

PHP

LE MODÈLE MVC AVEC PHP

« Le MVC tout comme l'orientation objet du code, semble être devenu un standard dans le développement d'applications web, avec la réputation d'être une bonne pratique de conception. » Julien Pauli, Developpez.com

Ce cours porte sur le motif appelé MVC pour modèle – vue – contrôleur. L'objet de ce cours est de présenter ce modèle ainsi que plusieurs de ses dérivés. Enfin, dans le cadre du cours de programmation en PHP, nous terminerons en étudiant un Framework utilisant ce motif : CodeIgniter.

Le modèle MVC

RETOUR AUX SOURCES

Retour aux sources

Tout au long de ce cours, nous prendrons comme exemple une application web de gestion de favoris.

Liste des favoris

Titre	Description	Liens
IUT de Provence, site d'Arles	Site de l'IUT d'Arles (IUT de Provence, site d'Arles)	IUT de Provence, site d'Arles supprimer
La ferme du web	Site web traitant des actualités web (AJAX, FRAMEWORK PHP, etc.).	La ferme du web supprimer
VDM	Les vies de merdes...	VDM supprimer
Korben	Site d'information sur le web.	Korben supprimer
MétéoFrance	Site web de Météo France.	MétéoFrance supprimer

Retour aux sources

```
<?php
$action = (isset($_REQUEST['action'])?$_REQUEST['action']:'list_favoris');
$link = mysql_connect("localhost", "user_cours_mvc", "cours_mvc");
mysql_set_charset('utf8',$link);
mysql_select_db('db_cours_mvc', $link);
$favoris=array();

switch($action) {
    case 'annuler' : $action='list_favoris';
    case 'list_favoris' :
        $query = "select * from `t_favoris_fav`";
        $favoris = mysql_query($query,$link);
        break;
    case 'delete_favoris':
        if (isset($_REQUEST['fav_id'])) {
            $fav_id = $_REQUEST['fav_id'];
            $query = sprintf('select * from `t_favoris_fav` where `fav_id`=%d', $fav_id);
            $result = mysql_query($query,$link);
            $favoris = mysql_fetch_object($result);
        } else header("Location: index.php");
        break;
    case 'delete':
        if (isset($_REQUEST['fav_id'])) {
            $fav_id = $_REQUEST['fav_id'];
            $query = sprintf('delete from `t_favoris_fav` where `fav_id`=%d', $fav_id);
            mysql_query($query,$link);
        }
        header("Location: index.php");
        break;
}
?>
```

Ce code est la 1^{er} partie du fichier **index.php**

Dans cette partie, on s'occupe :

- Des actions de l'utilisateur
- De produire les données relatives

Retour aux sources

```
<!DOCTYPE ...

<div id="container">
  <?php if ($action=="list_favoris") {?>
    <h1>Liste des favoris</h1>

    <div id="body">
      <table>
        <tr>
          <th>Titre</th>
          <th>Description</th>
          <th>Liens</th>
          <th>&nbsp;</th>
        </tr>

        <?php while($fav = mysql_fetch_object($favoris)) { ?>
          <tr>
            <td><?php echo $fav->FAV_TITRE;?></td>
            <td><?php echo $fav->FAV_DESCRIPTION;?></td>
            <td><a href="<?php echo $fav->FAV_LIEN;?>" target="_blank"><?php echo $fav->FAV_TITRE;?></a></td>
            <td><a href="index.php?action=delete_favoris&fav_id=<?php echo $fav->FAV_ID;?>">supprimer</a></td>
          <?php } ?>
        </table>
      </div>
      <?php }elseif ($action=="delete_favoris") {?>
        <h1>Supprimer un favori</h1>

        <div id="body">
          <form method="post">
            <div>
              <p>Voulez-vous vraiment supprimer ce favori ?</p>
              <br><strong><?php echo $favoris->FAV_TITRE;?></strong></p>
              <p><?php echo $favoris->FAV_LIEN;?></p>
              <div>
                <input type="submit" name="action" value="delete"/>
                <input type="submit" name="action" value="annuler"/>
              </div>
            </div>
          </form>
        </div>
        <?php } ?>
      </div>
    </div>
```

Ce code est la 2^{ème} partie
du fichier **index.php**

Dans cette partie, on s'occupe
d'afficher le contenu.

Retour aux sources

Bien que « relativement » propre, le code précédent pose plusieurs problèmes, parmi:

- ⊙ Mélange entre le PHP et l'HTML - pas de distinction entre le client et le serveur
- ⊙ Mélange entre les actions relatives à la gestion des actions de l'utilisateur et la gestion des données.

→ Dans une problématique métier, ce code est difficilement exploitable par les différentes entités qui auront à travailler dessus, probablement en même temps (Développeur frontoffice et backoffice, Designer, responsable éditorial, etc...)

Retour aux sources

Pour permettre une meilleur maintenance de ce code, il est nécessaire de mieux séparer le contrôle des actions, la manipulation des données ainsi que la façon dont on les affiche.

On peut alors séparer le code suivant en 4 fichiers :

- ⦿ `mvc.php`
Contient la logique fonctionnelle : intercepte et traite les actions demandées
- ⦿ `m-favoris.php`
Contient la connaissance métier : ensemble des fonctions de manipulation des favoris (CRUD).
- ⦿ `v-delete.inc.php` :
Affiche le formulaire de confirmation de suppression à partir des données générée par le script `mvc.php`.
- ⦿ `v-list.inc.php`
Affiche la liste des favoris à partir des données générées par le script `mvc.php`.

Retour aux sources

mvc.php

```
<?php
require "mvc/m-favoris.php";
$action = (isset($_REQUEST['action'])?$_REQUEST['action']:'list_favoris');

switch($action) {
    case 'annuler' : $action='list_favoris';
    case 'list_favoris' :
        $favoris = get_all_favoris();
        require "mvc/v-list.inc.php";exit;
        break;
    case 'delete_favoris':
        if (isset($_REQUEST['fav_id'])) {
            $favoris = get_favori($_REQUEST['fav_id']);
            require "mvc/v-delete.inc.php";exit;
        } else header("Location: mvc.php");
        break;
    case 'delete':
        if (isset($_REQUEST['fav_id'])) {
            delete_favori($_REQUEST['fav_id']);
        }
        header("Location: mvc.php");
        break;
    default:
        require "mvc/v-list.inc.php";exit;
        exit;
}
?>
```

