

Architecture MVC en PHP - Vue

Partie 1 : généralités

Partie 2 : contrôleur

Partie 3 : vue

Partie 4 : modèle

Partie 5 : contrôleur principal et routage

Fonctionnement de la vue 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 composant vue dans l'architecture MVC puis de mettre à jour des vues existantes pour les améliorer.

Chaque restaurant est associé à des données dans la base : des photos, des types de cuisine, etc. L'affichage des données associées sur la fiche d'un restaurant donne plus d'informations aux utilisateurs. Ils peuvent ainsi choisir le bon restaurant où dîner.

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.

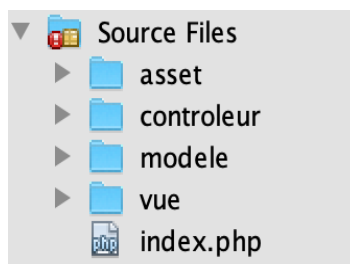
Ressources à utiliser téléchargeable

- 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 **MVCTP3** à la racine de votre serveur web. Copier le contenu du dossier « **site** » dans ce dossier. Importer le dossier **MVCTP3** 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"; //le SGBRD est installé sur le serveur
```

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](#).

Partie A : Analyse d'une vue existante

Dans les trois exercices suivants vous devrez analyser le script de vue `vueDetailResto.php` permettant d'afficher les détails d'un restaurant. Cette vue est appelée par le contrôleur `detailResto.php`.

Question 1 - Analyse du script de vue `vueDetailResto.php` : les photos

Documents à utiliser

- fichiers fournis en ressources
- annexes 1, 2, 5 et 6

1.1. Quelles fonctions définies dans le modèle sont utilisées dans le contrôleur `detailResto.php`.

1.2. Quelles annexes présentent l'affichage du résultat d'exécution des fonctions du modèle trouvées à la question précédente ?

Rappel sur les tableaux associatifs

Un tableau associatif permet de stocker plusieurs informations dans une seule variable. L'accès aux informations ne se fait pas grâce aux numéros de cellules comme dans un tableau classique, mais avec un nom de cellule.

Exemple :

Tableau associatif : `$tabEtudiant`

prenom	"Lionel"
nom	"Romain"
age	48

À la place d'indices, le tableau associatif fonctionne à l'aide de clés qui sont des chaînes de caractères.

La case nommée '*prenom*' contient la chaîne "lionel".

La case nommée '*nom*' contient la chaîne "Romain".

La case nommée '*age*' contient l'entier 48.

Les cellules d'un tableau associatif peuvent contenir des données de types différents.

L'affectation et l'accès à une cellule du tableau associatif fonctionnent de la même manière que pour un tableau classique : `$tabEtudiant['prenom'] = "Patrice" ;` // permet d'affecter la chaîne "patrice" à valeur qui se réfère à la clé '*prenom*'.

`echo $tabEtudiant['prenom'] ;` // permet d'afficher la valeur de la clé '*prenom*'.

Après exécution de la fonction `getPhotosByIdR()` dans le contrôleur `detailResto.php`, la structure de la variable `$lesPhotos` peut être schématisée sous cette forme :

0	idP	6
	cheminP	cidrerieDuFronton.jpg
	idR	4
1	idP	14
	cheminP	cidrerieDuFronton2.jpg
	idR	4
2	idP	15
	cheminP	cidrerieDuFronton3.jpg
	idR	4

- 1.3. Comment est composée chacune des 3 cellules du tableau \$lesPhotos ?
- 1.4. Repérer dans l'annexe 1 l'instruction PHP permettant d'afficher le nom du fichier de la première photo.
- 1.5. Quel est le rôle de l'instruction `if(count($lesPhotos) > 0)` ? Quel est son rôle dans le code de la vue ?
- 1.6. Quelle est la syntaxe générale permettant d'accéder à un champ (idP, CheminP ou idR) contenu dans la variable \$lesPhotos ?

Question 2 - vueDetailResto.php : les types de cuisine

