

# Architecture MVC en PHP - Contrôleur principal et routage

Partie 1 : généralités

Partie 2 : contrôleur

Partie 3 : vue

Partie 4 : modèle

Partie 5 : contrôleur principal et routage

## Fonctionnement du contrôleur principal dans le patron de conception MVC

### Rappel du contexte

R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

L'objectif de ce document est d'analyser le fonctionnement du contrôleur principal dans l'architecture MVC puis d'intégrer de nouveaux modules fournis.

### Ressources à utiliser

- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

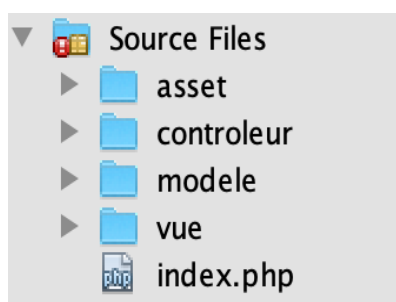
### Ressources à utiliser téléchargeable

- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

### Préparations *[ ne pas recréer / importer une base de données si elle a été créée précédemment ]*

Après avoir créé la base de données nommée « resto » (encodage utf8mb4) et importé le fichier base.sql sur PhpMyAdmin, créer un nouveau dossier nommé **MVCTP5** à la racine de votre serveur web. Copier le contenu du dossier « site » dans ce dossier. Importer le dossier **MVCTP5** dans Visual Studio Code.

L'arborescence devrait être semblable celle-ci :



Avant de commencer le TP, le site doit être paramétré afin qu'il utilise votre base de données. Dans le script modele/bd.inc.php, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";  
$mdp = "votre mot de passe mariaDB";  
$bd = "nom de votre base de données";  
$serveur = "localhost"; //notre SGBRD est exécute en local
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à [cette adresse](#).

## A - Analyse du fonctionnement du contrôleur principal

### Question 1 - Analyse de l'existant

#### Documents à utiliser

- fichiers fournis en ressources
- annexes 1, 2 et 3

Rappel : transmission de paramètres en méthode GET

Lorsque des paramètres sont transmis en méthode GET, ils sont visibles dans la barre d'URL. Les paramètres sont spécifiés en fin d'URL après le point d'interrogation.

`http://serveur/dossier/script.php?prenom=jean&age=20`

Dans l'exemple ci-dessus, les paramètres transmis sont : `prenom` et `age`.

`prenom` a pour valeur `jean`

`age` a pour valeur `20`

**1.1.** Relever les URL en navigant successivement sur les écrans de connexion, de recherche et d'accueil. indiquer quelle partie de l'URL change, en spécifiant le nom de paramètre et la valeur envoyés en méthode GET.

Écran	URL	Paramètre	Valeur
connexion			
recherche			
accueil			

Le fichier `index.php` est utilisé par défaut par le serveur web Apache lorsqu'une URL pointant sur un répertoire est appelé par le navigateur. C'est donc lui qui a reçu les paramètres envoyés en méthode GET dans la question 1.1.

**1.2.** A l'aide de l'annexe 1 et de la réponse à la question précédente, indiquer les valeurs pouvant être prises par la variable `$action`.

La fonction `redirigeVers()` visible en annexe 2 est appelée par le contrôleur principal de l'application nommé `index.php`. Cette fonction a un rôle de routage (d'aiguillage). Elle donne accès aux différents contrôleurs de l'application.

**1.3.** Schématiser la structure de la variable `$lesActions` créée dans la fonction `redirigeVers()`.

**1.4.** Quelle partie de cette structure est similaire aux données trouvées en question 1.1 ?

Lorsque l'on accède à la variable `$lesActions`, on peut à partir d'une action obtenir un nom de fichier. Par exemple l'instruction suivante place la chaîne `"listeRestos.php"` dans la variable `$fichier`.

```
$fichier = $lesActions['liste'];
```

**1.5.** À partir de vos réponses aux questions précédentes, indiquer quelles valeurs peuvent être transmises à la fonction `redirigeVers()` ?

**1.6.** Pour chacune des valeurs transmises à la fonction indiquer quelle sera la valeur retournée.

**1.7.** Que se passe-t-il si l'action transmise à la fonction `redirigeVers()` n'existe pas dans la variable `$lesActions` ? Quelle valeur est retournée ?

Lorsque la fonction `redirigeVers()` est appelée dans le fichier `index.php`, elle retourne le nom d'un fichier contrôleur à inclure. Les deux lignes suivantes issues du script `index.php` prennent en charge la sélection du contrôleur demandé, et son chargement.

```
$fichier = redirigeVers($action);  
include RACINE . "/contrôleur/$fichier";
```

**1.8.** Après avoir consulté l'annexe 3, expliquer à quoi sert la condition utilisant l'instruction `array_key_exists()` dans la fonction `redirigeVers()`.

