1er mode de séparation

Design Pattern : Model-View-Controller

Separation of concerns. (Séparation des préoccupations)

L'interface utilisateur a tendance à changer , c'est donc une bonne idée de pouvoir apporter des changements dans l'interface utilisateur sans affecter les parties du programme qui ne sont pas liées à l'interface utilisateur. La partie métier ou la partie fonctionelle.

Pluggable user interface (Interface utilisateur connectable)

Parfois, la même fonctionnalité de base est nécessaire pour différents types d'interface utilisateur. Vous pouvez télécharger un fichier sur le Web en utilisant un utilitaire de ligne de commande ou un navigateur Web graphique. Si vous programmez à la fois l'utilitaire et le navigateur, il est logique qu'ils utilisent le même code pour effectuer le véritable travail de téléchargement.

Division du travail

Souvent, des personnes différentes sont affectées à l'interface utilisateur et à la logique du domaine. Dans le cas des applications web, la mise en page et le style des pages web sont souvent réalisés par des concepteurs web qui ne sont pas des programmeurs.

Les templates

Un template est un modèle web - une page écrite dans un langage qui consiste principalement en un balisage HTML, mais qui a aussi une façon d'exprimer la façon dont le contenu généré dynamiquement doit être inclus. Un moteur de modèle est une bibliothèque logicielle qui nous permet de générer du code HTML à partir des modèles et de spécifier le contenu dynamique à inclure.

Exemple des programmes php qu'on a créé il y a du code HTML du code php avec accès à la base de données , des includes éventuels, du javascript, du css…etc…

Nous allons dans un premier temps tenter de séparer les préoccupations , d'un côté l'affichage , de l'autre côté la logique métier en php procédural pur sans moteur de template, php étant lui-même au départ un moteur de template.

L'avantage réside dans la compréhension et la maintenance du code. Il faut laisser la partie affichage le plus possible sans code métier pour un meilleur découplage et pour faciliter l'accès aux designers.

Développement de type MVC

Le design pattern Modèle-Vue-Contrôleur (Model-View-Controler) en php procédural.

- Structuration du code claire
- Maintenance
- Réutilisabilité du code
- La compréhension du MVC facilite l'apprentissage des frameworks.

Principe:

Le modèle (model):

représente les **données** et l'**état** des données
Le modèle ne contient pas de HTML
Le modèle est en interaction avec la base de données (SQL)

La vue (view):

Présentation des données à l'utilisateur

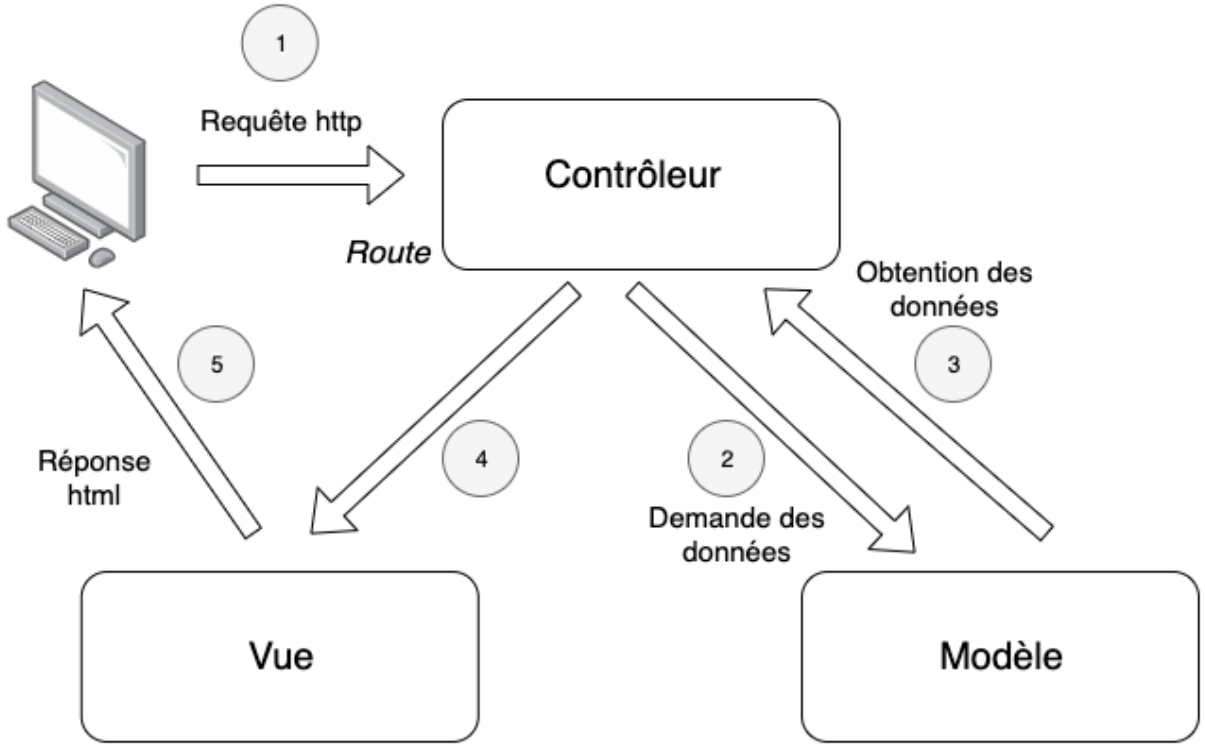
Affichage : HTML (css / javascript client , uniquement interactions dans le DOM)

