# db_include.php

Versions 1.2.2 20070506 et 2.0.0 20110312
(c) 2004~2011 Sébastien Adam - sebastien.adam.webdev@gmail.com - http://www.sebastienadam.be/

## Table des matières



## Introduction.

Les scripts PHP doivent souvent accéder à des bases de données. Le serveur de base de données le plus répandu est sans conteste MySQL. J'ai développé cette librairie afin de facilité les accès aux bases de données. De plus si un jour vous décidez de passer sur un autre serveur de base de données, il ne faudra pas réécrire tout votre programme. Il suffit de changer cette librairie.

## Pré-requis.

Pour fonctionner, cette librairie doit être installé sur un serveur "web" pouvant interpréter les scripts PHP et possédant accès à un serveur de base de données MySQL.

## Configuration.

Aucune configuration préalable n'est nécessaire.

# Utilisation (Version 2.X.X).

## Valeurs de retour

Les valeurs de retours des différentes méthodes renseignent généralement du succès où de l'échec de l'exécution de ces dernières. Toutefois, des informations plus détaillés sur ce qui s'est passé son accessible au travers de deux méthodes :

- `db::getError([widthSQL])` renseigne le message d'erreur de la fonction, éventuellement suivit du message d'erreur retourné par le serveur de bases de données.
- `db::getErrno()` renseigne une valeur numérique correspondant au type d'erreur rencontrée.

## Chargement de la librairie

Pour charger la librairie, il suffit de l'inclure dans votre script PHP.

```
<?php
require_once("db_include.php");
?>
```

Il est fortement conseillé d'utiliser la fonction "`require_once()`" plutôt que "`require()`" afin d'éviter tout problème de redéfinition de la classe.

## Connexion à une base de données

La connexion à une base de données se fait en créant un objet de la classe "`db`". Nous pouvons soit passer les paramètres de connexion au constructeur, soit utiliser la méthode `db::connect()`. L'objet va d'abord se connecter au serveur de bases de données puis à la base de données elle-même.

```
<?php
$connid = new db("<server>", "<user>", "<password>", "<db>");
?>
```

```
<?php
$connid = new db();
$connid->connect("<server>", "<user>", "<password>", "<db>");
?>
```

- `<server>` représente le nom du serveur de base de données.
- `<user>` représente le nom d'utilisateur utilisé pour s'identifier sur le serveur de bases de données.
- `<password>` représente le mot de passe de l'utilisateur.
- `<db>` représente le nom de la base de données à utiliser.

En cas de succès, la méthode `db::connect()` va retourner la valeur booléenne "`true`". Sinon, elle retournera la valeur booléenne "`false`".

## Exécution d'une requête

L'exécution d'une requête de fait grâce à la méthode "`db::query("<query>")`".

```
<?php
$connid->query("<query>");
?>
```

- `<query>` représente la requête à exécuter.

En cas de succès, la méthode va retourner le résultat de la requête. Ce résultat dépend du type de requête. En cas de requête de type "select", la méthode va retourner l'entièreté des données dans un tableau. Le premier indice du tableau est numérique et représente le numéro de la ligne. Chaque ligne est un tableau dont les indices sont les intitulés des colonnes. Si cette technique facilite, pour moi, le traitement des données, elle nécessite beaucoup de ressources mémoire au niveau du serveur en cas de traitement de table de grande taille. Sinon, la méthode va retourner la valeur booléenne "`true`" en cas de succès.

En cas d'échec, la méthode va retourner la valeur booléenne "`false`".

## Validation des opérations

Normalement, les méthodes retournent les valeurs booléennes "`true`" ou "`false`" suivant que l'action s'est correctement effectuée ou non. Mais parfois, aucune valeur de contrôle n'est retournée. Pour tester si une action s'est déroulée correctement, la méthode "`db::succeeded()`" peut être utilisée.

```
<?php
$connid->succeeded();
?>
```

Cette méthode retourne les valeurs booléennes "`true`" ou "`false`" suivant que la dernière action s'est correctement effectuée ou non.

Si une erreur survient, les méthodes "`db::getError([<widthSQL>])`" ou "`db::getErrno()`" peuvent être utilisées pour obtenir plus de détails su ce qui s'est passé.

```
<?php
$connid->getError([<widthSQL>]);
$connid->getErrno();
?>
```

- `<widthSQL>` est un paramètre booléen optionnel déterminant s'il faut retourner le message d'erreur généré par le serveur MySQL en même temps que celui généré par la classe ou non. Sans paramètre, "`db::getError([<widthSQL>])`" ne retourne pas le message généré par le serveur MySQL.

