Mini-Projet POO
Création d'un Blog

Formation Développeur Web & Web Mobile

Objectif

Vous allez créer un **blog**, entièrement articulé autour de la **Programmation Orientée Objets**. Les différents concepts seront modélisés à travers des classes (Blog, Post, User, Comment, ...). Dans un premier temps sans base de données (avec un jeu de données prédéfini qui vous sera fourni), ce blog pourrait sans difficulté majeure évoluer pour se greffer dans un second temps sur une véritable base de données (ou sur tout autre système de persistance *adhoc* – voir les améliorations possibles en fin de projet).

Enoncé

Dans ce mini-projet vous allez coder vous-même un **micro-blog**. Bien sûr, il ne s'agit pas de refaire Wordpress ! D'autant plus que nous travaillerons sans base de données. Dans un premier temps on se contentera d'afficher les données de notre blog avant de pouvoir en créer. Mais la structure du blog est suffisamment intéressante pour mériter un petit focus.

Un peu de terminologie d'abord :

- Un **blog** est un **ensemble de posts**, appartenant à un **utilisateur**. Ce blog possède donc un **titre**, un utilisateur défini comme propriétaire, et un tableau contenant tous les posts.
- Un **post** (ou article) est rédigé par un **utilisateur**. Il contient un **titre** et un **corps** et est publié à une certaine **date**. Chaque post dispose d'un **nombre de vues** et d'éventuels **commentaires**, laissés par des utilisateurs, contenus dans un tableau.
- Un **commentaire** est un **message textuel** laissé par un **utilisateur**. Il est **daté**, et est donc rattaché à un post.
- Un **utilisateur** représente un utilisateur sur le site et identifié par un **nom d'utilisateur**, un **prénom** et un **nom de famille**. Le propriétaire du blog, et les auteurs des posts et des commentaires sont tous des utilisateurs.

Autour de ces concepts, on va pouvoir construire des pages afin d'en afficher le contenu.

Mini-Projet POO
Création d'un Blog

Formation Développeur Web & Web Mobile

Votre travail à réaliser :

Dans un répertoire « *classes* » vous allez créer les classes suivantes :

- Blog
- Post
- Comment
- User

Chacune de ces classes contiendra les propriétés et les méthodes nécessaires à son bon fonctionnement telles que définies ci-dessous.

La classe Blog :

Propriétés privées :

- **title** pour le titre du blog,
- **owner** pour le propriétaire du blog (qui sera un User)
- un tableau **posts** qui contiendra les posts du blog.

Méthodes publiques :

- les **getters/setters** s'appelant *getNomPropriété* et *setNomPropriété*,
- un **constructeur**
- une méthode **addPost** prenant un Post en paramètre et qui a pour but d'ajouter ce post au tableau de *posts* du blog.

La classe Post :

Propriétés privées :

- **title**, le titre du post
- **message**, le texte du post
- **author,** l'auteur de l'article (qui doit être un User)
- **date,** pour la date de publication de l'article (construite avec DateTime)
- **views,** donnant le nombre de vues de l'article
- ainsi qu'un tableau **comments** contenant tous les commentaires du post

Méthodes publiques :

- les **getters/setters** s'appelant *getNomPropriété* et *setNomPropriété*
- le **constructeur**
- et la méthode **addComment**, prenant un Comment en paramètre et l'ajoutant au tableau *comments*

La classe Comment :

Propriétés privées :

- **message**, le texte du commentaire
- **author,** l'auteur du commentaire (qui doit être un User)
- **date,** pour la date de publication du commentaire (construite avec DateTime)

Méthodes publiques :

- les **getters/setters** s'appelant *getNomPropriété* et *setNomPropriété*
- un **constructeur** prenant dans l'ordre le texte du commentaire, son auteur et la date de publication

La classe User :

Propriétés privées :

- **username**, **firstname**, **lastname** pour le pseudo, le prénom et le nom de famille de l'utilisateur

Méthodes publiques :

- les **getters/setters** s'appelant *getNomPropriété* et *setNomPropriété*
- un **constructeur** prenant dans l'ordre le username, le firstname et le lastnam de l'utilisateur.

Il s'agit là de ce que vous devez implémenter **au minimum**. Si vous avez besoin de rajouter des méthodes, vous êtes libres de le faire mais chaque classe devra contenir ces éléments là pour fonctionner, sinon vous obtiendrez des erreurs.

Cet ensemble de classes formera ce que nous appelons un « Modèle objet ». Ce modèle sera l'un des composants essentiels du design pattern MVC (Modèle Vue Controleur) qui

est le design pattern le plus utilisé dans le monde pour construire des applications et des sites complexes.

Une fois ce modèle complètement codé, vous pourrez créer les pages suivantes :

- index.php
- post.php

index.php

Cette page affichera le nom du blog, ainsi que le titre de tous les posts contenus dans le blog sous forme de liste, avec le nombre de vues de chaque posts ainsi que le nom de l'auteur de ce post.

Chaque titre de post sera un lien vers la page **post.php ?id=xxx** où xxx sera l'identifiant du post (son index dans le tableau *posts* de l'objet *blog*).