un peu de PHP (le minimum possible)

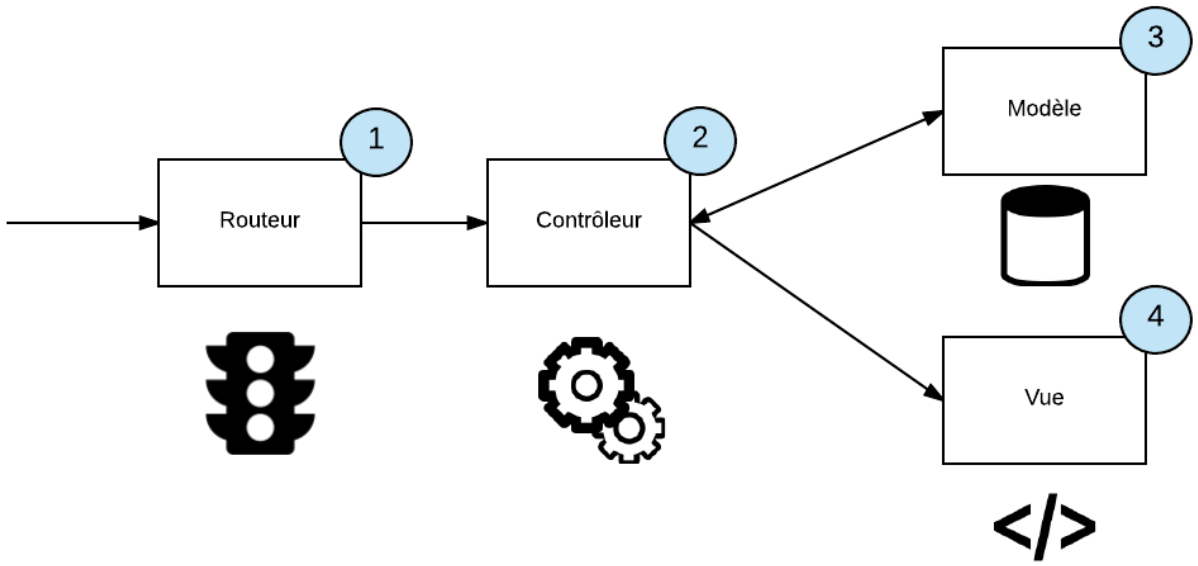
Le contrôleur (controller):

Assure la relation entre le modèle et la vue
Exécute des **actions** pour répondre à la demande de l'utilisateur.

Utilise le modèle pour interroger la base de données



Chaque action correspond à l'affichage d'un page web en fonction d'une route



En fonction de la route (URL) on va sélectionner un controller sur lequel on va effectuer une action pour afficher une page.

Pour accéder aux différentes pages du site web l'url doit contenir au moins le nom du controller et au moins une action sur ce controller.

on a une URL du type

http://localhost:8080/afpa/mvc1/index.php?controller=etudiant&action=creer

http://localhost:8080/afpa/mvc1/index.php?action=controller/action

Il est recommandé de faire de la réécriture d'URL.

transformer:

http://localhost:8081/index.php?action=etudiant/ajouter

en

http://localhost:8080/afpa/mvc1/etudiant/ajouter

exemple avec modification:

http://localhost:8080/afpa/mvc1/etudiant/modifier/1

REQUËTE

Le fichier index.php va faire office de routeur sur les différents contrôleurs. Il faut absolument passer par ce routeur.

