Rapport de projet Développement Web

Site d’annonces gratuites

ABDULSALAM Mohammad Kabir

Table of Contents

1. Introduction

Pour le projet du module de développement de site web, il est demandé de créer un site internet personnel basé sur le duo PHP/MySQL et/ou JavaScript. On a choisi de travailler à la conception d'un site d'annonces gratuites.

2. Objectif

L'objectif de ce site est de permettre à tout internaute de poster une annonce d'un produit (neuf ou occasion) ou de rechercher un produit (neuf ou occasion) parmi les annonces déjà postés. Les annonces pourront être modifié ou supprimer par le titulaire de l'annonce. Ils contiennent aussi une photo (format png ou jpeg), les informations nécessaires de l'annonceur (téléphone, email etc.) afin que les utilisateurs puisse contacter ces derniers et la date de publication. L'ajoute d'une photo n'est pas nécessaire. Le téléphone de l'annonceur non plus mais son adresse mail est obligatoire.

3. Réalisation

3.1 Partie base de données

Pour la réalisation de ce projet il est nécessaires de créer une base de données pour stocke les informations des utilisateurs inscrits et les annonces déposé dans le site. Cette base de donnees est crée en MySQL.

Schéma Entité Relation(MCD) et Schéma Relationnel(MLD)

Veuillez regarder le fichier PDF intitule SCHEMA MCD ET MLD pour plus d'informations.

3.2 Partie PHP

Pour cette partie, on a décidé de créer le site web en utilisant la programmation oriente objet du PHP. La puissance de ce dernier provient principalement de l'héritage et le polymorphisme. Ce qui permet la facile réutilisation du code pour étendre les fonctionnalités du site web. Dans notre cas chaque offre déposée sur le site web, en plus de leurs propres attributs, contient les informations suivantes :

- L'id de l'offre
- Le prix de l'offre
- Le titre de l'offre

- La description de l'offre
- L'id de l'annonceur de l'offre
- L'id du type de l'offre
- L'id du département ou l'offre se situe
- L'id de la région ou l'offre se situe
- La photo de l'offre
- La date de la publication de l'offre
- L'heure de la publication de l'offre.

Donc on peut créer une superclasse qui contient ces informations (Multimédia dans notre cas – mauvais choix de nom) et toutes les autres classes dérivent de ce superclasse. Donc les offres de type immobilier, véhicules etc. peuvent dériver de ce classe.

En plus, on a utilise une règle « Un classe, un rôle ». C'est-à-dire chaque classe a un but. Par exemple la classe Multimedia aura pour but de représenter un multimédia et pas de le gérer. Pour le gérer on a fait appelle a une autre classe qui a pour but d'enregistrer, modifier, supprimer ou accéder a un objet de type Multimedia. Cette organisation nous a permis de ne mettre aucune requête MySQL dans les fichiers PHP qui désignent le site. Donc s'il y a un problème dans fonctionnement du site (requêtes mal préparer etc.…) ou on veut changer une requête na a qu'a aller vers ce fichier pour le changer. Ca nous évite de nous rappeler sur quelle ligne et dans quel fichier a-t-on mis une requête de ce type.

4. PHP

4.1 Les classes utilises

4.1.1 Constructor.class.php

Une classe abstraite qui contient les constructeurs communs pour les autres classes.

Les méthodes principales

1. **public function __construct (array $donnees)** – Le constructeur par défaut pour tous les classes. Cette prend en paramètre un tableau associatif. Les clés de ce tableau les noms des colonnes (champs) dans les tableaux de la base de donnees. Cette méthode utilise la méthode hydrate sur ce tableau associatif.

2. **public function hydrate (array $donnees)** - Fournit les données nécessaires à un objet pour son bon fonctionnement. Cette méthode utilise les mutateurs de la classe concerne pour hydrater l'objet. Le paramètre « $donnees » est un tableau associatif.

4.1.2 Utilisateur.class.php

Cette classe a pour rôle de représenter un utilisateur présent en base de donnees. Elle n'a en aucun cas pour rôle de les gérer. Elle dérive de la classe « Constructor ».

Les attributs

1. private $id – L'id de l'utilisateur
2. private $nom – Le nom de l'utilisateur
3. private $passwd – Le mot de passe de l'utilisateur
4. private $telephone – Le telephone de l'utilisateur
5. private $mail – Le mail de l'utilisateur
6. private $type_util – Le type de l'utilisateur (particulier ou professionnel)
7. private $status – Le statut de l'utilisateur (pending or activated)
8. private $activationkey – La clef d'activation de l'utilisateur

Les méthodes

Il y a que les getters et setters pour cette classe qui ont comme but de changer ou récupérer les attributs d'un utilisateur.

