1 Quelques révisions SQL

Reprenons l'exercice en fin du cours précédent mais avec un choix technique plus propre, c'est-à-dire en utilisant SQL au lieu d'un fichier.

Exercice 1 : Faire une page web qui centralise un inventaire de commandes à faire, par exemple pour tout un service administratif. Chaque client (i.e. employé du service administratif en question) pouvant choisir dans une liste de produits donnée, donner le nombre d'exemplaire qu'il veut, puis valider son choix. Après une page de confirmation, sa commande est ajoutée dans l'inventaire des commandes à effectuer.

Faire également une page qui affiche proprement cet inventaire.

On va commencer par mettre en place la base de données SQL qui centralisera les informations :

- Si ce n'est déjà fait, créez un utilisateur qui ne soit pas `root` sous `phpMyAdmin`.
- Créez sous `phpMyAdmin` une table qui permette de mémoriser un inventaire similaire à celui de l'exercice (table appartenant à l'utilisateur précédent).
- Écrivez la requête SQL qui permet d'ajouter un produit dans l'offre (avec 0 exemplaires à commander à l'initialisation, naturellement)
  `INSERT INTO ... VALUES ...`
- Écrivez la requête SQL qui indique combien d'exemplaires d'un produit donné devraient être achétés maintenant
  `SELECT ... FROM ... WHERE ...`
- Écrivez la ou les requête(s) SQL qui ajoute(nt) `n` exemplaires d'un produit donné dans l'inventaire
  `UPDATE ... SET ...=(SELECT..FROM..WHERE..)+n WHERE ...`
- Écrivez la requête SQL qui supprime un produit de l'offre.
  `DELETE FROM ... WHERE ...`

2 Dialogues avec MySQL

On utilise MySQL sous PHP en quatre étapes :

1. se connecter au serveur MySQL,
2. lui envoyer des requêtes et récupérer les résultats,
3. exploiter les résultats sous une forme utilisable,
4. et enfin fermer la connexion.

Du fait qu'on ne sait jamais si le client va ou non abandonner sa session, il faut faire ces quatres étapes pour *chaque page* PHP. Il faut en effet s'assurer que l'on se déconnecte toujours de la base de données, même si le client ne poursuit pas sa session jusqu'au bout.

2.1 Pour se connecter :

On utilise la fonction `mysql_connect` :

```
$sql=mysql_connect($host,$user,$passwd);
```

Le résultat de `mysql_connect` est de type `resource` (comme lors de l'ouverture de fichier) et s'utilise de la même façon en le mémorisant dans une variable (`$sql` dans l'exemple), et de même, il vaut `0` si la connexion a échoué.

Ensuite, il est probable que l'utilisateur (`$user dans l'exemple`) possède plusieurs bases de données et il faut donc en choisir une pour les requêtes ultérieures :

```
mysql_select_db($nomDeBase,$sql);
```

qui retourne un booléen (`FALSE` si la base est inaccessible). De ce fait, si l'on veut obtenir un message d'erreur si la connexion ou la sélection échoue, on utilise l'astuce suivante :

```
mysql_select_db($nomDeBase,$sql) or exit("Base inaccessible");
```

Cette astuce est utile pour la mise au point mais il faudra sans doute faire un meilleur traitement en cas d'échec pour le logiciel final. De plus, Attention de ne pas laisser d'informations non souhaitée dans le message d'erreur puisque n'importe quel client peut a priori tomber dessus...

Nota : dans les nouvelles versions de PHP,

```
$sql=mysqli_connect($host,$user,$passwd,$nomDeBase);
```

fait les deux actions (connection et sélection de la base) en une seule fois. Le résultat de type `resource` vaut `0` si la connexion a échoué et on peut faire la même astuce puisque 0 est confondu avec `FALSE` en PHP.

Quelques conseils de bonne structuration du code :

- C'est une bonne habitude de *ne pas* définir les valeurs des paramètres `$host`, `$user`, `$passwd` et `$nomDeBase` dans les fichiers PHP qui font appel à `mysql_connect()` ou `mysqli_connect()`, mieux vaut les définir dans un autre fichier PHP qui affecte les variables globales, puis faire un `include_once()` de ce fichier.
- C'est généralement une bonne habitude de concevoir chacune des requêtes SQL préalablement, puis de la mémoriser dans une variable, par exemple `$req`, avant de la faire exécuter à MySQL.

2.2 Pour exécuter une requête et récupérer son résultat :

On utilise la fonction `mysql_query` qui exécute la requête qu'on lui donne en premier argument, dans la base qu'on lui donne en second argument :

```
$resultat=mysql_query($req,$sql);
```

La variable `$resultat` contient alors une ressource (type `resource`) qui permettra de lire la table qui est le résultat de la requête.

Nota : dans les nouvelles versions de PHP, on utilise `mysqli_query()` qui *inverse* l'ordre des arguments...

```
$resultat=mysqli_query($sql,$req);
```

2.3 Pour exploiter le résultat :

Le résultat d'une requête SQL de type `resource` n'est pas facile à utiliser tel quel. On peut le transférer dans un tableau, bien plus compatible avec le concept de *table* de SQL. La fonction `mysql_fetch_array` permet de lire *successivement* chacune des lignes de la table de résultats, c'est-à-dire chacun des `n`-uplets de la table. Le résultat de cette fonction est donc un tableau contenant les noms des colonnes comme éléments et la valeur de chacun de ces éléments est la valeur correspondante du `n`-uplet. Chaque nouvel appel de la fonction `mysql_fetch_array` passe au `n`-uplet suivant, de manière similaire à la lecture d'un fichier ligne à ligne.

Cela conduit généralement à une boucle while pour parcourir tout le tableau résultat de la requête :

```
while( $nuplet=mysql_fetch_array($resultat) )
  {
  ...
  }
```

On utilise ici encore le fait que `mysql_fetch_array` renvoit `0`, donc `FALSE`, lorsqu'il n'y a plus de `n`-uplet à lire dans la table.

Nota : dans les nouvelles versions de PHP, on utilise `mysqli_fetch_array` au lieu de `mysql_fetch_array`...

Par ailleurs, `mysql_num_rows($resultat)` fournit le nombre de `n`-uplets de la table résultat.

2.4 Pour fermer la connexion :

Il suffit d'invoquer

```
mysql_close($sql);
```

ou bien dans les nouvelles versions de PHP :

```
mysqli_close($sql);
```

Par ailleurs, `mysql_ping($sql)`, ou dans les nouvelles versions de PHP `mysqli_ping($sql)`, renvoit un booléen qui dit si la connexion est toujours active.

On peut maintenant finir l'exercice proprement...