Modèle-Vue-Contrôleur (introduction)

Un exemple web avec PHP

IUT de Fontainebleau

15 mars 2015

Sommaire

Sommaire

Modèle MVC

Principe de conception d'applications (WEB ou pas) qui sépare les fonctions nécessaires en trois composants distincts :

Modèle	⇝	Gère les données
Vue	⇝	Gère la présentation (UI)
Contrôleur	⇝	Agit

Il existe de nombreux framework PHP basés sur cette architecture.

Le modèle

Gestion des données

- Gére les échanges avec la base de données.
- Abstrait sous forme de classes les entités (les tables) de la bd.
- On peut utiliser un ORM (object-relational mapping) : doctrine, pdoMap, Propel, etc ...

La vue

Gestion de la présentation des données

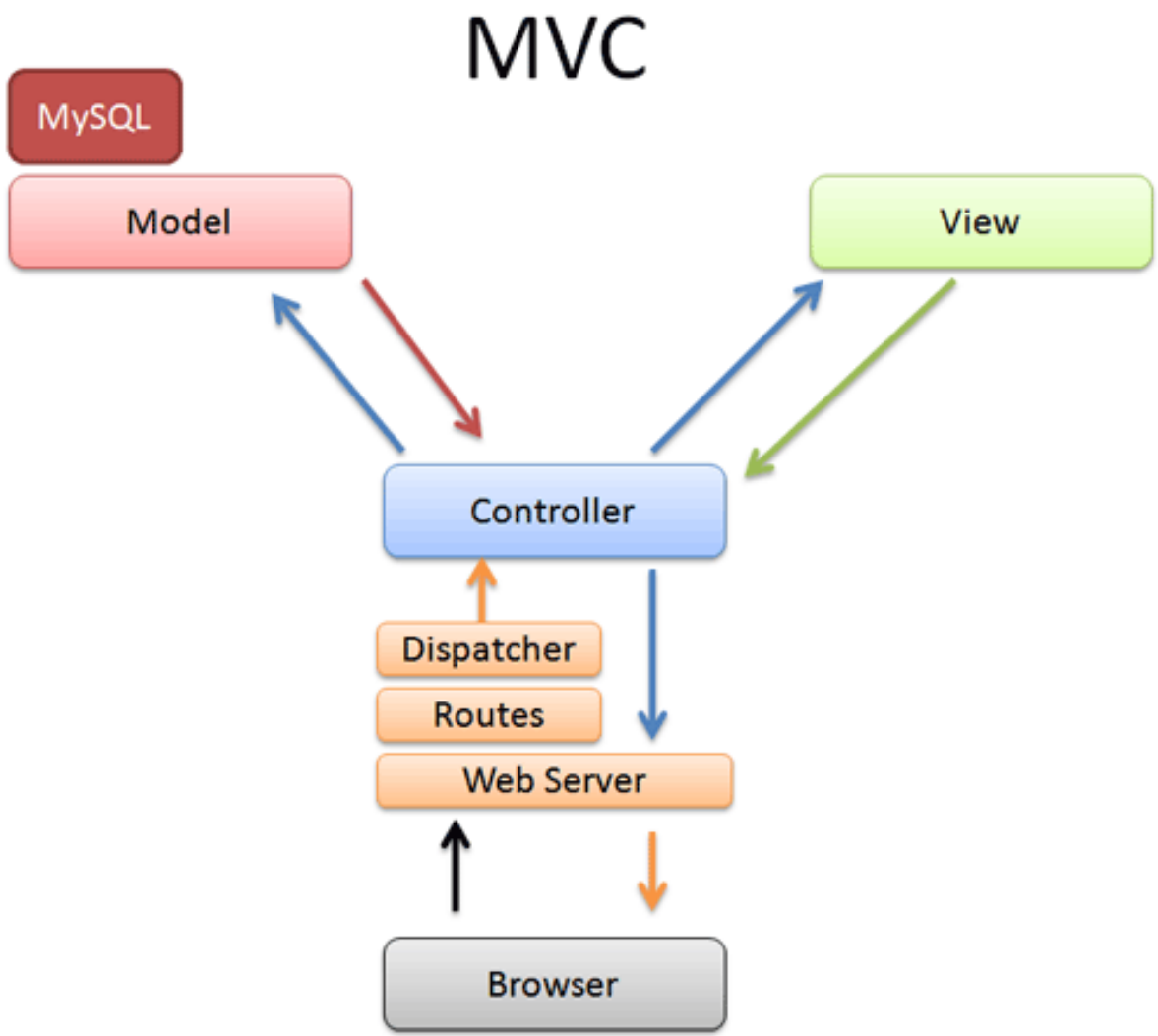
- Affichage Html essentiellement (mais pas que).
- Avec des parties utilisant des variables php, remplis par le contrôleur à l'aide du Modèle (détail d'un film, etc ...).
- On peut utiliser des templates.