Documents à utiliser

- fichiers fournis en ressources
- annexes 1, 2, 7 et 8

- 2.1. Quel contrôleur produit la variable \$lesTypesCuisine ? Préciser le nom de la fonction et du modèle.
- 2.2. Schématiser le contenu de la variable \$lesTypesCuisine
- 2.3. Combien de types de cuisine sont contenus dans cette variable ?
- 2.4. Quelle est la syntaxe permettant d'accéder au libelle d'un type de cuisine contenu dans la variable ?

Question 3 : vueDetailResto.php - le restaurant

Documents à utiliser

- fichiers fournis en ressources
- annexes 1, 2, 3 et 4

- 3.1. Quel contrôleur produit la variable \$unResto ? Préciser le nom de la fonction et du modèle.
- 3.2. Schématiser le contenu de la variable \$unResto
- 3.3. Quelles sections de code dans la vue utilisent cette variable ?
- 3.4. Combien de restaurants sont contenus dans cette variable ?
- 3.5. Quelle est la syntaxe permettant d'accéder au nom d'un restaurant contenu dans la variable ?

Synthèse

La vue utilise des données créées ou récupérées par le contrôleur. Certaines fonctions du modèles sont parfois appelées directement dans la vue. Ces données sont utilisées de manière brute, sans traitement. Si un traitement est à faire, il doit être déporté dans le contrôleur.

Un code propre devrait toujours récupérer les données dans le contrôleur, et parfois construire des variables complexes. Faire appel à des fonctions du modèle dans les vue est une manière de simplifier le travail, mais si trop de code est présent dans la vue, il est probable que sa conception est à revoir.

Le seul code PHP qui devrait être présent dans la vue devrait être directement lié à la vue. Ainsi on ne doit pas trouver de code permettant de gérer la logique applicative, ni d'accès direct aux données de la base.

Le choix des langages HTML et PHP dans la vue n'est qu'un exemple. Il est possible d'utiliser des frameworks de vue ou des moteurs de templates pour faciliter l'écriture du code : Twig avec le framework Symfony par exemple.

D'une manière générale, une vue contient essentiellement du code HTML qui intègre du code PHP (et pas l'inverse). C'est la manière de procéder la plus propre, on évite ainsi :

- les multiples appels à la fonction echo,
- les guillemets parfois complexes à gérer lors des concaténations, etc.

Pour pouvoir utiliser les variables créées dans le contrôleur, il faut connaître leur structure. Deux possibilités dans ce cas :

- la documentation du modèle et des fonctions est telle qu'il est facile de comprendre comment accéder aux informations ;
- la documentation n'est pas assez claire ou inexistante. Dans ce cas PHP nous donne des outils de debuggage (print_r, echo, etc) permettant de connaître la structure et le contenu des variables.

Partie B : Adaptation de vues

Question 4 : Adaptation du menu général

Documents à utiliser

- fichiers fournis en ressources
- annexe 9

Lors de la connexion de l'utilisateur sur le site, le lien connexion présent dans le menu général doit être modifié pour faire appel à la consultation du profil utilisateur à la place du formulaire de connexion. Le menu est présent dans la vue entete.html.php. Il est composé d'une liste HTML (voir annexe 9)



4.1. Rappeler quelle fonction du modèle authentication.inc.php permet de connaître l'état de connexion du visiteur du site.

4.2. Quel type de donnée est renvoyé par cette fonction ? Donner des exemples.

Lorsque l'utilisateur est connecté sur le site, l'option "Connexion" du menu doit être remplacée par l'option "Mon Profil" qui pourra être produite par le code ci-dessous :

