Développement WEB semaine 4

MVC et Framework

Luc Dartois

luc.dartois@u-pec.fr

Objectifs

Modèle-vue-contrôleur

- Appréhender le motif d'architecture MVC,
- Identifier ce qui relève du modèle, vue ou contrôleur

Framework

- S'initier à ce qu'est un framework
- Connaître les bases du Framework CodeIgniter

L'architecture logicielle

Motif d'architecture

Le Modèle-vue-contrôleur (ou MVC) est un motif d'architecture logicielle, i.e. une structure symbolique du code.
$\rightarrow$ Le comment faire.

Modèle-vue-contrôleur

MVC

Le but est de décomposer une application dans des *modules*, suivant leur rôle dans l'application. On a les modules :

- Modèle, qui contient les données,
- Vue, qui contient la présentation graphique,
- Contrôleur, qui contient la logique d'évolution de l'appli.

Modèle-vue-contrôleur

MVC

Le but est de décomposer une application dans des *modules*, suivant leur rôle dans l'application. On a les modules :

- Modèle, qui contient les données,
- Vue, qui contient la présentation graphique,
- Contrôleur, qui contient la logique d'évolution de l'appli.

Avantages

- La séparation permet une clarté de code,
- Chaque module est plus ou moins indépendant, permettant une flexibilité. On peut ainsi aisément :
  - Tester les modules séparément,
  - Changer une charte graphique sans toucher aux fonctionnalités,
  - Migrer des bases de données pour update,
- On peut adapter facilement notre code à d'autres projets ou d'autres systèmes (Version mobile en ne changeant que la vue).

Le Modèle

Le modèle :

- connaît les données (en se connectant au SGBD),
- sait agir dessus (valider, lire, écrire),
- est indépendant de la vue et du contrôleur,
- ne sait donc pas comment les données sont affichées

La Vue

La vue :

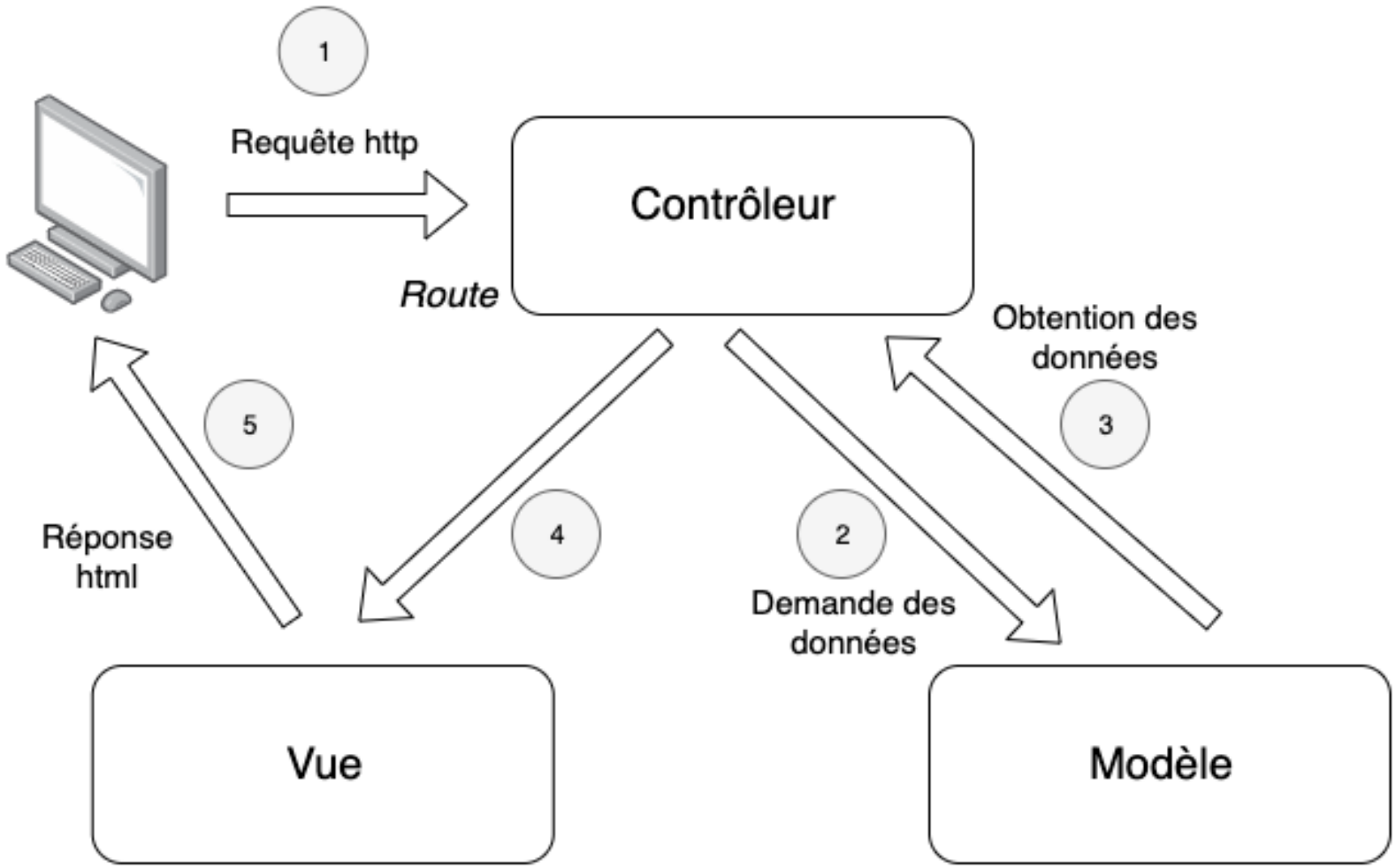
- représente la partie visible de l'interface graphique,
- affiche les données fournies par le *modèle*,
- est responsable de la gestion des sorties.