"`db::getErrno()`" retourne une valeur numérique. Cette valeur est généralement celle fournie par le serveur MySQL. Dans certains cas, c'est la classe elle-même qui retourne une valeur parmi les valeurs suivantes :

- `0` (zéro) : aucune erreur n'est survenue.
- `db::E_ALRCONN` : une tentative de connexion a été effectuée alors qu'une connexion est déjà active.
- `db::E_NOCONN` : une requête a été tentée alors qu'aucune connexion n'est active.
- `db::E_NOIN` : les données d'entrées fournies à la dernière méthode exécutée n'étaient pas suffisantes.

## Comptage du nombre de lignes affectées par une requête

Pour déterminer le nombre de lignes affectées par une requête, il suffit d'utiliser la méthode "`db::affected_rows()`".

```
<?php
db->affected_rows();
?>
```

La fonction retourne le nombre de lignes affectées par la dernière requête.

## Clôture de la connexion à base de données

La clôture de la connexion à une base de données est réalisée au moyen la méthode "`db::close()`"

```
<?php
db->close();
?>
```

En cas de succès, la fonction retourne la valeur booléenne "`true`". Sinon elle retourne la valeur booléenne "`false`".

## Exemple d'utilisation

Voici un exemple de code qui permet d'afficher le contenu d'une table :

```
<?php
$server   = "myserver";
$user     = "mylogin";
$password = "mypassord";
$db       = "mydb";
$DBH = new db($server, $user, $password, $db);
if($DBH->succeeded())
  {
  $query = "select * from matable";
  $result = $DBH->query($query);
  if($DBH->succeeded())
    {
    $heads = array_keys($result[0]);
    echo "<table>\n<thead>\n";
    echo "<tr>";
    foreach($heads as $head)
      echo "<th>$head</th>";
    echo "</tr>\n</thead>\n<tbody>\n";
    foreach($result as $line)
      {
      echo "<tr>";
      foreach($line as $col)
        echo "<td>$col</td>";
      echo "</tr>\n";
      }
    echo "</tbody>\n</table>\n";
    }
  else
    {
    echo "<p>Echec de l'exécution de la requête. Raison de l'échec :</p>\n";
    echo "<pre>".$DBH->getError(true)."</pre>\n";
    }
  $DBH->close();
  }
else
  {
    echo "<p>Echec de la connexion à la base de données. Raison de
l'échec:</p>\n";
  echo "<pre>".$DBH->getError(true)."</pre>\n";
  }
?>
```

# Utilisation (version 1.X.X)

## *Valeurs de retour*

Les valeurs de retours des différentes fonctions renseignent généralement du succès où de l'échec de l'exécution de ces dernières. Toutefois, il des informations plus détaillés sur ce qui s'est passé sont sauvegardées dans deux variables :

- `$cfg["error"]` renseigne le message d'erreur de la fonction, éventuellement suivit du message d'erreur retourné par le serveur de bases de données.
- `$cfg["errno"]` renseigne une valeur numérique correspondant au type d'erreur rencontrée.

## *Chargement de la librairie*

Pour charger la librairie, il suffit de l'inclure dans votre script PHP.

```
<?php
require_once("db_include.php");
?>
```

Bien que la librairie gère les tentatives de chargements multiples, il est conseillé d'utiliser la fonction "`require_once()`" plutôt que "`require()`".

## *Connexion à une base de données*

La connexion à une base de données se fait grâce à la fonction "`db_connect()`". La fonction va d'abord se connecter au serveur de bases de données puis à la base de données elle-même.

```
<?php
$connid = db_connect("<server>", "<user>", "<password>", "<db>");
?>
```

- `<server>` représente le nom du serveur de base de données.
- `<user>` représente le nom d'utilisateur utilisé pour s'identifier sur le serveur de bases de données.
- `<password>` représente le mot de passe de l'utilisateur.
- `<db>` représente le nom de la base de données à utiliser.

En cas de succès, la fonction va retourner l'identifiant de connexion à la base de données.