Le contrôleur

Gére les requêtes des utilisateurs, retourne une réponse avec l'aide mutuelle des couches Modèle et Vue.

- Le "cerveau" de l'application
- En fonction des requêtes du client, il fournit la bonne réponse en piochant dans le modèle et la vue pour afficher le bon résultat à l'utilisateur.
- Autant de classes que nécessaires, regroupés en entités logiques (Films, Uilisateurs, etc ...)
- Chaque contrôleur posséde des actions : lister, insérer, supprimer, etc ...
- Chaque contrôleur a ses vues associées.

MVC WEB


MVC
MySQL
Model
View
Controller
Dispatcher
Routes
Web Server
Browser

Avantages

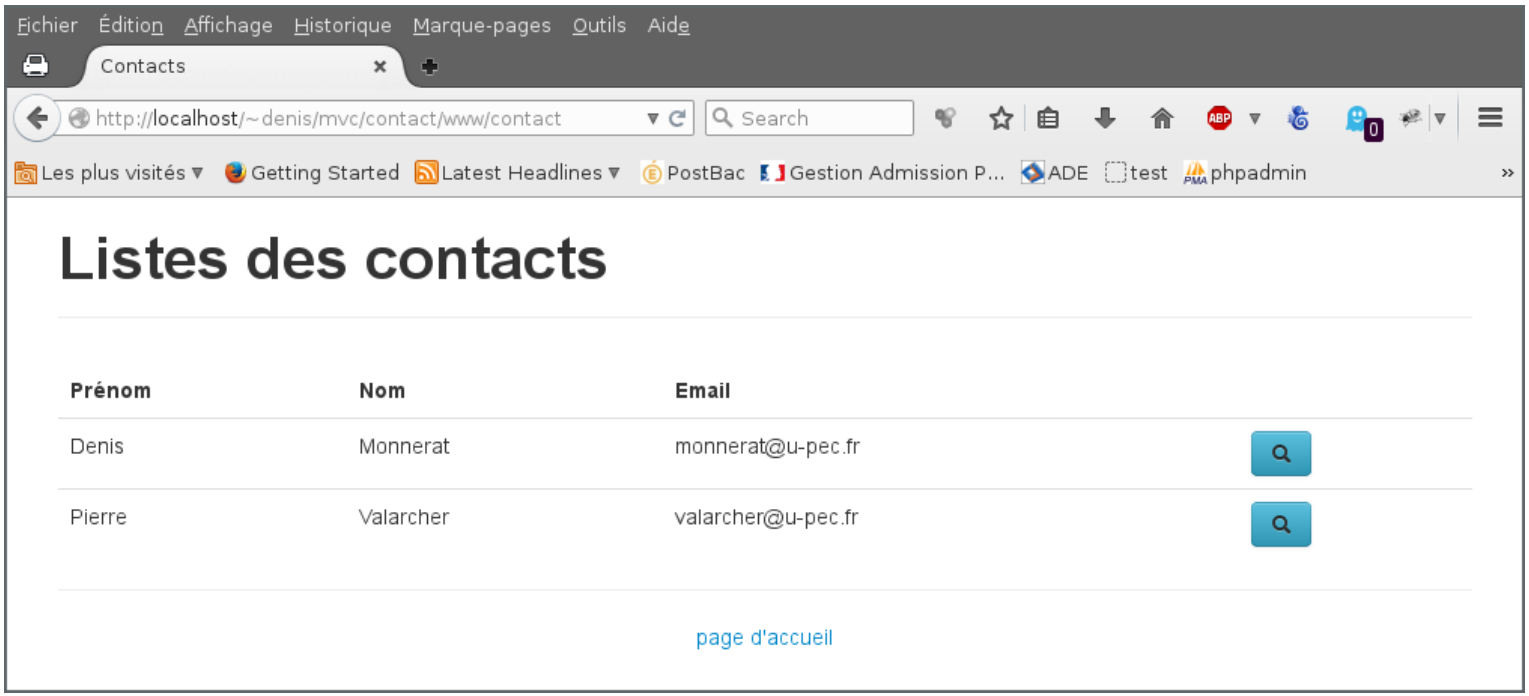
Modèle et Vue sont séparés

- ⇝ L'application est plus flexible, plus claire, facilement évolutive.
- ⇝ On peut les changer, remplacer, tester séparément.
- ⇝ Le "design" peut-être confié à un désigner.
- ⇝ Abstraction de l'application par rapport aux données.

Sommaire

