Projet site web R3st0.fr

Itération n°1 – Ticket n°2 – étude de l’existant

Table des matières

I - Rappel du contexte....2
II - Étude des fonctionnalités existantes....2
III - Architecture MVC en PHP....3
III.1 Analyse de l'affichage de la liste des restaurants....3
III.2 Analyse d’un script du dossier test....4
III.3 Les couches « vue et « modèle »....5
III.4 La couche contrôleur....5
III.4.1 - Le contrôleur listeRestos.php....5
III.4.2 - Le contrôleur detailResto.php....5
III.4.3 - Synthèse....6
IV - Synthèse globale....6

I - Rappel du contexte

R3st0.fr est un site web de critique de restaurants. 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 de manière globale l'organisation du site.

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités, le site final est consultable en interne à l'adresse suivante : https://www3.jolsio.net/test2slam/public/siteresto (adresse interne uniquement)

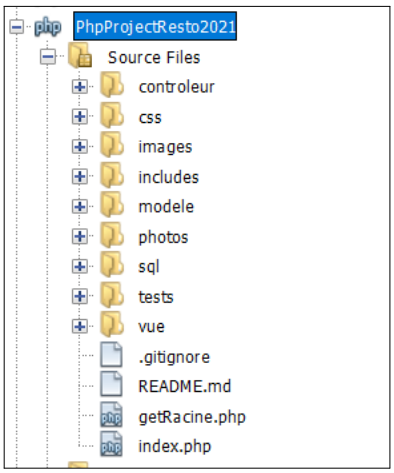
II - Étude des fonctionnalités existantes

Ressources à utiliser
- le code source du site, sous la forme d'un projet NetBeans, est à télécharger depuis Moodle.

Travail à faire

1. Paramétrer le projet pour que celui-ci soit exécutable sur le serveur web local ;
2. Créer la base de données « resto2 » en exécutant les scripts du dossier « sql » du projet ;
3. Paramétrer l'accès à la base de données par l'application :
   Dans le script modele/dao/Bdd.class.php, compléter les lignes suivantes afin d'indiquer les bonnes informations :

```
private static $login = "";        // login utilisateur de la BDD
private static $mdp = "";          // mdp  utilisateur de la BDD
private static $bd = "";           // nom de la BDD
private static $serveur = "";      // nom de domaine du serveur de BDD
```



Arborescence des dossiers du projet

4. Réaliser un diagramme de cas d'utilisation des fonctionnalités existantes

III - Architecture MVC en PHP

Découverte générale du patron de conception MVC.

III.1 Analyse de l'affichage de la liste des restaurants

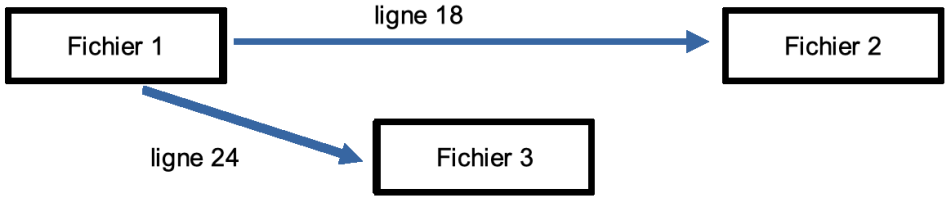
Travail à faire

1. A l'aide de la documentation officielle PHP, rappeler le rôle des mots clés suivants : include_once, include, require_once, require, use

2. Déterminer la valeur de la variable $fichier à la ligne précédant la balise de fin de PHP dans index.php ; *moyen : affichage intermédiaire ou bien mode débogage + point d'arrêt*

3. En partant du fichier index.php, schématiser l'ensemble des fichiers utilisés (require ou require_once) pour afficher la liste des restaurants (URL : index.php?action=liste) ;
   Le schéma comportera :
   - des rectangles portant le nom de chaque fichier,
   - des flèches pointant vers le fichier inclus. Sur la flèche, indiquer le n°de ligne de code de l'inclusion

Exemple :



Commentaire : Le fichier 1 inclus le fichier 2 à la ligne 18

Pour vous aider, utilisez le mode débogage et avancez pas à pas en utilisant step over (F8) ou step into (F7) pour les include.

