5XPOO-1

Framework et POO côté serveur

Benjamin Delbar

Benjamin Delbar

Cours 2

Récap

Exercice

Récapitulatif

MVC en Orienté objet

Model : Représentation des données (objets métiers)

Contrôleur : Reçoit les requêtes et les traitent

View : Vues affichée au client

Récapitulatif

Requête client

Routeur

Contrôleur

DAO

Database

Objet métier

Récapitulatif

Requête client

GET / POST / ...

Par ex : monsite.com/articles/5?lang=fr&color=red

monsite.com => domaine

/articles/5 => url

lang=fr; color=red => paramètres GET

Récapitulatif

Arrive sur le serveur

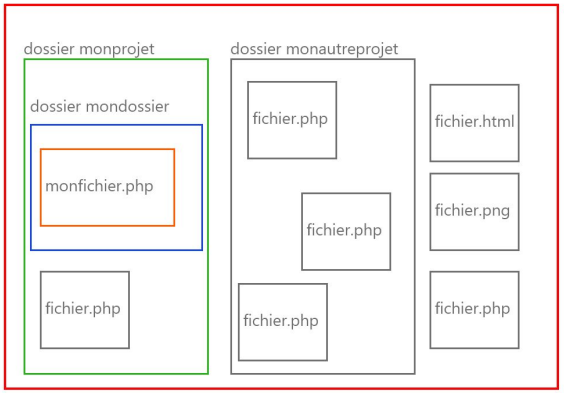
1) Réécriture d’url
2) Activation du routeur
3) Envoi de la requête au contrôleur voulu
4) Traitement par le contrôleur

Récapitulatif

CLIENT

localhost/monprojet/mondossier/monfichier.php

SERVEUR



Récapitulatif

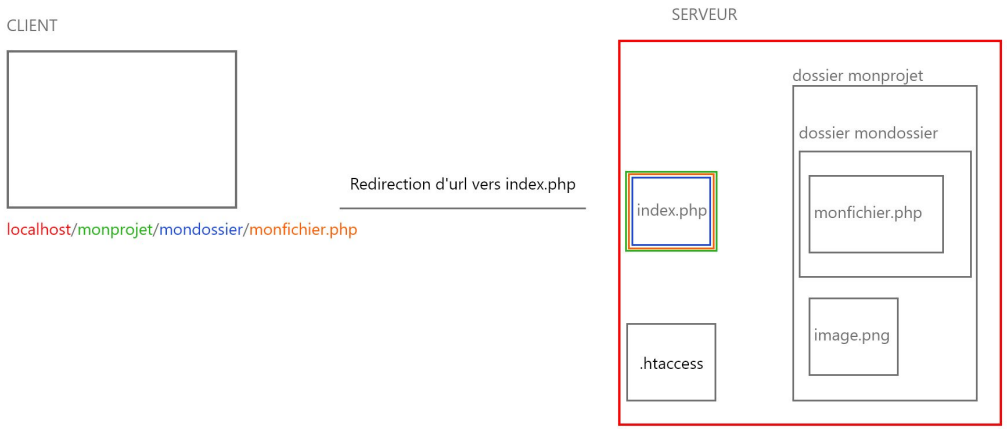
.htaccess

On y écrit les règles de réécriture

Tout rediriger vers index.php

```
RewriteEngine On
RewriteRule ^ index.php [L]
```

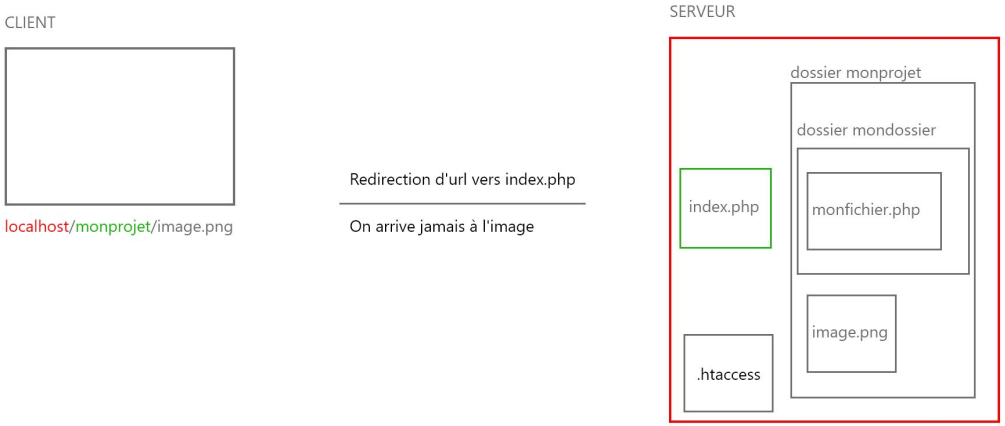
Récapitulatif



Récapitulatif

Et si on veut envoyer une image / un fichier css ?

