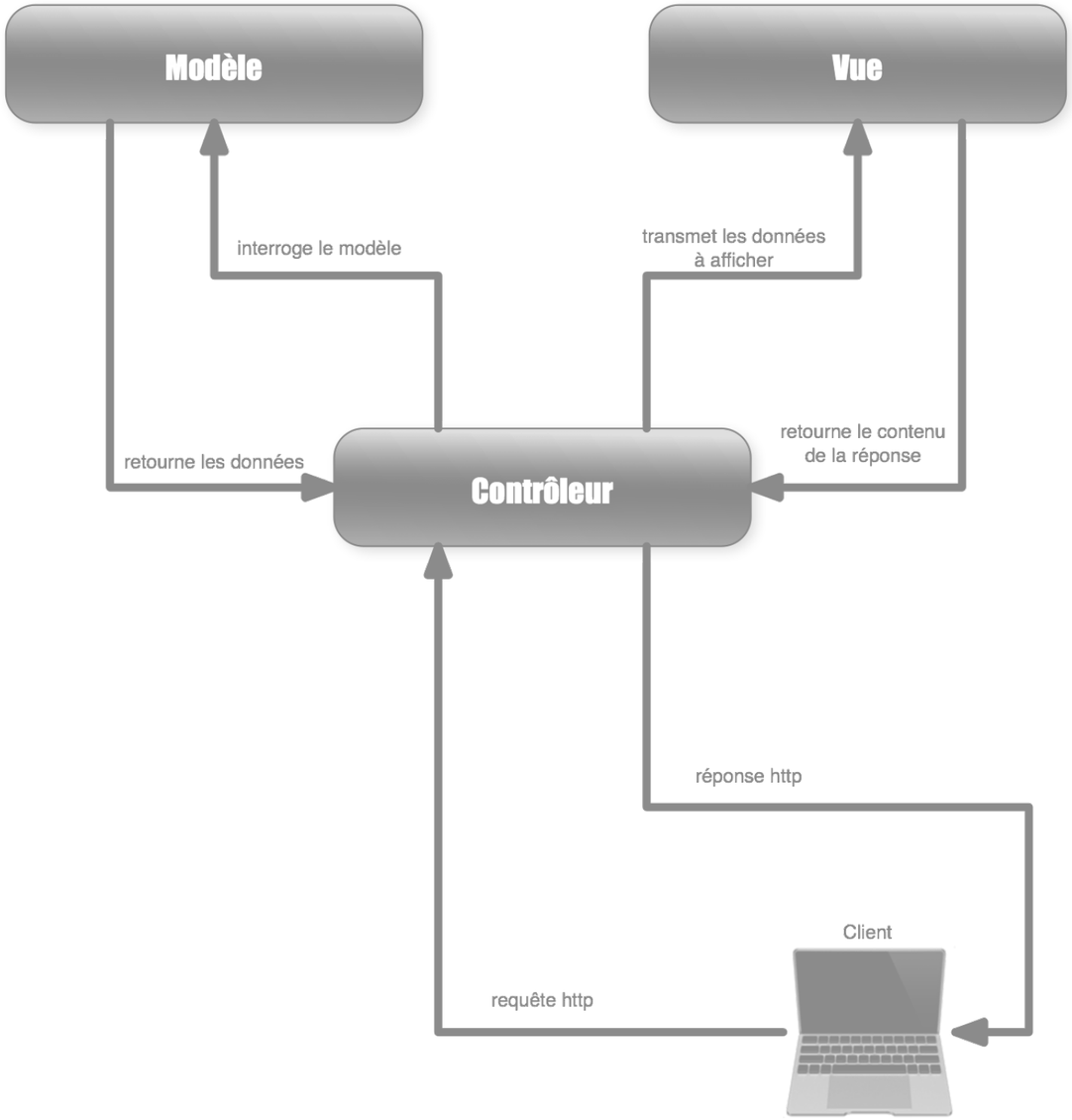
ARCHITECTURE MVC
SANS OBJET

LE MODÈLE MVC



LE CONTRÔLEUR

» un fichier php

» son rôle, transformer une requête en réponse

» il fait appel au modèle pour récupérer les données

» il transmet les données à la vue

LE MODÈLE

- il est chargé de communiquer avec la source de données
- il s'occupe également de la logique métier
  - validation des données
  - calculs sur les données
  - application des règles métier