Le Contrôleur

Le contrôleur :

- fait le lien entre le modèle et la vue,
- est celui qui interagit avec l'utilisateur,
- invoque le modèle pour gérer les données,
- fournit les données à afficher à la vue,
- est responsable de la gestion des entrées.

Fonctionnement du MVC[1]



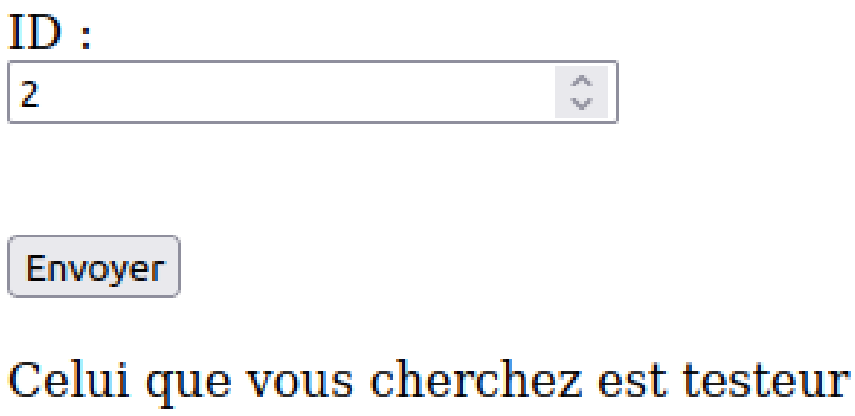
[1] crédits schéma: wikimedia.org

MVC sur un exemple

Exemple

On va illustrer le motif MVC sur un exemple simple. Il s'agit ici d'un formulaire demandant un entier et renvoyant le nom associé dans une base de données.

Exemple de la vue :



Exemple : Modele

Le modèle est simplement un ensemble de fonction faisant le lien avec la BD :

```
<?php
function connect(){
$user = "testeur";
$password = "MDPtesteur123456";
$database = "example_database";
$table = "mvc";
$db = mysqli_connect("localhost",$user, $password,$database);
return $db;
}
function get($i,$db){
$req="SELECT nom FROM mvc WHERE id=$i";
$res=mysqli_query($db,$req);
$nom=mysqli_fetch_assoc($res)['nom'];
return $nom;
}?>
```

Exemple : Controleur

Le contrôleur récupère les données de la vue, les traite via le modèle, et renvoie à la vue :

```
<?php
include_once('modele.php');
$co=connect();
$i=$_POST['id'];
$nom=get($i,$co);
header('Location:'.'http://localhost/MVC/vue.php?nom='.$nom);
?>
```

Exemple : Vue

La vue est la seule page visible. Elle affiche le formulaire et les résultats.

```
<form method="post" action="controleur.php">
  <label for="id">ID :</label><br>
  <input type="number" name="id"><br>
  <br><br>
  <input type="submit" value="Envoyer">
</form>

<?php
if(isset($_GET['nom'])){
echo "<br>";
$nom= $_GET['nom'];
if($nom==''){ echo "Numéro Invalide";  }
else{ echo "Celui que vous cherchez est $nom"; }
}
?>
```

Qu'est-ce qu'un framework ?