```
<li>
    <a href="./?action=profil">
        
        Mon Profil
    </a>
</li>
```

4.3. Quel élément de la liste du menu général est affiché lorsque l'utilisateur n'est pas connecté (comportement par défaut) ?

4.4. Adapter le code de la vue entete.html.php en utilisant la fonction trouvée à la question 1 afin de faire en sorte que l'option "Mon Profil" soit affichée lorsque l'utilisateur est connecté. L'option "Connexion" est affichée lorsque aucun utilisateur n'est connecté.

Remarques :

- on observe que l'URL pointée par le lien ainsi que le texte affiché sont différents dans les deux cas ;
- les fonctions du modèle authentication.inc.php sont accessibles depuis n'importe quelle page du site, ce fichier étant inclus dans index.php. Il n'est donc pas nécessaire d'ajouter une inclusion dans la vue.

Question 5 : Profil utilisateur - affichage des types de cuisine préférés par l'utilisateur

Documents à utiliser

- fichiers fournis en ressources
- annexes 10 et 12

Pour tester le code que vous allez mettre en place vous devez être connecté sur le site et consulter la page "Mon Profil" accessible en cliquant sur le lien dans le menu général.

Le contrôleur associé à cette fonctionnalité est `monProfil.php`.

L'instruction suivante crée la variable `$mesTypeCuisineAimes` :

```
$mesTypeCuisineAimes = getTypesCuisinePreferesByMailU($mailU);
```

5.1. Afficher la variable `$mesTypeCuisineAimes` dans le contrôleur à l'aide de la fonction `print_r()` puis schématiser sa structure.

5.2. Quelle syntaxe permet d'atteindre le libelle d'un type de cuisine contenu dans la variable `$mesTypeCuisineAimes` ?

5.3. Repérer dans la vue le code HTML permettant d'afficher les types de cuisine.

5.4. Quelle partie de code est répétée ?

5.5. Écrire le code permettant de parcourir la variable `$mesTypeCuisineAimes` et d'afficher pour chaque type le libellé associé.

5.6. Modifier le code produit à la question précédente afin de faire en sorte que chaque type de cuisine soit affiché en respectant les balises HTML trouvées à la question 5.3. Intégrer ce code à la vue `vueMonProfil.php`.

Le code HTML généré visible depuis le navigateur doit se rapprocher de celui de l'annexe 10.

Question 6 : Profil utilisateur - affichage des restaurants aimés par l'utilisateur

Documents à utiliser

- fichiers fournis en ressources
- annexes 11 et 12

Pour tester le code que vous allez mettre en place vous devez être connecté sur le site et consulter la page "Mon Profil" accessible en cliquant sur le lien éponyme dans le menu général. Le contrôleur associé à cette fonctionnalité est `monProfil.php`.

L'instruction suivante crée la variable `$mesRestosAimes` :

```
$mesRestosAimes = getRestosAimesByMailU($mailU);
```

L'objectif de cette vue est d'afficher la liste des restaurants aimés par l'utilisateur. Cette liste apparaît sous forme de liens permettant d'accéder à aux détails de chaque restaurant.

Dans le code suivant figurent des informations provenant de la variable `$mesRestosAimes`.

```
<a href="./?action=detail&idR=4">Cidrerie du fronton</a>
```

On retrouve ainsi l'identifiant et le nom du restaurant.

L'identifiant du restaurant est passé en paramètre pour le contrôleur qui gère la consultation des données du restaurant. Chaque lien doit respecter cette structure.

6.1. En procédant comme pour l'exercice précédent, expliquer comment accéder dans la variable `$mesRestosAimes` à l'identifiant et au nom de chaque restaurant.

6.2. Écrire le code permettant de parcourir la variable `$mesRestosAimes` et d'afficher pour chaque restaurant le nom associé.

6.3. Adapter le code de la vue `vueMonProfil.php` afin de remplacer l'affichage statique des trois restaurants par un affichage dynamique en fonction des restaurants contenus dans la variable `$mesRestosAimes`.

Annexe 1 - vueDetailResto.php

