

Adopter un style de programmation clair avec le modèle MVC

Par vincent1870



www.openclassrooms.com

*Licence Creative Commons 7 2.0
Dernière mise à jour le 13/03/2011*

Sommaire

Sommaire	2
Adopter un style de programmation clair avec le modèle MVC	3
Présentation théorique	3
Le contrôleur	4
Le modèle	4
La vue	4
Organiser les fichiers sur le FTP	4
Organisation simple	5
Une autre organisation possible	5
Redirections	6
En conclusion	6
Application pratique : un système de news	6
Le système de base	6
Correction	8
Résumé de la structure FTP	10
D'autres approches	10
Un système de templates !	10
Séparer les modèles	10
Utiliser un framework	11
Partager	11



Adopter un style de programmation clair avec le modèle MVC

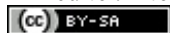


Par [vincent1870](#)

Mise à jour : 13/03/2011

Difficulté : Intermédiaire

Durée d'étude : 3 jours



Bonjour à tous !

Je vais aujourd'hui m'attaquer à un problème que tout programmeur a rencontré un jour ou l'autre dans sa carrière : comment organiser le code de son site de manière à être le plus clair possible ? Il existe de nombreuses solutions, chacune ayant ses avantages et ses inconvénients.

Je vais ainsi vous présenter une façon de faire que j'aime beaucoup. Ce n'est sans doute pas la meilleure, ni la pire, juste une parmi tant d'autres. Essayez, et forgez-vous votre opinion après cela.

Le modèle dont je vais vous parler se nomme MVC pour les intimes (nous allons voir dans le premier chapitre ce que cela signifie).

Difficulté : après avoir lu tous les cours de M@teo21 et quelques tutos rédigés par des Zér0s, vous pourrez vous en sortir. Ce n'est pas très dur, mais il vaut mieux avoir une bonne connaissance en PHP avant de se lancer dans ce tuto, je pense. Je vous donnerai des liens vers d'autres tutos par la suite si vous bloquez sur tel ou tel point.

Ceci étant dit, je vous souhaite une bonne lecture. 😊

Sommaire du tutoriel :



- Présentation théorique
- Organiser les fichiers sur le FTP
- Application pratique : un système de news
- D'autres approches

Présentation théorique



Note aux puristes : le modèle présenté ne sera peut-être pas très rigoureux à vos yeux. Ce n'est qu'une approche n'ayant pas pour but de dégoûter le lecteur à sa première lecture. J'ai repris le terme de **modèle MVC** car il s'en approche beaucoup, et parce que le terme est très connu. 😊

Tout d'abord, vous devez vous demander ce que signifie cet acronyme énigmatique MVC. MVC sont les initiales de

Modèle Vue Contrôleur.

Comme vous pouvez vous en douter, l'organisation se fera en trois parties. Au lieu d'un fichier utilisé par une organisation de la programmation classique (tout dans un fichier), le modèle MVC divisera le tout en trois fichiers.



Mais cela va beaucoup alourdir mon FTP ! Et puis, ce n'est pas très pratique d'avoir trois fichiers, alors qu'on peut faire n'en faire qu'un... 😊

À cette question, je répondrai qu'il faut savoir ce que l'on veut. 🤖 Le gain réside dans le fait que votre code sera plus lisible qu'avec un seul fichier. C'est le but de cette fragmentation. L'inconvénient d'avoir plusieurs fichiers n'en est pas un, je pense. Ce n'est pas dérangeant.

Le contrôleur

Nous allons commencer par le contrôleur, et vous allez comprendre pourquoi. Quand vous demanderez un fichier dans votre barre d'adresse, ce sera en fait le contrôleur que vous appellerez. C'est le fichier qui contient toute la logique du code. Vous commencerez généralement par y inclure votre modèle (nous verrons plus loin de quoi il s'agit). Vous procéderez à quelques vérifications d'ordre général (comme les autorisations), puis vous pourrez appeler des fonctions avant d'inclure la vue.

