Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

MySQL Functions (PDO_MYSQL) ¶

Introduction ¶

PDO_MYSQL is a driver that implements the PHP Data Objects (PDO) interface to enable access from PHP to MySQL 3.x, 4.x and 5.x databases.

PDO_MYSQL will take advantage of native prepared statement support present in MySQL 4.1 and higher. If you're using an older version of the mysql client libraries, PDO will emulate them for you.

Warning

Beware: Some MySQL table types (storage engines) do not support transactions. When writing transactional database code using a table type that does not support transactions, MySQL will pretend that a transaction was initiated successfully. In addition, any DDL queries issued will implicitly commit any pending transactions.

Installation ¶

The common Unix distributions include binary versions of PHP that can be installed. Although these binary versions are typically built with support for the MySQL extensions, the extension libraries themselves may need to be installed using an additional package. Check the package manager than comes with your chosen distribution for availability.

For example, on Ubuntu the *php5-mysql* package installs the ext/mysql, ext/mysqli, and PDO_MYSQL PHP extensions. On CentOS, the *php-mysql* package also installs these three PHP extensions.

Alternatively, you can compile this extension yourself. Building PHP from source allows you to specify the MySQL extensions you want to use, as well as your choice of client library for each extension.

When compiling, use **--with-pdo-mysql[=DIR]** to install the PDO MySQL extension, where the optional *[=DIR]* is the MySQL base library. As of PHP 5.4, mysqlnd is the default library. For details about choosing a

library, see Choosing a MySQL library.

Optionally, the **--with-mysql-sock[=DIR]** sets to location to the MySQL unix socket pointer for all MySQL extensions, including PDO_MYSQL. If unspecified, the default locations are searched.

Optionally, the **--with-zlib-dir[=DIR]** is used to set the path to the libz install prefix.

```
$ ./configure --with-pdo-mysql --with-mysql-sock=/var/mysql/mysql.sock
```

SSL support is enabled using the appropriate PDO_MySQL constants, which is equivalent to calling the » MySQL C API function mysql_ssl_set(). Also, SSL cannot be enabled with **PDO::setAttribute** because the connection already exists. See also the MySQL documentation about » connecting to MySQL with SSL.

Changelog

Version	Description
5.4.0	mysqlnd became the default MySQL library when compiling PDO_MYSQL. Previously, libmysqlclient was the default MySQL library.
5.4.0	MySQL client libraries 4.1 and below are no longer supported.
5.3.9	Added SSL support with mysqlnd and OpenSSL.
5.3.7	Added SSL support with libmysqlclient and OpenSSL.

Predefined Constants ¶

The constants below are defined by this driver, and will only be available when the extension has been either compiled into PHP or dynamically loaded at runtime. In addition, these driver-specific constants should only be used if you are using this driver. Using driver-specific attributes with another driver may result in unexpected behaviour. PDO::getAttribute() may be used to obtain the **PDO_ATTR_DRIVER_NAME** attribute to check the driver, if your code can run against multiple drivers.

PDO::MYSQL_ATTR_USE_BUFFERED_QUERY (integer)

If this attribute is set to **TRUE** on a PDOStatement, the MySQL driver will use the buffered versions of the MySQL API. If you're writing portable code, you should use PDOStatement::fetchAll() instead.

Example #1 Forcing queries to be buffered in mysql

```
<?php
if ($db->getAttribute(PDO::ATTR_DRIVER_NAME) == 'mysql') {
    $stmt = $db->prepare('select * from foo',
        array(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY => true));
} else {
    die("my application only works with mysql; I should use \$stmt->fetchAll() instead");
}
?>
```

PDO::MYSQL_ATTR_LOCAL_INFILE (integer)

Enable *LOAD LOCAL INFILE*.

Note, this constant can only be used in the `driver_options` array when constructing a new database handle.

PDO::MYSQL_ATTR_INIT_COMMAND (integer)

Command to execute when connecting to the MySQL server. Will automatically be re-executed when reconnecting.

Note, this constant can only be used in the `driver_options` array when constructing a new database handle.

PDO::MYSQL_ATTR_READ_DEFAULT_FILE (integer)

Read options from the named option file instead of from *my.cnf*. This option is not available if mysqlnd is used, because mysqlnd does not read the mysql configuration files.

PDO::MYSQL_ATTR_READ_DEFAULT_GROUP (integer)

Read options from the named group from *my.cnf* or the file specified with **MYSQL_READ_DEFAULT_FILE**. This option is not available if mysqlnd is used, because mysqlnd does not read the mysql configuration files.