Le premier post du tableau de posts du blog aura l'id 0, car c'est son index dans le tableau.

post.php

Cette page servira à mettre en forme un post spécifique, dont l'identifiant sera passé en paramètre GET de la page, de la forme *id=xxx* où xxx est l'index du post dans le tableau de *posts* du blog.

Le premier post aura l'id 0, le second post aura l'id 1, etc ...

Ils seront affichés en saisissant les urls : **post.php?id=0**, **post.php?id=1**, **post.php?id=2**, ...

Si l'id est absent, ou si l'id saisi ne correspond pas à un post existant, à vous de mettre en place ce qu'il faut pour ne pas générer d'erreur, et pour indiquer à l'utilisateur la nature du problème.

Vous êtes libre de mettre en forme le post comme vous le souhaitez mais vous devrez exploiter au mieux les informations qui lui sont attachées, ainsi eu les commentaires associés à ce post.

Vous prévoirez aussi un lien, disponible sur toutes les pages, permettant de revenir sur la page **index.php**.

Mini-Projet POO
Création d'un Blog

Formation Développeur Web & Web Mobile

Technique :

Dans les deux fichiers **index.php** et **post.php** vous allez importer les classes que vous aurez écrites ainsi que le fichier **datas.php** qui vous est fourni. Ce fichier contiendra les données de base de votre blog ! Avec déjà plusieurs posts inclus, des utilisateurs, des commentaires ...

Si vous obtenez des erreurs en chargeant ce dernier fichier dans votre navigateur (avec un *require* ou un *require_once*) ça signifie que votre modèle ne correspond pas aux attentes. Il est peut-être incomplet ou bien vous n'avez pas nommé les méthodes comme il faut. Regardez bien le contenu de ce fichier pour comprendre quel nom donner à vos méthodes ou ce qui manque.

Pour tester si les données du blog se chargent bien, après avoir effectué tous les *require*, vous pouvez mettre le code suivant dans une page que vous affichez dans votre navigateur (par exemple dans votre page *index.php*) :

```
<pre><?php print_r($blog) ?></pre>
```

Si vous obtenez la même chose que dans le fichier **blog.txt** fourni, alors tout va bien, vous pourrez créer vos pages **index.php** et **post.php** sans problème.

Ne passez pas à la suite tant qu'il vous reste des erreurs à ce stade.

Dans le fichier **index.php**, vous devez affichez le titre (avec lien) de chaque post. Vous allez donc faire une boucle sur le tableau de posts du blog, accessible par son getter *$blog->getPosts()*. Attention, chaque élément sera un objet de type Post !

De même, dans le fichier **post.php**, quand vous afficherez les commentaires vous itèrerez sur le tableau de commentaires de ce post accessible via la méthode *getComments()*. Chaque élément sera de type Comment.

Concernant le rendu visuel, et tout l'aspect *front-end* des pages à créer, vous êtes totalement libres de créer votre propre feuille de style CSS ou bien d'utiliser un *template* trouvé sur internet. Ce qui sera important, ce sera la bonne intégration des éléments du blog dans cette interface. Recherchez l'expérience utilisateur, même si le rendu est minimaliste. Pensez aussi à produire un code modulaire, avec un découpage en plusieurs fichiers (*header, footer, sidebar, navbar etc* ...).

Pour aller plus loin

Si vous avez bien aimé ce mini-projet, vous aurez peut-être envie d'aller encore plus loin. Voici quelques améliorations intéressantes à faire.

- Avec les méthodes json_encode() et json_decode() de PHP vous pouvez transformer un objet (ou un tableau) en une chaine de caractères structurée (au format JSON, pour JavaScript Objet Notation) ou l'inverse, c'est-à-dire reconstruire un objet d'après sa chaine de caractères JSON. Vous savez lire et écrire dans les fichiers en PHP. Faîtes donc en sorte d'enregistrer la base de données (actuellement dans *datas.php*) dans un fichier **datas.json** afin qu'à chaque chargement de page ce soit ce fichier qui soit lu.
- L'avantage de cette méthode, c'est qu'elle vous permettra de rajouter de nouvelles fonctionnalités à votre blog telles que l'ajout d'articles ou de commentaires. En effet, vous pourrez rajouter des formulaires pour saisir des commentaires ou des posts, et ces nouveaux éléments pourront **persister**.
- Lorsque vous soumettrez le formulaire, vous chargerez le fichier json contenant vos données, vous ajouterez dans votre **objet blog** les nouveaux éléments (comme ça a été fait dans **datas.php**) et vous enregistrerez le nouveau blog dans le même fichier json. Ainsi, aux chargements suivants, votre blog contiendra tous les éléments que vous aurez rajoutés depuis l'interface de votre blog.
- Vous avez compris comment réutiliser ces concepts dans d'autres projets ? Dans la pratique vous utiliserez très souvent ce concept de modélisation objet pour manipuler les données de votre application. Ce modèle vous permettra aussi de faire le lien entre la base de données (et ses données) et les données dans le langage backend que vous utilisez – ici PHP.

JSON est le format d'échange de données le plus utilisé sur internet pour transmettre des données structurées entre des composants (notamment sous la forme de réponses aux requêtes ajax). Il permet aussi, comme c'est ici le cas, de faire office de pseudo base de données.