Gestionnaire de contacts mail

Le but de concevoir une toute petite application web permettant de gérer (insertion, modification, édition, suppression) sa liste de contacts mails constitués d'un nom, prénom et d'une adresse mail.



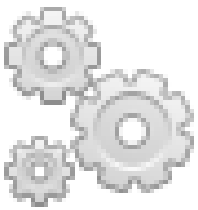
Les routes de notre site

On choisit un système d'url segmenté de la forme

monsite.com/class/function/ID

comme par exemple

monsite.com/contact/afficher/2



Cela impose de configurer le serveur WEB (apache) et d'utiliser l'URL rewriting, qui transformera l'url précédente en

monsite.com/index.php ?query=contact/afficher/2

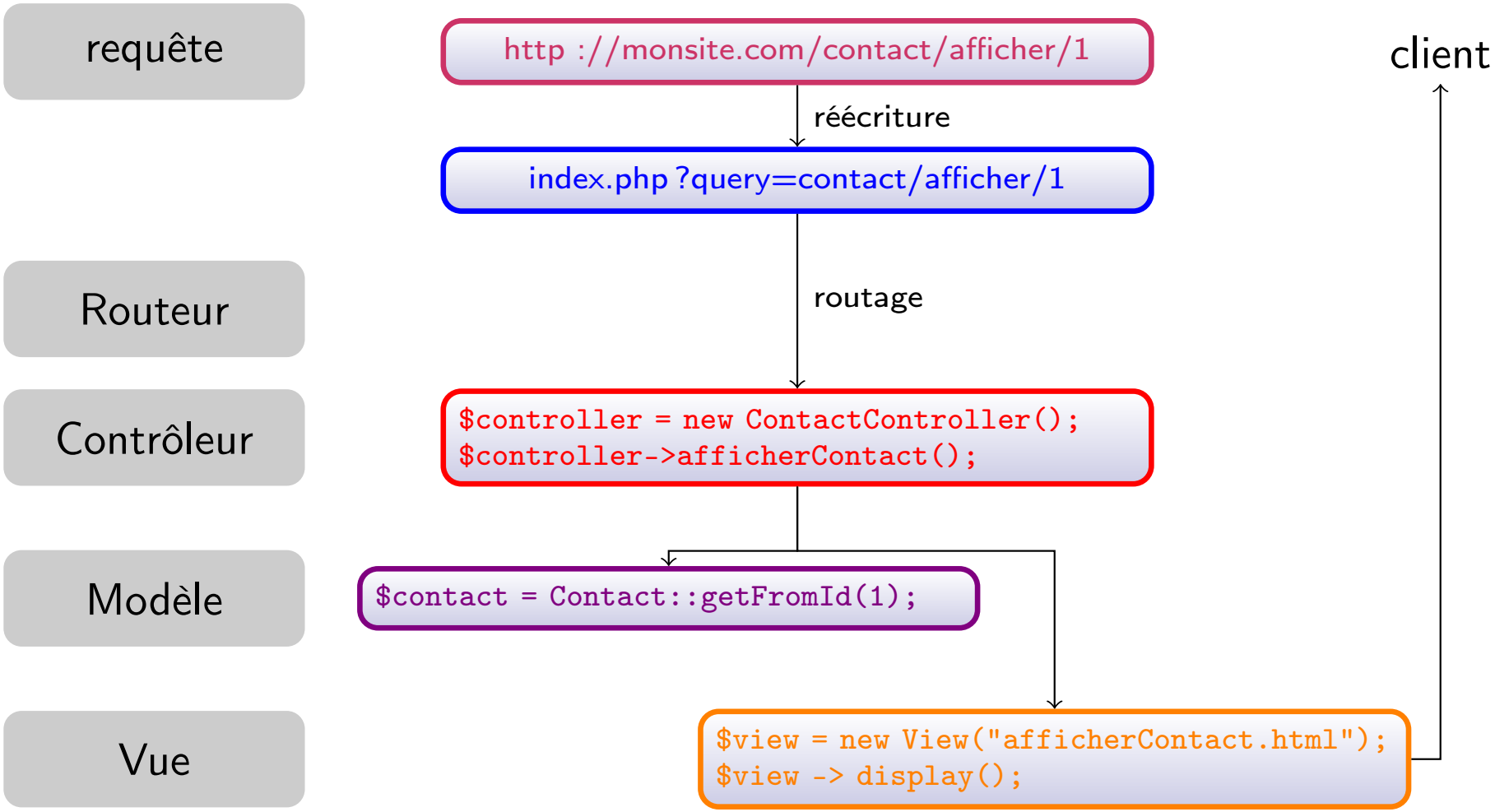
.htaccess pour apache

```
# Mise en place de la ré-écriture
Options +FollowSymLinks
RewriteEngine On

# Adresse de base de réécriture
RewriteBase /contact/

# Règles
RewriteCond %{SCRIPT_FILENAME} !-f
RewriteCond %{SCRIPT_FILENAME} !-d
RewriteRule (.*) index.php?query=$1 [QSA,L]
```