PDO::MYSQL_ATTR_MAX_BUFFER_SIZE (integer)

Maximum buffer size. Defaults to 1 MiB. This constant is not supported when compiled against mysqlnd.

PDO::MYSQL_ATTR_DIRECT_QUERY (integer)

Perform direct queries, don't use prepared statements.

PDO::MYSQL_ATTR_FOUND_ROWS (integer)

Return the number of found (matched) rows, not the number of changed rows.

PDO::MYSQL_ATTR_IGNORE_SPACE (integer)

Permit spaces after function names. Makes all functions names reserved words.

PDO::MYSQL_ATTR_COMPRESS (integer)

Enable network communication compression. This is also supported when compiled against mysqlnd as of PHP 5.3.11.

PDO::MYSQL_ATTR_SSL_CA (integer)

The file path to the SSL certificate authority.

This exists as of PHP 5.3.7.

PDO::MYSQL_ATTR_SSL_CAPATH (integer)

The file path to the directory that contains the trusted SSL CA certificates, which are stored in PEM format.

This exists as of PHP 5.3.7.

PDO::MYSQL_ATTR_SSL_CERT (integer)

The file path to the SSL certificate.

This exists as of PHP 5.3.7.

PDO::MYSQL_ATTR_SSL_CIPHER (integer)

A list of one or more permissible ciphers to use for SSL encryption, in a format understood by OpenSSL. For example: *DHE-RSA-AES256-SHA:AES128-SHA*

This exists as of PHP 5.3.7.

PDO::MYSQL_ATTR_SSL_KEY (integer)

The file path to the SSL key.

This exists as of PHP 5.3.7.

PDO::MYSQL_ATTR_MULTI_STATEMENTS (integer)

Disables multi query execution in both PDO::prepare() and PDO::query() when set to **FALSE**.

Note, this constant can only be used in the `driver_options` array when constructing a new database handle.

This exists as of PHP 5.5.21 and PHP 5.6.5.

Runtime Configuration ¶

The behaviour of these functions is affected by settings in *php.ini*.

PDO_MYSQL Configuration Options

Name	Default	Changeable
pdo_mysql.default_socket	"/tmp/mysql.sock"	PHP_INI_SYSTEM
pdo_mysql.debug	NULL	PHP_INI_SYSTEM

For further details and definitions of the PHP_INI_* modes, see the Where a configuration setting may be set.

Here's a short explanation of the configuration directives.

`pdo_mysql.default_socket` string

Sets a Unix domain socket. This value can either be set at compile time if a domain socket is found at configure. This ini setting is Unix only.

`pdo_mysql.debug` boolean

Enables debugging for PDO_MYSQL. This setting is only available when PDO_MYSQL is compiled against mysqlnd and in PDO debug mode.

Table of Contents ¶

add a note

User Contributed Notes 12 notes

up
down
10

curt at webmasterbond dot com ¶
5 years ago
Today's PHP snapshot now has SSL support for PDO. Follow the directions here (http://dev.mysql.com/doc/refman/5.0/en/secure-create-certs.html) to set up MySQL and then use the following connection options:

```
<?php
$pdo = new PDO(
    'mysql:host=hostname;dbname=ssldb',
    'username',
    'password',
    array(
        PDO::MYSQL_ATTR_SSL_KEY    =>'/path/to/client-key.pem',
        PDO::MYSQL_ATTR_SSL_CERT=>'/path/to/client-cert.pem',
        PDO::MYSQL_ATTR_SSL_CA    =>'/path/to/ca-cert.pem'
    )
);
?>
```

up
down
6
brian at diamondsea dot com ¶
8 years ago
SQLSTATE[HY000]: General error: 2014 Cannot execute queries while other unbuffered queries are active. ...

After spending hours trying to track down why we were getting this error on a new server, after the same code ran fine on other servers, we found the problem to be an old MySQL _client_ library running on our web server, and a latest-version MySQL _server_ running on the database server's box.

Upgraded the MySQL client on the web server to the current revision and the problem went away.
up
down
6
davey at php dot net ¶
9 years ago
To use "PDO::MYSQL_ATTR_USE_BUFFERED_QUERY" you should call PDO::setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);

It will not work when passed into PDO::prepare()
up
down
0
proyecto2016 ¶
1 month ago
A way to doing a PDORepository

```
<?php
    abstract class PDORepository{
        const USERNAME="root";
        const PASSWORD="";
        const HOST="localhost";
        const DB="parcial";
```

```
        private function getConnection(){
            $username = self::USERNAME;
            $password = self::PASSWORD;
            $host = self::HOST;
            $db = self::DB;
            $connection = new PDO("mysql:dbname=$db;host=$host", $username, $password);
            return $connection;
        }
        protected function queryList($sql, $args){
            $connection = $this->getConnection();
            $stmt = $connection->prepare($sql);
            $stmt->execute($args);
            return $stmt;
        }
    }

?>
```

up
down
0

miller_kurt_e at yahoo dot com ¶
8 years ago

SQLSTATE[HY000]: General error: 2014 Cannot execute queries while other unbuffered queries are active. ...