Comme vous le voyez, c'est la partie du système qui fait le lien avec les deux autres parties.



Petite précision : nous verrons plus tard qu'un fichier `index.php` se chargera de faire le lien entre les informations demandées par l'URL et les fichiers. Ce fichier `index.php` sera lui-même un contrôleur (appelé *FrontController*), par opposition aux autres contrôleurs (appelés les *BackController*). Son rôle est de procéder aux initialisations nécessaires, et d'appeler le *BackController*.

Le modèle

Ce fichier ne contiendra que des fonctions. Le modèle a pour but de gérer l'organisation des données. Chacune de ces fonctions effectuera une action bien précise. **Vous ne pourrez effectuer des requêtes SQL que dans ces fonctions !** Le contrôleur et la vue ne devront contenir aucun appel à MySQL. Réfléchissez-donc bien aux fonctions que vous y mettrez.



Quand je dis *requêtes SQL*, cela inclut aussi bien MySQL, que PostgreSQL, ou tout autre SGBDR. De même, les bases de données ne sont pas le seul moyen de gérer des données. Le modèle pourra donc tout aussi bien contenir des fonctions gérant des fichiers XML, par exemple. 🤖

Vous aurez au final sans doute des fonctions du type : `ajouter_news`, `supprimer_news`, `ajouter_commentaire_news` pour un script de news.

Vous pouvez bien évidemment aussi stocker des fonctions classiques dans ce fichier, mais en général, ce type de fonctions est souvent réutilisé dans d'autres fichiers, et vous préférerez les mettre dans un fichier à part, inclus lors de l'initialisation de votre script.

La vue

La vue contient le code xHTML. C'est la seule partie qui doit en contenir. Vous pouvez à la limite faire quelques `echo` depuis le contrôleur, mais pour du débogage ou l'affichage d'erreurs. Sinon, ce sera la vue qui s'en chargera. Elle s'occupe donc de l'interface homme / machine.

La vue est un moteur de templates, dans le sens où logique du code et affichage sont séparés. Le code PHP doit être très limité dans ce fichier. Personnellement, je n'emploie que des `if`, `foreach` et des `echo` dans la majorité des cas. Après, vous pouvez y joindre un moteur de templates dans le sens courant du terme (voir plus loin), mais cela donne un peu plus de boulot au contrôleur. Ici, cela revient à employer PHP comme moteur de templates (ce qui devrait faire plaisir à certains 🤖).

Enfin, notez que restreindre la vue à du code xHTML est réducteur. La vue s'occupe de l'interface pour l'utilisateur. Il peut donc en théorie s'agir d'un peu tout et n'importe quoi, comme du XML par exemple, voire du texte brut ! Mais l'affichage de xHTML est, vous vous en doutez, le plus employé dans le cadre de la conception d'un site web, même si vous serez sans doute amenés à afficher des flux RSS (en XML donc). Certains sites proposent aussi une API retournant des résultats dans d'autres formats que le xHTML, comme du Yaml, XML, Json, etc.

Organiser les fichiers sur le FTP

Maintenant que les bases sont posées, commençons à réfléchir à l'organisation des fichiers sur le FTP. Personnellement, j'entrevois plusieurs solutions. Dans tous les cas, le principe sera de séparer les trois éléments : **modèle**, **vue** et **contrôleur** (parce que ça ne fait pas de mal de les réécrire une fois de plus 😊).

Organisation simple

La solution la plus simple est de créer trois dossiers : **modeles**, **vues** et **controleurs**. Dans chacun de ces dossiers, vous mettrez l'élément correspondant.

À partir de là, vous avez deux solutions pour gérer les appels de fichiers. Vous pouvez donc soit appeler directement le contrôleur depuis l'URL, avec une adresse de la sorte :

<http://monsiteamoi.net/controleurs/forum.php>,

soit l'appeler en passant par un **index.php** faisant le lien entre les contrôleurs avec une adresse de la forme :

<http://monsiteamoi.net/index.php?page=forum>.