Une requête et son cheminement.



Point d'entrée : `index.php`.

Execution de `Kernel::run()`

- Analyse l'url (réécrite par apache).
- Initialise le bon contrôleur.
- Exécute la bonne méthode.

```php
class Kernel {
  public static function run() {
    // Autoload des classes
    spl_autoload_register('Kernel::autoload');
    // Analyser la requete
    $query = isset($_GET["query"]) ? $_GET["query"] : "";
    $route = Router::analyze( $query );
    // Instancier le controleur et
    // executer l'action
    $class = $route["controller"]."Controller";
    if(class_exists($class)) {
      $controller = new $class ($route);
      $method = array($controller, $route["action"]);
      if( is_callable( $method ))
        call_user_func($method);
    }
    // Gestion des erreurs
  }
}
```

Avec la méthode autoload qui permet d'inclure les fichiers en fonctions des besoins.

```
class Kernel {

  public static function autoload($class) {
    if(file_exists(ROOT."/app/kernel/$class.php"))
      require_once(ROOT."/app/kernel/$class.php");
    else if(file_exists(ROOT."/app/controller/$class.php"))
      require_once(ROOT."/app/controller/$class.php");
    else if(file_exists(ROOT."/app/model/$class.php"))
      require_once(ROOT."/app/model/$class.php");
  }
}
```