Retour aux sources

m-favoris.php

```
<?php
$link = mysql_connect("localhost", "user_cours_mvc", "cours_mvc");
mysql_set_charset('utf8',$link);
mysql_select_db('db_cours_mvc', $link);

function get_all_favoris() {
    global $link;
    $query = "select * from `t_favoris_fav`";
    $result = mysql_query($query,$link);

    $favoris = array();
    while($fav = mysql_fetch_object($result)) $favoris[] = $fav;

    return $favoris;
}

function get_favori($fav_id) {
    global $link;
    $query = sprintf('select * from `t_favoris_fav` where `fav_id`=%d', $fav_id);
    $result = mysql_query($query,$link);
    return mysql_fetch_object($result);
}

function delete_favori($fav_id) {
    global $link;
    $query = sprintf('delete from `t_favoris_fav` where `fav_id`=%d', $fav_id);
    return mysql_query($query,$link);
}
?>
```

Retour aux sources

v-list.inc.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Liste des favoris</h1>

  <div id="body">
    <table>
      <tr>
        <th>Titre</th>
        <th>Description</th>
        <th>Liens</th>
        <th>&nbsp;</th>
      </tr>

      <?php foreach($favoris as $fav) { ?>
      <tr>
        <td><?php echo $fav->FAV_TITRE;?></td>
        <td><?php echo $fav->FAV_DESCRIPTION;?></td>
        <td><a href="<?php echo $fav->FAV_LIEN;?>" target="_blank"><?php echo
$fav->FAV_TITRE;?></a></td>
        <td><a href="mvc.php?action=delete_favoris&fav_id=<?php echo $fav-
>FAV_ID;?>">supprimer</a></td>
      <?php } ?>
    </table>
  </div>
</div>
</body>
</html>
```

v-delete.inc.php

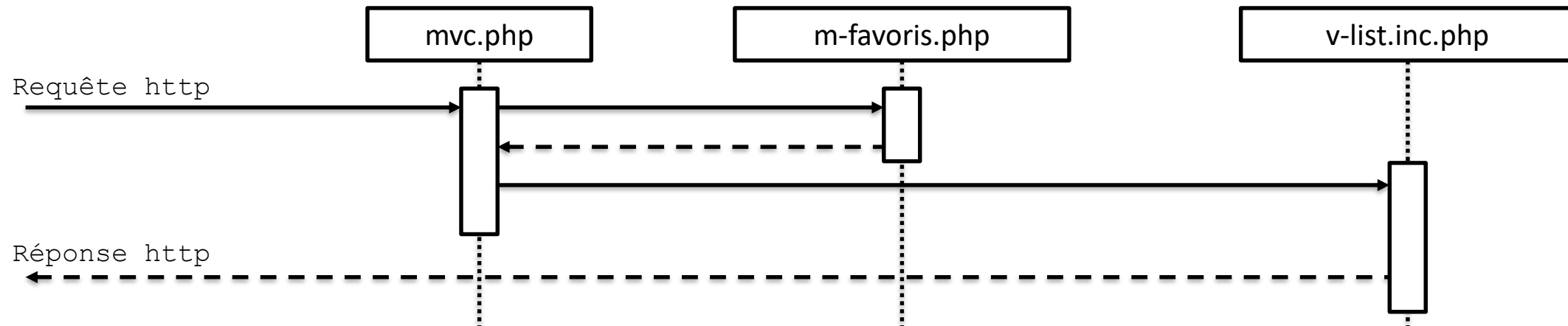
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Supprimer un favori</h1>

  <div id="body">
    <form method="post">
      <div>
        <p>Voulez-vous vraiment supprimer ce favoris ?
        <br><strong><?php echo $favoris->FAV_TITRE;?></strong></p>
        <p><?php echo $favoris->FAV_LIEN;?></p>
        <div>
          <input type="submit" name="action" value="delete"/>
          <input type="submit" name="action" value="annuler"/>
        </div>
      </div>
    </form>
  </div>
</div>
</body>
</html>
```

Retour aux sources

Le diagramme de séquence d'un appel à la page mvc.php est le suivant :



En séparant notre code en 3 couches qui se connaissent, on a nettement minimisé les interconnections entre les différents composants de notre application facilitant ainsi :

- ⦿ le travail collaboratif sur ce code
- ⦿ La mise à jour d'une des couches, indépendamment des autres.

Bien que encore éloigné d'un MVC, on c'est rapproché de sa philosophie.

