Fonctions PHP pour mySQL

Les fonctions PHP pour MySQL commencent toujours par " **mysql_** ", en voici ci−dessous la liste exhaustive. En règle générale je n'utilise pas plus de 6 ou 7 fonctions, c'est d'ailleurs sur celles−ci que je mettrai l'accent plus bas dans cet exercice, ceci dit, je vous conseille tout de même les tester toutes.

Fonctions	Descriptions
mysql_affected_rows()	Retourne le nombre de rangée affectées par la dernière requête faite sur la base de données.
mysql_close()	Ferme la connexion à une base de données.
mysql_connect()	Établit une connexion vers la base de données spécifiée dans les arguments. Si cette base se trouve sur un port différent, faites suivre le nom de la machine par (**:**) puis le numéro du port (ex. **:8080**). Cette connexion est automatiquement fermée à la fin de l'exécution du script sauf si une fonction **mysql_close()** intervient auparavant.
mysql_create_db()	Permet de créer une nouvelle base de données.
mysql_data_seek()	Déplace le pointeur interne de la rangée du résultat vers la rangée spécifiée. Utilisez cette fonction avec **mysql_fetch_row()** pour passer à la rangée spécifiée et récupérer les données.
mysql_db_query()	Permet d'exécuter une requête **SQL** sur une ou plusieurs tables d'une base de données. **Attention cette fonction est obsolète, ne l'utilisez plus, utilisez mysql_query !**
mysql_drop_db()	Permet de supprimer une base de données. Dans ce cas toutes les données sont perdues.
mysql_errno()	Retourne le numéro de l'erreur générée lors de la dernière action sur la base de donnée.
mysql_error()	Retourne la description textuelle d'une erreur générée par la dernière action sur une base de données.
mysql_fetch_array()	Retourne un tableau qui représente tous les champs d'une rangée dans le résultat. Chaque appel récupère la prochaine rangée et ce jusqu'à ce qu'il n'y en ait plus. Chaque valeur de champ est stockée de deux façons: elle est indexée par un offset qui commence par '**0**', et indexée par le nom du champ.
mysql_fetch_assoc()	Comme mysql_fetch_array() mais indexée uniquement par le nom du champ. Dans la plupart des cas, utiliser cette fonction ou mysql_fetch_row() de préférence à mysql_fetch_array().
mysql_fetch_field()	Récupère l'information attachée à un champs du résultat. Ces champs sont numérotés à partir de zéro.
mysql_fetch_lengths()	Retourne un tableau d'une longueur spécifiée pour chaque champ de résultat.
mysql_fetch_object()	Cette fonction est semblable à **mysql_fetch_array()** et à **mysql_fetch_row()**. Mais à la place, elle retourne un objet. Chaque champ du résultat correspond à une propriété dans l'objet renvoyé. Chaque appel à **mysql_fetch_object()** retourne la rangée suivante, ou **False** s'il ne reste plus de rangée.
mysql_fetch_row()	Comme mysql_fetch_array() mais indexée uniquement par le numéro d'ordre du champ. C'est la méthode la plus rapide pour obtenir des résultats à partir d'une requête. Dans la plupart des cas, utiliser cette fonction ou mysql_fetch_assoc() de préférence à mysql_fetch_array().
mysql_field_flags()	Permet d'obtenir une description des options rattachées au champs spécifié.
mysql_field_len()	Retourne la longueur maximale du champ spécifié.
mysql_field_name()	Retourne le nom d'une colonne. L'argument champ correspond à un offset numéroté à partir de zéro.
mysql_field_seek()	Déplace le pointeur interne du champ vers le champs spécifié. Le prochain appel vers **mysql_field_seel()** retournera l'information de ce champs.
mysql_field_table()	Retourne le nom de la table pour le champ spécifié.
mysql_field_type()	Retourne le type d'un champ particulier dans le résultat obtenu.

