TP 11 : Modèle MVC sur un site marchand

Objectifs du TP:

- Appliquer le modèle MVC à un autre site.
- Voir un objet **DAO** qui fait une abstraction supplémentaire à PDO
- Construire un objet lié à un autre, à partir de la BD

1. Partie visualisation d'un site marchand en MVC

Objectif : programmer une application Web qui affiche les articles du site de vente "Bricomachin" selon le patron MVC en utilisant un DAO pour les accès à une base de données. Dans cet exercice, on ne travaille que sur la partie affichage des données, sans modifications.

1.1 Analyse

Chaque article à vendre est défini par une *référence*, un *libellé*, un *prix* et une *catégorie*. Les catégories ont un *nom* et un *identifiant* unique. Elles ont une structure d'arbre. Chaque catégorie possède alors l'identifiant de sa catégorie parent. La racine de l'arbre a pour parent elle même.

Vous devez programmer une application selon le modèle MVC. Une partie de l'application est déjà programmée. Les répertoires : `view`, `model`, `controler`, `framework` et `data` existent déjà. Dans le répertoire `model`, le fichier `dao.class.php` contient la classe d'accès à la base de données. Il existe également deux fichiers pour les classes `Article` et `Categorie`. Le répertoire `data` contient les données. Le fichier `bricomachin.db` devra contenir la base de données. Cette base de données doit être chargée à partir des deux fichiers texte qu'il faut importer avec `sqlite3`.

1.2 Récupération du code existant

- Récupérez la version incomplète de l'application dans le fichier data/bricomachin.zip
- Décompressez et installez ces fichiers dans votre répertoire Web.
- Les images sont disponibles à l'adresse suivante : https://www-info.iut2.univ-grenoble-alpes.fr/intranet/enseignements/ProjWeb/data/bricomachin/img/. Il ne faut pas les télécharger mais simplement utiliser des liens dans le HTML (URL).

1.3 La base de donnée.

Allez dans le répertoire `data`. Examinez le fichier `create.sql`. Il contient la définition de la base de donnée.

```
CREATE TABLE article (
    ref INTEGER PRIMARY KEY,
    libelle TEXT,
    categorie INTEGER,
    prix REAL,
    image TEXT,
    FOREIGN KEY(categorie) REFERENCES categorie(id)
);
```

La table `article` décrit un article à vendre avec sa référence unique sa description (libellé), son prix, le nom du fichier image de cet article, et une référence à une catégorie.

```
CREATE TABLE categorie (
    id INTEGER PRIMARY KEY,
    nom TEXT,
    pere INTEGER,
    FOREIGN KEY(pere) REFERENCES categorie(id)
);
```

La table `categorie` contient la liste des catégories possibles. Chaque catégorie est liée à une autre catégorie : il s'agit de son noeud parent dans l'arbre. La racine de l'arbre est liée à elle même (cf. "Produits" ci-dessous). Le fichier `categorie.txt` contient la liste des catégories au format texte (CSV avec le caractère séparateur de champs `|`). Par exemple :

```
1|Produits|1
9|Outillage|1
10|Outillage à main|9
11|Clé et douille|10
```

La catégorie `Clé et douille` d'identifiant `11` est la fille de la catégorie `10` de nom `Outillage à main`. La catégorie `Produits` a pour parent elle même : c'est donc la racine de la hiérarchie. Le chemin dans l'arbre qui mène de la racine à la catégorie `Clé et douille` est le suivant :

```
 Produits > Outillage > Outillage à main > Clé et douille
```

Le fichier `article.txt` contient la liste de tous les articles de la base. Par exemple :

```
67400732|Clé à molette en chrome vanadium DEXTER, 28 mm|11|17.10|67400732.jpg
65996084|Coffret à douilles CRV DEXTER 54 pièces|11|49.95|65996084.jpg
```

L'image `67400732.jpg` correspond à une clé à molette d'identifiant 67400732 à 17,10€ et de catégorie "Clé et douille".

Les images sont disponibles à d'adresse : https://www-info.iut2.univ-grenoble-alpes.fr/intranet/enseignements/ProjWeb/data/bricomachin/img/. **Attention** : Ne recopiez pas ces images localement dans votre répertoire, mais utilisez l'URL.