Un *cadre* PHP est un logiciel PHP fournissant de nombreuses fonctionnalités et règles. Elles sont censées :

- aider au développement d'applications,
- permettre de bonnes performances,
- diriger le développement suivant des patrons de conception,
- faire du code clair, ...

Quel framework utiliser ?

Pour cette initiation, nous utiliserons *CodeIgniter 3* car il est (plus) simple à prendre en main. Néanmoins de nombreux framework existent, tels que Laravel, Symfony, ...

Une approche MVC

MVC

La plupart des framework PHP s'appuient sur le motif MVC. Ils proposent des méthodes pour créer les Modèles, Vues et Contrôleur, et des moyens de les séparer proprement et les relier facilement.

Une approche MVC et orientée objet

MVC

La plupart des framework PHP s'appuient sur le motif MVC. Ils proposent des méthodes pour créer les Modèles, Vues et Contrôleur, et des moyens de les séparer proprement et les relier facilement.

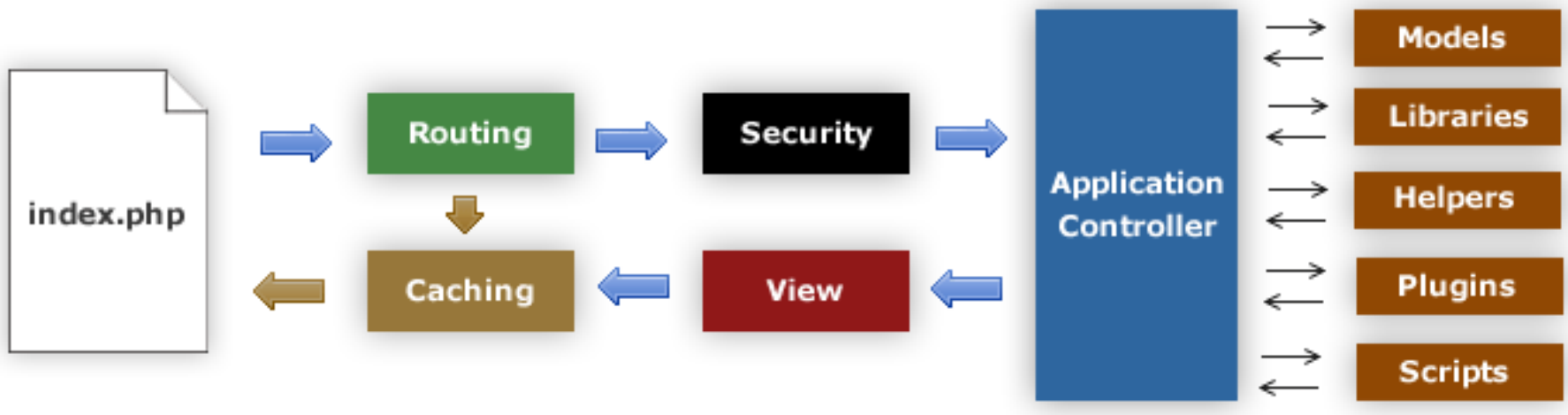
Orienté Objet

Ils utilisent aussi souvent la partie Orientée Objet de PHP, que nous avons jusque là négligée.
→ Chacun des Modèles, Vues et Contrôleurs sont alors des classes *héritant* du template de base du framework, qui fournissent des méthodes de base.

CodeIgniter 3

CodeIgniter (https://codeigniter.com) fait partie des Framework simples à installer et prendre en main.
Bien que la dernière version soit la version 4., nous utiliserons la version 3., qui est plus simple d'accès.

CodeIgniter 3



- index.php est le point d'entrée du site, il initialise les ressources de CI.
- Le Routing examine la requête, et renvoie directement la version du Caching si il y en a une,
- Ou envoie à la fonction Sécurity, qui filtre les données
- Le contrôleur charge les modèles et les ressources nécessaires
- Il calcule la vue, qui est mise en cache et envoyé à l'utilisateur.