Dans le premier cas, il faudra que votre contrôleur et / ou votre vue inclue(nt) les éléments communs à votre site (connexion à la BDD, en-tête, pied de page, etc.). Dans le deuxième cas, c'est l'**index.php** qui effectue toutes les initialisations, qui vérifie si le contrôleur existe bien (pour empêcher un piratage) et l'inclut si tel est le cas. Pour ma part je vous conseille la seconde solution qui permet d'éviter des duplications de code inutiles. Vous pouvez commencer à coder directement dans votre fichier, puisque toutes vos initialisations sont faites dans l'**index.php** ! De plus, combiné avec de l'*url rewriting*, les adresses pourront être plus jolies (pour les visiteurs mais aussi les moteurs de recherche).

Voici un exemple d'un fichier d'**index** type :

Code : PHP

```
<?php
//On démarre la session
session_start();

//On se connecte à MySQL
mysql_connect('localhost', 'root', '');
mysql_select_db('tests');

//On inclut le logo du site et le menu
include 'vues/logo.php';
include 'vues/menu.php';

//On inclut le contrôleur s'il existe et s'il est spécifié
if (!empty($_GET['page']) &&
is_file('controleurs/'.$_GET['page'].'.php'))
{
    include 'controleurs/'.$_GET['page'].'.php';
}
else
{
    include 'controleurs/accueil.php';
}

//On inclut le pied de page
include 'vues/pied.php';

//On ferme la connexion à MySQL
mysql_close();
```



Ce fichier est très simple et pas encore très sécurisé, mais il peut très bien vous servir de base de travail si vous décidez de procéder avec cette méthode.

Une autre organisation possible

Cette variante est similaire à l'organisation utilisée avec un moteur de templates. D'ailleurs, si vous comptez en adjoindre un (voir la dernière partie), vous devriez songer à l'utiliser.

La principale différence est que vous allez mettre tous les contrôleurs à la racine du site. Vous noterez que pour les appeler, vous aurez encore le choix entre les deux méthodes (voir ci-dessus pour plus de détails). Il vous faudra ensuite deux autres dossiers pour y mettre les vues et les modèles. Vous les nommerez comme vous le désirez, mais sachez que la majorité des moteurs de templates exigent qu'on les nomme 'templates' (original 🤖), quoique ce paramètre peut très souvent être modifié. Vous vous retrouvez alors avec une structure quelque peu différente, avec plus de fichiers à la racine, mais qui pourra peut-être paraître plus logique à certains.

Redirections

Certains m'ont fait la remarque du problème posé par les redirections, j'en parle donc ici. Comme vous le savez, une redirection se fait habituellement en PHP avec un :

Code : PHP

```
<?php header('Location: cible.php'); ?>
```

Or un tel appel de fonction **ne peut pas être mis si quelque chose a déjà été envoyé au navigateur**. Pour cela, et dans l'optique de garder l'organisation actuelle, une seule solution : la tamponisation de sortie ([tutoriel d'Alex ici](#)). Globalement, vous mettez un `ob_start` en début de script, un `ob_end_flush` en fin, et vous pouvez, entre les appels à ces deux fonctions, insérer divers éléments habituellement interdits en milieu de fichiers PHP comme des `setcookie` et des `header`.

Cependant, je n'emploierai pas cette technique par la suite, pour simplifier les choses. Sachez simplement qu'elle existe et qu'elle peut être très utile. 😊

En conclusion

Vous choisissez l'organisation que vous voulez, voire une autre de votre cru. Le principal est que vous soyez à l'aise avec. Un petit conseil pendant que j'y suis : protégez les répertoires des vues et des modèles avec un `.htaccess`. Si vous employez la méthode à base de `index.php?page=`, protégez aussi le répertoire des contrôleurs (si celui-ci n'est pas la racine 🤖). Cela pourra éventuellement éviter quelques piratages.

