



# PHP OO MODÈLE MVC



Aurélien Delorme  
OCT 2021



Modèle MVC

Les composants d'un modèle MVC

Exemple de projet MCV

Le routeur

Les contrôleurs

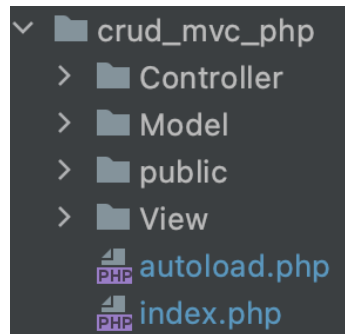
Le modèle (lui on le connaît déjà)

La vue

Démonstration

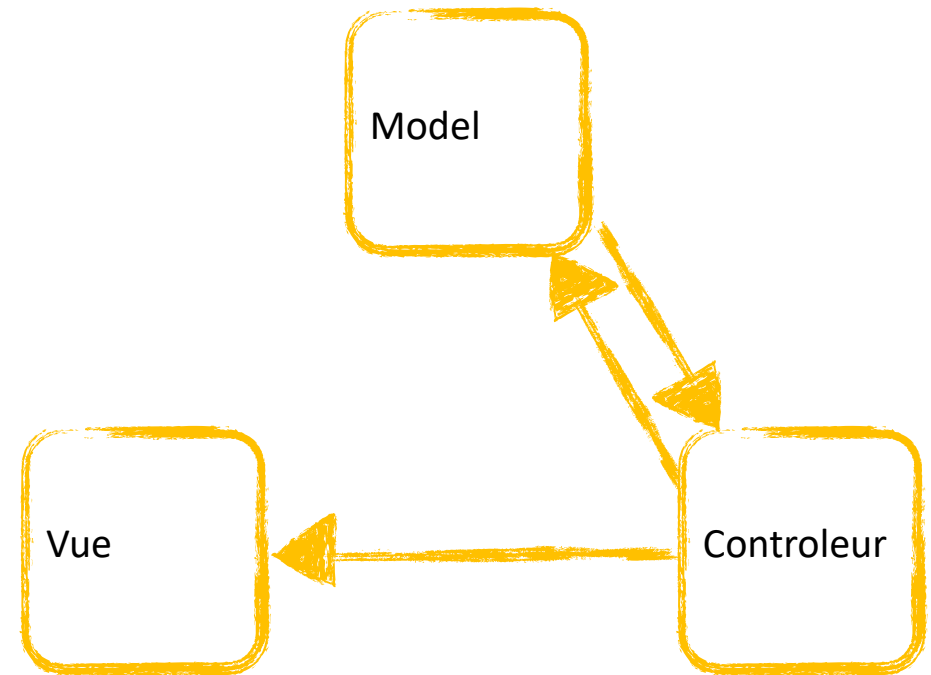


## Modèle MVC



L'utilisation d'un modèle MVC nous permettra :

- De **structurer** notre projet
- D'utiliser une **méthode de développement standardisé**
- Rendre la maintenance plus facile
- Permet d'utiliser un **pattern de développement évolutif**
- Dissocie le SQL du PHP du HTML/CSS/JS



## Les composants d'un modèle MVC

## MODEL

Gestion de notre BDD

Contient les objets  
représentant notre BDD

Contient des Managers  
permettant de requêtes  
sur notre BDD. Ils feront  
des requêtes SQL

## VUE

Permet la gestion de  
l'affichage de notre site  
internet.

Correspond à la partie  
visible par l'utilisateur.

Nous y retrouverons  
essentiellement du  
html

## CONTROLLER

Gère les traitements  
applicatif.

Devra contrôler le  
contenu de la requête  
HTTP, interroger le model  
si nécessaire et renverra  
notre Vue.

## Le routeur

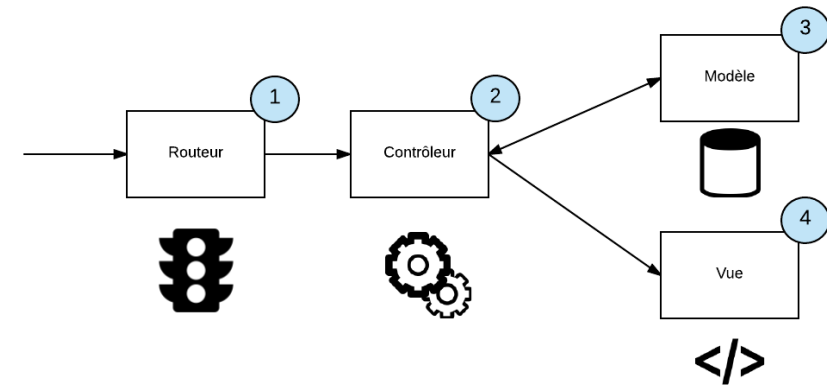
Point d'entrée de notre application.

Toutes les requêtes traitées par notre application passeront par ce point d'entrée.

Il permettra de rediriger la requête vers le bon contrôleur.

Utilise les paramètres envoyés dans l'URL

Nous utiliserons la variable super globale \$\_GET pour identifier la demande HTTP



```
<?php
if (empty($_GET)) {
    header( header: 'Location: index.php?controller=default&action=homepage');
}

if($_GET['controller'] === 'contact' ){
    $controller = new ContactController();
    if($_GET['action'] == 'addForm'){
        $controller->addForm();
    }
}

if($_GET['controller'] === 'default' ){
    $controller = new DefaultController();
    if($_GET['action'] == 'addForm'){
        $controller->home();
    }
}
}
```

## Le Contrôleur

Controller &gt; ContactController.php

Notre contrôleur sera un objet PHP qui traitera la demande envoyé par le routeur.