URL et Classes

Si `http://localhost/CI3/` est la racine de mon application, alors aller à cette adresse lance l'index.php de mon application.
Il est néanmoins possible de lancer une méthode précise d'un contrôleur donné avec l'url :

```
localhost/CI3/index.php/Fiche/view/12
```

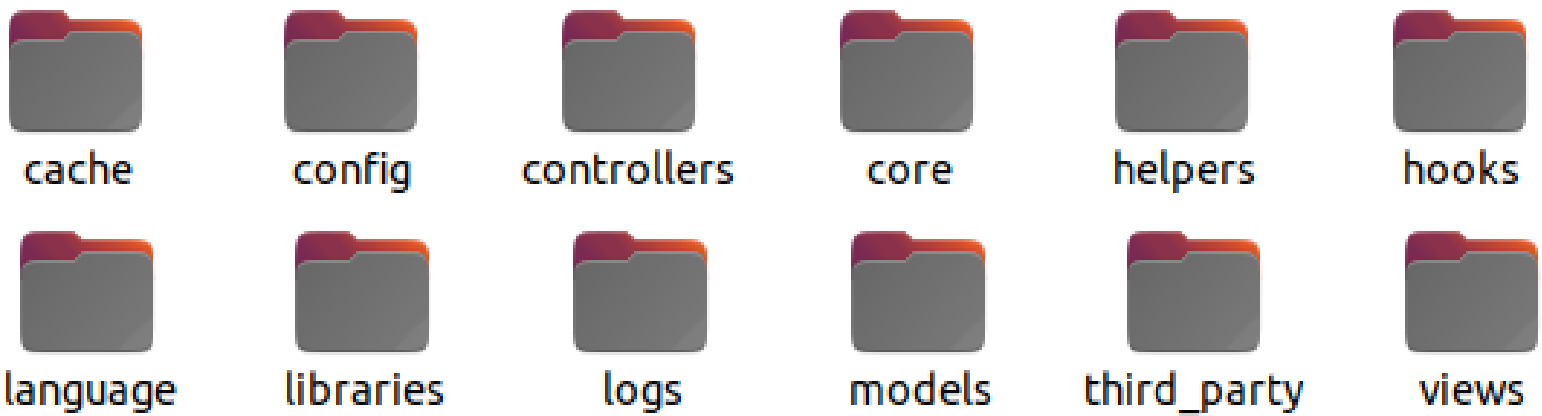
Ici, on a alors :

- localhost/CI3/ : adresse de mon site,
- index.php : point d'entrée du site,
- Fiche : un contrôleur (i.e. une classe Fiche héritant de CI_Controller),
- view : une méthode de ma classe Fiche,
- 12 : le paramètre de ma méthode view.

La méthode view de la classe Fiche générant la fiche d'un film dont l'identifiant est donné en paramètre (plus de détails dans l'exemple).

Installer CI3

- CI3 se télécharge à l'adresse https://codeigniter.com/download.
- L'installation est juste une décompression de l'archive dans votre public_html (par exemple dans un dossier CI3).
- Votre application est alors accessible à votre adresse `dwarves.iut-fbleau.fr/~login/CI3/`
- L'archive comprend notamment un dossier application, qui est le seul endroit où l'on créera/modifiera des fichiers !



Configurer CI3

Dans le dossier application/config/ :

config.php

On indique l'url de notre site en définissant :

`$config['base_url'] = '/CI3/';`

database.php

On indique notre serveur de base de données :

```
$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'dwarves.iut-fbleau.fr',
    'username' => 'login',
    'password' => 'mdp',
    'database' => 'login',
```

routes.php

On indique le contrôleur d'entrée du site :