Pour plus d'infos sur la création et l'utilisation des `.htaccess` : [le tutoriel officiel de M@teo21](#) et [un tutoriel de kozo sur les usages de ce fichier bien particulier](#).

Application pratique : un système de news

Cette partie sera un mini-TP. Je vais vous fournir un système de news tout fait, et vous allez me l'adapter en suivant le modèle MVC que je vous ai décrit plus haut.

À l'assaut du MVC ! 🧑🏻‍💻

Le système de base

Voici donc le code PHP servant à afficher les news. Vous noterez qu'il est très simple : la difficulté n'est pas le but de ce TP. 😊

Il emploie une table MySQL news caractérisée ainsi :

- id / int / *clé primaire* / auto-increment
- titre / varchar
- auteur / varchar
- date / datetime

- contenu / text.

Voici le code pour créer la table facilement, avec quelques données d'exemple :

Code : SQL

```
CREATE TABLE `news` (
  `id` SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  `titre` VARCHAR( 255 ) NOT NULL ,
  `auteur` VARCHAR( 255 ) NOT NULL ,
  `date` DATETIME NOT NULL ,
  `contenu` TEXT NOT NULL
) ENGINE = MYISAM ;

INSERT INTO `news` (
  `id` ,
  `titre` ,
  `auteur` ,
  `date` ,
  `contenu`
)
VALUES (
  NULL , 'Une première news', 'vincent1870', '2007-12-30 18:38:02',
  'Bienvenue à tous sur ce beau site !<br /> <br /> Bon surf ! ;)'
), (
  NULL , 'Et une deuxième', 'Arthur', '2007-12-11 18:38:44', 'Hello
  !<br /> What happened ?'
);
```



Je considérerai dans le script d'affichage que les données entrées dans la BDD sont déjà sécurisées (insérées avec htmlspecialchars), converties en xHTML avec nl2br et éventuellement une fonction de parsing. Elles sont de plus dépourvues de slashes gênants.

Le code de l'[index.php](#) :

Code : PHP

```
<?php
//On démarre la session
session_start();

//On se connecte à MySQL
mysql_connect('localhost', 'root', '');
mysql_select_db('tests');

//On inclut le fichier s'il existe et s'il est spécifié
if (!empty($_GET['page']) && is_file($_GET['page'].'.php'))
{
    include $_GET['page'].'.php';
}
else
{
    include 'accueil.php';
}

//On ferme la connexion à MySQL
mysql_close();
```

Et le code d'une page [news.php](#), située à la racine du site pour le moment :

Code : PHP

```

<h1>Les news du site</h1>

<?php
$req = mysql_query("SELECT id, auteur, titre, DATE_FORMAT(date,
'%d/%m/%Y %H') AS date_formatee, contenu
FROM news
ORDER BY date DESC");
while ($data = mysql_fetch_assoc($req))
{
    echo '
<div class="news">
<h2>'.$data['titre'].'</h2>
<p>News postée le '.$str_replace(' ', ' à ',
$data['date_formatee']).' par '.$data['auteur'].'</p>
<p>'.$data['contenu'].'</p>
</div>';
}
?>

```

Ce code est très simple, vous devriez le comprendre sans peine. Si ce n'est pas le cas, vous pouvez relire [la partie sur la récupération de données du tuto de M@teo21](#) et [le tuto de Machin sur les DATETIME de MySQL](#).

Bon, maintenant, à vous de me faire un joli modèle MVC. Vous utiliserez le premier système décrit avec :

- trois dossiers : modeles, vues et controleurs ;
- un fichier `index.php`, comme je l'ai fait ici, en le modifiant bien sûr.

Correction

Puisque vous avez fini, voici venue l'heure de la correction, commentée bien entendu. 😊

Pour l'`index.php`, je reprends celui que je vous avais donné dans la partie précédente. Il se situera à la racine du site.