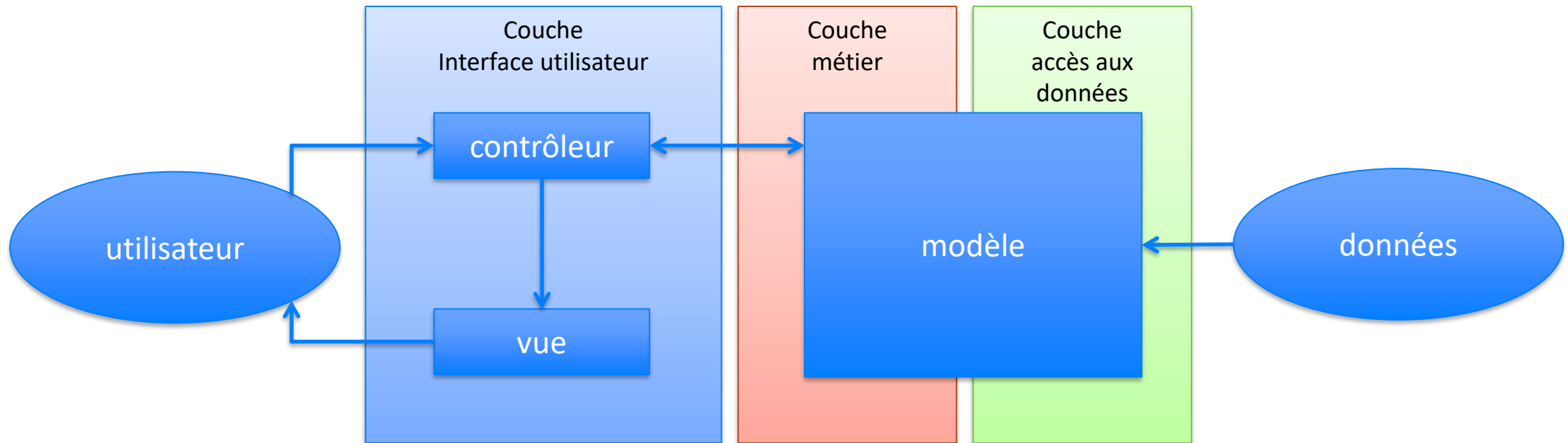
Le modèle MVC

PRINCIPE

Principe du modèle MVC

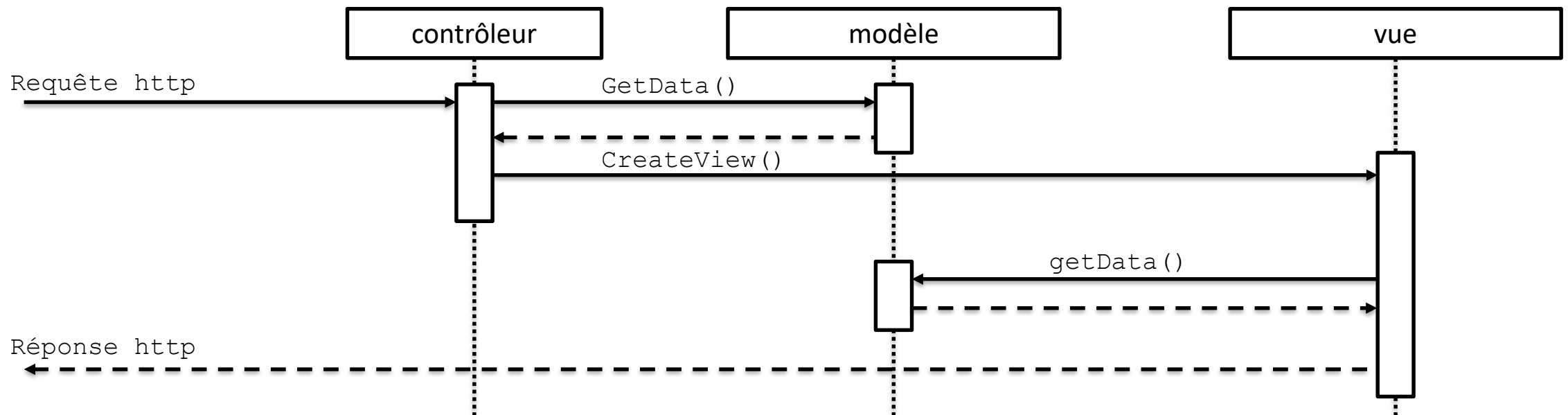
Le modèle **MVC** (Modèle-Vue-Contrôleur) cherche à séparer nettement les couches présentation, traitement et accès aux données.

Architecture d'une application MVC classique :



Principe du modèle MVC

Le diagramme de séquence d'un MVC simple est le suivant :



Remarque :

notez que la vue connaît le modèle et qu'elle peut donc lui demander des données.

LE MODÈLE

Le modèle

Le modèle

- ⦿ est un ensemble d'objets qui représentent le domaine du problème. Ces objets implémentent la logique métier.
- ⦿ Ils ignorent tout de l'interface utilisateur, et ne connaissent rien de la vue ou du contrôleur.

La couche modèle est constituée :

- ⦿ **Des classes métiers**
Qui aident à la gestion de la logique fonctionnelle de l'application : par exemple, les classes qui s'occupent de la gestion de la base de données, ou encore, les classes qui s'occupent du routage.
- ⦿ **Et, des classes d'accès aux données**
Qui gère l'accès aux données, souvent en base de données :
par exemple, une classe favoris dans notre exemple avec l'ensemble des méthodes classique (CRUD: create, read, update and delete).

Le modèle

Un mot sur l'ORM et le scaffolding

Object-relational mapping (mapping objet-relationnel) :

Un mapping objet-relationnel (en anglais object-relational mapping ou ORM) est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé.

(Source wikipedia)

L'ORM permet de simplifier l'accès aux données. Chaque tuple devient une instance d'objet. Les méthodes de modification sont uniformisées, ce qui permet de réaliser les mêmes traitements sur une donnée qu'elle provienne d'une base de données ou de n'importe quelle autre source.

Un autre avantage de l'ORM est de rendre l'accès aux données complètement indépendant du SGBD utilisé. Il devient donc très simple de changer de SGBD au cours du développement de l'application.

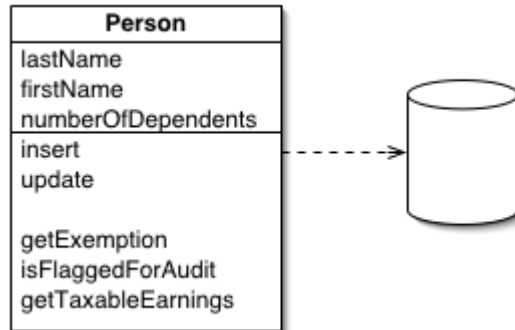
Le modèle

Un mot sur l'ORM et le scaffolding

Il existe plusieurs motifs du type ORM, entre autre :

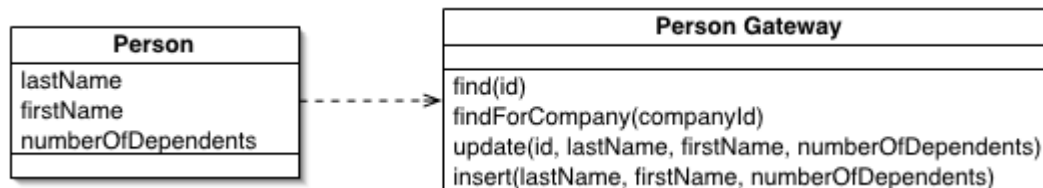
- ◉ **ActiveRecord**

On regroupe la structure de l'objet et les méthodes pour le manipuler.



- ◉ **table gateway**

On sépare la structure de l'objet et les méthodes pour le manipuler.



Plusieurs framework d'ORM existent en php : doctrine, propel, codeIgniter, ZenFramework, etc.

Le modèle

Un mot sur l'ORM et le scaffolding

Scaffolding (échafaudage) :

L'échafaudage ou **scaffolding** en anglais est une manière de concevoir des logiciels liés à une base de données. Cette technique est souvent fournie avec le patron de conception Modèle-Vue-Contrôleur, dans lequel le programmeur écrit une spécification décrivant comment la base de données sera utilisée. Le compilateur génère le code source de création, lecture, mise-à-jour et suppression (CRUD) des données en base pour l'application.

(Source wikipédia)

Le scaffolding permet de générer automatiquement les méthodes permettant de manipuler les éléments de la base de données à partir d'une description de cette dernière. Celle-ci étant souvent fournie par un fichier xml. L'avantage de cette approche est qu'elle permet de maintenir facilement les CRUD d'une base de données malgré sa modification.

Attention cependant, pour de gros sites nécessitant une optimisation des accès à la base de données, le scaffolding n'est pas nécessairement le plus adapté.

Le modèle

Un mot sur l'ORM et le scaffolding

Il existe plusieurs framework permettant de mettre en œuvre le scaffolding en PHP parmi les plus connus :

- ◉ RoR (Ruby On Rails)
- ◉ Symfony (avec Propel et Doctrine)
- ◉ CodeIgniter(scaffolding abandonné aujourd'hui)
- ◉ CakePHP
- ◉ etc.