`$route['default_controller'] = 'Films';`

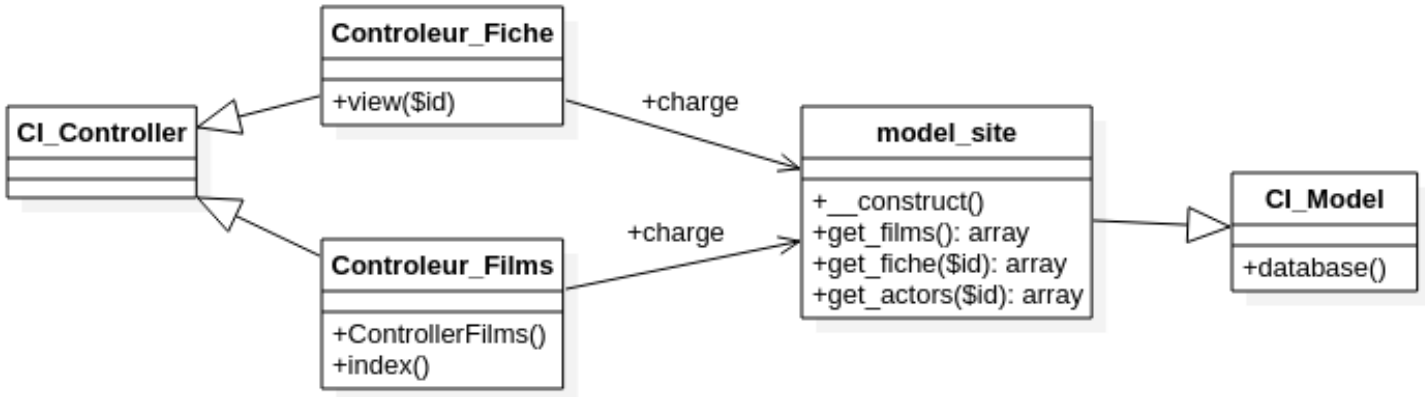
CI3 : Exemple

On possède une base de données de cinéma. On souhaite avoir un site permettant d'afficher la liste des films, mais aussi de pouvoir afficher les informations concernant un film précis dans une fiche de film.

Films

Titre	Année	Genre	Réalisateur
58 minutes pour vivre	1990	Action	Renny Harlin
Alien	1979	Science-fiction	Ridley Scott
American Beauty	1999	Comédie	Sam Mendes
Bad Lieutenant	1992	Drame	Abel Ferrara
Blade Runner	1982	Action	Ridley Scott
De bruit et de fureur	1988	Drame	Jean-Claude Brisseau
Eyes Wide Shut	1999	Thriller	Stanley Kubrick
Fenêtre sur cour	1954	Suspense	Alfred Hitchcock
Gladiator	2000	Drame	Ridley Scott
Godzilla	1998	Action	Roland Emmerich
Impitoyable	1992	Western	Clint Eastwood
Jackie Brown	1997	Policier	Quentin Tarantino

Structure du projet



Le mini-(mini-)projet comporte 5 fichiers :

- Un modèle : model_site qui comprend les fonctions d'accès à la base de données,
- 2 contrôleurs qui s'occuperont de charger les données grâce au modèle :
  - Films : point d'entrée du site, d'où la méthode index(),
  - Fiche : Qui sera utilisé via la fonction view, qui prend en argument l'identifiant d'un film.
- 2 vues, qui mettent en page les données fournies par leur contrôleur :
  - La vue Films,
  - La vue Fiche d'un film.

CI3-Exemple : Le modèle

```php
class Model_site extends CI_MODEL {

    public function __construct(){
            $this->load->database();
    }

    public function get_films()
    {
    $res=$this->db->query("SELECT Film.titre, Film.annee, Film.idFilm,
                                Film.genre, Artiste.nom, Artiste.prenom
                           from Film INNER JOIN Artiste
                           where Film.idMes=Artiste.idArtiste
                           ORDER by Film.titre");
    return $res->result_array();
    }

    public function get_fiche($id){
            $req=$this->db->query("SELECT * from Film INNER JOIN Artiste
                          where idFilm=$id AND Film.idMes=Artiste.idArtiste");
                return $req->row();
    }

    public function get_actors($id){
    $req=$this->db->query("SELECT prenom,nom,nomRole from Artiste INNER JOIN Role
                          where Role.idFilm=$id AND Role.idActeur=Artiste.idArtiste");
    return $req->result_array();
    }
}
```

CI3-Exemple : Le modèle (2)

```php
public function get_films()
{
$res=$this->db->query("SELECT Film.titre, Film.annee, Film.idFilm,
                              Film.genre, Artiste.nom, Artiste.prenom
                       from Film INNER JOIN Artiste
                       where Film.idMes=Artiste.idArtiste
                       ORDER by Film.titre");
return $res->result_array();
}
```

- La syntaxe "orienté objet" n'est pas très éloignée de celle de Java (public, class, extends,...). Le reste est bien sûr du PHP.
- La fonction __construct() est un constructeur. Elle sert ici à créer une connexion avec la BD.
- L'opérateur $->$ sert la même fonction que le . dans Java : On appelle un attribut ($this$->$db) ou une méthode ($req$->$result_array()) d'un objet.