mysql_free_result()	Libère la mémoire associée au résultat spécifié. Elle n'est toutefois pas strictement nécessaire, car cette mémoire est automatiquement vidée lorsqu'un script termine son exécution.
mysql_insert_id()	Après l'insertion d'un nouvel enregistrement avec un champ **auto_increment**, la fonction **mysql_insert_id()** retourne l'**ID** qui vient d'être affecté au nouvel enregistrement.
mysql_list_dbs()	Interroge le serveur pour obtenir une liste de bases de données. Elle retourne un pointeur de résultat qui pourra être exploité avec **mysql_fetch_row()** et d'autres fonctions similaires.
mysql_list_fields()	Retourne un pointeur de résultat correspondant à une requête sur une liste de champs pour la table spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.
mysql_list_tables()	Retourne le pointeur de résultat d'une liste de tables pour la base de données spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.
mysql_num_fields()	Retourne le nombre de champs dans un résultat.
mysql_num_rows()	Retourne le nombre de rangées dans un résultat. Anciennement mysql_numrows()
mysql_pconnect()	Cette fonction opère de la même manière que **mysql_connect()**, sauf que la connexion ne se referme pas à la fin de l'exécution du script sauf si un **mysql_close()** se trouve en fin de script.
mysql_query()	Permet d'exécuter une requête **SQL** sur une ou plusieurs tables d'une base de données. Si la requête exécute une instruction: **INSERT**, **DELETE** ou **UPDATE**, une valeur booléenne sera retournée (**0** ou **1**). Dans le cas d'une requête de type **SELECT**, vous obtiendrez un identifiant de résultat.
mysql_result()	Retourne la valeur du champ spécifié dans la rangée concernée. L'argument champ peut être un numéro et, dans ce cas, il sera considéré comme un champ offset. Il peut également désigner le nom de la colonne, avec éventuellement celui de la table. Enfin, il peut également renvoyer à un alias.
mysql_select_db()	Sélectionne la base de données par défaut.

Fonctions : mysql_connect(), _select_db() , _query(), _num_rows(), _close()

Ces fonctions sont le minimum que vous utiliserez à chaque fois que vous interrogerez une base de données MySQL, voyez le code ci−dessous (nous avons repris notre table clients_tbl de l'exercice n°11).

Code PHP

```php
<?php
$db = mysql_connect('sql.free.fr', 'login', 'password');  // 1
mysql_select_db('nom_de_la_base',$db);                    // 2
$req = mysql_query('SELECT * FROM clients_tbl');          // 3
$res = mysql_num_rows($req);                               // 4

echo 'Il y a '.$res.' enregistrement(s) dans la table Clients.';  // 5

mysql_close($db);  // 6
?>
```

Donne à l'écran

Il y a 5 enregistrement(s) dans la table Clients.

Explication du code ci−dessus :

1. On se connecte à la base de données :
   Host : Par exemple "sql.free.fr", vous pouvez également utiliser "localhost" par défaut.
   Login : Ensuite vous devez mettre le "login" pour accéder à la base (chez les hébergeurs gratuits c'est souvent le même login que l'accès FTP).
   Password : Et pour finir le "mot de passe", là aussi il s'agit très souvent du même password que l'accès FTP.

2. On sélectionne la base de données, en effet je vous rappelle que MySQL est un serveur de bases de données, donc il peut contenir plusieurs bases. Bien sûr dans votre cas si vous êtes chez un hébergeur gratuit, vous n'avez en général droit qu'à une seule base, mais MySQL ne le sait pas ;), il faut donc lui spécifier sur quelle base vous souhaitez vous connecter.
   Notez : Chez Free.fr et Nexen le nom de la base est souvent le même que le login de connection.

3. La fonction mysql_query() permet de passer une requête SQL vers la base de données, c'est évidement l'un des attraits intéressants de PHP (notez que nous initialisons au passage la variable $req qui contient la requête).

4. La fonction mysql_num_rows() permet de compter le nombre d'enregistrements que retourne la requête "$req" dans notre cas : 5 puisqu'il s'agit d'un simple "select " sur la table sans aucune condition, nous initialisons donc une variable $res qui contient : 5.

5. Il ne reste plus qu'à afficher le nombre de résultats avec un echo() de $res.

6. Et pour finir on referme la connexion − ouverte avec mysql_connect − avec la fonction mysql_close(). Cette fonction n'est pas vraiment obligatoire avec un mysql_connect(), car par défaut la connexion sera coupée automatiquement à la fin de l'exécution du script.