Pour plus d'informations sur ces méthodes de programmation :

<http://www.martinfowler.com/eaCatalog/index.html>

<http://pn-mougel.developpez.com/tutoriels/php/orm/>

Le modèle MVC

LA VUE

La vue

La vue

- est un élément sur l'écran : il peut s'agir d'un bouton, d'un formulaire, d'un tableau, etc... dont le but est d'afficher les données du modèle.
- peut communiquer avec le modèle.

Attention, la vue ne contient pas la logique de présentation. Cette logique est partagée entre le modèle et la vue.

Il existe d'autres motifs permettant cette représentation comme:

- AM-MVC (Application Model MVC)
- MVPC (Model-View-Presenter-Controller)

Le modèle MVC

LE CONTRÔLEUR

Le contrôleur

Le rôle du contrôleur est de gérer le comportement (intercepter les entrées utilisateur).

Il peut être partagé par plusieurs vues. Il peut aussi choisir quelle vue afficher.

Il existe différent motif pour le contrôleur. Parmi les plus connus, on peut citer :

- ◉ PageControl :
C'est le plus naturel – on a un contrôleur par page, donc potentiellement plusieurs points d'entrées.
- ◉ Frontcontrol + motif Command :
Le FrontContrôleur reçoit et analyse la requête et invoque la méthode *launch()* d'une action (motif Command).
A la fin de son exécution, c'est l'action qui demandera le rendu et l'affichage d'une vue (ou une redirection, etc.). On a alors, un et un seul point d'entrée dans l'application.

Le contrôleur

La mauvaise utilisation du terme « contrôleur » est l'une des principales cause de confusion et d'incompréhension du modèle MVC et conduit à tout appeler MVC.

Le contrôleur du MVC a pour rôle de gérer les actions de l'utilisateur et de les traduire en commandes pour les objets métiers. Il ne fait pas la médiation entre le modèle et la vue.

Le modèle MVC

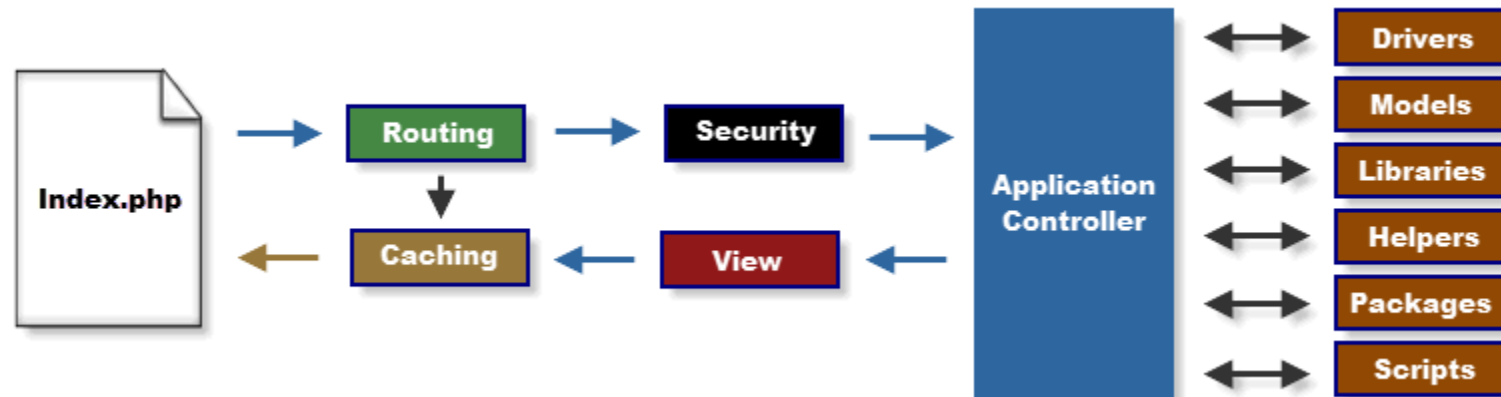
ETUDE D'UN FRAMEWORK PHP UTILISANT LE MODÈLE MVC : CODEIGNITER (V2.X)

CodeIgniter

Architecture

De nombreux Frameworks utilisent aujourd'hui le modèle **MVC**, car le but principal de ce dernier est de séparer les couches logiques d'une application.

Architecture de CI



Le fichier index.php sert de contrôleur frontal. Il initialise les ressources de base nécessaires à l'exécution de CodeIgniter. Le routeur examine la requête HTTP. Si un fichier cache existe, il est envoyée directement au navigateur, sans passer par l'exécution normale du système. Avant le chargement du contrôleur d'application, la requête HTTP et les données soumises par les utilisateurs sont filtrées.

Le contrôleur charge le modèle, les bibliothèques de base, les helpers et les autres ressources nécessaires pour traiter la demande spécifique. La vue est rendue puis envoyée au navigateur Web. Si le cache est activé, la vue est mis en cache afin d'être servie lors de demandes suivantes.

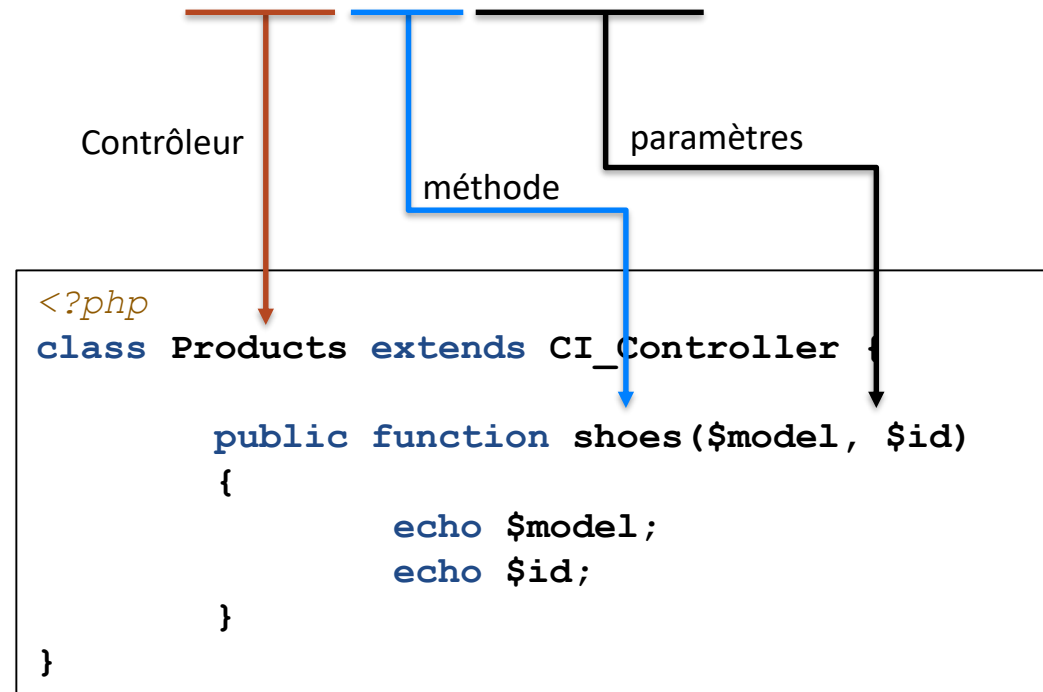
CodeIgniter

Routing

Le **routing** consiste à associer un code à exécuter à une URL reçue sur le serveur.

