulté le 24 août 2010. http://struts.apache.org/2.0.14/docs/iterator.html

Tags, expressions OGNL et EL expressions avec Struts 2. Consulté le 24 août 2010. http://www.olivier-guillou.fr/oneandcie/ognl-et-el-expressions-avec-struts-2-acces-portee-utilisation

Tutoriel Struts2 – Réaliser un formulaire identification login avec session et OGNL en utilisant MVC 2. Consulté le 24 août 2010. http://www.olivier-guillou.fr/oneandcie/tutoriel-struts2-realiser-un-formulaire-de-login-avec-session

Problème de digest en SHA1 sur java 1.6. Consulté le 24 août 2010. http://www.developpez.net/forums/d393355/java/general-java/java-mobiles/java-me/probleme-digest-sha1-java-1-6-a/

- **CMS :**

Le fonctionnement d'un CMS en image | Higherweb. Consulté le 24 août 2010. http://www.higherweb.fr/le_blog/2008/11/le-fonctionnement-dun-cms-en-image/

Image concernant la structure d'un CMS. Consulté le 24 août 2010. http://ressources.info.free.fr/spip/IMG/jpg/CMS_Diagram_l.jpg

Petit comparatif Joomla et Drupal : Concept, Performances. Consulté le 24 août 2010. http://tuto.dashcircle.com/index.php/home/3-administration-web/85-petit-comparatif-joomla-contre-drupal

Concepts et terminologie | Drupal France et francophonie. Consulté le 24 août 2010. http://drupalfr.org/node/4679

Joomla vs Drupal - essais de performance. Consulté le 24 août 2010. http://forum.joomla.org/viewtopic.php?f=158&t=94653

- **QCodo™ :**

Qcodo Development Framework. Consulté le 24 août 2010. http://www.qcodo.com/

- **Les frameworks et leurs évolutions :**

Vidéo "Parole d'expert": Le framework Symfony par son créateur Fabien Potencier. Consulté le 24 août 2010. http://www.intelli-n.tv/Paroles-d-Expert/Video-Parole-d-expert-Le-framework-Symfony-par-son-createur-Fabien-Potencier

Frameworks open source : attention aux dépendances ! Consulté le 24 août 2010. http://www.indexel.net/applications/frameworks-open-source-attention-aux-dependances.html

PHP : L'avenir du langage et ses frameworks d'ici fin 2010 - Vincent Composieux - Développeur web PHP5 Symfony / Zend Framework. Consulté le 24 août 2010. http://vincent.composieux.fr/2009/11/22/lavenir-de-php-et-ses-frameworks/

Vers une convergence des frameworks PHP ? | Industrialisation PHP. Consulté le 24 août 2010. http://www.industrialisation-php.com/vers-une-convergence-des-frameworks-php/

Symfony, un framework nouvelle génération. Consulté le 24 août 2010. http://www.mti.epita.fr/blogs/2010/06/11/symfony-un-framework-nouvelle-generation/

Zend Framework / Zend Framework 2.0 et Symfony 2.0 en 2010. Consulté le 24 août 2010. http://www.z-f.fr/forum/viewtopic.php?id=4410

Zend Framework 2.0 : première version de développement. Consulté le 24 août 2010. http://www.paperblog.fr/3507615/zend-framework-20-premiere-version-de-developpement/

Struts 1 vs Struts 2. Consulté le 24 août 2010. http://blog.valtech.fr/wordpress/2007/06/28/struts-1-vs-struts-2/

"Struts 1 ou Struts 2". Consulté le 24 août 2010. http://blog.lecacheur.com/2007/11/15/struts-1-ou-struts-2/

- **Glossaire :**

Wikipédia – API ou interface de programmation. Consulté le 24 août 2010. http://fr.wikipedia.org/wiki/Interface_de_programmation

DOS - Wikibooks. Consulté le 24 août 2010. http://fr.wikibooks.org/wiki/DOS

Les composants Java beans. Consulté le 24 août 2010. http://www.jmdoudoux.fr/java/dej/chap025.htm

Wikipédia – Open source. Consulté le 24 août 2010. http://fr.wikipedia.org/wiki/Open_source