4. Après avoir observé le code de la classe RestoDAO.class.php, relever l'ensemble des requêtes SQL contenues dans ses méthodes.

Nom de la méthode	**Requête SQL**

5. Quels sont les points communs de ces requêtes SQL ?

III.2 Analyse d'un script du dossier test

Cette section n'est exécutée que lorsque l'on veut effectuer un test unitaire, indépendamment de l'affichage du site web.
Exécutez le script suivant : testRestoDAO.php et observez le code correspondant.

Travail à faire

1. Pour chaque méthode DAO testée, observer le résultat affiché, la requête SQL exécutée, puis expliquer son rôle d'une manière très générale.
   Par exemple, la méthode getRestoByIdR() :
   la requête SQL permet de récupérer les informations d'un restaurant à partir de son identifiant passé en paramètre ($idR). Le résultat d'exécution du test unitaire est le suivant :

```
1- getOneById
Le restaurant n° 6
/tests/dao/testRestoDAO.php:33:
object(modele\metier\Resto)[3]
  private int 'idR' => int 6
  private ?string 'nomR' => string 'Le Bistrot Sainte Cluque' (length=24)
  private ?string 'numAdr' => string '9' (length=1)
  private ?string 'voieAdr' => string 'Rue Hugues' (length=10)
  private ?string 'cpR' => string '64100' (length=5)
  private ?string 'villeR' => string 'Bayonne' (length=7)
  private ?float 'latitudeDegR' => null
  private ?float 'longitudeDegR' => null
  private ?string 'descR' => string 'description' (length=11)
  private ?string 'horairesR' => string '<table>
    <thead>
        <tr>
            <th>Ouverture</th><th>Semaine</th><th>Week-end</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td class="label">Midi</td>
            <td class="cell">de 11h45 à 14h30</td>
            <td class="cell">de 11h45 à 15h00</td>
        </tr>
        [. . .]
    </tbody>
</table>' (length=694)
  private array 'lesPhotos' =>
    array (size=1)
      0 =>
        object(modele\metier\Photo)[7]
          private int 'idP' => int 8
          private string 'cheminP' => string 'leBistrotSainteCluque.jpg' (length=25)
  private array 'lesCritiques' =>
    array (size=1)
      0 =>
        object(modele\metier\Critique)[6]
          private ?int 'note' => int 4
          private ?string 'commentaire' => string 'Cuisine de qualité.' (length=20)
          private modele\metier\Utilisateur 'leUtilisateur' =>
            object(modele\metier\Utilisateur)[5]
              private int 'idU' => int 5
              private string 'mailU' => string 'nicolas.harispe@gmail.com' (length=25)
              private ?string 'mdpU' => string '$1$zvNDSFQSdfqsDfQsdfsT.' (length=24)
              private ?string 'pseudoU' => string 'Nico40' (length=6)
              private array 'lesRestosAimes' =>
                array (size=0)
                  empty
```

   Cette fonction permet donc de récupérer les informations d'un restaurant donné et d'instancier un objet de la classe métier Resto.
2. Quelle méthode de la classe RestoDAO est utilisée dans listeRestos.php ?

III.3 Les couches « vue et « modèle »

L'application web fournie en ressources respecte le « design pattern » MVC ou Modèle Vue Contrôleur. Dans ce patron de conception, les différentes "couches" permettant de construire un site sont gérées de façon indépendante. Découvrons à quoi servent les couches "Vue" et "Modèle".

Répondre aux questions suivantes après avoir observé le contenu des scripts dans les dossiers « controleur » et « vue » du projet.

Travail à faire

1. Trouve-t-on des éléments de CSS ou de HTML dans les fichiers des dossiers controleur et modele ?
2. Dans quel dossier sont contenus les fichiers produisant du code HTML et CSS ?
3. Le code PHP contenu dans vueListeRestos.php est-il toujours visible après réception par le navigateur ? Par quoi est-il remplacé ?
4. Rappeler le point commun entre toutes les méthodes de classe RestoDAO. du dossier modele/dao.
5. Trouve-t-on dans d'autres fichiers que ceux du dossier modele/dao des références à la base de données ?