Code : PHP

```

<?php
//On démarre la session
session_start();

//On se connecte à MySQL
mysql_connect('localhost', 'root', '');
mysql_select_db('tests');


//On inclut le contrôleur s'il existe et s'il est spécifié
if (!empty($_GET['page']) &&
is_file('contrôleurs/'.$_GET['page'].'.php'))
{
    include 'contrôleurs/'.$_GET['page'].'.php';
}
else
{
    include 'contrôleurs/accueil.php';
}

//On ferme la connexion à MySQL
mysql_close();

```

Il se connecte à la BDD et choisit le bon contrôleur. Dans notre cas, pour visualiser les news, il vous faudra donc faire un <http://monsite.com/index.php?page=news>.

Notez que je n'ai pas inclus le code xHTML de base, ce sera donc à vous de le trouver et de le mettre comme des grands. Il ne

présente pas d'intérêt dans ce tuto. Vous n'avez qu'à le mettre dans une vue et à inclure cette vue depuis l'index. Enfantin ! 

On continue avec le contrôleur [news.php](#), situé donc dans [contrôleurs/news.php](#) :

Code : PHP

```
<?php
/* C'est en tout début de fichier que l'on vérifie les
autorisations. Les
news sont visibles par tous, mais si vous voulez en restreindre
l'accès, c'est
ici que cela se passe. */

//On inclut le modèle
include(dirname(__FILE__).'../modeles/news.php');

/* On effectue ici diverses actions, comme supprimer des news, par
exemple. ;)
Il n'y en aura aucune dans ce tutoriel pour rester simple, mais
libre à vous d'en rajouter. */

//On récupère les news
$news = recuperer_news();

//On inclut la vue
include(dirname(__FILE__).'../vues/news.php');
?>
```

Ce code est vraiment le contrôleur le plus simple que vous pourrez avoir : il inclut le modèle, récupère les news et appelle la vue. Trivial, je pense. La seule chose que vous pourriez ne pas connaître est le `dirname(__FILE__)`. C'est une fonction permettant de retourner le chemin vers un fichier, le fichier actuel (le contrôleur) en l'occurrence, désigné par `__FILE__`, qui est une constante définie par PHP. Vous noterez que je vous ai mis en commentaire les endroits où vous pourrez effectuer diverses actions et vérifications, pour améliorer ce script.

Allez, maintenant, au tour du modèle ([modeles/news.php](#)) :

Code : PHP

```
<?php
function recuperer_news()
{
    $news = array();

    $req = mysql_query("SELECT id, auteur, titre,
DATE_FORMAT(date, '%d/%m/%Y %H') AS date_formatee, contenu
FROM news
ORDER BY date DESC");
    while ($data = mysql_fetch_assoc($req))
    {
        $news[] = $data;
    }

    return $news;
}
```

Ce code récupère toutes les news dans la base de données, les stocke dans un array et les retourne à l'appelant. Toute l'astuce était dans le tableau. Je pense que ceux qui auront bloqué l'auront fait à ce moment. Il faut juste penser à stocker les données dans un tableau avant de les afficher. Cela rajoute certes une étape, mais c'est le prix de la clarification du code.

Et la dernière étape, la vue, qui devrait passer toute seule si vous avez compris ce qui précède (dans [vues/news.php](#)) :

Code : PHP

```
<h1>Les news du site</h1>
```

```
<?php
foreach($news as $n)
{
    echo '
<div class="news">
<h2>'.$n['titre'].'</h2>
<p>News postée le '.$str_replace(' ', ' à ', $n['date_formatee']).'
par '.$n['auteur'].'</p>
<p>'.$n['contenu'].'</p>
</div>';
}
?>
```

Comme vous le voyez, la vue parcourt le tableau et affiche les infos, exactement comme dans le fichier de support. En réalité, nous n'avons fait que diviser un fichier en trois, mais cela procure une belle clarté, je trouve : pas vous ? 😎

Résumé de la structure FTP

Le FTP se présente normalement à la fin comme cela :

- /
- [index.php](#)
- [controleurs](#)
 - [news.php](#)
- [modeles](#)
 - [news.php](#)
- [vues](#)
 - [news.php](#)
 - et les autres fichiers d'affichage comme [logo.php](#), [menu.php](#) et [pied.php](#) si besoin.