Wikipédia – File Transfer Protocol. Consulté le 24 août 2010. http://fr.wikipedia.org/wiki/File_Transfer_Protocol

Wikipédia – Shell (informatique). Consulté le 24 août 2010. http://fr.wikipedia.org/wiki/Shell_(informatique)

Url - Définition. Consulté le 24 août 2010. http://www.graphic-evolution.fr/definition/url-820.html

Wikipédia – PL/SQL. Consulté le 24 août 2010. http://fr.wikipedia.org/wiki/PL/SQL

18. Annexes

18.1 Questionnaire pour les entreprises

Questionnaire concernant les frameworks pour les entreprises

Bonjour,

Voici le questionnaire concernant les frameworks pour les applications web auquel vous avez accepté de répondre suite à notre entretien téléphonique.

Vos réponses seront exploitées de façon anonyme dans le but de réaliser un chapitre de mon mémoire de bachelor concernant le choix d'un framework pour l'entreprise.

Il comporte 16 questions (selon vos réponses vous ne répondrez pas à toutes ces questions) et il peut être effectué en 10 minutes.

Bien évidemment, si vous le souhaitez, je vous transmettrais mon dossier dès la fin de mon travail (début septembre).

Je vous remercie encore de votre participation.

Bien à vous.

Arielle Moro

* Required

Question n°1 : Quel est le type de votre entreprise ? *

- ○ Micro-entreprise (< 10 salariés)
- ○ Petite entreprise (de 10 à 49 salariés)
- ○ Moyenne entreprise (de 50 à 249 salariés)
- ○ Grande entreprise (250 salariés et plus)

Question n°2 : Est-ce une entreprise spécialisée dans l'informatique ? *

- ○ Oui (allez à la question n°4)
- ○ Non (allez à la question n°3)

Question n°3 : Quel est votre principal secteur d'activité ?

- ○ Bancaire
- ○ Comptabilité
- ○ Marketing - Communication
- ○ Other:

Question n°4 : Faites-vous du développement informatique au sein de votre entreprise ou de votre service informatique? *

- ○ Oui (allez à la question n°5)
- ○ Non (allez à la question n°8)

Question n°5 : Utilisez-vous un framework pour créer des applications web ?

- ○ Oui (allez à la question n°9)
- ○ Non (allez à la question n°6)
- ○ Non, j'utilise un CMS (allez à la question n°7)

Question n°6 : Pourquoi n'en avez-vous pas choisi ?

Question n°7 : Quel CMS utilisez-vous ? Quels sont ses avantages et ses inconvénients ?

Question n°8 : Si vous n'utilisez pas de framework ou que vous ne faites pas de développement, dans le cas où vous devez créer dans l'avenir une application, choisirez-vous un framework pour la développer ?

- ○ Oui
- ○ Non
- ○ Non, un CMS me conviendrait mieux

Question n°9 : Si vous utilisez un framework, lequel utilisez-vous ?

- [ ] Symfony (PHP)
- [ ] Zend (PHP)
- [ ] Struts (Java)
- [ ] Un framework développé par l'entreprise ou le service informatique (allez à la question n°10)
- [ ] Other:

Question n°10 : Pourquoi avez-vous choisi de développer un framework vous-même ?

Question n°11 : Si vous utilisez un framework, décrivez brièvement la raison du choix de ce framework :

Question n°12 : Selon vous, existe-t'il des risques à utiliser un framework ?

Risques liés au développement, risques liés aux attaques d'applications web....

- ○ Oui
- ○ Non

Question n° 13 : Selon vous, quels sont les avantages et les inconvénients d'un framework ?

Question n°14 : Classez par ordre d'importance les critères suivants relatifs aux frameworks : sécurité (1), maintenance (2), tests unitaires et fonctionnels (3), outils de débug (logs...) (4), routage (5), internationalisation (gestion de langues) (6), extensibilité (7), système d'authentification - session (8), facilité de déploiement (9):
Recopiez uniquement les chiffres (du critère le plus au moins important)

Question n°15 : Avez-vous été amené à étoffer votre framework ?
C'est-à-dire créer de nouveaux composants bien précis pour le développement de vos applications.

