Module Programmation Web S3
Mini-projet PHP

On souhaite réaliser un mini-gestionnaire de blog. Cet outil doit permettre d'afficher des billets (messages) classés par catégories et bien sur d'en ajouter. Les fonctionnalités minimales attendues sont les suivantes :

1. affichage d'un billet complet : titre, date, message complet, catégorie (colonne centre)
2. affichage de la liste des billets, avec pour chacun d'eux : titre, date, début du texte (colonne centre)
3. affichage de la liste des catégories (menu gauche), affichage des billets d'une catégorie (colonne centre)
4. accès direct aux billets par menu (menu droit)
5. ajout d'un billet : saisie du titre, du message et de la catégorie.

I. Les Contrôleurs :

L'objectif de cette 1ère partie est d'écrire les programmes de contrôle de l'application, c'est à dire les programmes qui sont appelés par les clients (via une url) et réalisent la gestion de l'interaction et le déroulement de la logique de l'application. Ces contrôleurs n'interagissent pas avec la base de données et ne réalisent pas eux-mêmes la génération de code html. Ces 2 aspects seront traités dans les parties II et III.

Le principe est celui-ci : chaque action offerte à l'utilisateur correspond d'une part à une url (pour invoquer l'action) et d'autre part à une méthode d'un contrôleur. Le rôle des contrôleurs est d'assurer que les méthodes sont appelées de façon adéquate lors de la réception d'une requête émise par un client.

Dans notre application, on aura 2 contrôleurs : un contrôleur pour gérer les actions des visiteurs du blog (afficher les messages, afficher les messages d'une catégorie, afficher le détail d'un message) et un contrôleur pour gérer les actions de gestion du blog (ajout d'un message, ajout d'une catégorie). Chaque contrôleur reçoit en paramètre :

- l'action à effectuer
- les paramètres de cette action

Ainsi, on aura des urls du type :

- http:// .... /blog.php?a=list // tous les messages
- http://.... /blog.php?a=detail&id=3 // détail du message 3
- http://... /blog.php?a=cat&id=2 // messages de la catégorie 2
- http://.... /admin.php?a=addM // ajout d'un message : affichage du formulaire
- http://.... /admin.php?a=saveM // ajout d'un message : envoi des données
- http://.... /admin.php?a=addC // ajout d'une catégorie : affichage du formulaire
- http://.... /admin.php?a=saveC // ajout d'une catégorie : affichage du formulaire

Un contrôleur est lui-même est une classe qui comprend notamment une méthode pour chaque action à réaliser.

1. Construire le contrôleur gérant les actions des visiteurs et la navigation sur le blog. La classe « `BlogController` » implante les méthodes « `BlogController::listAction()` », « `BlogController::detailAction()` », « `BlogController::catAction()` ». Pour l'instant, ces méthodes se contentent d'afficher leur nom et paramètres ; elles seront complétées lors de la réalisation des parties II et III.
2. Ajouter une méthode dans le contrôleur, nommée « `callAction`() » dont le rôle est d'analyser les paramètres reçus dans la requête et d'appeler la méthode adéquate. Cette méthode reçoit en paramètre le tableau contenant les paramètres de la requête http. Le programme « blog.php » crée le contrôleur et appelle cette méthode. Cette méthode utilise un tableau pour faire la correspondance entre action dans l'url et nom de la méthode à appeler.

```
<?php
 include 'BlogController.php' ;
 $c=new BlogController ;
 $c->callAction( $_GET ) ;
```

3. Construire sur le même modèle le contrôleur pour les actions de gestion du blog.
4. Construire la classe abstraite « `Controller` ». Elle définit la méthode « `callAction`() » ; les 2 classes concrètes « `BlogController` » et « `AdminController` » héritent de cette classe.

