

Base de données et requêtes

Catégorie : Fiches techniques

Publié par [Fooups](#) le 16/04/2005

Si le module que vous souhaitez créer fait appel à des données stockées dans des tables de la Bdd, vous serez amené, dès le début de sa création, à mettre en place vos tables et à écrire les requêtes nécessaires à l'extraction, ou l'insertion, de ces données.

Ce document a pour but de vous expliquer les grandes lignes à suivre, le détail des méthodes utilisables se trouvant dans le [manuel du développeur](#).

1) Les tables

Nous supposons que vous êtes familier avec l'organisation des données dans les tables, les différents types de champs, les index, etc.

Si ce n'est pas le cas, approfondissez la question, car il est primordial de créer correctement les tables avant de coder, toute modification ultérieure risquant de se heurter sur de nombreuses portions de code.

Normalisez vos tables: il existe des règles précises relatives à la répartition des données dans les tables et le contenu des différents champs. Si vous ignorez ce que sont les formes normales d'une base, il existe une littérature abondante sur le net (ou en librairie) sur ces formes normales: mais rassurez-vous, cela n'a rien de complexe et peut-être vos tables sont déjà normalisées sans que vous le sachiez!

Envisagez la création d'une table 'categories' si votre application s'y prête (c'est souvent le cas): xoops dispose de différentes méthodes de la classe [xoopsTree](#) permettant de gérer les catégories et sous-catégories dans lesquelles vous aurez classé vos données.

Enfin, je ne saurais que trop vous conseiller de créer une classe dérivée de la classe [xoopsObject](#): vos requêtes seront 'concentrées' dans cette classe, ce qui permet de heurter les modifications de table beaucoup plus facilement, et vous ne travaillez plus avec des requêtes dans votre code, mais ferez appel aux méthodes getVar, setVar, initVar, etc.

2) Conventions

Respectez les conventions d'écriture, votre code n'en sera que plus lisible et compréhensible par vous-même, mais également par vos utilisateurs.

Certaines de ces conventions sont officielles, d'autres sont des conventions d'usage.

- Les instructions sql doivent être écrites en majuscules (SELECT, GROUP BY, etc.)
- Utilisez (de préférence) \$sql pour écrire la requête dans une chaîne de caractères.
- Utilisez (de préférence) \$result pour le résultat de la requête.
- Utilisez (de préférence) \$myrow pour parcourir les lignes du tableau \$result.

3) Requêtes

N'utilisez jamais directement les fonctions mysql_, n'utilisez que les méthodes de l'objet xoopsDB : [\\$xoopsDB->prefix\(\)](#), [\\$xoopsDB->query\(\)](#), etc.

* Pour vous affranchir des instructions de connexion à la Bdd.

* Pour bénéficier du mode debug sql.

* Pour une éventuelle compatibilité de xoops avec d'autres bases de données. Ecrivez votre requête dans une chaîne de caractères (\$sql) :

```
$sql = 'SELECT * FROM '.$xoopsDB->prefix('mymodule_table1').' WHERE champ1 = '.$variable.'  
AND champ2 = '.$variable2';
```

Cette façon de procéder présente plusieurs avantages:

- une meilleure lisibilité du code
- la possibilité d'afficher votre requête avec un `echo $sql` pendant la phase de mise au point
- la possibilité de concaténer votre requête suivant le résultat d'un test

```
$sql = "SELECT cat_id, count(*) FROM ".$xoopsDB->prefix("mymodule_table1 ");  
if ( !empty($order)) $sql .= " ORDER BY ".$order;  
else $sql .= " ORDER BY ".$weight;
```

Puis exécutez la requête

```
$result = $xoopsDB->query($sql)
```

Pendant la phase de développement, mettez un test pour vérifier son exécution

```
if ( !$result ) { die( 'Erreur sql : '.$sql ); }
```

L'astuce ci-dessus présente plusieurs avantages si votre requête ne s'exécute pas :

- votre requête s'affichera et vous permettra de voir s'il manque un élément (ex WHERE champ1= AND.)
- vous pourrez faire un copier /coller de cette requête et l'exécuter avec phpmyadmin pour voir le résultat.

Dans votre code définitif, ne laissez pas ce test, mais supprimez le, ou mieux modifiez le pour faire une redirection sur un message d'erreur, par exemple .:

```
if (!$result) $msg = _MD_ERROR_INSERT;  
else $msg = _MD_INSERT_OK;  
redirect_header("index.php", 3, $messagesent);
```

Prévoyez également le cas où votre requête SELECT ne retourne aucun résultat, qui, s'il n'est pas traité, peut aboutir à une page blanche

4) Requetes SELECT

Il existe de nombreuses méthodes pour exploiter les résultats.

Vous trouverez différents exemples à la page [xoopsDB](#) du manuel du développeur.

N'oubliez pas que les résultats sont obtenus sous forme d'un tableau qu'il faudra parcourir pour en extraire les résultats. Une bonne compréhension de la notion des tableaux en Php vous est donc recommandée !

5) Requetes INSERT et UPDATE

Avant d'insérer des données dans une table, et particulièrement si elles proviennent d'un formulaire, il est impératif de contrôler la validité de ces données.

Exemple pour une donnée numérique: `$number = intval($number);`

Pour du texte, utilisez les différentes méthodes de la classe MyTextSanitizer.

Enfin, n'oubliez pas que votre texte, pour être inséré, doit être placé entre simples quotes.

Vous pourrez utiliser pour cela la méthode `$xoopsDB->quoteString($mytext)`

6) queryF

J'avoue ne pas tout connaître à propos des requêtes et de queryF, et ne vous livrerai donc que ce que je sais :

- il arrive que les requêtes INSERT, UPDATE ou DELETE ne s'exécutent pas en utilisant la méthode `$xoopsDB->query($sql)`.
- on peut utiliser dans ce cas la méthode `$xoopsDB->queryF($sql)`, le F signifiant force.
- attention, l'utilisation non contrôlée de cette méthode peut entraîner une faille de sécurité, il est impératif de vérifier les données avant insertion. **7) Déclaration**

Si vous utilisez les méthodes de la classe `xoopsDB` dans une fonction, n'oubliez pas de la déclarer globale:

```
function my_function($param) {
global $xoopsDB;
```

8) Mise au point

Vous disposez pour la mise au point de vos requêtes de la console de débogage que vous activez dans

Administration>>Système>>Préférences>>Paramètres généraux>>Mode de mise au point>>MySQL/BlocsPersonnellement, je préfère rester en mode Mise au point Php et insérer des `echo $sql.`

À la suite des mes requêtes, plutôt que d'avoir cette fenêtre qui surgit à chaque page ...

Par contre, cette console de débogage est extrêmement utile pour connaître toutes les requêtes qui s'exécutent à l'appel d'une page, vous permettant ainsi, entre autres, de faire la chasse aux requêtes inutiles !