Synthèse

6. Expliquer globalement le rôle des scripts contenus dans les dossiers suivants
   - modele/dao
   - modele/metier
   - vue

III.4 La couche contrôleur

III.4.1 - Le contrôleur listeRestos.php

Travail à faire

1. Quelle méthode DAO est utilisée dans ce contrôleur ? Que permet-elle de récupérer dans la base de données ?
2. Les données récupérées sont-elles affichées à l'écran ? L'affichage est-il fait dans le contrôleur ?
3. Quels scripts sont inclus dans les dernières lignes du contrôleur ? Quels sont leurs rôles ?

III.4.2 - Le contrôleur detailResto.php

Travail à faire

1. Quelle donnée est transmise au contrôleur en méthode GET ?
2. Rappeler le rôle de la méthode getOneById. Pourquoi cette méthode a-t-elle besoin d'un paramètre ?

La variable $unResto reçoit le résultat de l'appel à getOneById.

3. Explorer les fichiers "vue" inclus à la fin du fichier contrôleur et repérer les lignes où cette variable est utilisée.
4. Le contrôleur gère-t-il directement l'accès aux données ?

III.4.3 - Synthèse

1. Où sont affichées les données créées ou récupérées dans le contrôleur ?

Les données utilisées par le contrôleur peuvent provenir de 2 sources : l'utilisateur ou la base de données.

2. Comment ces données sont transmises au contrôleur :
   - de l'utilisateur vers le contrôleur ?
   - de la base de données vers le contrôleur ?
3. Résumer le rôle de la couche « contrôleur »

IV - Synthèse globale

MVC est l'acronyme de Modèle Vue Contrôleur. Dans le domaine du développement d'applications, MVC est un patron de conception. C'est une bonne pratique de développement d'une application.

Son application apporte plusieurs avantages :

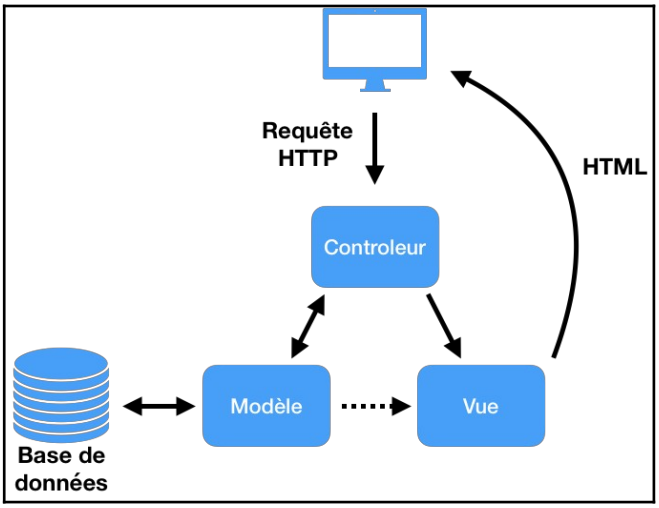
• faciliter le travail en équipe : les composants peuvent être écrits par différentes personnes ;
• faciliter le test unitaire : chacun des composants peut être testé séparément ;
• maintenance et évolutivité : il est possible de revoir le code, ou de changer de technologie sur un des composants (la vue par exemple) sans devoir modifier le reste du code.

Chaque fonctionnalité, par exemple l'affichage de la liste des restaurants, fait ainsi appel aux composants :

• Modèle : fonction d'accès à la base de données ;
• Vue : affichage des données à l'utilisateur ;
• Contrôleur : composant chargé de la logique applicative : récupération les données saisies par l'utilisateur, appel des fonctions du modèle, traitement des données, puis appel des vues pour l'affichage.

Le patron de conception MVC contraint et guide le programmeur dans sa manière de coder une application. Il lui impose des règles, mais en retour il permet de faciliter la maintenance, et le travail en équipe.
Principes de base du MVC :

• le contrôleur et le modèle ne procèdent à aucun affichage : c'est le rôle de la vue ;
• les requêtes SQL sont exclusivement situées dans les méthodes de la couche DAO ;
• les vues ne doivent contenir que du code d'affichage de données provenant du contrôleur, ou éventuellement du modèle (des classes métier).



Pattern MVC