II. Gestion des données

Les données du blog sont stockées dans une base de données contenant 2 tables : « billet » et « categorie ». Vous trouverez la structure complète de cette base dans l'ENT. L'objectif de cette partie est de créer les classes d'accès aux données du blog dans la base. Vous devez utiliser une classe de gestion de la connexion, et une classe par table construite suivant le modèle « Active Record ». Vous aurez ainsi 3 classes : « Base », « Billet », « Categorie ».

Afin d'éviter tout problème d'encodage de caractères, tout doit être en UTF-8 :

- la base de donnée doit être déclarée en UTF-8 (valeur interclassement utf8_general_ci)
- les pages HTML générées doivent avoir un meta charset=utf-8
- PDO doit préciser à la base qu'il travaille en utf-8
    - `$db->exec("SET CHARACTER SET utf8");`
- les requêtes doivent être des requêtes préparées.

En cas d'absence de l'une de ces propriétés, vous risquez de rencontrer de nombreux soucis (dans vos requêtes, affichages, etc …)

1. Créer la base sur le server Webetu en utilisant le script SQL fourni dans l'ent.
2. Créer la classe « `Base` » gérant la connexion. La classe offre une méthode statique publique : `Base::getConnection`() qui permet d'obtenir une connexion à la base. Les paramètres de connexion doivent être stockés dans un fichier. Il faut créer une connexion PDO persistante.
3. Construire la classe « Categorie » qui gère l'accès à la table des categories. Pour démarrer cette réalisation, utilisez le squelette de la classe ainsi que le programme de test de la classe fournie dans l'ENT. Dans le squelette, complétez les parties indiquées. Votre classe DOIT fonctionner avec le programme de test fourni. Cette classe réalise un « Active Record », et implante les méthodes :
    - Les accesseurs magiques __get() et __set() pour manipuler les attributs de l'objet,
    - Les méthodes `delete()` puis `insert()` pour gérer la correspondance objet-table,
    - `findById(), findAll()` pour rechercher des lignes dans la table et les récupérer sous forme d'objet.
4. Programmer la méthode « `Categorie::findByTitre()` » qui permet de retrouver des catégorie à partir de leur titre.
5. Sur le même modèle, programmez la classe Billet.

III. Gestion de l'Affichage :

Créer une classe regroupant les fonctionnalités d'affichage du blog :

1. une méthode pour l'affichage général, recevant en paramètre le contenu central et les 2 menus (droite et gauche).
2. une méthode pour afficher 1 billet : génère l'affichage de ce billet comme prévu en (a) ci-dessus. Cette méthode ne génère qu'un fragment html et le renvoie en résultat. Ce résultat est ensuite passé à la fonction 1 , pour affichage dans la colonne centrale.
3. une méthode pour afficher une liste de billets : génère un tableau html (uniquement le tableau) correspondant à l'affichage (b). Même contraintes que ci-dessus.
4. une méthode pour générer le menu de la colonne droite (d),
5. une méthode qui génère le fragment html correspondant au menu de la colonne gauche (c).

IV. Extensions

Etendre votre blog pour gérer des utilisateurs. L'objectif est double :

1. contrôler l'accès à l'ajout de billets et de catégories
2. identifier les auteurs des billets

Pour cela, créez une table des utilisateurs, et stockant au moins pour chaque utilisateur : userid, login, password, mail.

Ajoutez également une référence à l'auteur dans la table des billets. Dans le même esprit que les questions précédentes, ajoutez la gestion des utilisateurs dans votre application :

- affichage de l'auteur dans l'affichage détaillé d'un billet,
- possibilité, en cliquant sur l'auteur, d'accéder à son profil,
- gestion de l'ajout d'utilisateurs

Ajoutez ensuite le contrôle d'accès : pour toutes les fonctions « back-office », faites un contrôle de droits d'accès. Prévoyez un formulaire de connexion et une variable de session pour stocker le profil des utilisateurs identifiés. Vérifiez le contenu de cette variable de session chaque fois qu'il est nécessaire de contrôler l'accès.