Système d'information - 2020/2021

TP 4 : Microblog - MVC, DTO...

Exercice 1 : Partir sur de bonnes bases

Afin que tout le monde reparte sur une base de code "saine" et identique, vous allez déployer une version du microblog "toute prête toute propre" qui implémente toutes les fonctionnalités demandées lors du précédent semestre.

a) Récupérez le code du microblog à l'endroit indiqué par votre encandrant.
b) Vous y trouverez (entre autre) un fichier blog.sql à importer dans votre base de données sur mmi.unilim.fr via phpmyadmin. Cela y créera une table Blog nécessaire au fonctionnement de l'application.
c) Editez le fichier model.php et dans la fonction de connexion à la base de données, modifiez les paramètres du constructeur PDO en fonction de vos propres identifiants.
d) Uploadez sur votre hébergement MMI l'ensemble des fichiers php ainsi que le répertoire contenant les images. Ce répertoire ne doit pas être renommé et son emplacement doit demeurer au même niveau que les fichiers php.
e) Testez et vérifiez le bon fonctionnement du microblog. Si vous avez le temps, adaptez le css que vous aviez pour votre précédente version.

Exercice 2 : DTO : Data Transfert Object

Dans une application PHP/MySQL, il faut distinguer :

— la représentation des données dans la base. C'est une version persistente des données, au sens ou elles perdurent d'une utilisation à une autre de l'application. Dans notre cas, il s'agit de la table Blog.
— la représentation des données au sein de l'application. Cette représentation est celle utilisée par le programme et est de fait limitée à sa durée d'utilisation. Dans notre cas il s'agit du tableau associatif retourné par la fonction `getAllTweets`.

Les données sont les mêmes, mais leur représentation diffère. Il est important que ces représentations soient indépendantes. Par exemple si on modifie la base de données, il ne faut pas que ça ait de lourdes conséquences sur le code PHP.

Dans notre microblog, les clés du tableau associatif retourné par `getAllTweets` sont de fait les noms des attributs de la table Blog. Donc si on change ces attributs, il faudra le répercuter dans tout le code partout où on les utilise. Ici c'est un cas simple, c'est faisable, mais imaginez sur une grosse application ! Et puis de manière générale, on souhaite pouvoir coder sans avoir à se souvenir des attributs des tables de la base !

Les Data Transfert Object (DTO) correspondent à un patron de conception (design pattern) qui répond (entre autre) à ces problèmes. Les DTO sont des objets qui ne font que stocker les données au sein du programme. Leurs méthodes se limitent le plus souvent à de simples accesseurs qui permettent au programme d'accéder aux données contenus.

a) Ecrire une classe `Tweet` qui représente un message.
   1) Les propriétés seront privées. On représentera l'identifiant, le contenu et la date d'un message.
   2) Ajoutez un constructeur prenant en paramètre un identifiant, un message, une date.
   3) Ajoutez un accesseur pour chaque propriété, par exemple `getId(), getMessage(), getDate()`
b) Modifiez la fonction `getAllTweets` de sorte à ce qu'elle retourne un tableau d'objets `Tweet`. Attention, vous retournerez un tableau associant chaque message à son identifiant (ça sera pratique plus tard)
c) Modifiez la vue sachant que `$tweets` contient à présent des objets `Tweet` et vérifiez que tout fonctionne
d) Justifiez l'affirmation suivante : Maintenant, si l'on modifie la base de données (structure, nom des attributs...), les répercussions sur le code PHP seront limitées.