Exemple de routing avec codeIgniter

http://example.com/index.php/products/shoes/sandals/123



Fichier `application/controllers/products.php`

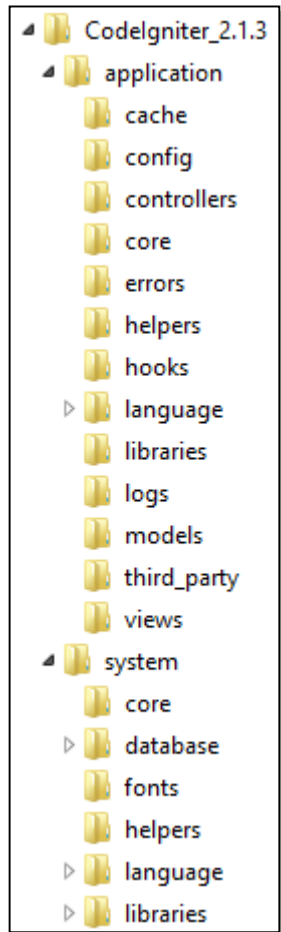
Dans codeIgniter, l'URL fera toujours référence à une fonction d'un contrôleur.

Index.php est le contrôleur frontal et le contrôleur appelé via l'URL est le contrôleur d'application.

Le lien entre l'URL, le contrôleur et la méthode se fait par leur nom : attention aux conventions de nommage !

CodeIgniter

arborescence



CI utilise une arborescence imposée (à gauche). On y retrouve les répertoires :

- ⦿ **application**

Dossier principal de notre application. Ce dossier contient l'essentiel des fichiers liés à notre application web, notamment, les contrôleurs (controllers), les modèles (models) et les vues (views).

Il est possible d'avoir plusieurs sous-répertoires dans ce dossier. Chaque répertoire correspond alors à une application. Cela permet de maintenir une seule instance de CI pour un ensemble de sites web.

- ⦿ **system**

Contient l'essentiel du cœur de CI. En principe, on n'a pas à y toucher.

CodeIgniter

exemple

Pour mettre en œuvre CI, nous allons reprendre notre exemple de gestion de Favoris.

Liste des favoris			
Titre	Description	Liens	
IUT de Provence, site d'Arles	Site de l'IUT d'Arles (IUT de Provence, site d'Arles)	IUT de Provence, site d'Arles	editer supprimer
La ferme du web	Site web traitant des actualités web (AJAX, FRAMEWORK PHP, etc.).	La ferme du web	editer supprimer
VDM	Les vies de merdes...	VDM	editer supprimer
Korben	Site d'information sur le web.	Korben	editer supprimer
MétéoFrance	Site web de METEO France.	MétéoFrance	editer supprimer
ajouter un lien			

CodeIgniter

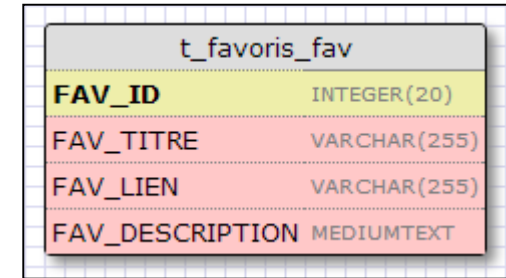
exemple - Mise en place de la BDD

-- Base de données DB_BONAPPETIT

```
DROP DATABASE IF EXISTS `db_cours_mvc`;
CREATE DATABASE IF NOT EXISTS `db_cours_mvc` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `db_cours_mvc`;

CREATE USER 'user_cours_mvc'@'%' IDENTIFIED BY 'cours_mvc';
GRANT USAGE ON * . * TO 'user_cours_mvc'@'%' IDENTIFIED BY 'cours_mvc' WITH MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0 ;
GRANT ALL PRIVILEGES ON `db_cours_mvc` . * TO 'user_cours_mvc'@'%;

-- Table des favoris
-- DROP TABLE IF EXISTS `T_FAVORIS_FAV`;
CREATE TABLE IF NOT EXISTS `T_FAVORIS_FAV` (
  `FAV_ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `FAV_TITRE` varchar(255) NOT NULL,
  `FAV_LIEN` varchar(255) NOT NULL,
  `FAV_DESCRIPTION` longtext NOT NULL,
  PRIMARY KEY (`FAV_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```



FAV_ID	INTEGER(20)
FAV_TITRE	VARCHAR(255)
FAV_LIEN	VARCHAR(255)
FAV_DESCRIPTION	MEDIUMTEXT

CodeIgniter

exemple - Création du contrôleur
controllers/main.php

```
class Main extends CI_Controller {  
    public function index() {  
        $this->list_all();  
    }  
  
    public function list_all() {  
        ...  
    }  
  
    public function create_favoris() {  
        ...  
    }  
  
    public function update_favoris($fav_id) {  
        ...  
    }  
  
    public function delete_favoris($fav_id) {  
        ...  
    }  
}
```

Dans le répertoire **controllers** de notre application, on crée le fichier **main.php** avec le code suivant.

La class **Main** hérite de la classe **CI_Controllers** propre à CI.

Chaque méthode de ce contrôleur gèrera une action de notre site :
lister, ajouter, éditer et supprimer nos favoris.

Pour accéder à celles-ci, il suffira d'utiliser l'URL correspondante :
[http://localhost/mon_site/index.php/\[contrôleur\]/\[méthode\]/\[param1\]/\[param2\]/\[..\]](http://localhost/mon_site/index.php/[contrôleur]/[méthode]/[param1]/[param2]/[..])

Par exemple, pour accéder à l'action **list_all**, on utilisera l'URL suivante :
http://localhost/mon_site/index.php/main/list_all

Ou encore, pour supprimer le favoris dont l'ID vaut 2 :
http://localhost/mon_site/index.php/main/delete_favoris/2

CodeIgniter

exemple - Création des vues

Il y a 4 vues à créer.

On crée les vues dans le répertoire **views** en tachant de mélanger le moins possible HTML et PHP, même si le Framework permet déjà de minimiser leur couplage.

```
<ul>
  <?php foreach($todo_list as $item) {?>
    <li><?php echo $item;?></li>
  <?php } ?>
</ul>
```

Remarque :

On peut utiliser les *short open tags* sachant que CI est capable de les réécrire si jamais ils ne sont pas activés sur le serveur (version < php 5.4), et la *syntaxe alternative*.