Récapitulatif

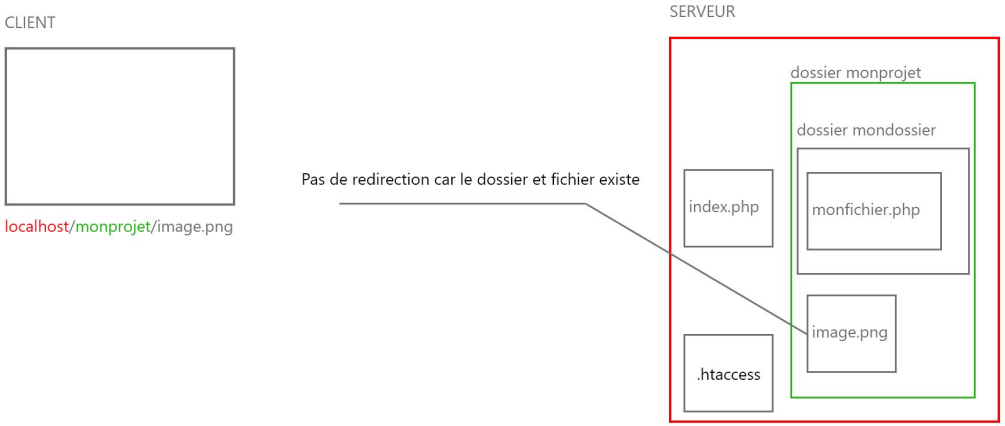


Récapitulatif

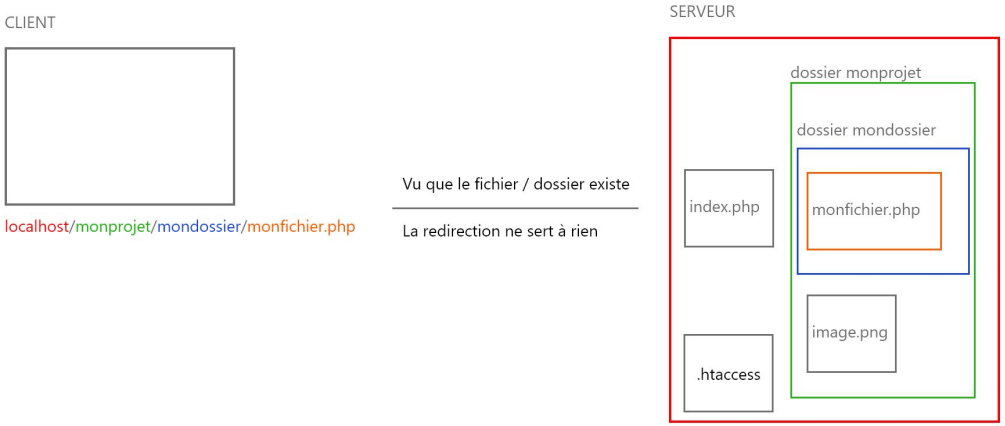
Rediriger vers index SAUF si le fichier / dossier existe!

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
```

Récapitulatif



Récapitulatif



Récapitulatif

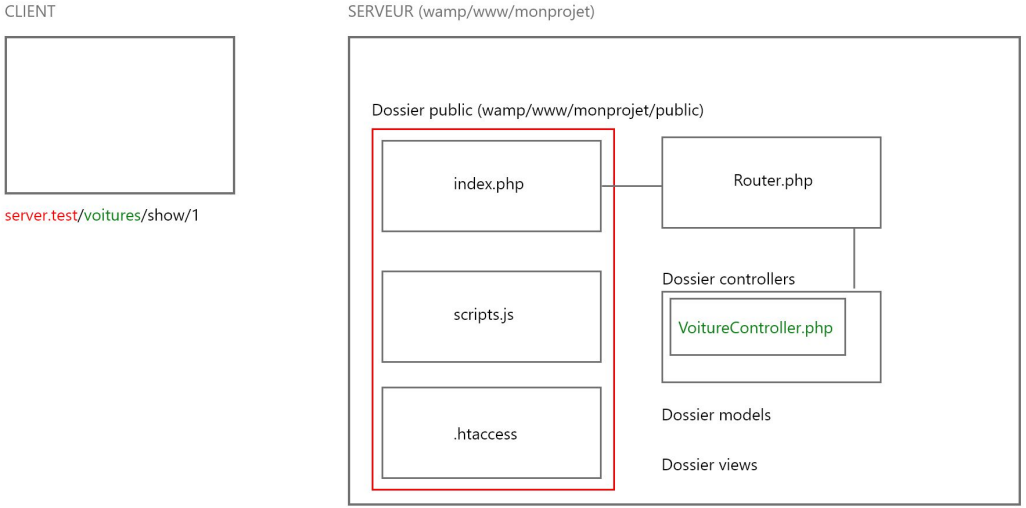
Et si on isolait les fichiers de code des fichiers en "accès libre" ?

C'est ce qui se fait naturellement avec les hébergements web
On définit que la racine du site est un dossier du serveur et non la racine du serveur

En local, on peut créer des virtual hosts

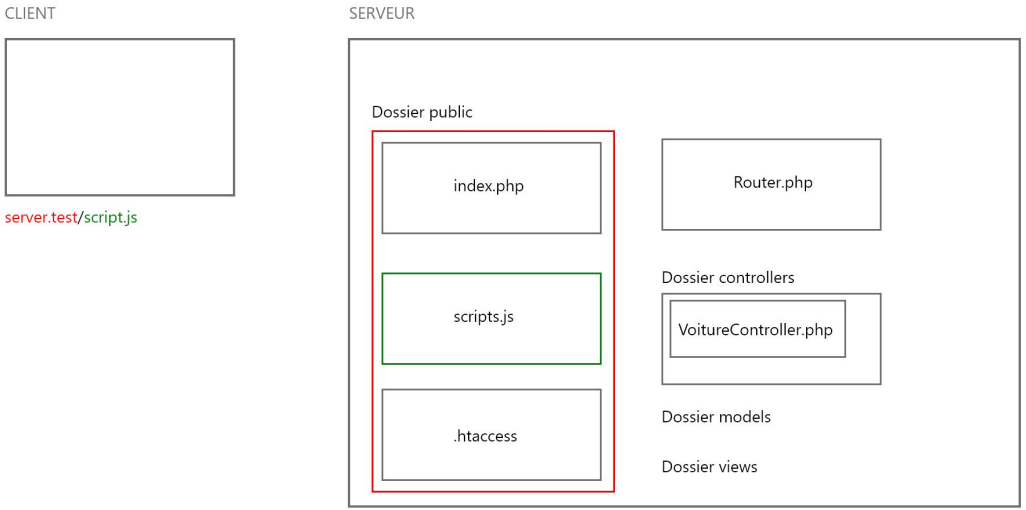
wamp/www/monprojet/public => server.test

Récapitulatif



Ce qui est en dehors du dossier public est inaccessible par le client directement

Récapitulatif



Ce qui est en dehors du dossier public est inaccessible par le client directement

Récapitulatif

Notre HTACCESS