This one can be a royal pain to deal with. Never stack statements to be executed in one go. Nobody ever mentions this possibility in all the posts I've seen dealing with this error.

This example is a Zend Framework example but the theory is the same.

As in:

```
<?php
$sql = <<<____SQL
     CREATE TABLE IF NOT EXISTS `ticket_hist` (
       `tid` int(11) NOT NULL,
       `trqform` varchar(40) NOT NULL,
       `trsform` varchar(40) NOT NULL,
       `tgen` datetime NOT NULL,
       `tterm` datetime,
       `tstatus` tinyint(1) NOT NULL
     ) ENGINE=ARCHIVE COMMENT='ticket archive';
     CREATE TABLE IF NOT EXISTS `request_hist` (
       `rqid` int(11) NOT NULL,
       `rqtid` int(11) NOT NULL,
       `rqsid` int(11) NOT NULL,
       `rqdate` datetime NOT NULL,
       `rqcode` tinyint(1) NOT NULL,
       `rssid` int(11) NOT NULL,
       `rsdate` datetime,
       `rscode` tinyint(1)
     ) ENGINE=ARCHIVE COMMENT='request archive';
     CREATE TABLE IF NOT EXISTS `relay_hist` (
       `rqid` int(5) NOT NULL,
```

```
        `sdesc` varchar(40) NOT NULL,
        `rqemail` varchar(40) NOT NULL,
        `sid` int(11) NOT NULL,
        `rlsid` int(11) NOT NULL,
        `dcode` varchar(5) NOT NULL
      ) ENGINE=ARCHIVE COMMENT='relay archive';
____SQL;
$result = $this->db->getConnection()->exec($sql);
?>
```

This will run fine but PDO will balk with the 'unbuffered' error if you follow this with another query.

Instead do:

```
<?php
$sql = <<<____SQL
     CREATE TABLE IF NOT EXISTS `ticket_hist` (
       `tid` int(11) NOT NULL,
       `trqform` varchar(40) NOT NULL,
       `trsform` varchar(40) NOT NULL,
       `tgen` datetime NOT NULL,
       `tterm` datetime,
       `tstatus` tinyint(1) NOT NULL
     ) ENGINE=ARCHIVE COMMENT='ticket archive';
____SQL;
$result = $this->db->getConnection()->exec($sql);

$sql = <<<____SQL
     CREATE TABLE IF NOT EXISTS `request_hist` (
       `rqid` int(11) NOT NULL,
       `rqtid` int(11) NOT NULL,
       `rqsid` int(11) NOT NULL,
       `rqdate` datetime NOT NULL,
       `rqcode` tinyint(1) NOT NULL,
       `rssid` int(11) NOT NULL,
       `rsdate` datetime,
       `rscode` tinyint(1)
     ) ENGINE=ARCHIVE COMMENT='request archive';
____SQL;
$result = $this->db->getConnection()->exec($sql);

$sql = <<<____SQL
     CREATE TABLE IF NOT EXISTS `relay_hist` (
       `rqid` int(5) NOT NULL,
       `sdesc` varchar(40) NOT NULL,
       `rqemail` varchar(40) NOT NULL,
       `sid` int(11) NOT NULL,
       `rlsid` int(11) NOT NULL,
       `dcode` varchar(5) NOT NULL
     ) ENGINE=ARCHIVE COMMENT='relay archive';
____SQL;
$result = $this->db->getConnection()->exec($sql);
?>
```

Chopping it into individual queries fixes the problem.

up
down
-1

dibakar at talash dot net ¶

10 years ago

PDO is much better option for calling procedures, views or triggers of mysql 5.x versions from PHP instead of using mysqli extension. Following is a simple demo script which can help anybody on how to call and use mysql procedures through php

```
try {
        $dbh = new PDO('mysql:host=xxx;port=xxx;dbname=xxx', 'xxx', 'xxx', array(
PDO::ATTR_PERSISTENT => false));

        $stmt = $dbh->prepare("CALL getname()");

        // call the stored procedure
        $stmt->execute();

        echo "<B>outputting...</B><BR>";
        while ($rs = $stmt->fetch(PDO::FETCH_OBJ)) {
            echo "output: ".$rs->name."<BR>";
        }
        echo "<BR><B>".date("r")."</B>";

    } catch (PDOException $e) {
        print "Error!: " . $e->getMessage() . "<br/>";
        die();
    }
```

up
down
-1

rmottey at gmail dot com ¶

9 years ago

I have been getting the error below when performing multiple queries within a single page.

