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

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Introduction »
« odbc_tables

PHP Data Objects ¶

- PDOStatement::rowCount — Returns the number of rows affected by the last SQL statement
- PDOStatement::setAttribute — Set a statement attribute
- PDOStatement::setFetchMode — Set the default fetch mode for this statement

- PDOException — The PDOException class
- PDO Drivers
  - CUBRID (PDO) — CUBRID Functions (PDO_CUBRID)
  - MS SQL Server (PDO) — Microsoft SQL Server and Sybase Functions (PDO_DBLIB)
  - Firebird (PDO) — Firebird Functions (PDO_FIREBIRD)
  - IBM (PDO) — IBM Functions (PDO_IBM)
  - Informix (PDO) — Informix Functions (PDO_INFORMIX)
  - MySQL (PDO) — MySQL Functions (PDO_MYSQL)
  - MS SQL Server (PDO) — Microsoft SQL Server Functions (PDO_SQLSRV)
  - Oracle (PDO) — Oracle Functions (PDO_OCI)
  - ODBC and DB2 (PDO) — ODBC and DB2 Functions (PDO_ODBC)
  - PostgreSQL (PDO) — PostgreSQL Functions (PDO_PGSQL)
  - SQLite (PDO) — SQLite Functions (PDO_SQLITE)
  - 4D (PDO) — 4D Functions (PDO_4D)

add a note

User Contributed Notes 32 notes

up
down
66
djlopez at gmx dot de ¶
10 years ago
Please note this:

Won't work:
```
$sth = $dbh->prepare('SELECT name, colour, calories FROM ?  WHERE calories < ?');
```

THIS WORKS!
```
$sth = $dbh->prepare('SELECT name, colour, calories FROM fruit WHERE calories < ?');
```

The parameter cannot be applied on table names!!
up
down
18
pokojny at radlight dot com ¶
10 years ago
I wanted to extend PDO class to store statistics of DB usage, and I faced some problems. I wanted to count number of created statements and number of their executings. So PDOStatement should have link to PDO that created it and stores the statistical info. The problem was that I didn't knew how PDO creates PDOStatement (constructor parameters and so on), so I have created these two classes:

```
<?php
/**
* PHP Document Object plus
*
* PHP Document Object plus is library with functionality of PDO, entirely written
* in PHP, so that developer can easily extend it's classes with specific functionality,
* such as providing database usage statistics implemented in v1.0b
*
* @author Peter Pokojny
* @license http://opensource.org/licenses/gpl-license.php GNU Public License
*/
    class PDOp {
```

```
        protected $PDO;
        public $numExecutes;
        public $numStatements;
        public function __construct($dsn, $user=NULL, $pass=NULL, $driver_options=NULL) {
            $this->PDO = new PDO($dsn, $user, $pass, $driver_options);
            $this->numExecutes = 0;
            $this->numStatements = 0;
        }
        public function __call($func, $args) {
            return call_user_func_array(array(&$this->PDO, $func), $args);
        }
        public function prepare() {
            $this->numStatements++;

            $args = func_get_args();
            $PDOS = call_user_func_array(array(&$this->PDO, 'prepare'), $args);

            return new PDOpStatement($this, $PDOS);
        }
        public function query() {
            $this->numExecutes++;
            $this->numStatements++;

            $args = func_get_args();
            $PDOS = call_user_func_array(array(&$this->PDO, 'query'), $args);

            return new PDOpStatement($this, $PDOS);
        }
        public function exec() {
            $this->numExecutes++;

            $args = func_get_args();
            return call_user_func_array(array(&$this->PDO, 'exec'), $args);
        }
    }
    class PDOpStatement implements IteratorAggregate {
        protected $PDOS;
        protected $PDOp;
        public function __construct($PDOp, $PDOS) {
            $this->PDOp = $PDOp;
            $this->PDOS = $PDOS;
        }
        public function __call($func, $args) {
            return call_user_func_array(array(&$this->PDOS, $func), $args);
        }
        public function bindColumn($column, &$param, $type=NULL) {
            if ($type === NULL)
                $this->PDOS->bindColumn($column, $param);
            else
                $this->PDOS->bindColumn($column, $param, $type);
        }
        public function bindParam($column, &$param, $type=NULL) {
            if ($type === NULL)
                $this->PDOS->bindParam($column, $param);
            else
                $this->PDOS->bindParam($column, $param, $type);
        }
        public function execute() {
```

```
            $this->PDOp->numExecutes++;
            $args = func_get_args();
            return call_user_func_array(array(&$this->PDOS, 'execute'), $args);
        }
        public function __get($property) {
            return $this->PDOS->$property;
        }
        public function getIterator() {
            return $this->PDOS;
        }
    }
?>
```