D'autres approches

Je vois plusieurs améliorations possibles. Je vais vous en lister quelques-unes ici : à vous de prendre, ou de laisser. Si vous avez d'autres idées, vous pouvez les proposer dans les commentaires, je me ferai un plaisir de les ajouter si je le juge utile. 😊

Un système de templates !

Comme je le dis depuis le début, c'est l'amélioration la plus évidente à apporter. Les codes de logique et de présentation sont déjà séparés, vous n'avez plus que quelques changements à réaliser pour ajouter le moteur.

Il en existe un grand nombre, je vous donne quelques idées ici, à vous de chercher un peu ensuite (Google est votre ami 😊)...

- [Gagatemplate](#) ;
- [Smarty](#) ;
- [Talus' TPL](#).

Un tutoriel sur l'utilisation des Gagatemplates existe sur ce site ([celui-ci](#)), je vous conseille de le lire. Les généralités pourront vous aider à mieux cerner les avantages et inconvénients d'un tel système. Si vous envisagez de les utiliser, alors la suite vous fournira une initiation simple, faite par son concepteur.

Séparer les modèles

Une idée peut être de séparer vos modèles par action. Actuellement, vous avez un modèle par contrôleur ou bien par thème ([groupes.php](#), [membres.php](#), etc.). Si vous le désirez, vous pouvez mettre une fonction par fichier. Vous aurez alors des fichiers du style [groupes/ajouter.php](#) ou [membres/editer_profil.php](#).

C'est à vous de trouver s'il y a un intérêt pour votre site. Cela rend la gestion des erreurs un peu plus facile, puisque vous savez de quel fichier précis vient l'erreur, donc de quelle fonction. Certains trouveront lourd de devoir inclure le modèle spécifique avant chaque action... À vous de voir. 😊

Utiliser un framework

Un *framework* est un ensemble de scripts PHP vous permettant de faciliter la création de sites web. En (très) gros, ils comportent des méthodes pour gérer les accès à SQL, les sessions, le cache, des templates, etc., sans que vous n'ayez rien à écrire. Or, ils se trouvent (tiens donc 🤖) que la majorité de ces *frameworks* utilisent le modèle MVC. Bien sûr, vous n'aurez pas à utiliser ce dernier tel quel, mais la lecture de ce tuto vous en aura appris le principe. 😊

Voici quelques *frameworks* que vous pourrez utiliser :

- [Symfony](#) (en anglais) ;
- [Zend Framework](#) (en anglais, mais avec [une documentation en français](#)) ;
- [Code Igniter](#) (en anglais) ;
- [CakePHP](#) (en anglais).

Les deux premiers *frameworks* cités sont sans doute les frameworks les plus utilisés dans le monde professionnel, et sont vraiment très bien conçus ! Les deux suivants sont plus légers, mais aussi bien plus simples à prendre en main. On pourrait aussi parler de [Django](#). Attention, il s'utilise avec le langage Python ! Il sort donc un peu du cadre de tutoriels PHP puisqu'il faut connaître le langage Python pour l'utiliser. Il est cependant très connu, c'est pourquoi j'en parle.



Notez que tous les ajouts spécifiés dans cette partie ne sont en aucun cas obligatoires. Ils s'éloignent un peu du modèle MVC standard. Cependant, si l'un d'entre eux vous tente, n'hésitez pas !

Je vous recommande [ce sujet du forum PHP](#), traitant des techniques de programmation en PHP, qui est dans la continuité de ce tuto. En cas de problème sur le tuto, les commentaires sont là pour ça.

J'espère que ce tutoriel vous a plu et donné envie de mieux organiser le code de votre site !

Merci à [ptipilou](#) pour sa relecture attentive de mon tutoriel et sa zCorrection online. 😊

Partager



Ce tutoriel a été corrigé par les [zCorrecteurs](#).