Travail à faire :

1. Créer la base de donnée dans le fichier `bricomachin.db` dans le répertoire `data` avec le logiciel `sqlite3`. Utilisez le fichier `create.sql` pour créer les tables.
2. Charger les catégories depuis le fichier `categorie.txt` dans la table correspondante. Attention, le séparateur de champs est le caractère `|`. Utilisez la commande de sqlite :

```
.separator |
```

Notez que le séparateur influence aussi le format de sortie des requêtes envoyée dans l'interface en ligne de commande (CLI: Commande Line Interface). Par défaut l'affichage dans la CLI est avec des séparateurs sans entêtes, mais il est possible d'avoir un meilleur affichage avec la commande : `.mode` par exemple :

```
.mode table
```

Pour plus d'information : https://database.guide/4-tabular-output-modes-in-sqlite/

3. Charger les articles depuis le fichier `article.txt` dans la table correspondante.
4. Vérifier à l'aide de quelques requêtes simples, que la base de données est correctement configurée.
5. IMPORTANT : ajoutez les droits d'écriture sur la base de données et sur le répertoire qui la contient.

1.4 Modèle : classe DAO

Un unique objet de la classe DAO permet de "représenter" la base de données. Le DAO permet donc de s'abstraire de la couche PDO, qui elle même est déjà une couche logicielle qui permet de s'abstraire du véritable logiciel de base de données utilisé (ici Sqlite). La classe DAO se trouve dans le fichier `model/dao.class.php`.

Travail à faire :

Examinez le codage de la classe DAO. Cette classe DAO en une classe singleton. Ce patron de conception n'autorise l'existance que d'un seul objet pour cette classe. Le constructeur est alors privé, et une méthode statique (méthode de classe) est disponible pour avoir accès à cet unique objet. Si cet objet n'existe pas encore, il est alors créé lors du premier appel.

- Quel est le nom de l'attribut de cette classe qui va conserver la référence sur l'objet singleton ? A quoi sert le mot clé "static" ?
- Quel est le nom de la méthode utilisée pour accéder à l'objet DAO ? Quel est son statut ?
- Pourquoi le constructeur est privé ?
- A quoi ce patron de conception est utile pour le DAO ?
- Que signifie `self::$instance` dans la méthode `get()` ?

1.5 Modèle : classe Categorie

Examinez la classe Catégorie du modèle. Elle est composée de getters et de méthodes pour gérer la persistance des données en utilisant une base de données avec la classe DAO.

Travail à faire :

Examinez la classe Categorie:

- Complétez les trois getters de la classe.
- Utilisez en ligne de commande le fichier `test/Categorie.test.php`.

```
~/VotrePath/Bricomachin/test$ php Categorie.test.php
```

Les premiers tests doivent fonctionner.

- Pourquoi la méthode `read(int $id): Categorie` est-elle statique ?
- Comment est réalisé l'accès à la base de données ?
- A quoi servent les deux tests d'erreur ?
- La méthode `public function readSubCategorie(): array` est une méthode de parcours de l'arbre des catégories. Elle retourne la liste des sous catégories d'une catégorie sous la forme d'une table d'objets Categorie. Réalisez cette méthode sachant qu'une catégorie A est fille d'une catégorie B si dans le champs `pere` de A se trouve l'identifiant de B. Réalisez cette méthode et la tester avec `Categorie.test.php`.

1.6 Modèle : classe Article

La classe Article modélise les articles de la base de données. Notez qu'elle possède une constante de classe qui est le chemin URL du répertoire qui contient les images à afficher. Un objet Article est lié à une catégorie. Dans la base de données, ce lien est matérialisé par la référence à un identifiant de catégorie. Lorsque l'on passe du modèle relationnel au modèle objet, ce lien doit devenir un lien entre objets. La lecture (read) d'un objet Article doit donc impliquer la lecture d'un objet Categorie. Pour être cohérent avec le modèle relationnel, il ne faudrait créer qu'un seul objet par catégorie. Dans cet exercice, nous ne tiendront pas compte de cette contrainte, cela signifie que deux articles d'une même catégorie seront associé à deux objets Categorie différents.