Le routeur va identifier quel controller utiliser et l'action à utiliser .

Le contrôleur récupère le données puis les renvoi à la vue

Exemple d'un système de gestion de cours.

1) Espace membres étudiants
    1) créer un compte
    2) se connecter et se déconnecter
    3) profil
    4) afficher cours
    5) s'inscrire à un cours

2) Gestion des cours
    1) CRUD cours (Create Read Update Delete).

Relation n à n entre les étudiants et les cours donc il faut créer une 3 ème table qui fait le lien.

Petite incursion dans la modélisation.

Dessiner le diagramme de tables.
Créer les tables en base de données.

…

Contrôleur frontal : le routeur (router)

En MVC on veut forcer la redirection pour centraliser vers 1 seul point d'accès. Donc ici on centralise vers index.php (les autres pages ne sont pas accessibles directement).

Pour cela , une seule porte d'entrée.

Comment rédiger l'utilisateur vers le contrôleur principal: index.php

Le serveur HTTP communément utilisé pour des sites web php est le serveur **Apache**, mais ceci ne fonctionnera pas avec un serveur **nginx** de plus en plus utilisé.

Réécriture de nos URLs désirées URL:"Uniform Resource Locator" (URI Uniform Resource Identifier pour internet).

On désire pouvoir écrire plus tard les URLs ainsi.

```
localhost:8080/afpa/formation/students/add
localhost:8080/afpa/formation/students/update/23
```

Routage pour des URL brutes (sans index.php sans ? …etc…).
C'est mieux pour le SEO

Comment réécrire les adresses vues ci-dessus

index.php

Si je remplace par n'importe quel nom j'ai => ERREUR 404 (logique)

Comment rediriger ?

Modifier un fichier **.htaccess** à la racine de notre site

```
RewriteEngine on
RewriteRule ^etudiants$ index.php
```

Tester la requête
formation = racine de notre site.
http://localhost:8080/afpa/formation/etudiants

Routeur

```
RewriteEngine on
RewriteRule  ^([A-Za-z0-9\-\_\/]*)$  index.php?action=$1
```

(expressions régulières), cette expression régulière [A-Za-z0-9\-_\/]* veut dire la 1ère séquence de chaine possédant ces caractères sera rédirigée vers index.php/action=[cette séquence matérialisée par $1, $1 étant la 1ere regex entre parenthèses]

Tout ce qu'on tapera redirigera vers index.php?action=(ce qui correspondra à la regexp)

Exemple

dans index.php faire un `print_r($_GET);`

```
<?php

if ( $_GET['action'] ) {


// voir la doc replace.

    $params = explode("/", $_GET['action'] ) ;

    echo "param1:". $params[0] ; // students (controller)
    echo "<br/>" ;
    echo "param2:". $params[1] ; // connecter (action)
    echo "<br/>" ;
    echo "param3:". $params[2] ; // 45 (paramètre)
    echo "<br/>" ;

}
?>
```

essayer avec l'url :

```
http://localhost:8080/formation/controllerStudents/
connecter/45
```

```
if ( $_GET['action'] ) {

    $params = explode("/", $_GET['action'] ) ;

    if( $params[0]  != "" ) {
        $controller = $params[0];
        $action = "" ;
        if (isset($params[1])) { $action = $params[1] ; }

        // on a controller et action
        // on inclus donc ce controller

         require_once('controllers/'.$controller.'.php');

        // le folder controller n'est pas trouvable (relatif)
        //  vu qu'on a
         // modifié l'URL
        //  pour  savoir  où  on  se  trouve  affiche  de  l'URL
complet
        echo $_SERVER['SCRIPT_FILE'];
        // on va supprimer index.php de l'URL (au début) d'où
        require_once(RACINE.'controllers/'.
$controller.'.php');

        if(function_exists($action)){
            if ( isset($params[2] && isset($params[3] ) ) {
                $action($params[2],$params[3]);
            } else if (isset($params[2]) {
                $action($params[2]);
        }
        else { $actions(); }
        }
    else {
        echo "page page default"
    }

}
} else {
    echo "aucune action ";
}
```