LA VUE

» Elle s'occupe de la présentation des données

» Elle ne contient aucune logique de traitement, juste de l'affichage

PREMIÈRE APPROCHE : LE CONTRÔLEUR

```php
//fichier client-list.php

//inclusion du modèle,
//une bibliothèque de fonctions
//pour accèder aux données
require "models/client.php";

//récupération des données
//avec un appel à une fonction du modèle
$clientList = getClientList();

//inclusion de la vue, les variables
//définies dans le contrôleur
//sont accessibles dans la vue
require "views/client-list.php");
```

PREMIÈRE APPROCHE : LE MODÈLE

```php
//Exécute une requête et retourne les données
function getClientList() {
    $pdo = getPDO();
    $rs = $pdo->query("SELECT * FROM clients");

    return $rs->fetchAll(PDO::FETCH_ASSOC);
}
```

PREMIÈRE APPROCHE : LA VUE

```
<!DOCTYPE html>
<html>
    <head>
        <title>Liste des clients</title>
    </head>
    <body>
        <ul>
        <?php foreach($clientList as $client): ?>
            <li><?= $client["nom"]; ?>
       <?php endforeach; ?>
       </ul>
    </body>
</html>
```

CONCLUSION

» On échange un seul fichier complexe contre plusieurs fichiers simples

» Dans chaque vue, on répète le squelette html (factorisation possible)

PROBLÉMATIQUE DE LA VUE

- ne pas polluer la logique métier avec celle de la présentation
- adopter une architecture modulaire
  - on peut utiliser la même logique avec des vues différentes
  - on peut utiliser la même vue dans des contrôleurs différents
- factoriser les éléments récurrents dans un gabarit

PREMIÈRE SOLUTION
LE GABARIT

Le fichier inclus est stocké dans une variable

```
<!-- fichier views/layout.php -->
<html>
<head>...</head>
<body>
    <div>
        <?php require "{$page}.php"; ?>
    </div>
</body>
</html>
```

PREMIÈRE SOLUTION
LA VUE

Ne contient que la partie variable

```
<!-- fichier views/clientList.php -->
<ul>
    <?php foreach($clientList as $client): ?>
        <li><?= $client["nom"]; ?>
    <?php endforeach; ?>
</ul>
```

PREMIÈRE SOLUTION
LE CONTRÔLEUR

Détermine la vue et appelle le gabarit

```php
//fichier client-list.php

$page = "views/clientList";

require "views/layout.php";
```

CONCLUSION

C'est mieux mais il est possible de factoriser la gestion de la vue dans des fonctions

» Récupérer le contenu de l'interprétation PHP dans une variable

» Définir les variables disponibles pour la vue

FACTORISATION DE LA VUE 1

Quelques fonctions dans un fichier lib/mvc.php

```php
function getViewContent($view, array $data = []){
    //Mise en tampon du résultat de l'interpréteur PHP
    ob_start();

    //Exportation des variables
    extract($data);

    //inclusion de la vue
    require "views/{$view}.php";

    //Récupération du contenu du tampon dans une variable
    $viewContent = ob_getClean();

    return $viewContent;
}
```

FACTORISATION DE LA VUE 2

```php
function getRenderedView($view, array $data = [], $layout="default-layout"){
    //Récupération du contenu de la vue
    $viewContent = getViewContent($view, $data);

    //Ajout de la vue aux données
    $data["viewContent"] = $viewContent;

    //Obtention du layout
    $rendered = getView($layout, $data);

    return $rendered;
}
```

FACTORISATION DE LA VUE 3
LE GABARIT

```
<!-- fichier views/default-layout.php -->
<html>
<head>...</head>
<body>
    <div>
        <?=$viewContent?>
    </div>
</body>
</html>
```