```
<?php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    die('Erreur : '.basename(__FILE__));
}
?>
<h1><?= $unResto['nomR']; ?>
</h1>
<span id="note">
</span>
<section>
    Cuisine <br />
    <ul id="tagFood">
        <?php for ($j = 0; $j < count($lesTypesCuisine); $j++) { ?>
            <li class="tag"><span class="tag">#</span><?=
$lesTypesCuisine[$j]["libelleTC"] ?></li>
        <?php } ?>
    </ul>
</section>
<p id="principal">
    <?php if (count($lesPhotos) > 0) { ?>
        " alt="photo du
restaurant" />
        <?php } ?>
        <br />
        <?= $unResto['descR']; ?>
    </p>
<h2 id="adresse">
    Adresse
</h2>
<p>
    <?= $unResto['numAdrR']; ?>
    <?= $unResto['voieAdrR']; ?><br />
    <?= $unResto['cpR']; ?>
    <?= $unResto['villeR']; ?>
</p>

<h2 id="photos">
    Photos
</h2>
<ul id="galerie">
    <?php for ($i = 0; $i < count($lesPhotos); $i++) { ?>
        <li> " alt="" /></li>
    <?php } ?>
</ul>

<h2 id="horaires">
    Horaires
</h2>
<?= $unResto['horairesR']; ?>

<h2 id="crit">Critiques</h2>

<ul id="critiques">
</ul>
```


Annexe 2 - controleur detailResto.php

```
<?php

/**
 *    Controleur secondaire : detailResto
 */

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

require_once RACINE . "/modele/bd.resto.inc.php";
require_once RACINE . "/modele/bd.typecuisine.inc.php";
require_once RACINE . "/modele/bd.photo.inc.php";

// creation du menu burger
$menuBurger = array();
$menuBurger[] = ["url"=>"#top", "label"=>"Le restaurant"];
$menuBurger[] = ["url"=>"#adresse", "label"=>"Adresse"];
$menuBurger[] = ["url"=>"#photos", "label"=>"Photos"];
$menuBurger[] = ["url"=>"#horaires", "label"=>"Horaires"];
$menuBurger[] = ["url"=>"#crit", "label"=>"Critiques"];

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

// appel des fonctions permettant de recuperer les donnees utiles a l'affichage
$unResto = getRestoByIdR($idR);

$lesTypesCuisine = getTypesCuisineByIdR($idR);
$lesPhotos = getPhotosByIdR($idR);

// traitement si necessaire des donnees recuperees

// appel du script de vue qui permet de gerer l'affichage des donnees
$titre = "detail d'un restaurant";
include RACINE . "/vue/entete.html.php";
include RACINE . "/vue/vueDetailResto.php";
include RACINE . "/vue/pied.html.php";
?>
```

Annexe 3 - extrait du modele bd.resto.inc.php

```
<?php
include_once "bd.inc.php";

function getRestoByIdR($idR) {
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select * from resto where idR=:idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);

        $req->execute();

        $resultat = $req->fetch(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
    }
}
```

```

        die();
    }
    return $resultat;
}

...

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
    ...

    echo "getRestoByIdR(idR) : \n";
    print_r(getRestoByIdR(1));
    ...
}
?>

```

Annexe 4 - extrait du résultat d'exécution de bd.resto.inc.php

```

getRestoByIdR(idR) :
Array
(
    [idR] => 1
    [nomR] => l'entrepote
    [numAdrR] => 2
    [voieAdrR] => rue Maurice Ravel
    [cpR] => 33000
    [villeR] => Bordeaux
    [latitudeDegR] => 44.7948
    [longitudeDegR] => -0.58754
    [descR] => description
    [horairesR] => <table>...</table>
)

```

Annexe 5 - extrait du modele bd.photo.inc.php

```

<?php
include_once "bd.inc.php";
function getPhotosByIdR($idR) {
    $resultat = array();
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select * from photo where idR=:idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);
        $req->execute();
        $ligne = $req->fetch(PDO::FETCH_ASSOC);
        while ($ligne) {
            $resultat[] = $ligne;
            $ligne = $req->fetch(PDO::FETCH_ASSOC);
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
}

```