**1.9.** Quel script contrôleur est appelé par `index.php` lorsque la variable GET `action` n'est pas renseignée ?

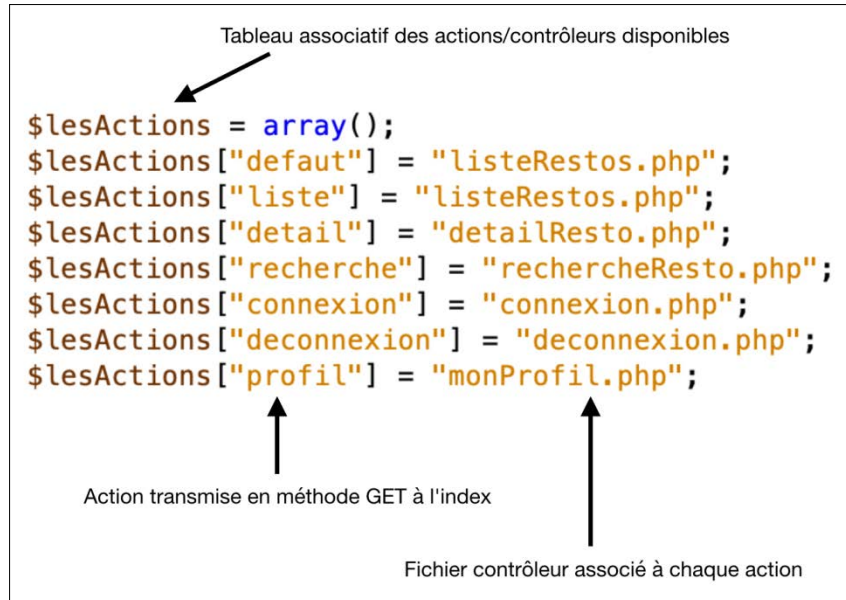
**1.10.** Quel script contrôleur est appelé par `index.php` lorsque la variable GET `action` contient le mot clé `"liste"`.

## Synthèse

La variable `$lesActions` de la fonction `redirigeVers()` contient l'ensemble des contrôleurs accessibles sur le site. Un contrôleur est l'élément central d'une fonctionnalité.

Lorsqu'une nouvelle fonctionnalité est développée sur l'application, il faut l'intégrer au contrôleur principal en ajoutant une nouvelle valeur dans la variable `$lesActions`.

Chaque clé dans la variable `$lesActions` est unique et correspond à un script contrôleur.



*Structure de la variable `$lesActions`*

Lorsqu'on ajoute une nouvelle fonctionnalité dans l'application, il faut créer le script contrôleur dans le dossier du même nom, puis déclarer cette nouvelle fonctionnalité dans la fonction `redirigeVers()` en ajoutant l'action correspondante (clé) et le fichier associé.

L'instruction ci-dessous permet d'ajouter une nouvelle action associée à un contrôleur dans la variable `$lesActions`.

```
$lesActions['uneAction'] = "scriptControleur.php" ;
```

### Accéder à une fonctionnalité sur l'application web

Le fichier `index.php` est la porte d'entrée pour l'accès aux différentes fonctionnalités. C'est lui qui réceptionne l'action envoyée en méthode GET puis charge le contrôleur associé.

Dès lors, la navigation sur l'application web ne fait apparaître que le fichier `index.php` dans l'URL. Chaque fonctionnalité est demandée par l'intermédiaire de la variable action transmise.

Lorsqu'on souhaite accéder à une fonctionnalité par l'intermédiaire d'un lien, l'URL pointée doit faire référence à l'action souhaitée.



*Accès au contrôleur de connexion*

```

<ul id="menuGeneral">
  <li><a href=".."?action=accueil">Accueil</a></li>
  <li><a href=".."?action=recherche">Recherche</a></li>
  <li></li>

  <li id="logo"><a href=".."?action=accueil"></a></li>
  <li></li>
  <li><a href=".."?action=ccgu">CCGU</a></li>
  <?php if(isLoggedIn()){ ?>
  <li><a href=".."?action=profil">Mon Profil</a></li>
  <?php }
  else{ ?>
  <li><a href=".."?action=connexion">Connexion</a></li>
  <?php } ?>
</ul>

```

Action transmise au fichier index.php

*Le nom du fichier contrôleur est masqué à l'utilisateur qui ne voit que l'action.*

Dans l'exemple ci dessus, le nom du fichier index.php n'est pas visible, on peut soit l'indiquer, soit faire référence à l'emplacement "/" qui désigne le fichier index.php dans la configuration du serveur web Apache .