- ○ Oui
- ○ Non

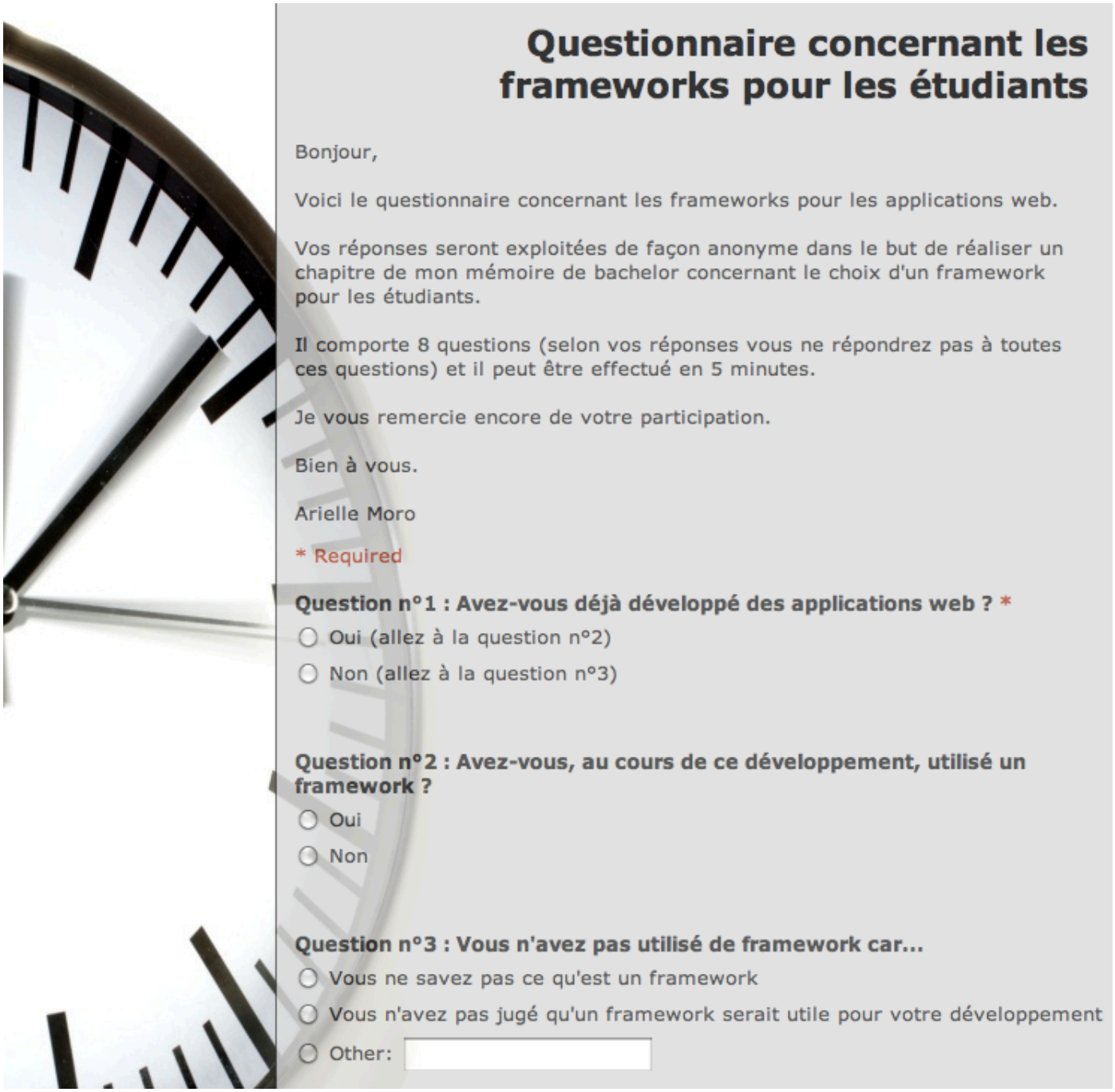
Question n°16 : Si vous deviez améliorer un framework tel que Symfony ou Zend, quelles seraient ces améliorations ?
Améliorations techniques ou structurelles

Submit

Powered by Google Docs

Report Abuse - Terms of Service - Additional Terms

18.2 Questionnaire pour les étudiants



Questionnaire concernant les frameworks pour les étudiants

Bonjour,

Voici le questionnaire concernant les frameworks pour les applications web.