4.1.3 UtilisateurManager.class.php

Cette classe a pour rôle de gérer un Utilisateur.

Les attributs

1. private $db – L'instance de PDO (la base de donnees)

Les méthodes principales

1. **public function __construct ($bdd**) – Ce méthode créer une instance de UrilisateurManager avec une connexion a la base de données « $bdd »

2. **public function add (Utilisateur $user)** – Ce méthode prend en paramètre un variable de type « Utilisateur » et ajoute ce dernier a la base de donnees. En principe il ajoute un utilisateur à la base de donnees. Retourne vrai si l'utilisateur a été ajoute avec succès, faux sinon

3. **public function login ($mail, $password) –** Ce méthode permet de vérifier si un utilisateur qui a « $mail » comme adresse mail et « $password » comme mot de passe existe dans la base de donnees. Il est utilise pour authentifier un utilisateur dans la section login du site. Retourne les informations de tel utilisateur s'il existe, faux sinon.

4. **public function getUserFromId ($id)** – Permet d'avoir les informations d'un utilisateur qui a « $id » come son id. Retourne cette utilisateur s'il existe, faux sinon.

5. **public function getUserFromActivationKey ($key) –** Permet d'avoir les informations d'un utilisateur qui a « $key » come son clé activation. Retourne cette utilisateur s'il existe, faux sinon.
6. **public function getUserFromMail ($mail) –** Permet d'avoir les informations d'un utilisateur qui a « $mail » come son mail. Retourne cette utilisateur s'il existe, faux sinon.

7. **public function sendActivationMail (Utilisateur $user) –** Envoie un mail activation a le mail de l'utilisateur « $user » pour que ce dernier puisse activer son compte. Retourne vrai si réussite, faux sinon.

8. **public function activateUser (Utilisateur $user) –** Prend un paramètre un « Utilisateur » et tente d'activer son compte. Retourne vrai s'il a réussi a active le compte de l'utilisateur, faux sinon.

9. **public function updateUserPasswd (Utilisateur $user) –** Permet de mettre a jour le mot de passe de l'utilisateur. Si l'utilisateur a décidé de changer son mot de passe ou bien qu'il l'a oublie.

4.1.4 Region.class.php

Cette classe a pour rôle de représenter une région présente en base de donnees. Elle n'a en aucun cas pour rôle de les gérer. Elle dérive de la classe « Constructor ».

Les attributs

1. private $id_region – L'id de la région.
2. private $nom_region – le nom de la région.

Les méthodes

Les méthodes sont les accesseur et mutateurs de ces attributs.

4.1.5 Departement.class.php

Cette classe a pour rôle de représenter un département présent en base de donnees. Elle n'a en aucun cas pour rôle de les gérer. Elle dérive aussi de la classe « Constructor ».

Les attributs

1. private $id_dept – L'id du département (Le code postal).
2. private $nom_dept – Le nom du département.

Les méthodes

Les méthodes sont les accesseur et mutateurs de ces attributs.

4.1.6 RegionDeptManager.class.php

Cette classe a pour rôle de gérer une région ou un département.

Les attributs

1. private $db – L'instance de PDO (la base de donnees)

Les méthodes principales

1. **public function __construct ($bdd**) – Ce méthode créer une instance de UrilisateurManager avec une connexion a la base de données « $bdd ».

2. **public function getRegions ()** – Retourne un tableau contenant la liste de toutes les régions et leurs identifiants respectifs.

3. **public function getDepartements ($id) –** Retourne un tableau contenant la liste de toutes les départements et leurs identifiants respectifs.

4.1.7 Multimedia.class.php

Cette classe a pour rôle de représenter une multimédia (Informatique, Téléphonie, Jeux Vidéos et Consoles etc.) présente en base de donnees. Elle n'a en aucun cas pour rôle de les gérer. Elle dérive de la classe « Constructor ».

Remarque : Cette classe est une superclasse qui contient tout les donnes nécessaires pour une offre de type immobilier, véhicule etc. Ainsi les autres classes pour représenter une offre dériveront de cette classe.

Les attributs

1. $id_annonce - L'id de l'offre
2. $prix - Le prix de l'offre
3. $titre- Le titre de l'offre
4. $description - La description de l'offre
5. $id_utilisateur- L'id de l'annonceur de l'offre
6. $id_type- L'id du type de l'offre
7. $id_departement- L'id du département ou l'offre se situe
8. $id_region- L'id de la région ou l'offre se situe
9. $photo1 - La photo de l'offre
10. $date_offre- La date de la publication de l'offre
11. $time_offre - L'heure de la publication de l'offre.

Les méthodes

Les méthodes sont les accesseurs et mutateurs de ses attributs.