```

    return $resultat;
}

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
    ...
    echo "\n getPhotosByIdR(4) : \n";
    print_r(getPhotosByIdR(4));
    ...
}
?>

```

Annexe 6 - extrait du résultat d'exécution de bd.photo.inc.php

```

getPhotosByIdR(4) :
Array
(
    [0] => Array
        (
            [idP] => 6
            [cheminP] => cidrerieDuFronton.jpg
            [idR] => 4
        )

    [1] => Array
        (
            [idP] => 14
            [cheminP] => cidrerieDuFronton2.jpg
            [idR] => 4
        )

    [2] => Array
        (
            [idP] => 15
            [cheminP] => cidrerieDuFronton3.jpg
            [idR] => 4
        )
)

```

Annexe 7 - extrait du modele bd.typecuisine.inc.php

```
<?php
include_once "bd.inc.php";
function getTypesCuisineByIdR($idR){
    $resultat = array();
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select typeCuisine.* from typeCuisine,proposer
where typeCuisine.idTC = proposer.idTC and proposer.idR = :idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);
        $req->execute();

        $ligne = $req->fetch(PDO::FETCH_ASSOC);
        while ($ligne) {
            $resultat[] = $ligne;
            $ligne = $req->fetch(PDO::FETCH_ASSOC);
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
    ...
    echo "getTypesCuisineByIdR(idR) : \n";
    print_r(getTypesCuisineByIdR(4));
    ...
}
?>
```

Annexe 8 - extrait du résultat d'exécution de bd.typecuisine.inc.php

```
getTypesCuisineByIdR(idR) :
Array
(
    [0] => Array
        (
            [idTC] => 1
            [libelleTC] => sud ouest
        )

    [1] => Array
        (
            [idTC] => 8
            [libelleTC] => sandwich
        )

    [2] => Array
        (
            [idTC] => 11
```

```

        [libelleTC] => grillade
    )
)

```

Annexe 9 - extrait de la vue entete.html.php - menu général

```

<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>
    <li><a href="./?action=connexion">Connexion</a></li>
</ul>

```

Annexe 10 - extrait code code généré affichant les types de cuisine préférés par l'utilisateur

```

<ul id="tagFood">
    <li class="tag"><span class="tag">#</span>sud ouest</li>
    <li class="tag"><span class="tag">#</span>viande</li>
    <li class="tag"><span class="tag">#</span>grillade</li>
</ul>

```

Annexe 11 - extrait code code généré affichant les restaurants aimés par l'utilisateur

```

les restaurants que j'aime : <br />
    <a href="./?action=detail&idR=4">Cidrerie du fronton</a><br />
    <a href="./?action=detail&idR=6">Le Bistrot Sainte Cluque</a><br />
    <a href="./?action=detail&idR=8">La table de POTTOKA</a><br />

```

Annexe 12 - vue à modifier : vueMonProfil.php

```

<h1>Mon profil</h1>

Mon adresse électronique : <?= $util["mailU"] ?> <br />
Mon pseudo : <?= $util["pseudoU"] ?> <br />

<hr>

les restaurants que j'aime : <br>
    <a href="./?action=detail&idR=4">Cidrerie du fronton</a><br>
    <a href="./?action=detail&idR=6">Le Bistrot Sainte Cluque</a><br>
    <a href="./?action=detail&idR=8">La table de POTTOKA</a><br>
<hr>
les types de cuisine que j'aime :
<ul id="tagFood">
    <li class="tag"><span class="tag">#</span>sud ouest</li>
    <li class="tag"><span class="tag">#</span>viande</li>

```

```
    <li class="tag"><span class="tag">#</span>grillade</li>
</ul>
<hr>
<a href="./?action=deconnexion">se deconnecter</a>
```