Vos réponses seront exploitées de façon anonyme dans le but de réaliser un chapitre de mon mémoire de bachelor concernant le choix d'un framework pour les étudiants.

Il comporte 8 questions (selon vos réponses vous ne répondrez pas à toutes ces questions) et il peut être effectué en 5 minutes.

Je vous remercie encore de votre participation.

Bien à vous.

Arielle Moro

* Required

Question n°1 : Avez-vous déjà développé des applications web ? *

- ○ Oui (allez à la question n°2)
- ○ Non (allez à la question n°3)

Question n°2 : Avez-vous, au cours de ce développement, utilisé un framework ?

- ○ Oui
- ○ Non

Question n°3 : Vous n'avez pas utilisé de framework car...

- ○ Vous ne savez pas ce qu'est un framework
- ○ Vous n'avez pas jugé qu'un framework serait utile pour votre développement
- ○ Other:

Question n°4 : Si vous avez déjà utilisé un framework, lequel avez-vous utilisé ?

- [ ] Symfony (Php)
- [ ] Zend (Php)
- [ ] Struts (Java)
- [ ] Other:

Question n°5 : Pensez-vous qu'il soit bien de débuter une toute première application avec un framework très évolué. Avec un framework moins élaboré on comprend parfois mieux la structure d'un framework. *

- ○ Oui
- ○ Non
- ○ Other:

Question n°6 : Avez-vous déjà utilisé un CMS ? *

- ○ Oui
- ○ Non

Question n°7 : Si oui, lequel avez-vous utilisé ?

- ○ Joomla
- ○ Drupal
- ○ Other:

Question n°8 : Quels sont les points forts et les points faibles d'un framework selon vous ? Et quels sont les points à améliorer ?

18.3 Fiches

8.5 Fiche n°1 : Notions préliminaires

1. Programmation orientée objet

Paradigme objet : Héritage, liaison dynamique, masquage, encapsulation.

Classe : attribut(s) + méthode(s).

Objet : instance d'une classe.

Classe abstraite : attribut(s) + méthode(s) concrète(s) et abstraite(s) (doit être étendue par une autre classe).

Interface : signature(s) de méthode(s) (doit être implémentée par une autre classe).

2. Base de données

Base de données : composée d'objet(s) de schéma tels que une ou plusieurs table(s), vue(s), procédure(s)...

Table : composée d'un ou plusieurs champ(s) (corps de la table) dont au moins une clé primaire et qui contient des enregistrements.

Clé primaire : champ d'identification d'une enregistrement d'une table (unique).

Clé étrangère : champ qui permet de faire des liaisons entre deux ou plusieurs tables (correspond généralement à la clé primaire d'une autre table).

Modèles principaux pour réaliser une bonne base de données : modèle conceptuel de données (MCD), modèle logique de données (MLD) et modèle relationnel de données.

3. Design Patterns

Définition : Brique d'architecture logicielle réutilisable qui répond à un problème récurrent.

4. Tests d'application

Principaux tests d'application : Tests unitaires et tests fonctionnels.

9.7 Fiche n°2 : Qu'est-ce qu'un framework ?

1. Définition

Un Framework est un outil de haut niveau qui est composé d'une multitude de composants qui aide à l'élaboration d'une application le plus souvent (mais il existe aussi des frameworks qui ont d'autres finalités tel que le mapping relationnel-objet). Le framework possède une architecture qui lui est propre et qu'il faudra respecter tout le long du développement.