CI3-Exemple : Contrôleur Films

```php
class Films extends CI_Controller{

    public function Controlefilms(){
        $this->load->model('Model_site');
        $films=$this->Model_site->get_films();
        $data=array('films' => $films);
        $this->load->view('header');
        $this->load->view('films',$data);
        $this->load->view('footer');
    }

    public function index()
    {
        $this->Controlefilms();
    }
}
```

La fonction index() est appelée lorsque l'on arrive initialement sur le site.
Elle ne fait que lancer la fonction ControleFilms().

CI3-Exemple : Contrôleur Films

```php
class Films extends CI_Controller{

    public function Controlefilms(){
        $this->load->model('Model_site');
        $films=$this->Model_site->get_films();
        $data=array('films' => $films);
        $this->load->view('header');
        $this->load->view('films',$data);
        $this->load->view('footer');
    }

    public function index()
    {
        $this->Controlefilms();
    }
}
```

La fonction ControleFilms() charge le modèle et récupère les données. Les données sont mises dans le tableau $data qui est fourni à la vue films générant le code html de la page.

CI3-Exemple : Vue Films

```
<h2>Films</h2>
<table>
    <thead>
        <tr>
            <th>Titre</th>
            <th>Année</th>
            <th>Genre</th>
            <th>Réalisateur</th>
        </tr>
    </thead>
    <tbody>

<?php

foreach($films as $film){
    echo "
    <tr>
        <td><a href='/CI3/index.php/Fiche/view/{$film['idFilm']}'>
                {$film['titre']}</a></td>
        <td>{$film['annee']}</td>
        <td>{$film['genre']}</td>
        <td>{$film['prenom']} {$film['nom']}</td>
    </tr>";

}

?>
```

La vue Films a accès au tableau $films fourni par $data.
À noter : La balise <a> génère un lien :
`index.php/Fiche/view/idFilm`

CI3-Exemple : Contrôleur Fiche

```php
class Fiche extends CI_Controller{

    public function view($id = 12){

        $this->load->model('Model_site');
        $film=$this->Model_site->get_fiche($id);
        $actors=$this->Model_site->get_actors($id);
        $data=array('film' => $film,'actors' => $actors);
        $this->load->view('header');
        $this->load->view('fiche',$data);
        $this->load->view('footer');
    }

}
```

Nous avons ici que la fonction view, prenant en argument un $id.
Elle est appelée lorsque l'on suit le lien précédent :
`index.php/Fiche/view/idFilm`
Elle charge les données du film et des acteurs du film, les mets dans la variable $data et les envoie à la vue fiche.

CI3-Exemple : Vue Fiche

```php
<h2><?php echo $film->titre; ?></h2>
<img src="<?php echo $film->urlImage; ?>">
<table>
<?php
    echo "<tr><td><b>Année :</b>  {$film->annee}</td></tr>";
    echo "<tr><td><b>Réalisateur :</b>  {$film->prenom} {$film->nom}</td></tr>";
    echo "<tr><td><b>Genre :</b>  {$film->genre}</td></tr>";
    echo "<tr><td><b>Résumé :</b>  {$film->resume}</td></tr>";
    echo "<tr><td><b>Pays :</b>  {$film->codePays}</td></tr>";
    echo "<tr><td><b>Casting :</b><br> ";
    foreach ($actors as $act) {
        echo "{$act['prenom']} {$act['nom']} as {$act['nomRole']}, <br> ";
    }
    echo "</td></tr>";
?>
</table>
<a href="/CI3/">Retour</a>
```

À noter que la variable $film fournie par data est un objet ! On accède à ses attributs avec $->$. C'est du à la fonction get_fiche du modèle qui renvoie un req->$row() (car on n'attend qu'une seule ligne de résultat).
Le lien de retour est simplement la racine de notre application.