```
<ul>
  <? foreach ($todo_list as $item) :?>
    <li><?= $item;?></li>
  <? endforeach;?>
</ul>
```

CodeIgniter

exemple - Création des vues

Il y a 4 vues à créer.

On crée les vues dans le répertoire **views** en tachant de mélanger le moins possible HTML et PHP, même si le Framework permet déjà de minimiser leur couplage.

home - La page d'accueil qui contient le liste des favoris

Liste des favoris

Titre	Description	Liens
IUT de Provence, site d'Arles	Site de l'IUT d'Arles (IUT de Provence, site d'Arles)	IUT de Provence, site d'Arles editer supprimer
La ferme du web	Site web traitant des actualités web (AJAX, FRAMEWORK PHP, etc.).	La ferme du web editer supprimer
VDM	Les vies de merdes...	VDM editer supprimer
Korben	Site d'information sur le web.	Korben editer supprimer
MétéoFrance	Site web de METEO France.	MétéoFrance editer supprimer
ajouter un lien		

CodeIgniter

exemple - Création des vues

Il y a 4 vues à créer.

On crée les vues dans le répertoire **views** en tachant de mélanger le moins possible HTML et PHP, même si le Framework permet déjà de minimiser leur couplage.

home - La page d'accueil qui contient la liste des favoris

new - La page qui contient le formulaire de création d'un nouveau favori.



Nouveau favori

Titre

URL (http://)

Description

CodeIgniter

exemple - Création des vues

Il y a 4 vues à créer.

On crée les vues dans le répertoire **views** en tachant de mélanger le moins possible HTML et PHP, même si le Framework permet déjà de minimiser leur couplage.

home - La page d'accueil qui contient le liste des favoris

new - La page qui contient le formulaire de création d'un nouveau favori.

update - La page qui contient le formulaire permettant d'éditer un favori existant.



The screenshot shows a web form titled "Home" with the following fields and buttons:

- Titre**: A text input field containing the value "Korben".
- URL (http://)**: A text input field containing the value "http://korben.info/".
- Description**: A text area containing the text "Site d'information sur le web.".
- Buttons**: Three buttons at the bottom: "sauvegarder", "annuler", and "reset".

CodeIgniter

exemple - Création des vues

Il y a 4 vues à créer.

On crée les vues dans le répertoire **views** en tachant de mélanger le moins possible HTML et PHP, même si le Framework permet déjà de minimiser leur couplage.

home - La page d'accueil qui contient le liste des favoris

new - La page qui contient le formulaire de création d'un nouveau favori.

update - La page qui contient le formulaire permettant d'éditer un favori existant.

delete - La page qui contient le formulaire de confirmation de suppression.

Supprimer un favori

Voulez-vous vraiment supprimer ce favoris ?

VDM : <http://www.viedemerde.fr/>

Les vies de merdes...

CodeIgniter

exemple - Création des vues – *views/home.php*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Liste des favoris</h1>

  <div id="body">
    <table>
      <tr>
        <th>Titre</th>
        <th>Description</th>
        <th>Liens</th>
        <th>&nbsp;</th>
      </tr>
      <?php foreach($favoris as $fav) : ?>
      <tr>
        <td><?php echo $fav->FAV_TITRE;?></td>
        <td><?php echo $fav->FAV_DESCRIPTION;?></td>
        <td><?php echo anchor($fav->FAV_LIEN, $fav->FAV_TITRE, 'target="_blank"'); ?></td>
        <td>
          <?php echo anchor("/main/update_favoris/$fav->FAV_ID", 'editer'); ?>
          <?php echo anchor("/main/delete_favoris/$fav->FAV_ID", 'supprimer'); ?>
        </td>
      </tr>
      <?php endforeach; ?>
    </table>

    <p><?php echo anchor('/main/create_favoris', 'ajouter un lien'); ?></p>
  </div>
</div>

</body>
</html>
```

CodeIgniter

exemple - Création des vues – *views/new.php*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Nouveau favori</h1>

  <div id="body">
    <?php echo validation_errors(); ?>
    <form method="post">
      <div><label>Titre</label><input type="text" name="fTitre" value="<?php echo set_value('fTitre'); ?>"></div>
      <div><label>URL (http://)</label><input type="text" name="fLien" value="<?php echo set_value('fLien'); ?>"></div>
      <div><label>Description</label></div>
      <textarea name="fDescription"><?php echo set_value('fDescription'); ?></textarea>
      <div>
        <input type="submit" name="action" value="sauvegarder">
        <input type="submit" name="action" value="annuler">
        <input type="reset" name="action" value="reset">
      </div>
    </form>
  </div>
</div>

</body>
</html>
```


CodeIgniter

exemple - Création des vues – *views/update.php*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Home</h1>

  <div id="body">
    <?php echo validation_errors(); ?>
    <form method="post">
      <div><label>Titre</label><input type="text" name="fTitre" value="<?php echo set_value('fTitre',$favoris->FAV_TITRE);?>"></div>
      <div><label>URL (http://)</label><input type="text" name="fLien" value="<?php echo set_value('fLien',$favoris->FAV_LIEN);?>"></div>
      <div><label>Description</label></div>
      <textarea name="fDescription"><?php echo set_value('fDescription',$favoris->FAV_DESCRIPTION);?></textarea>
      <div>
        <input type="submit" name="action" value="sauvegarder">
        <input type="submit" name="action" value="annuler">
        <input type="reset" name="action" value="reset">
      </div>
    </form>
  </div>
</div>

</body>
</html>
```

CodeIgniter

exemple - Création des vues – *views/delete.php*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home</title>
</head>
<body>

<div id="container">
  <h1>Supprimer un favori</h1>

  <div id="body">
    <form method="post">
      <div class="alert alert-danger">
        <p>Voulez-vous vraiment supprimer ce favori ?
        <br><strong><?php echo $favoris->FAV_TITRE;?></strong> : <?php echo anchor($favoris->FAV_LIEN, $favoris->FAV_LIEN,
'target="_blank"'); ?></p>
        <p><?php echo $favoris->FAV_DESCRIPTION;?></p>
        <div>
          <button class="btn btn-danger" type="submit" name="action" value="delete">Oui</button>
          <button class="btn" type="submit" name="action" value="annuler">Non</button>
        </div>
      </div>
    </form>
  </div>
</div>

</body>
</html>
```

CodeIgniter

exemple - Création du modèle – *models/mFavoris.php*

