UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2023/2024
***Resp. :* C. Rabat**

Travaux dirigés n° 3

PHP et les bases de données

Exercice 1 (MyPDO)

Nous souhaitons écrire une classe permettant de gérer les connexions à la base de données d'un site en *MySQL/MariaDB*. L'URL du SGBD, le nom de la base de données, le login et le mot de passe utilisateur sont spécifiés dans un fichier PHP `DBConfig.php`.

1. Donnez le contenu du fichier `DBConfig.php` (aidez-vous des annexes).
2. Rappelez les précautions à prendre concernant ce fichier.
3. Écrivez la classe `MyPDO` contenant uniquement la méthode statique `getInstance` qui permet de créer ou retourner l'instance de l'objet PDO permettant d'accéder à la base de données.
4. Nous supposons l'existence d'une table `user` contenant entre autres le nom et le prénom. Donnez un script permettant d'utiliser la classe `MyPDO` pour afficher un utilisateur spécifique puis la liste de toutes les utilisateurs.

Exercice 2 (Retour sur les articles)

Nous reprenons les articles vus dans le TD précédent. Ils sont maintenant stockés dans une base de données et associés à un fournisseur (un nom, une adresse). De plus, ils sont classés dans une ou plusieurs catégories (un intitulé et une description) et il est possible de leur associer des mots-clefs (un simple intitulé).

1. Donnez le MCD de l'application.
2. Donnez le MLD et le MPD correspondants.
3. Donnez les modifications à apporter dans la classe `Item` qui nous permettra de représenter un enregistrement.
4. Écrivez la classe `ItemModel` correspondant à la classe CRUD (pour `Create`, `Read`, `Update` et `Delete`) de la classe `Item`. Elle contient également une méthode `fromArray` permettant de créer un objet `Item` à partir d'un tableau associatif.
   N.B. : utilisez la classe `MyPDO` de l'exercice précédent.

Exercice 3 (Publicité)

Nous souhaitons compléter notre site marchand que nous avions développé au TD précédent. En plus d'afficher la liste des articles, nous souhaitons en plus les mettre en avant un par un en haut de la page. Ils sont affichés dynamiquement (avec un changement toutes les 3 secondes, par exemple). Nous afficherons le titre, la description, le prix et le nom du fournisseur. Pour cela, nous utilisons AJaX et les éléments développés dans l'exercice précédent.

1. Indiquez tous les éléments à mettre en place pour une telle solution.
2. Donnez l'ordre d'appel et toutes les requêtes HTTP échangées sous la forme d'un diagramme d'échange.
3. Donnez le contenu du script PHP et de la partie *Javascript*.

Annexes - Manuel PHP

Classe **PDO** (version simplifiée) :

```
public __construct(string $dsn [, string $username [, string $passwd
                  [, array $options ]]])
public int exec(string $statement)
public string lastInsertId([ string $name = NULL ])
public PDOStatement prepare(string $statement
                           [, array $driver_options = array() ])
public PDOStatement query(string $statement)
```

Représente une connexion entre PHP et un serveur de base de données (SGBD). À noter que le paramètre `$dsn` du constructeur correspond à la chaîne suivante (pour une connexion à un SGBD *mySQL*) :

```
mysql:host=localhost;dbname=mabase;charset=utf8
```

Classiquement, les options utilisées (paramètre `$driver_options`) sont les suivantes :

```
$driverOptions = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
  ];
```

Classe **PDOStatement** (version simplifiée) :

```
public bool bindColumn(mixed $column, mixed &$param [, int $type
                      [, int $maxlen [, mixed $driverdata ]]])
public bool bindParam(mixed $parameter, mixed &$variable
                     [, int $data_type = PDO::PARAM_STR [, int $length
                     [, mixed $driver_options ]]])
public bool bindValue(mixed $parameter, mixed $value
                     [, int $data_type = PDO::PARAM_STR ])
public bool closeCursor(void)
public int columnCount(void)
public void debugDumpParams(void)
public bool execute([ array $input_parameters ])
public mixed fetch([ int $fetch_style
                   [, int $cursor_orientation = PDO::FETCH_ORI_NEXT
                   [, int $cursor_offset = 0 ]]])
public array fetchAll([ int $fetch_style [, mixed $fetch_argument
                     [, array $ctor_args = array() ]]])
public mixed fetchColumn([ int $column_number = 0 ])
public mixed fetchObject([ string $class_name = "stdClass" [, array $ctor_args ]])
public mixed getAttribute(int $attribute)
public array getColumnMeta(int $column)
public bool nextRowset(void)
public int rowCount(void)
public bool setAttribute(int $attribute, mixed $value)
public bool setFetchMode(int $mode)
```

Représente une requête préparée et, une fois exécutée, le jeu de résultats associé.