4.1.8 Immobilier.class.php

Cette classe a pour rôle de représenter une offre de type immobilier (c'est-à-dire locations, colocations, ventes immobilière etc.) présente en base de donnees. Elle n'a en aucun cas pour rôle de les gérer. Elle dérive de la classe « Multimedia» et par conséquent la classe « Constructor ».

Les attributs supplémentaires

1. private $surface – le surface de la pièce
2. private $nbPieces – le nombre de pièces
3. private $classe_energie – le classe d'énergie (A, B, C, D, E)
4. private $ges – le GES, Gaz à effet de serre (A, B, C, D, E)

Les méthodes

Les méthodes sont les accesseurs et mutateurs de ses attributs.

4.1.9 OffresManager.class.php

Cette classe a pour rôle de gérer les offres c'est-à-dire les autres classes (Multimedia, Immobilier).

Les attributs

1. private $db – L'instance de PDO (la base de donnees)

Les méthodes principales

1. **public function __construct ($bdd**) – Ce méthode créer une instance de UtilisateurManager avec une connexion a la base de données « $bdd ».

2. **public function add ($data) –** le variable $data une instance d'un objet (Multimedia ou Immobilier). Donc c'est une offre et cette méthode permet d'ajouter une offre dans la base de donnees. La fonction PHP « getClass ($data) » est utilise pour avoir le nom de classe de l'objet « $data » et c'est en fonction de cette classe on va insérer dans le table approprie dans la base de donnees. Retourne vrai si réussite, faux sinon.

3. **public function getIdType ($param) –** Retourne le id du type de l'offre.

4. **public function getNumberOfOffers ($table, $id) –** Retourne le nombre de l'offres présents dans un table « $table ». Le $id est un paramètre pour savoir si on veut récupérer tout les offres présents dans la basse de donnees ou juste dans un table de la base de donnees.

5. **public function getUsersOffer ($user_id, $offer_id, $offerType) –** Retourne l'offre qui a $id_utilisateur « $user_id », $id_annonce « $offer_id » et $id_type « $offerType ».

6. **public function getDeptName ($id) –** Retourne le nom du département avec l'id « $id ».
7. **public function getRegionName ($id) –** Retourne le nom de la région avec l'id « $id ».
8. **public function getNameType ($id) –** Retourne le nom du type de l'offre avec l'id « $id ».

4.2 Le conf.php

Il contient une class appelle DBConf. Les informations de configuration de base de données sont stockées ici. C'est ce fichier il faudra changer si on veut migrer vers une autre base de donnees.

Les attributs

1. private $databaseURL – Le URL de la base de donnees.
2. private $databaseUserName – le nom d'utilisateur de la base de donnees.
3. private $databasePWord – le mot de passe de la base de donnees.
4. private $databaseName – le nom de la base de donnees.

Les méthodes

Les méthodes sont les accesseurs et de ses attributs.

4.3 Les Fonctions

1. **function verif_tel ($tel) –** Vérifie la syntaxe d'un numéro de telephone taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
2. **function verif_mail ($mail)** – Vérifie la syntaxe d'un mail taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
3. **function verif_passwd ($passwd) -** Vérifie la syntaxe du mot de passe taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
4. **function verif_nom ($nom) -** Vérifie la syntaxe du nom/pseudo taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
5. **function verif_titre ($titre) –** Vérifie la syntaxe d'un titre de l'offre taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
6. **function verif_prix ($prix) –** Vérifie la syntaxe du prix taper par l'utilisateur. Retourne vrai si la syntaxe est correcte, faux sinon.
7. **function checkRegistrationForm () -** Vérifie le formulaire d'inscription pour déterminer si l'utilisateur l'a bien rempli. Retourne vrai si tout les champs sont bien rempli, faux sinon.
8. **function generateUniqueId () –** génère une suite de nombre et lettres unique.
9. **function checkOfferForm () –** Vérifie le formulaire de dépôt des offres pour déterminer si l'utilisateur l'a bien rempli. Retourne vrai si tout les champs sont bien rempli, faux sinon.
10. **function pageNumber ($arr) –** Permet de faire la pagination d'une page.
11. **function date_in_french($date) –** Mettre la date en format français.
12. **function iplog() –** Enregistre l'adresse IP du client, la page où le client consulté et l'heure d'accès.

5. Conclusion

On a travaille sur un projet très intéressant base sur une situation réelle de la vie courante. La modélisation d'une base de donnees très optimale et une rigueur dans l'implémentation du PHP sont nécessaires pour la réussite de ce projet. Ce qui nous a permis d'améliorer nos connaissances dans ces domaines.

Ce projet nous a aussi permis de découvrir un nouveau langage de programmation, JavaScript (AJAX).