```
<?php
class mFavoris extends CI_Model {

    function __construct() {
        // Call the Model constructor
        parent::__construct();
    }

    public function get_all() {
        $query = $this->db->get('T_FAVORIS_FAV');

        if( $query->num_rows() > 0 ) {
            return $query->result();
        } else return array();
    }

    public function get($fav_id) {
        $query = $this->db->get_where('T_FAVORIS_FAV', array('FAV_ID' => $fav_id));

        if( $query->num_rows() > 0 ) {
            $favoris = $query->result();
            return $favoris[0];
        } else return null;
    }

    public function create($data) {
        return $this->db->insert('T_FAVORIS_FAV', $data);
    }

    public function update($fav_id,$data) {
        $this->db->where('FAV_ID', $fav_id);
        return $this->db->update('T_FAVORIS_FAV', $data);
    }

    public function delete($fav_id) {
        $this->db->where('FAV_ID', $fav_id);
        return $this->db->delete('T_FAVORIS_FAV');
    }
}

/* End of file mFavoris.php */
/* Location: ./application/models/mFavoris.php */
```

Dans le répertoire **models** de notre application, on crée le fichier **mFavoris.php** avec le code suivant.

La class **mFavoris** hérite de la classe **CI_Model**, propre à CI et implémente les méthodes métiers permettant de travailler sur la base de données. C'est notre CRUD.

On utilise ici les ActiveRecord fournies par CI.

CodeIgniter

exemple – Retour au contrôleur
controllers/main.php

```
class Main extends CI_Controller {  
    public function index() ...  
  
    public function list_all() {  
        $data = array();  
        $this->load->model('mFavoris');  
        $data['favoris'] = $this->mFavoris->get_all();  
        $this->load->helper('url');  
        $this->load->view('home', $data);  
    }  
  
    public function create_favoris() ...  
    public function update_favoris($fav_id) ...  
    public function delete_favoris($fav_id) ...  
}
```

On implémente l'action ***list_all*** du contrôleur ***main***.

Celle-ci fait appelle au model ***mFavoris*** pour récupérer les favoris stockés dans la base de données.

CodeIgniter

exemple – Retour au contrôleur
controllers/main.php

```
class Main extends CI_Controller {
    public function index()...
    public function list_all()...

    public function create_favoris() {
        $this->load->helper(array('form', 'url'));
        if ($this->input->post('action')=="annuler") redirect('main');

        $this->load->library('form_validation');
        $this->form_validation->set_rules('fTitre', 'Titre', 'required');
        $this->form_validation->set_rules('fLien', 'URL', 'required');
        $this->form_validation->set_rules('fDescription', 'Description', 'required');

        if ($this->form_validation->run() == FALSE) $this->load->view('new');
        else {
            $new_fav = array(
                'FAV_TITRE' => $this->input->post('fTitre'),
                'FAV_LIEN' => $this->input->post('fLien'),
                'FAV_DESCRIPTION' => $this->input->post('fDescription')
            );
            $this->load->model('mFavoris');
            $this->mFavoris->create($new_fav);
            redirect('main');
        }
    }

    public function update_favoris($fav_id)...
    public function delete_favoris($fav_id)...
}
```

On implémente l'action **create_favoris** du contrôleur **main**.

Celle-ci fait appel au model **mFavoris** pour créer un nouveau favoris en base de données.

Note : on utilise la capacité du Framework à gérer les données du formulaire via l'*helper form* et la méthode *post()*. Comme pour `$_POST`, le lien est fait par l'attribut *name* dans le formulaire.

CodeIgniter

exemple – Retour au contrôleur
controllers/main.php

```
class Main extends CI_Controller {
    public function index()...
    public function list_all()...
    public function create_favoris()...

    public function update_favoris($fav_id) {
        $this->load->helper(array('form', 'url'));
        if ($this->input->post('action')=="annuler") redirect('main');

        $this->load->model('mFavoris');
        $favoris = $this->mFavoris->get($fav_id);
        if (!$favoris) redirect('main');

        $data = array();
        $data['favoris'] = $favoris;

        $this->load->library('form_validation');
        $this->form_validation->set_rules('fTitre', 'Titre', 'required');
        $this->form_validation->set_rules('fLien', 'URL', 'required');
        $this->form_validation->set_rules('fDescription', 'Description', 'required');

        if ($this->form_validation->run() == FALSE) {
            $this->load->view('update', $data);
        } else {
            $update_fav = array(
                'FAV_TITRE' => $this->input->post('fTitre'),
                'FAV_LIEN' => $this->input->post('fLien'),
                'FAV_DESCRIPTION' => $this->input->post('fDescription')
            );

            $this->mFavoris->update($fav_id, $update_fav);
            redirect('main');
        }
    }

    public function delete_favoris($fav_id) {}
}
```

On implémente l'action **update_favoris** du contrôleur **main**.

Celle-ci fait appelle au model **mFavoris** pour récupérer les données associées au favori demandé.

Après validation du formulaire, le controleur fait de nouveau appel au model pour mettre à jour les données.

CodeIgniter

exemple – Retour au contrôleur
controllers/main.php

```
class Main extends CI_Controller {
    public function index()...
    public function list_all()...
    public function create_favoris()...
    public function update_favoris($fav_id)...

    public function delete_favoris($fav_id) {
        $this->load->helper('url');
        $this->load->model('mFavoris');
        $favoris = $this->mFavoris->get($fav_id);

        if (!$favoris) redirect('main');

        if ($_POST) {
            if ($this->input->post('action')==="delete") {
                $this->mFavoris->delete($fav_id);
            }
            redirect('main');
        } else $this->load->view('delete', array('favoris' => $favoris));
    }
}
```

On implémente l'action ***delete_favoris*** du contrôleur ***main***.

Celle-ci fait appelle au model ***mFavoris*** pour récupérer les favoris stockés dans la base de données.

Après confirmation de la suppression, le controleur utilise le model pour mettre en œuvre la suppression du favori en base de données.

Le modèle MVC

CODEIGNITER + SMARTY

CI + Smarty

Cet exemple s'appuie sur l'application de gestion des favoris. Pour cela, elle combine l'utilisation de différents framework :

CodeIgniter :

mise en œuvre de la logique fonctionnelle et du modèle

<http://codeigniter.com/>

Smarty :

création et gestion des vues

<http://www.smarty.net/>

Bootstrap :

framework html/css

<http://twitter.github.com/bootstrap/>

CI + Smarty

main.tpl

```

<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]>
<!--> <html class="no-js"> <!--> <![endif]-->
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title></title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width">

  <link rel="stylesheet" href="{base_url('assets/bootstrap/css/bootstrap.min.css')}">
  <link rel="stylesheet" href="{base_url('assets/FortAwesome/css/font-awesome.css')}">
  <style>
    body {
      padding-top: 60px;
      padding-bottom: 40px; }
  </style>
  <link rel="stylesheet" href="{base_url('assets/bootstrap/css/bootstrap-responsive.min.css')}">
  <link rel="stylesheet" href="{base_url('assets/bootstrap/css/main.css')}">