Travail à faire :

- Complétez les getters et la méthode `getImageURL` qui retourne l'URL de l'image. Utiliser la constante `URL` de la classe. La syntaxe pour l'usage d'une constante de classe est

```
self::CONSTANTE
```

- Complétez la méthode de classe `read` qui crée un objet Article correspondant aux données de la BD avec l'identifiant `ref`.

```
public static function read(int $ref): Article
```

- Complétez la méthode de classe `count` qui retourne le nombre total d'articles dans la base. Utiliser le fonction COUNT de SQL.

```
public static function count() : int
```

- Complétez la méthode de classe `readPage` permet de lire une série d'articles qui vont être affichés dans une page. Ces articles sont triés par référence. Les pages doivent être numérotés à partir de 1.

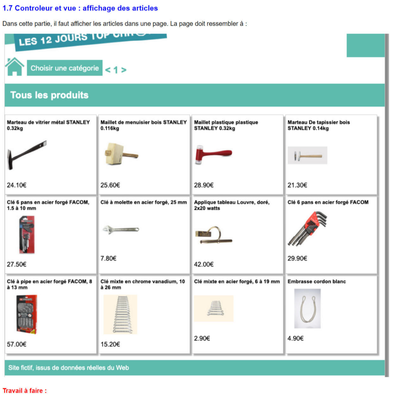
```
public static function readPage(int $page,int $pageSize): array
```

La notion de page est réalisée dans les requêtes SQL avec OFFSET et LIMIT. La valeur LIMIT correspond à la taille d'une page, et OFFSET au début de cette page. N'oubliez pas d'ordonner les résultats sur la valeur de la référence.

- Testez ces fonctions avec le test en ligne de commande `article.test.php`.

1.7 Controleur et vue : affichage des articles

Dans cette partie, il faut afficher les articles dans une page. Le page doit ressembler à :



Travail à faire :

Il y a deux tâches principales : compléter la vue `articles.view.php` et le controleur `afficherArticles.ctrl.php`. Conseil : développez les fonctionnalités une à une en les testant.

Voici la listes des fonctionalités à développer :

- Affichage de la liste des produits de la page courante : Cela consiste à récupérer la liste des produits et à passer cette liste à la vue. Les articles doivent être affichés avec le HTML suivant :

```
<article>
<h2>Marteau de vitrier métal STANLEY 0.32kg</h2>
<img src="https://www-info.iut2.univ-grenoble-alpes.fr/intranet/enseignements/ProgWeb/data/bricomachin/img/54331060.jpg"/>
<p>24.10€</p>
</article>
```

- Déplacement dans les pages : cela consiste à donner des valeurs correctes aux variables `$pagePrec` et `$pageSuiv`.

1.8 Selection d'une catégorie

Le bouton 'Choisir une catégorie' renvoit vers un nouveau couple de vue / controleur qui permet de descendre la hiérarchie des catégories à l'aide d'un formulaire et de boutons. Tant que la catégorie possède des sous catégories, il faut rester dans cette boucle de sélection. Dès qu'une catégorie terminale est choisie, il faut revenir à la page principale et activer le filtre des catégories. La vue de la sélection doit ressembler à :

Travail à faire :

Vous devez modifier le couple vue / controleur `categorie.view.php` / `choisirCategorie.ctrl.php` pour obtenir les fonctionalités suivantes :

- Un clic sur un bouton de catégorie fait afficher la liste des sous catégories et change le titre de la page qui devient le nom de la catégorie père. Par exemple si l'on clique sur "Décoration & Eclairage" on doit avoir l'affichage suivant :

- Un clic sur une catégorie qui n'a pas de sous catégorie fait revenir à l'affichage des articles avec un filtrage sur cette catégorie. Par exemple, un clic sur "Rideau" retourne à la page de l'affichage des articles avec le titre "Rideau" et un affichage des articles uniquement de la catégorie "Rideau".

- Les boutons de navigation doivent alors passer de pages en pages en gardant la catégorie. Par exemple pour les rideaux :