2. Architecture Modèle-Vue-Contrôleur

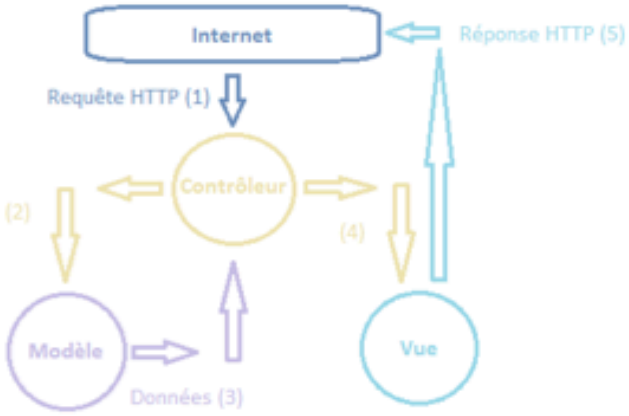


Figure 16 : Architecture Modèle-Vue-Contrôleur (fiche n°1)

3. Mapping relationnel-objet

Le mapping relationnel-objet permet de relier une base de données relationnelle avec les notions objets liées au langage de programmation objet choisi. Il intervient lors de la génération de classes par le framework.

4. Points positifs et négatifs

Points positifs : Gagner du temps, améliorer la maintenance, l'organisation de l'application, avoir des normes de développement, mieux répartir le travail.

Points négatifs : Temps d'apprentissage non négligeable, poids important, mise à jour plus ou moins difficile.

10.4 Fiche n°3 : Analyse de frameworks Php et Java

Comparaison des frameworks

Critères / Frameworks	Symfony™	Zend™	Struts™
Prise en main du framework	5/5	4/5	4/5
Installation	5/5	2/5	4/5
Routing	5/5	5/5	5/5
Mapping relationnel-objet	5/5	3/5	5/5
Formulaires	5/5	5/5	4/5
Sessions	3/5	3/5	5/5
Organisation du code	5/5	5/5	4/5
Interactions avec la base de données	5/5	5/5	5/5
Documentation sur les APIs	4/5	4/5	4/5
Outils de développement	5/5	3/5	5/5
Feuilles de style CSS	5/5	5/5	5/5
Composants élaborés	5/5	5/5	3/5
Internationalisation	5/5	5/5	5/5
Note finale	**4.8/5**	**4.2/5**	**4.4/5**

Table 5 : Comparaison des frameworks Symfony™, Zend™ et Struts™ (fiche n°3)

Ce tableau met en évidence que ces trois solutions sont relativement appropriées pour réaliser une bonne application web. Les différences majeures sont le langage de programmation et évidemment le poids du framework car, en choisissant une application Java, celui-ci est souvent plus important. C'est pour cela que l'on utilise un framework Java, le plus souvent, pour des applications web très spécifiques.

11.3 Fiche n°4 : Une solution alternative au framework : le CMS

1. Définition

Le CMS est un ensemble de logiciels servant à gérer du contenu de façon simplifié, par le biais d'interfaces. Avec ceci nous pouvons facilement réaliser et mettre à jour des sites Internet dynamiques. Ils permettent de travailler en équipe avec des gens qui n'ont pas forcément besoin d'avoir des notions très pointues en informatique, s'agissant de mettre à jour le contenu d'un site. Cependant un informaticien sera indispensable pour personnaliser le CMS, rajouter des plugins ou des modules.

2. Architecture

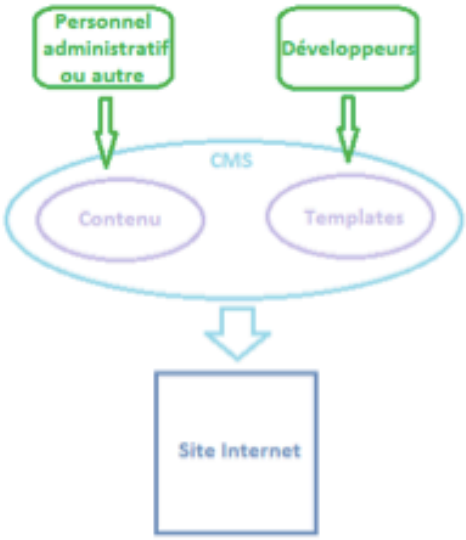


Figure 30 : Structure d'un CMS (fiche n°4)

3. Points positifs et négatifs

Points positifs : Création d'un site Internet en quelques minutes, maintenance facile et pouvant être effectuée par des non informaticiens (informaticiennes) (excepté les mises à jour très techniques ou la personnalisation du CMS).