  <script src="{base_url('assets/bootstrap/js/vendor/modernizr-2.6.1-respond-1.1.0.min.js')}"></script>
</head>
<body>
  {include 'header.tpl' scope=parent}

  <div class="container">
    <h1>{block name="titre"}main.TITRE{/block}</h1>
    {block name=output_area}
      zone principale
    {/block}

    <hr>
    <footer>
      <p>&copy; IUT de Provence, site Arles 2012 - {mailto address='brett.desbenoit@univ-amu.fr' encode='javascript' subject='AuBonPlat'}</p>
    </footer>
  </div> <!-- /container -->

  <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
  <script>window.jQuery || document.write('<script src="{base_url('assets/bootstrap/js/vendor/jquery-1.8.2.min.js')}"></script>')</script>
  <script src="{base_url('assets/bootstrap/js/vendor/bootstrap.min.js')}"></script>
  <script src="{base_url('assets/bootstrap/js/main.js')}"></script>
</body>
</html>

```

CI + Smarty

header.tpl

```
<div class="navbar navbar-fixed-top">
  <div class="navbar-inner">
    <div class="container">
      <a class="brand" href="{site_url()}"><i class="icon-bookmark"></i> MyFav'</a>
    </div>
  </div>
</div>
```


CI + Smarty

home.tpl











```
{extends 'main.tpl'}
{block name="titre"}Liste des favoris{/block}
{block name="output_area"}
<div id="body">
{if $favoris|default:''}
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Titre</th>
        <th>Description</th>
        <th>Liens</th>
        <th>&nbsp;</th>
      </tr>
    </thead>
    <tbody>
      {foreach $favoris as $fav}
        <tr>
          <td>{$fav->FAV_TITRE|truncate:20}</td>
          <td>{$fav->FAV_DESCRIPTION|truncate:60}</td>
          <td>{$fav->FAV_TITRE}&nbsp;<a href="{ $fav->FAV_LIEN}" target="_blank"><i class="icon-external-link"></i></a>
          </td>
          <td>
            &nbsp;<a href="{site_url('/main/update_favoris/'|cat:$fav->FAV_ID)}"><i class="icon-edit"></i></a>
            &nbsp;<a href="{site_url('/main/delete_favoris/'|cat:$fav->FAV_ID)}"><i class="icon-remove"></i></a>
          </td>
        </tr>
      {/foreach}
    </tbody>
  </table>
{else}
  <div> Vous n'avez encore aucune recette en cours de rédaction.</div>
{/if}
<p><a href="{site_url('/main/create_favoris/')}"><i class="icon-plus"></i> ajouter un lien</a></p>
</div>
{/block}
```

CI + Smarty

home.tpl

 MyFav'

Liste des favoris

Titre	Description	Liens
IUT de Provence,...	Site de l'IUT d'Arles (IUT de Provence, site d'Arles)	IUT de Provence, site d'Arles  
La ferme du web	Site web traitant des actualités web (AJAX, FRAMEWORK...	La ferme du web  
VDM	Les vies de merdes...	VDM  
Korben	Site d'information sur le web.	Korben  
MétéoFrance	Site web de METEO France.	MétéoFrance  

[+ ajouter un lien](#)

© IUT de Provence, site Arles 2012 - brett.desbenoit@univ-amu.fr

CI + Smarty

update.tpl

```
{extends 'main.tpl'}
{block name="titre"}<i class="icon-edit"></i> Editer un favori{/block}
{block name="output_area"}
<div id="body">
  {assign "error" validation_errors()}
  {if $error|default:''}<div class="alert alert-error">{$error}</div>{/if}
  <form method="post">
    <div><label>Titre</label><input type="text" name="fTitre" value="{set_value('fTitre',$favoris->FAV_TITRE|default:'')}"></div>
    <div><label>URL (http://)</label><input type="text" name="fLien" value="{set_value('fLien',$favoris->FAV_LIEN|default:'')}"></div>
    <div><label>Description</label></div>
    <textarea name="fDescription">{set_value('fDescription',$favoris->FAV_DESCRIPTION|default:'')}</textarea>
    <div>
      <button class="btn" type="submit" name="action" value="sauvegarder">Sauvegader</button>
      <button class="btn" type="submit" name="action" value="annuler">Annuler</button>
      <button class="btn" type="reset" name="action" value="raz">R.A.Z</button>
    </div>
  </form>
</bouton>
{/block}
```

CI + Smarty

update.tpl

 MyFav'

 **Editer un favori**

Titre

URL (http://)

Description





© IUT de Provence, site Arles 2012 - brett.desbenoit@univ-amu.fr

CI + Smarty

new.tpl

```
{extends 'update.tpl'}  
{block name="titre"}<i class="icon-plus"></i> Nouveau favori{/block}
```


CI + Smarty

new.tpl

 MyFav'

+ Nouveau favori

Titre

URL (http://)

Description

© IUT de Provence, site Arles 2012 - brett.desbenoit@univ-amu.fr

CI + Smarty

delete.tpl

```
{extends 'main.tpl'}
{block name="titre"}<i class="icon-remove"></i> Supprimer un favori{/block}
{block name="output_area"}
<div id="body">
  <form method="post">
    <div class="alert alert-danger">
      <p>Voulez-vous vraiment supprimer ce favoris ?
      <br><i class="icon-bookmark"></i>&nbsp;<strong>{$favoris->FAV_TITRE}</strong> :
      {$favoris->FAV_LIEN}&nbsp;<a href="{$favoris->FAV_LIEN}" target="_blank"><i class="icon-external-link"></i></a></p>
      <p>{$favoris->FAV_DESCRIPTION}</p>
    <div>
      <button class="btn btn-danger" type="submit" name="action" value="delete">Oui</button>
      <button class="btn" type="submit" name="action" value="annuler">Non</button>
    </div>
  </div>
</form>
</div>
{/block}
```

CI + Smarty

delete.tpl



CI + Smarty

le mot de la fin

Quid du contrôleur (controllers/main.php) ?

Quid du modèle (models/mFavoris.php) ?

CI + Smarty

le mot de la fin

Quid du contrôleur (`controllers/main.php`) ?

Il n'a pas changé banane !!!

Quid du modèle (`models/mFavoris.php`) ?

Il n'a pas changé non plus !!!!

Qu'est-ce que tu crois qu'on essaie de faire depuis 2h ?!!



CodeIgniter

conclusion

Dans cette partie, nous n'avons vu qu'une toute petite partie de CodeIgniter. Cependant, elle représente les fondements d'une application plus conséquente.

CI est un framework extrêmement bien documenté et facile d'accès. Pour plus d'informations :

http://www.codeigniter.com/user_guide/