FACTORISATION DE LA VUE 4
LE CONTRÔLEUR

```php
//fichier client-list.php

//importation de la bibliothèque
require "lib/mvc.php";

//affichage de la vue
echo getRenderedView("client-list");
```

PISTES D'AMÉLIORATIONS

» Gérer les erreurs (vue ou gabarit inexistant)

» Échapper automatiquement les variables

» Gérer l'héritage à plusieurs niveaux

» Gérer des blocs multiples

UN FRAMEWORK
PLUS MODERNE

» Point d'entrée unique de l'application

» Sécurisation du code source

» Réécriture d'url

ARBORESCENCE

- Projet
  - src
    - controllers
    - views
    - models
  - lib
    - mvc.php
  - web
    - index.php

LE POINT D'ENTRÉE

```php
//fichier index.php

//Définition des chemins
define("ROOT_PATH", dirname(__DIR__));
define("SRC_PATH", ROOT_PATH. "/src");
define("WEB_PATH", __DIR__);

//Inclusion de la bibliothèque
require ROOT_PATH."/lib/mvc.php";

//Définition du fichier
$controllerFile = getController();



require $controllerFile;
```

L'OBTENTION DU CONTRÔLEUR
DANS MVC.PHP

```php
function getController(){
    //Récupération du nom de la page
        $page = filter_input(
        INPUT_GET,"page",
        FILTER_SANITIZE_STRING
    ) ?? "default";

    //Définition du fichier
    $controllerFile = SRC_PATH."/controllers/{$page}.php";

    //Gestion d'un contrôleur inexistant
    if(! file_exists($controllerFile)){
        $controllerFile = SRC_PATH."/controllers/notFound.php";
    }

    return $controllerFile;
}
```

MODIFICATION DE LA GESTION DES VUES

DANS LIB/MVC.PHP

```php
function getViewContent($view, array $data = []){
    //Mise en tampon du résultat de l'interpréteur PHP
    ob_start();

    //Exportation des variables
    extract($data);

    //inclusion de la vue
    require SRC_PATH."/views/{$view}.php";

    //Récupération du contenu du tampon dans une variable
    $viewContent = ob_getClean();

    return $viewContent;
}
```

UTILISATION

```
index.php?page=contact => controllers/contact.php
```

FONCTION HELPER POUR LES LIENS

```php
function getPath($controller, $params){
    $url = "/index.php?page={$page}";
    foreach($params as $key=>$val){
        $url .= "&$key=$val";
    }

    return $url;
}
```

HÔTE VIRTUEL

SERVIR DES PAGES EN DEHORS DE HTDOCS

Dans le fichier conf/extra/httpd-vhosts.conf

```
<VirtualHost *:80>
    DocumentRoot "/projets/app/web"
    ServerName projet.local

    <Directory "/projets/app/web">
        Options All
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

FICHIER HOSTS

» /etc/hosts pour linux et Mac OS

» C:\Windows\System 32\drivers\etc\hosts pour Windows

```
127.0.0.1    projet.local
```

RÉÉCRITURE D'URL

```
<IfModule mod_rewrite.c>
    RewriteEngine on

    # La ressource ne correspond pas à un fichier présent sur le serveur
    RewriteCond %{REQUEST_FILENAME} !-f
    # La ressource ne correspond pas à un dossier présent sur le serveur
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^(.*)$ /index.php?page=$1 [L,QSA]

</IfModule>
```

UTILISATION

http://projet.local/contact=>index.php?page=contact

```php
//Fonction helper pour les liens
function getPath($controller, $params){
    $host = $_SERVER["HTTP_HOST"];
    $url = "{$host}/{$controller}";

    $keys = array_keys($params);
    $nbKeys = count($keys);
    if($nbKeys >0) $url .= "?";
    }

    for($i=0;$i < $nbKeys; $i++){
        if($i >0) $url .= "&";
        $url .= "$key=$val";
    }

    return $url;
}
```