```
RewriteEngine On

# Send Requests To Index.php
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [L]
```

Récapitulatif

Contrôleur

Récupère la requête du routeur => détecte l’action à effectuer

si nécessaire : appelle la couche d’accès aux données pour récupérer les données

Récapitulatif

Exemple : **/voitures/show/1**

Modele : Voitures

Action : show (montrer)

Paramètres : 1 (id)

=> Le contrôleur comprend qu'il doit montrer la voiture à l'ID 1

Récapitulatif

=> Le contrôleur comprend qu'il doit montrer la voiture à l'ID 1

Il va devoir

- Aller demander à la couche d'accès un objet "Voiture" qui comprend les données de la voiture à l'id 1
- Insérer les données dans la vue correspondante
- Renvoyez la vue au client

Récapitulatif

La couche d’accès aux données

Appelée **DAL** (Data Access Layer)

Contient des **DAO** (Data Access Object)

Objets servant à faire le pont entre les objets métiers (Voiture / Animal / Personne) et la Database

Récapitulatif

Un **DAO** (Data Access Object) s'occupe des opérations de **CRUD**

CRUD => Create Read Update & Delete

Récapitulatif

Exemple

/voitures/show/1

Le contrôleur instancie un VoitureDAO
$voitureDAO = new VoitureDAO();

Le contrôleur demande au VoitureDAO un objet Voiture comprenant les données de la voiture à l'ID 1
$voiture = $voitureDAO->fetch(1);

Le DAO se connecte à la database, récupère la voiture correspondante et instancie un objet voiture avec les données récupérées

Récapitulatif

PARTIE 1

1) Réaliser une DB comprenant une table voiture avec un numéro de châssis, une couleur et sa puissance en ch
2) Réaliser l'objet métier Voiture
3) Réaliser le DAO VoitureDAO permettant de récupérer (lire) une voiture de la base de données (*fetch*)

Récapitulatif

Objet métier voiture

- Attributs compris dans notre base de données
- Getters & Setters

Récapitulatif

Getters & Setters “magique”

__get & __set

Récapitulatif

Se connecter à la database dans notre DAO

```
public function __construct () {
        $this->table = “voitures”;
        $this->connection = new PDO('mysql:host=localhost;dbname=maDatabase;charset=utf8', 'root', '');
        $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
```

Récapitulatif

Récupérer une entrée de la base de données

```
public function fetch($id) {
        //Créer un statement
        //Exécuter le statement
        //Vérifier le contenu récupéré
}
```

Récapitulatif

Récupérer une entrée de la base de données

```
public function fetch ($id) {
        try {
                $statement = $this->connection->prepare("SELECT * FROM {$this->table} WHERE id = ?");
                $statement->execute([$id]);
                $result = $statement->fetch(PDO::FETCH_ASSOC);
                var_dump($result);
        } catch (PDOException $e) {
                var_dump($e);
        }
}
```

Récapitulatif

Créer un objet Voiture depuis les données brutes récupérées

```
public function create ($data) {
        return new Voiture(
                $data['id'],
                $data['serial'],
                $data['color'],
                $data['hp']
        );
}
```

Exercice

- Récupérer l'ensemble des voitures présentes en base de données fetchAll()
- Créer une voiture en base de données
- Supprimer une voiture en passant son ID => **delete($id)**