Classes have properties with original PDO and PDOStatement objects, which are providing the functionality to PDOp and PDOpStatement.
From outside, PDOp and PDOpStatement look like PDO and PDOStatement, but also are providing wanted info.

up
down
7

lkmorlan at uwaterloo dot ca ¶

6 years ago

This works to get UTF8 data from MSSQL:

```
<?php
$db = new PDO('dblib:host=your_hostname;dbname=your_db;charset=UTF-8', $user, $pass);
?>
```

up
down
5

paulius_k at yahoo dot com ¶

10 years ago

If you need to get Output variable from MSSQL stored procedure, try this :

```
-- PROCEDURE
CREATE PROCEDURE spReturn_Int @err int OUTPUT
AS
SET @err = 11
GO

$sth = $dbh->prepare("EXECUTE spReturn_Int ?");
$sth->bindParam(1, $return_value, PDO::PARAM_INT|PDO::PARAM_INPUT_OUTPUT);
$sth->execute();
print "procedure returned $return_value\n";
```

up
down
5

paul dot maddox at gmail dot com ¶

7 years ago

I decided to create a singleton wrapper for PDO that ensures only one instance is ever used.
It uses PHP 5.3.0's __callStatic functionality to pass on statically called methods to PDO.

This means you can just call it statically from anywhere without having to initiate or define the object.

Usage examples:

```
<?php
DB::exec("DELETE FROM Blah");
```

```
foreach( DB::query("SELECT * FROM Blah") as $row){
        print_r($row);
}
?>
```

Code:

```
<?php

class DB {

    private static $objInstance;

    /*
     * Class Constructor - Create a new database connection if one doesn't exist
     * Set to private so no-one can create a new instance via ' = new DB();'
     */
    private function __construct() {}

    /*
     * Like the constructor, we make __clone private so nobody can clone the instance
     */
    private function __clone() {}

    /*
     * Returns DB instance or create initial connection
     * @param
     * @return $objInstance;
     */
    public static function getInstance(  ) {

        if(!self::$objInstance){
            self::$objInstance = new PDO(DB_DSN, DB_USER, DB_PASS);
            self::$objInstance->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }

        return self::$objInstance;

    } # end method

    /*
     * Passes on any static calls to this class onto the singleton PDO instance
     * @param $chrMethod, $arrArguments
     * @return $mix
     */
    final public static function __callStatic( $chrMethod, $arrArguments ) {

        $objInstance = self::getInstance();

        return call_user_func_array(array($objInstance, $chrMethod), $arrArguments);

    } # end method

}
?>
```

up
down
0

Sbastien Gourmand ¶

3 years ago

Merge the prepare() and execute() in one function like a sprintf().
And like sprintf, I choose to use unnamed args (?) ;)

you could still use old insecure query() (not prepared) with renamed function :)

```
<?php
class MyPDO extends PDO{

    const PARAM_host='localhost';
    const PARAM_port='3306';
    const PARAM_db_name='test';
    const PARAM_user='root';
    const PARAM_db_pass='';

    public function __construct($options=null){

parent::__construct('mysql:host='.MyPDO::PARAM_host.';port='.MyPDO::PARAM_port.';dbname='.MyPDO::PARAM_db_name,
MyPDO::PARAM_user,
MyPDO::PARAM_db_pass,$options);
    }

    public function query($query){ //secured query with prepare and execute
        $args = func_get_args();
        array_shift($args); //first element is not an argument but the query itself, should removed

        $reponse = parent::prepare($query);
        $reponse->execute($args);
        return $reponse;

    }

    public function insecureQuery($query){ //you can use the old query at your risk ;) and should use secure
quote() function with it
        return parent::query($query);
    }

}

$db = new MyPDO();
$db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_OBJ);

$t1 = isset($_GET["t1"])?$_GET["t1"]:1; // need to be securised for injonction
$t2 = isset($_GET["t2"])?$_GET["t2"]:2; // need to be securised for injonction
$t3 = isset($_GET["t3"])?$_GET["t3"]:3; // need to be securised for injonction

$ret = $db->query("SELECT * FROM table_test WHERE t1=? AND t2=? AND t3=?",$t1,$t2,$t3);
//$ret = $db->insecureQuery("SELECT * FROM table_test WHERE t1=".$db->quote($t1));

while ($o = $ret->fetch())
{
    echo $o->nom.PHP_EOL;
}
?>
```