Points négatifs : Structure imposée des articles et du contenu ainsi que difficulté parfois à ajouter du code pur au sein du CMS.

12.4 Fiche n°5 : Choix d'un framework

1. Pour l'entreprise

Critères de sélection d'un framework (du plus au moins important) :

1. Sécurité
2. Maintenance
3. Présence d'outils aidant au développement
4. Présence de système d'authentification
5. Facilité de déploiement
6. Tests unitaires et fonctionnels facilités
7. Ajout d'extensions
8. Routage
9. Internationalisation

Classement des frameworks :

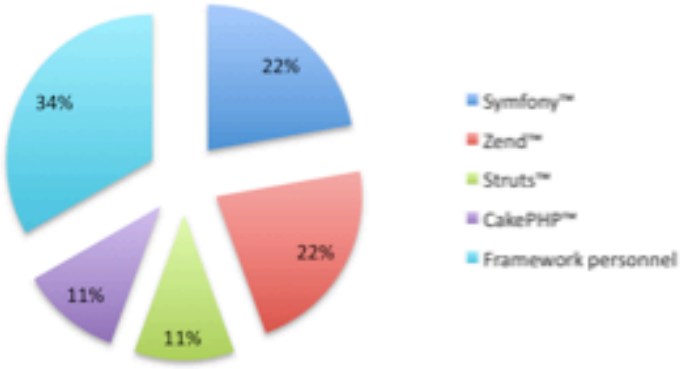


Figure 44 : Frameworks utilisés par les entreprises (fiche n°5)

2. Pour les étudiants

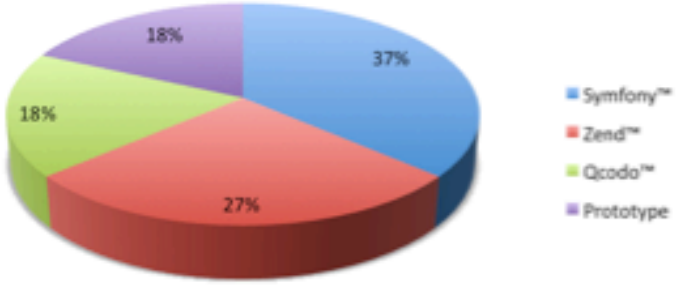
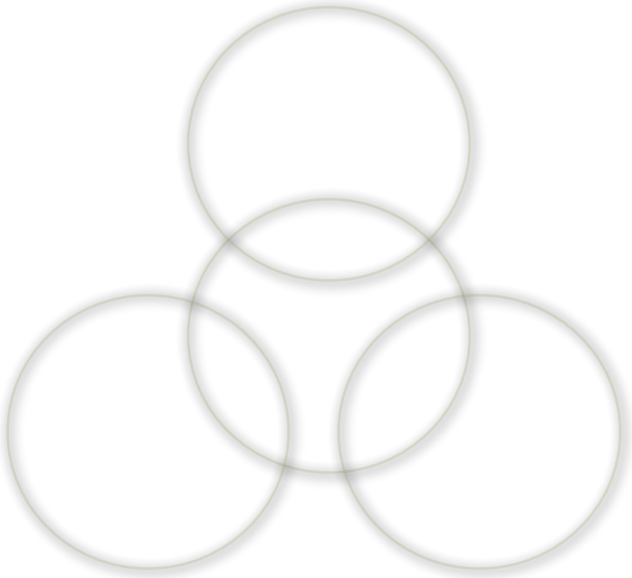


Figure 45 : Frameworks utilisés par les étudiants (fiche n°5)



Etudiante : Arielle Moro

arielle.moro@gmail.com

Directeur de mémoire : Peter Daehne

peter.daehne@hesge.ch

Haute Ecole de Gestion de Genève (HEG)

www.hesge.ch/heg

Site Internet du mémoire de bachelor

http://www.ariellemoro.com/TDFrameworks