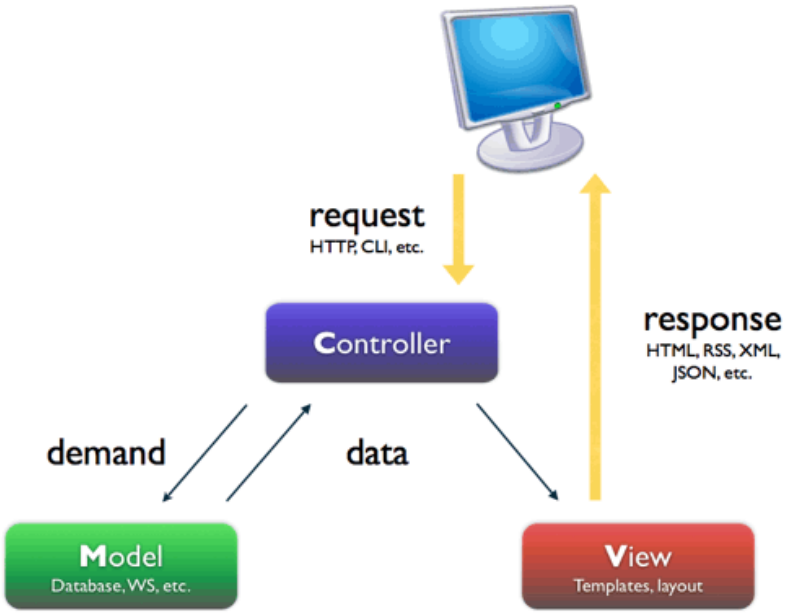
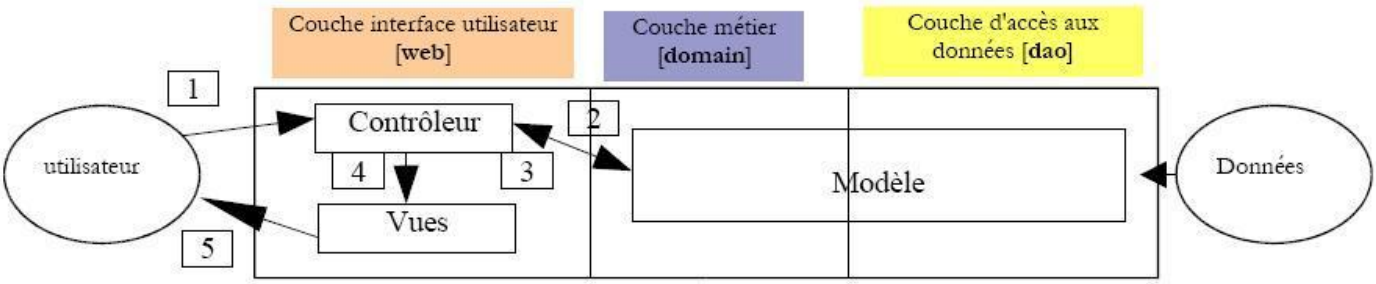
Modèle Model ⇔Vue ⇔Controller (MVC) dans les applications Web/PHP

Présentation classique :



Séparation des couches **présentation**, **traitement** et **accès aux données**

Une application web respectant ce modèle sera architecturée de la façon suivante :



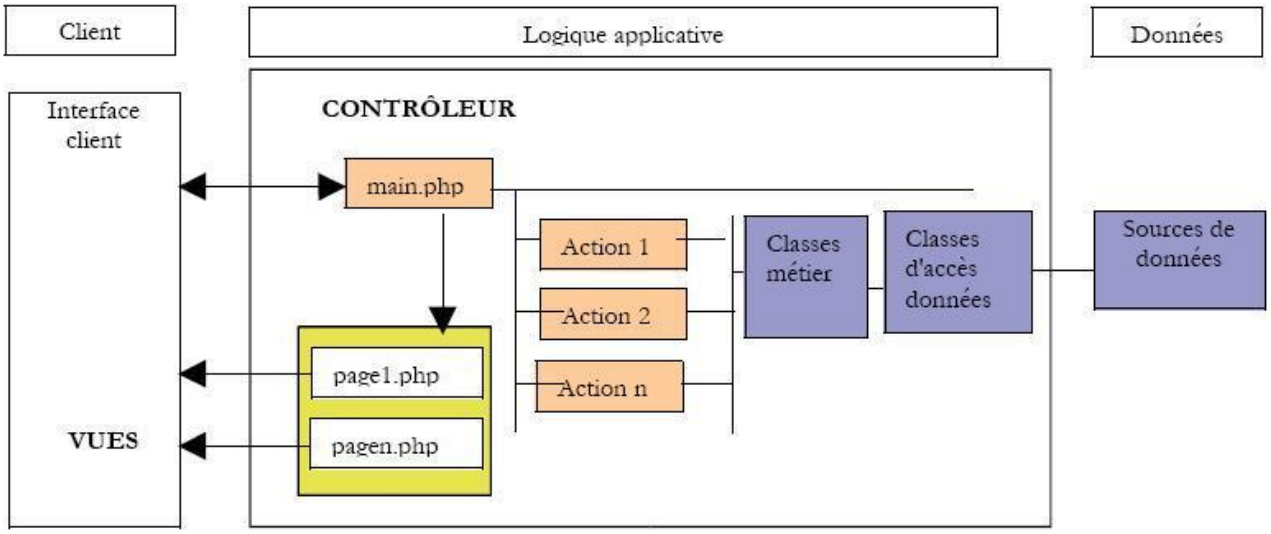
Le traitement d'une demande d'un client se déroule selon les étapes suivantes :

- Le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le C de MVC.
- Le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier, ce qu'on appelle le modèle M dans la structure MVC.
- Le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. Un exemple classique est
  • Une page d'erreurs si la demande n'a pu être traitée correctement
  • Une page de confirmation sinon
- Le contrôleur choisit la réponse (= vue) à envoyer au client. Celle-ci est le plus souvent une page contenant des éléments dynamiques. Le contrôleur fournit ceux-ci à la vue.
- La vue est envoyée au client. C'est le V de MVC.