Analyse de la route

Assuré par `Router::analyze`

Un extrait pour /afficher/contact/id

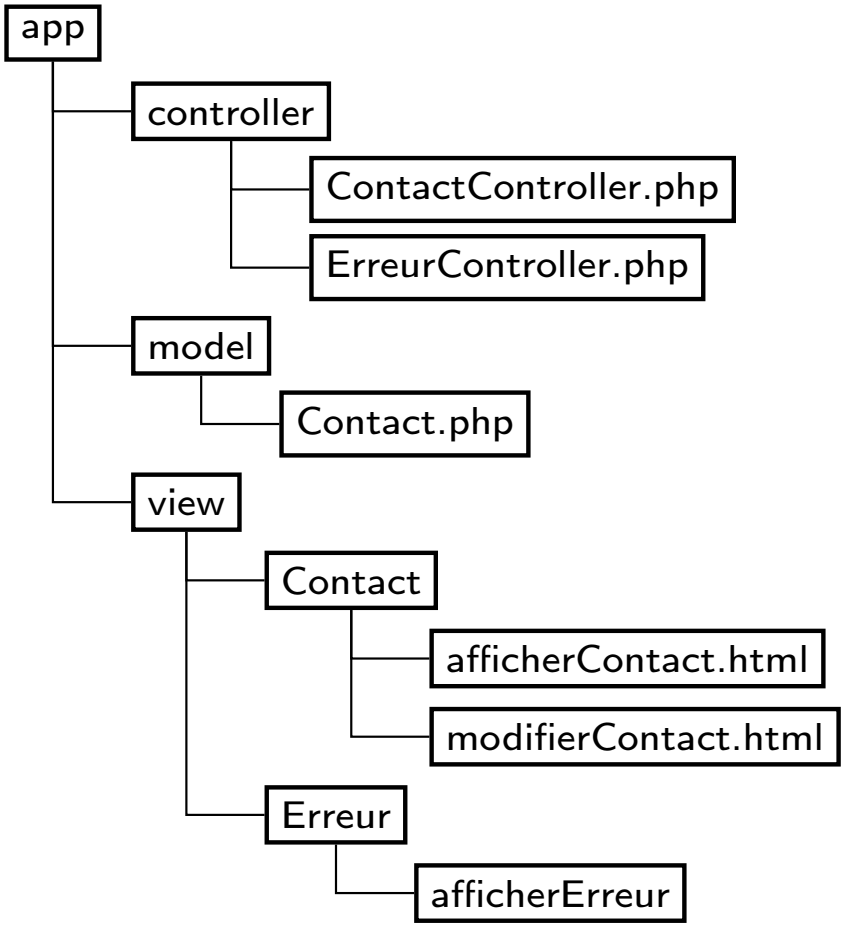
```
class Router {
  public static function analyze( $query ) {
    // un extrait
    $parts = explode("/", $query);

    if ((count($parts) == 3)
      && ($parts[1] == "afficher")
      && ($parts[0] == "contact")){

        $result["controller"] = "Contact";
        $result["action"] = "afficherContact";
        $result["params"]["id"] = $parts[2];
      }
    return $result;
  }
```

Strucutre des répertoires

En dehors de la racine du serveur (plus sûr).



Le modéle

On crée une classe pour les accés à la base de données ...

```
class Database {
   static protected $_instance = null;
   protected $_db;

   static public function getInstance() {
      if( is_null(self::$_instance) )
         self::$_instance = new Database();
      return self::$_instance;
   }
   public function query($sql){
     return $this->_db->query($sql);
   }
   public function prepare($sql){
     return $this->_db->prepare($sql);
   }

   protected function __construct() {
      $this->_db = new PDO(
         "mysql:host=localhost;dbname=mvc;charset=utf8",
         "root",
         "toto"
      );
   }
}
```

Le modéle

et les classes correspondantes aux entités.

```
class Contact extends Model {
  public $id, $nom, $prenom, $email;

  public static function getFromId( $id ) {
    $db = Database::getInstance();
    $sql = "SELECT * FROM contacts WHERE id = :id";
    $stmt = $db->prepare($sql);
    $stmt->setFetchMode(PDO::FETCH_CLASS, "Contact");
    $stmt->execute(array(":id" => $id));
    return $stmt->fetch();
  }

  public static function setFromId( $id,$data ) {
    $db = Database::getInstance();
    $sql = "UPDATE contacts set nom=:nom,prenom=:prenom,email=:email WHERE id = :id";
    $stmt = $db->prepare($sql);
    return $stmt->execute(array(
      ":id" => $id,
      ":nom"=>$data['nom'],
      ":prenom"=>$data['prenom'],
      ":email"=>$data['email']));
  }

  public static function getList() {
```

Le contrôleur

```
class ContactController extends Controller {
  public function afficherContact() {
    $id = $this->route["params"]["id"];
    $this->view->contact = Contact::getFromId($id);
    $this->view->display();
  }
  public function modifierContact()
  {
  $id = $this->route["params"]["id"];
  $data = $this->route["params"]["post"];
  if (count($data) != 0) {
    $this->view->success  =  Contact::setFromId($id,$data) ;
  }
  $this->view->contact = Contact::getFromId($id);
  $this->view->display();
  }


}
```

La vue

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8"/>
   <title>Contacts</title>
</head>
<body>
   <header>
      <h1>Contact</h1>
   </header>
   <h2><?php echo ($this->contact->prenom)." ".($this->contact->nom);?></h2>
   <h3>Email : <?php echo $this->contact->email; ?></p>
   <footer>
     <p><a href="">retour à la page d'acceuil</a></p>
   </footer>
</body>
</html>
```