Les contrôleurs ne sont jamais directement appelés dans l'URL, on doit toujours passer par le contrôleur principal, et donc par index.php. Il en est de même avec les formulaires, comme celui de connexion.

Action transmise au fichier index.php en méthode GET (visible dans l'URL)

```

<h1>Connexion</h1>
<form action=".."?action=connexion" method="POST">
  <input type="text" name="mailU" placeholder="Email de connexion" /><br />
  <input type="password" name="mdpU" placeholder="Mot de passe" /><br />
  <input type="submit" />
</form>

```

Données transmises en POST au fichier index.php puis au contrôleur connexion.php inclus

*Extrait de la vue affichant le formulaire de connexion*

On remarque que les données du formulaire sont transmises en méthode POST à index.php. Lors de l'inclusion du fichier contrôleur (connexion.php), ces données seront aussi accessibles dans le contrôleur.

Pour intégrer une nouvelle fonctionnalité au site web, il faut donc :

- créer un nouveau contrôleur dans le dossier approprié,
- créer la vue et le modèle associé si besoin,
- ajouter une action dans la variable \$lesActions associant le nom de l'action au nom du script contrôleur,
- donner l'accès à ce contrôleur par l'intermédiaire d'un lien ou d'un formulaire transmettant l'action en méthode GET.

## **B - Intégration de contrôleurs pré-existants**

### **Exercice 1 - CGU**

#### **Documents à utiliser**

- fichiers fournis en ressources
- annexe 4

Les conditions générales d'utilisations ont déjà été écrites. La vue et le contrôleur associés sont disponibles. De même, le menu général propose un lien vers celles ci.

Lorsque l'on clique sur ce lien les CGU ne sont pas affichées, le contrôleur par défaut est chargé à la place.

- 2.1. Repérer dans le menu général l'action correspondant au contrôleur ayant en charge l'affichage des CGU
- 2.2. Placer les fichiers fournis en ressource dans les dossiers appropriés.
- 2.3. Rédiger l'instruction à ajouter à la fonction `redirigeVers()` pour ajouter la nouvelle action dans la variable `$lesActions`.
- 2.4. Ajouter la nouvelle action à la fonction puis tester le bon fonctionnement du lien CGU dans le menu général.

## Exercice 2 - aimer un restaurant

### Documents à utiliser

- fichiers fournis en ressources
- Annexe 5

Lorsqu'un utilisateur connecté consulte la fiche descriptive d'un restaurant, il a la possibilité d'ajouter celui-ci dans la liste des restaurants qu'il aime. Cette action se fait en cliquant sur l'étoile située à droite du nom du restaurant. Tant que l'utilisateur n'a pas aimé ce restaurant, l'étoile est grisée.



*Lien sous forme d'une étoile pour aimer un restaurant*

Lorsqu'on clique sur ce lien, le traitement n'est pas effectué, rien n'est enregistré dans la table `aimer`. Le contrôleur par défaut est chargé à la place.

- 3.1. Quelle vue permet d'afficher la fiche descriptive d'un restaurant ?
  - 3.2. Repérer dans le code de la vue le lien correspondant à l'étoile.
  - 3.3. Quels sont les paramètres envoyés en méthode GET lorsque l'on clique sur le lien ? Préciser le nom et la valeur de chaque paramètre.
  - 3.4. À partir de vos connaissances et du script contrôleur fourni en ressource, déterminer dans quels scripts sont utilisées chacune des deux variables transmises par le lien en méthode GET (celles trouvées à la question précédente).
  - 3.5. Placer le fichier fourni en ressource dans le dossier approprié.
  - 3.6. Rédiger l'instruction à ajouter à la fonction `redirigeVers()` pour ajouter la nouvelle action dans la variable `$lesActions`.
  - 3.7. Ajouter la nouvelle action à la fonction puis tester le bon fonctionnement du contrôleur en cliquant sur l'étoile. (Pour tester, il faut être authentifié sur le site)
- Lorsqu'on clique sur l'étoile, on observe que la page affichée reste la même. La page chargée affiche aussi la même URL, pourtant le lien pointé est bien différent.
- 3.8. Le contrôleur `aimer.php` fait-il appel à une vue ?

En fin de fichier il est fait mention en commentaire du `referer`. La dernière instruction du contrôleur mentionne aussi ce terme :

```
header('Location: ' . $_SERVER['HTTP_REFERER']);
```

**3.9.** Rechercher sur le web la signification du terme « referer ».

Mettre en commentaire l'instruction `header()` et afficher à la place la valeur de la variable `$_SERVER['HTTP_REFERER']`.