Une telle architecture est souvent appelée "architecture 3-tier" ou à 3 niveaux.

L'architecture MVC est bien adaptée à des applications web écrites avec des langages orientés objet. On peut néanmoins faire un effort de structuration du code et de l'architecture de l'application afin de se rapprocher du modèle MVC :

- On mettra la logique métier de l'application dans des modules séparés des modules chargés de contrôler le dialogue demande-réponse. L'architecture MVC devient la suivante :



M=modèle	les classes métier, les classes d'accès aux données et la base de données
V=vues	les pages PHP
C=contrôleur	le script PHP de traitement des requêtes clientes, les scripts PHP [Action] de traitement des actions.

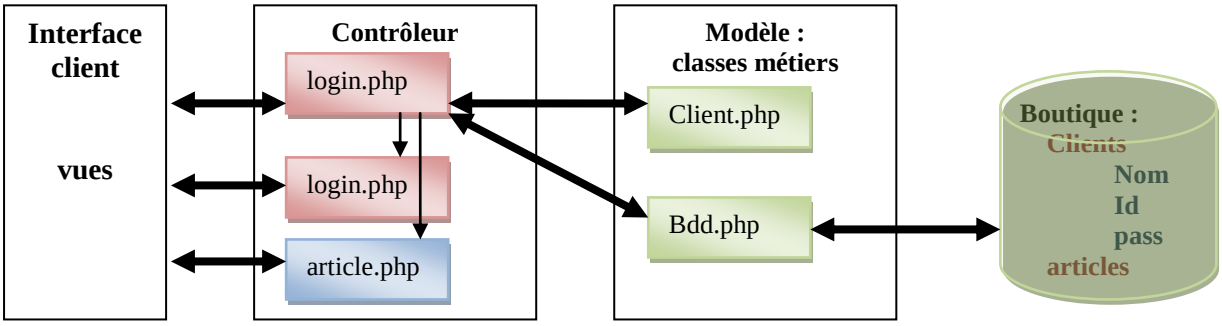
Dans le bloc [Logique Applicative], on pourra distinguer :

- Le programme principal ou contrôleur **[main.php]**, qui est la porte d'entrée de l'application.
- Le bloc [Actions], ensemble de scripts PHP chargés d'exécuter les actions demandées par l'utilisateur.
- Le bloc [Classes métier] qui regroupe les modules PHP nécessaires à la logique de l'application. Ils sont **indépendants du client**. Par exemple, la fonction permettant de calculer un impôt à partir de certaines informations qui lui sont fournies en paramètres, n'a pas à se soucier de la façon dont ont été acquises celles-ci.
- Le bloc [Classes d'accès aux données] qui regroupe les modules PHP qui obtiennent les données nécessaires au contrôleur, souvent des données persistantes (BD, fichiers…)
- Les générateurs [pagex.php] des vues envoyées comme réponse au client.

Dans les cas les plus simples, la logique applicative est souvent réduite à deux modules :

- Le module [contrôle] assurant le dialogue client-serveur : traitement de la requête, génération des diverses réponses
- Le module [métier] qui reçoit du module [contrôle] des données à traiter et lui fournit en retour des résultats. Ce module [métier] gère alors lui-même l'accès aux données persistantes.

TP : mise en œuvre d'un login



Gestion d'un login:

Contrôleur: gère l'arrivée du login : nom/password
Les transmet au modèle.
Gère le retour du modèle pour présenter la vue correspondante.

Modèle Retourne **true** si le login existe dans la base de données, sinon **false**.

Vues Code html/javascript et ou php de la page suivante.
Rejeu de la page de login

1] Coder une base de données mysql nommée '**boutique'**.

2] Coder et peupler une table '**Client'** munie des enregistrements suivants :
Id, **nom**, **password**.

3] Coder la classe **Client**.php

```
<?php
class Client
{
  private $nom;
  private $pass;

  public function toString()
  {
      echo 'Client: '.$this->nom.'/'.$this->pass.'<br>';
      return $this->nom.'/'.$this->pass;
  }
  public function initialise($n , $p)
  {
      $this->nom = $n;
      $this->pass = $p;
  }
  public function Client($n , $p)
  {
      $this->initialise($n , $p);
  }

}
```

4] Coder la classe **Bdd** réalisant la requête et retournant true ou false.

```php
<?php

class Bdd
{
  private $client;


// Ouverture bdd boutique sur localhost, root, ""
  public function ouvrir($serv , $user , $passuser , $nom)
  {
      …
  }
//ferme la connexion à la base de données 'boutique'
  public function fermer()
  {
      . . .
  }
//retourne true si le login existe dans la base de données,
//false sinon
  public function valideUser($n , $p)
  {
      . . .
  }
//retourne l'objet $this->client
  public function getClient()
  {
      . . .
   }
```

5] Tester.

6] Coder une table '**article'** muni des enregistrements suivants :

Nom, **prix**, **cheminPhoto**, **stock**

7] Compléter la classe **Bdd** permettant de récupérer dans la base de données tous les articles.

8] Coder une classe '**Article'** gérant les articles afin d'être exploitée dans le contrôleur.

9] Compléter le contrôleur et la vue permettant d'afficher la liste des articles stockés dans la base de données.

10] Tester.