En cas d'échec, la fonction va retourner la valeur booléenne "`false`". Les valeurs de `$cfg["errno"]` peuvent être interprétées comme suit :

- `dbinc_E_FUNC_NOIN` indique qu'un des paramètres est manquant.
- `dbinc_E_FUNC_CONN` indique que la connexion au serveur de bases de données a échoué.
- `dbinc_E_FUNC_DBSE`L indique que l'identification sur le serveur à réussi, mais la sélection de la base de données a échoué.

La valeur de `$cfg["error"]` contient une explication plus détaillée sur ce qu'il s'est passé.

## *Exécution d'une requête*

L'exécution d'une requête de fait grâce à la fonction "`db_query()`".

```
<?php
db_query("<query>", "<connid>");
?>
```

- `<query>` représente la requête à exécuter.
- `<connid>` représente l'identifiant de connexion à la base de données retourné par la fonction `db_connect()`.

En cas de succès, la fonction va retourner le résultat de la requête. Ce résultat dépend du type de requête. En cas de requête de type "select", la fonction va retourner l'entièreté des données dans un tableau. Le premier indice du tableau est numérique et représente le numéro de la ligne. Chaque ligne est un tableau dont les indices sont les intitulés des colonnes. Si cette technique facilite, pour moi, le traitement des données, elle nécessite beaucoup de ressources mémoire au niveau du serveur en cas de traitement de table de grande taille. Sinon, la fonction va retourner la valeur booléenne "`true`" en cas de succès.

En cas d'échec, la fonction va retourner la valeur booléenne "`false`". Les valeurs de `$cfg["errno"]` peuvent être interprétées comme suit :

- `dbinc_E_FUNC_NOIN` indique qu'un des paramètres est manquant.
- `dbinc_E_FUNC_SQLFAIL` indique que l'exécution de la requête a échoué. Cela peut provenir d'une erreur de syntaxe ou d'une violation d'une contrainte. Pour plus de détails sur ce qui s'est passé, il faut voir le contenu de la variable `$cfg["error"]`.
- `dbinc_E_FUNC_SQLNORES` indique que la requête n'a retourné aucune valeur.

La valeur de `$cfg["error"]` contient une explication plus détaillée sur ce qu'il s'est passé.

## Clôture de la connexion à base de données

La clôture de la connexion à une base de données se fait grâce à la fonction "`db_close()`"

```
<?php
db_close("<connid>");
?>
```

- `<connid>` représente l'identifiant de connexion à la base de données retourné par la fonction `db_connect()`.

En cas de succès, la fonction retourne la valeur booléenne "`true`". Sinon elle retourne la valeur booléenne "`false`". Les variables `$cfg["error"]` et `$cfg["errno"]` ne sont pas affectées par cette fonction.

## Comptage du nombre de lignes affectées par une requête

Pour déterminer le nombre de lignes affectées par une requête, il suffit d'utiliser la fonction "`db_affected_rows()`".

```
<?php
db_affected_rows("<connid>");
?>
```

- `<connid>` représente l'identifiant de connexion à la base de données retourné par la fonction `db_connect()`.

La fonction retourne le nombre de lignes affectées par la dernière requête.

## Exemple d'utilisation

Voici un exemple de code qui permet d'afficher le contenu d'une table:

```
<?php
$server   = "myserver";
$user     = "mylogin";
$password = "mypassord";
$db       = "mydb";
if($DB = db_connect($server, $user, $password, $db))
  {
  $query = "select * from matable";
  if($result = db_query($query, $DB))
    {
    $heads = array_keys($result[0]);
    echo "<table>\n<thead>\n";
    echo "<tr>";
    foreach($heads as $head)
      echo "<th>$head</th>";
    echo "</tr>\n</thead>\n<tbody>\n";
    foreach($result as $line)
      {
      echo "<tr>";
      foreach($line as $col)
        echo "<td>$col</td>";
      echo "</tr>\n";
      }
    echo "</tbody>\n</table>\n";
    }
  else
    {
    echo "<p>Echec de l'exécution de la requête. Raison de l'échec:</p>\n";
    echo "<pre>".$cfg["error"]."</pre>\n";
    }
  db_close($DB);
  }
else
  {
     echo  "<p>Echec  de  la  connexion  à  la  base  de  données.  Raison  de
l'échec:</p>\n";
  echo "<pre>".$cfg["error"]."</pre>\n";
  }
?>
```