En renvoyant un contrôleur, le routeur va exécuter une méthode du Contrôleur

Notre contrôleur **traitera nos formulaires**, appeler le modèle si l'on a besoin d'accéder à nos données (insertion, mise à jour, sélection, suppression).  
Renverra une vue ou une redirection

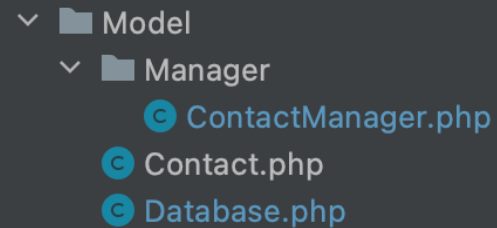
```
<?php

class ContactController{

    public function add()
    {
        $contact = new Contact( id: null, $_POST['name'], $_POST['tel'], $_POST['mail'], $_POST['picture']);
        $contactManager = new ContactManager();
        $contactManager->insert($contact);
        header( header: 'Location: /tp_open/menu.php?controller=default&action=home');
    }

    public function addForm(){
        require ('View/insertForm.php');
    }
}
```

## Le Modèle



```

v Model
  v Manager
    © ContactManager.php
    © Contact.php
    © Database.php

```

Contient la représentation objet de notre BDD ainsi que les managers qui permettront d'effectuer des requêtes MySQL

Contient nos managers (ils seront en charge d'effectuer des requêtes sur notre BDD)

Nos managers contiendront une classe abstraite permettant la connexion à notre BDD

Nous aurons un Manager par représentation objet de notre BDD. Nous en aurons souvent un pour chaque table de notre BDD

## Le Modèle






```
<?php
class Contact{
    private $id;
    private $name;
    private $tel;
    private $mail;
    private $picture;

    public function __construct($id,$name,$tel,$mail,$picture){
        $this->id = $id;
        $this->name = $name;
        $this->tel = $tel;
        $this->mail = $mail;
        $this->picture = $picture;
    }

    /**
     * Get the value of id
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set the value of id
     *
     * @return self
     */
    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }
}
```

Contact	
 id	int
 name	varchar(250)
 tel	varchar(250)
 mail	int
 picture	int

Exemple de code source de la représentation objet de notre BDD :

Cet objet représentera la table contact de notre BDD.

On retrouvera ici tous les champs de notre BDD. Ce seront les attributs de notre objet.

Nous retrouverons un constructeur permettant d'instancier de nouveaux objets

Nous retrouverons les accesseurs (méthodes publiques get / set). Ils permettront d'accéder à nos attributs privés



## Le Modèle

Model &gt; Manager &gt; ContactManager.php

```
<?php
class ContactManager extends DBManager {
    public function __construct()
    {
        parent::__construct();
    }

    public function selectAll()
    {
        $contacts = [];
        $sql = 'SELECT * FROM myContacts';
        foreach ($this->bdd->query($sql) as $row){
            $contacts[] = new Contact($row['id'],$row['name'],$row['tel'],$row['mail'],$row['picture']);
        }
        return $contacts;
    }

    public function insert(Contact $contact)
    {
        $name = $contact->getName();
        $tel = $contact->getTel();
        $mail = $contact->getMail();
        $picture = $contact->getPicture();
        $req = $this->bdd->prepare( query: "INSERT INTO myContacts (name, tel, mail, picture) VALUES (?, ?, ?, ?)");
        $req->bindParam( param: 1, &var: $name);
        $req->bindParam( param: 2, &var: $tel);
        $req->bindParam( param: 3, &var: $mail);
        $req->bindParam( param: 4, &var: $picture);
        $req->execute();
        $contact->setId($this->bdd->lastInsertId());
    }
}
```

Exemple de code source d'un Manager :

Il étendra de la classe DBManager qui sera en charge de connecter la base de données MySQL

Il effectuera des Requêtes MySQL en fonction du besoin (sélection, insertion, mise à jour).

Il transformera nos résultats MySQL en Objet grâce à la représentation objet vu dans le slide précédent



## La vue

```
<?php
    include ('stylesheet.html');
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>home</title>
</head>
<body>
<div class="d-flex justify-content-center">
<div class="col-9 border border-primary mt-5">
    <div class="d-flex justify-content-around border rounded mt-2 mb-2">
        <a class='pl-2' href="index.php?controller=contact&&action=addForm">
            <button style="..." type='submit' class="btn btn-primary mt-2 mb-2 text-uppercase">
                <i class="far fa-plus-square"></i>
            </button>
        </a>
        <a class='pl-2' href="#">
            <button style="..." type='submit' class="btn btn-secondary mt-2 mb-2 text-uppercase"><i class="fas fa-info-circle"></i></button>
        </a>
        <a class='pl-2' href="#">
            <button style="..." type='submit' class="btn btn-danger mt-2 mb-2 text-uppercase">
                <?php
                    echo $data->count();
                ?>
            </button>
        </a>
    </div>
    <h4 class="text-uppercase text-secondary border rounded p-1">ma liste de contacts</h4>
<!-- font-italic-->
<?php
    include ('Parts/ListContacts.php');
?>
</div>
```

Exemple de code source d'une Vue :

- Contient essentiellement du HTML
- Utilise les données renvoyées par le contrôleur
- Ne fait pas de traitement sur les données
- Représente la partie visible par l'utilisateur



# DEMONSTRATION





# ECHANGES / QUESTION