up
down
1

Matthias Leuffen ¶
11 years ago
Hi there,

because of ZDE 5.0 and other PHP-IDEs do not seem to support PDO from PHP5.1 in code-completion-database yet, I wrote a code-completion alias for the PDO class.
NOTE: This Class has no functionality and should be only included to your IDE-Project but NOT(!) to your application.

```
<?php
/**
* This is a Code-Completion only file
* (For use with ZDE or other IDEs)
*
* Do NOT include() or require() this to your code
*
* @author Matthias Leuffen
*/

    class PDO {

        /**
         * Error Constants
         *
         */
        const ERR_ALREADY_EXISTS = 0;
        const ERR_CANT_MAP = 0;
        const ERR_NOT_FOUND = 0;
        const ERR_SYNTAX = 0;
        const ERR_CONSTRAINT = 0;
        const ERR_MISMATCH = 0;
        const ERR_DISCONNECTED = 0;
        const ERR_NONE = 0;

        /**
         * Attributes (to use in PDO::setAttribute() as 1st Parameter)
         *
         */
        const ATTR_ERRMODE = 0;
        const ATTR_TIMEOUT = 0;
        const ATTR_AUTOCOMMIT = 0;
        const ATTR_PERSISTENT = 0;

        // Values for ATTR_ERRMODE
        const ERRMODE_EXCEPTION = 0;
        const ERRMODE_WARNING = 0;

        const FETCH_ASSOC = 0;
        const FETCH_NUM = 0;
        const FETCH_OBJ = 0;

        public function __construct($uri, $user, $pass, $optsArr) {
        }

        /**
         * Prepare Statement: Returns PDOStatement
         *
         * @param string $prepareString
```

```
         * @return PDOStatement
         */
        public function prepare ($prepareString) {
        }
        public function query ($queryString) {
        }
        public function quote ($input) {
        }
        public function exec ($statement) {
        }
        public function lastInsertId() {
        }
        public function beginTransaction () {
        }
        public function commit () {
        }
        public function rollBack () {
        }
        public function errorCode () {
        }
        public function errorInfo () {
        }
    }

    class PDOStatement {

        public function bindValue ($no, $value) {
        }
        public function fetch () {
        }
        public function nextRowset () {
        }
        public function execute() {
        }
        public function errorCode () {
        }
        public function errorInfo () {
        }
        public function rowCount () {
        }
        public function setFetchMode ($mode) {
        }
        public function columnCount () {
        }
    }
```

up
down
1
webform at aouie dot website ¶
10 years ago

If you use $dbh = new PDO('pgsql:host=localhost;dbname=test_basic01', $user, $pass); and you get the following error:
PHP Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[08006] [7] could not connect to server: Connection refused\n\tIs the server running on host "localhost" and accepting\n\tTCP/IP connections on port 5432?'
then as pointed out under pg_connect at: http://www.php.net/manual/en/function.pg-connect.php#38291
******
you should try to leave the host= and port= parts out of the connection string. This sounds strange, but this

is an "option" of Postgre. If you have not activated the TCP/IP port in postgresql.conf then postgresql doesn't accept any incoming requests from an TCP/IP port. If you use host= in your connection string you are going to connect to Postgre via TCP/IP, so that's not going to work. If you leave the host= part out of your connection string you connect to Postgre via the Unix domain sockets, which is faster and more secure, but you can't connect with the database via any other PC as the localhost.
******
Sincerely,
Aouie

up
down
0

nospam dot list at unclassified dot de ¶
9 years ago

When using persistent connections, pay attention not to leave the database connection in some kind of locked state. This can happen when you start a transaction by hand (i.e. not through the PDO->beginTransaction() method), possibly even acquire some locks (e.g. with "SELECT ... FOR UPDATE", "LOCK TABLES ..." or in SQLite with "BEGIN EXCLUSIVE TRANSACTION") and then your PHP script ends with a fatal error, unhandled exception or under other circumstances that lead to an unclean exit.

To use that database again, it may then be necessary to disable the persistence attribute to get a new database connection or restart the web server. (Persistent connections should not work with a PHP-CGI anyway.) It does not work (tested with PHP 5.2.3/WinXP and SQLite) to close a persistent database connection - it will not actually be closed but instead returned to PDO's connection pool.

The only thing you can do to resolve the lock as a regular user (I imagine) is to try and get all of your persistent connections in a single script and unlock the tables resp. end the transactions with the appropriate SQL statements ("UNLOCK TABLES" in MySQL, "ROLLBACK" for transactions). Should they fail, there is no problem, but one or some of them might succeed and thereby resolve your locking problem.

up
down
0

Konstantin at Tokar dot ru ¶
9 years ago

Example 5:

```
<?php
try {
  $dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
      array(PDO::ATTR_PERSISTENT => true));
.......
} catch (Exception $e) {
  $dbh->rollBack();
  echo "Failed: " . $e->getMessage();
}
?>
```

We must change the last two lines to catch the error connecting to the database:

```
} catch (Exception $e) {
  echo "Failed: " . $e->getMessage();
  $dbh->rollBack();
}
?>
```

up
down
0

bart at mediawave dot nl ¶
9 years ago

It seems MySQL doesn't support scrollable cursors. So unfortunately PDO::CURSOR_SCROLL wont work.