Création de l’arborescence

Création des dossiers controllers , models, views, assets (css, js, images) , index.php

dossier Racine
- index.php
- .htaccess

Gestion des cours
Ajouter un cours
Afficher tous les cours
modif
suppression

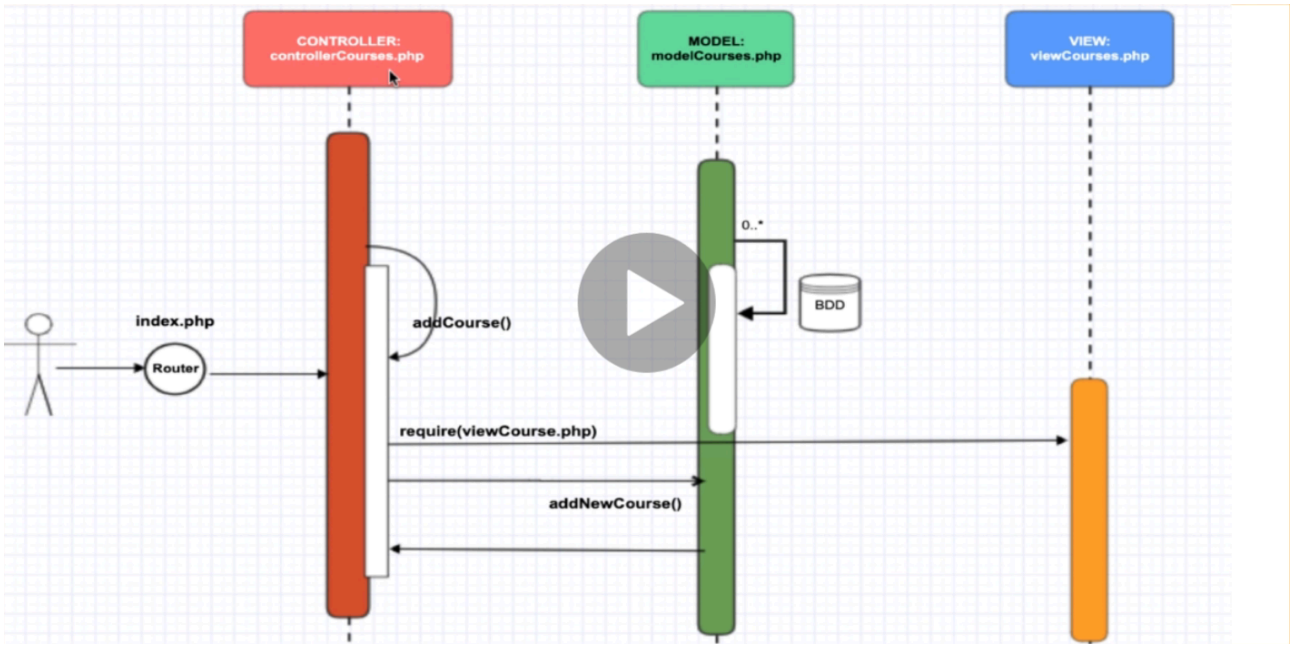
Pour chaque fichier model on va créer un fichier de connection:

model.php

```
function connexionBDD(){
       $host =
       $db_name =
       $user =
       $password =
       $connex = False ;
       if (!connexion) {
       try {
          $bddPDO = new PDO(…)
          return $bddPDO ;
       } catch(PDOException $ex) {
          echo $ex ;
       }
}
```

tester la connection

MVC -> ajout d’un cours



modelCours.php

```php
function addCours($pcode, $ptitre, $plangage ) {
    $bdd = connexionBDD() ;
    /*
     Une manière sécure d'effectuer une requete dans la base de
données
     requête préparée
    */
    $sql = $bdd->prepare('INSERT INTO courses(code,titre,langage)
VALUES (:code,:titre,:langage)');
    $sql->bindvalue(':code',$pcode);
    $sql->bindvalue(':code',$ptitre);
    $sql->bindvalue(':code',$plangage);

    $result = $sql->execute();
    return $result ;
}
```

Les requêtes préparées

1) Met la requête SQL en cache

2) Par défaut, PDO va appliquer un filtre pour vérifier le type du paramètre et utiliser sa fonction interne pour le "quote" de façon appropriée pour éviter la plupart des injections SQL. Il va au besoins échapper les caractères spéciaux selon les réglages du pilote PDO actif. PDO ne va pas te protéger des autres types d'attaques comme les injections javascript.

Pour se protéger des autres types d'injection, avant de les utiliser dans les requêtes préparées utiliser `htmlspecialchars()` ou n'importe quel autre moyen pour nettoyer les données.