Tester de nouveau l'action d'aimer sur un restaurant.

**3.10.** Que contient la variable `$_SERVER['HTTP_REFERER']` ?

**3.11.** Dédurre de vos observations le rôle de l'instruction ci-dessous :

```
header('Location: ' . $_SERVER['HTTP_REFERER']);
```

## Exercice 3 - inscription

Le formulaire d'inscription est accessible par l'intermédiaire d'un lien dans la section "connexion" du site.

**4.1** Analyser ce lien, puis à l'aide des fichiers en ressource apporter les modifications nécessaires pour rendre l'inscription fonctionnelle.

### Remarques :

- le modèle, la vue et le contrôleur sont déjà prêts. Ils sont soit fournis en ressource, soit déjà en place dans le code existant ;
- reprendre la même logique que pour les exercices précédents.

## Annexe 1 - index.php

```
<?php
/**
 *    Contrôleur principal
 */

require dirname(__FILE__) . "/contrôleur/config.php";

require RACINE . "/contrôleur/routage.php";
require_once RACINE . "/modele/authentication.inc.php";
if (isset($_GET["action"])) {
    $action = $_GET["action"];
}

//Ajoute un contrôleur secondaire ($fichier) en fonction du métier ($action) :
$fichier = redirigeVers($action);
require RACINE . "/contrôleur/" . $fichier;
?>
```

## Annexe 2 - routage.php

```
<?php
/**
 *    Module du routing (routage).
 *    Chaque action est récupérée par la méthode : $_GET
 */

function redirigeVers($action="default") {

    $lesActions = array();
    $lesActions["default"] = "listeRestos.php";
    $lesActions["liste"] = "listeRestos.php";
    $lesActions["detail"] = "detailResto.php";
    $lesActions["recherche"] = "rechercheResto.php";
    $lesActions["connexion"] = "connexion.php";
    $lesActions["deconnexion"] = "deconnexion.php";
    $lesActions["profil"] = "monProfil.php";

    $contrôler_id = $lesActions[$action];

    //si le fichier n'existe pas :
    if(! file_exists(__DIR__ . '/' . $contrôler_id) ) die("Le fichier : " .
    $contrôler_id . " n'existe pas !");

    //si la clé "action" existe dans notre tableau "lesActions" :
    if (array_key_exists($action, $lesActions)) {
        // le fichier à inclure sera retourné :
        return $contrôler_id;
    } else {
        return $lesActions["default"];
    }
}
?>
```

## Annexe 3 - extrait de la documentation PHP



## array\_key\_exists

(PHP 4 >= 4.0.7, PHP 5, PHP 7, PHP 8)

array\_key\_exists — Vérifie si une clé existe dans un tableau

### Description

array\_key\_exists ( mixed \$key , array \$array ) : bool

array\_key\_exists() retourne TRUE s'il existe une clé du nom de key dans le tableau array. key peut être n'importe quelle valeur valide d'index de tableau.

### Liste de paramètres

key

Valeur à vérifier.

array

Un tableau contenant les clés à vérifier.

### Valeurs de retour

Cette fonction retourne TRUE en cas de succès ou FALSE si une erreur survient.

## Annexe 4 - extrait du fichier entete.html.php

```
<ul id="menuGeneral">
  <li><a href="./?action=accueil">Accueil</a></li>
  <li><a href="./?action=recherche">Recherche</a></li>
  <li></li>

  <li id="logo"><a href="./?action=accueil"></a></li>
  <li></li>
  <li><a href="./?action=cgu">CGU</a></li>
  <?php if(isLoggedIn()){ ?>
  <li><a href="./?action=profil">Mon Profil</a></li>
  <?php }
  else{ ?>
  <li><a href="./?action=connexion">Connexion</a></li>
  <?php } ?>

</ul>
```

## Annexe 5 - contrôleur aimer.php

```
<?php

/**
 *      Controleur secondaire : aimer
 */

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : '.basename(__FILE__));
}

include_once RACINE . "/modele/bd.aimer.inc.php";
```

```
// recuperation des donnees GET, POST, et SESSION
$idR = $_GET["idR"];

// appel des fonctions permettant de recuperer les donnees utiles a l'affichage

$mailU = getMailULoggedOn();
if ($mailU != "") {
    $aimer = getAimerById($mailU, $idR);

// traitement si necessaire des donnees recuperees
    ;
    if ($aimer == false) {
        addAimer($mailU, $idR);
    } else {
        delAimer($mailU, $idR);
    }
}

// redirection vers le referer
header('Location: ' . $_SERVER['HTTP_REFERER']);
?>
```