Setting the attribute below did not seem to work for me.

So building on previous example i am initilizing my stmt variable on every query and a fetch all into an array. Seems to be working for me.

Error:
PDO Error 1.1: Array ([0] => xxx[1] => yyy[2] => Lost connection to MySQL server during query)

Fix:

(PDO::setAttribute("PDO::MYSQL_ATTR_USE_BUFFERED_QUERY", true);)

```
<?

try {
        $dbh = new PDO('mysql:host=xxx;port=xxx;dbname=xxx', 'xxx', 'xxx', array(
PDO::ATTR_PERSISTENT => false));
```

```
$stmt = $dbh->prepare("CALL getname()");

        // call the stored procedure
        $stmt->execute();
      // fetch all rows into an array.
       $rows = $stmt->fetchAll();
       foreach ($rows as $rs)
    {
          $id = $rs['id'];
         }
//initilise the statement
unset($stmt);
$stmt = $dbh->prepare("call secondprocedure(?);");
$stmt->bindValue(1, $id);
if ( ! $stmt->execute() )
{
    echo "PDO Error 1.1:\n";
    print_r($stmt->errorInfo());
    exit;
}
unset($stmt);
} catch (PDOException $e) {
        print "Error!: " . $e->getMessage() . "<br/>";
        die();
    }
?>
```

up
down
-2

Gerald Schneider ¶

3 years ago

This page suggests that the constant PDO::MYSQL_ATTR_FOUND_ROWS was always available (no note "exists as of X.X"), but I found the constant missing on an installation with PHP 5.2. After switching the PHP version to 5.3.27 on the webspace the constant was available.

up
down
-2

georgy dot garnov at gmail dot com ¶

2 years ago

PDO::MYSQL_ATTR_LOCAL_INFILE - will not work if you have
open_basedir in your php.ini

up
down
-8

david at manifestwebdesign dot com ¶

5 years ago

The SSL options are silently ignored in PHP 5.3.8, see https://bugs.php.net/bug.php?id=55870
Looks like it's addressed upstream, I just want to save others the hour and a half I just wasted :)

up
down
-8

info at westgatesearch dot com ¶

2 years ago

```
Here is a real world example of a PHP PDO MySQL Data Entry App with SQL Insert and redirect to a
report to show the data "just entered" as auto-generated by an expert system, WizzyWeb

$handler = new PDO('mysql:host=localhost;dbname=db', 'username', 'password');

$sql = "INSERT INTO Employees (EmployeeFirstName, EmployeeLastName, EmployeeOffice,
EmployeeDepartment, EmployeeEmailAddress, EmployeeExtension, EmployeeTitle) VALUES
(:EmployeeFirstName, :EmployeeLastName, :EmployeeOffice, :EmployeeDepartment, :EmployeeEmailAddress,
:EmployeeExtension, :EmployeeTitle)";

$query = $handler->prepare($sql);

$query->execute(array(
':EmployeeFirstName' => $EmployeeFirstName,
':EmployeeLastName' => $EmployeeLastName,
':EmployeeOffice' => $EmployeeOffice,
':EmployeeDepartment' => $EmployeeDepartment,
':EmployeeEmailAddress' => $EmployeeEmailAddress,
':EmployeeExtension' => $EmployeeExtension,
':EmployeeTitle' => $EmployeeTitle
));

// You can uncomment this code to see which PDO drivers are available and troubleshoot the PDO
database connection
//try {
//   print_r(PDO::getAvailableDrivers()); # Enumerates available PDO drivers available
//   $handler = new PDO('mysql:host=localhost;dbname=db1', 'username', 'password');
//   $handler->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//} catch(PDOException $e) {
//   echo $e->getMessage();
//   die();
//}

header("Location: http://www.wizzyweb.com/php/report.php");
```

up
down
-27
konrads dot smelkovs at gmail dot com ¶
9 years ago

```
A note for the eager:
There is no way how to get returned row count from an executed prepared statement without fetching
the rows.
```

add a note

- PDO Drivers
  - CUBRID (PDO)
  - MS SQL Server (PDO)
  - Firebird (PDO)
  - IBM (PDO)
  - Informix (PDO)
  - MySQL (PDO)
  - MS SQL Server (PDO)
  - Oracle (PDO)
  - ODBC and DB2 (PDO)