up
down
0
smileaf at smileaf dot org ¶
9 years ago

If you intend on extending PDOStatement and your using
setAttribute(PDO::ATTR_STATEMENT_CLASS, ...)
you must override the __construct() of your PDOStatement class.
failure to do so will result in an error on any PDO::query() call.
Warning: PDO::query() [function.PDO-query]: SQLSTATE[HY000]: General error: user-supplied statement does not accept constructor arguments

Here is a minimum PDO and PDOStatement class

```
<?php
class Database extends PDO {
    function __construct($dsn, $username="", $password="", $driver_options=array()) {
        parent::__construct($dsn,$username,$password, $driver_options);
        $this->setAttribute(PDO::ATTR_STATEMENT_CLASS, array('DBStatement', array($this)));
    }
}
class DBStatement extends PDOStatement {
    public $dbh;
    protected function __construct($dbh) {
        $this->dbh = $dbh;
    }
}
?>
```

up
down
0
php at moechofe dot com ¶
9 years ago

Simple example to extends PDO

```
<?php

class connexion extends PDO
{
  public $query = null;
  public function prepare( $statement, $driver_options = array() )
  {
    $this->query = $statement;
    return parent::prepare( $statement, $driver_options = array() );
  }
  public function last_query()
  {
    return $this->query;
  }
}

class connexion_statement extends PDOStatement
{
  protected $pdo;
  protected function __construct($pdo)
  {
     $this->pdo = $pdo;
  }
  // return first column of first row
```

```
  public function fetchFirst()
  {
    $row = $this->fetch( PDO::FETCH_NUM );
    return $row[0];
  }
  // real cast number
  public function fetch( $fetch_style = null, $cursor_orientation = null, $cursor_offset = null )
  {
    $row = parent::fetch( $fetch_style, $cursor_orientation, $cursor_offset );
    if( is_array($row) )
      foreach( $row as $key => $value )
        if( strval(intval($value)) === $value )
          $row[$key] = intval($value);
        elseif( strval(floatval($value)) === $value )
          $row[$key] = floatval($value);
    return $row;
  }
  // permit $prepare->execute( $arg1, $arg2, ... );
  public function execute( $args = null )
  {
    if( is_array( $args ) )
      return parent::execute( $args );
    else
    {
      $args = func_get_args();
      return eval( 'return parent::execute( $args );' );
    }
  }
  public function last_query()
  {
    return $this->pdo->last_query();
  }
}

$pdo = new connexion( ... );
$pdo->setAttribute( PDO::ATTR_STATEMENT_CLASS, array( 'connexion_statement', array($pdo) ) );

?>
```

up
down
0

keyvez at hotmail dot com ¶

10 years ago

PDO doesn't return OUTPUT params from mssql stored procedures

```
/* Stored Procedure Create Code: */
/*
CREATE PROCEDURE p_sel_all_termlength @err INT OUTPUT AS
SET @err = 2627
*/

/* PHP Code: */
<?php
    $Link = new PDO('mssql:host=sqlserver;dbname=database', 'username',
'password');

    $ErrorCode = 0;
```

```
    $Stmt = $Link->prepare('p_sel_all_termlength ?');
    $Stmt->bindParam(1,$ErrorCode,PDO::PARAM_INT,4);
    $Stmt->execute();
    echo "Error = " . $ErrorCode . "\n";
?>

/* PHP Output:
Error = 0
*/
```

up
down
-1
lkmorlan at uwaterloo dot ca ¶
6 years ago

You may also have to edit /etc/freetds.conf. Make sure the TDS version is recent, e.g., "tds version = 8.0".

up
down
-1
nicolas at serpe dot org ¶
10 years ago

I use PDO with the ODBC driver to query stored procedures in a MS SQL Server 2005 Database under Windows XP Professional with IIS 5 and PHP 5.1.4. You may have the same problems with a different configuration.

I experienced 2 very time consuming errors:

1. The first one is when you return the result of a SELECT query, and you get the following clueless message:
>>> Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[24000]: Invalid cursor state: 0 [Microsoft][SQL Native Client]Invalid cursor state (SQLFetchScroll[0] at ext\pdo_odbc\odbc_stmt.c:372)' in (YOUR_TRACE_HERE) <<<
Your exact message may be different, the part to pay attention to is "Invalid cursor state".

-> I found that I had this error because I didn't include "SET NOCOUNT ON" in the *body* of the stored procedure. By default the server returns a special piece of information along with the results, indicating how many rows were affected by the stored procedure, and that's not handled by PDO.

2. The second error I had was:
>>> Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[22003]: Numeric value out of range: 0 [Microsoft][SQL Native Client]Numeric value out of range (SQLFetchScroll[0] at ext\pdo_odbc\odbc_stmt.c:372)' in (YOUR_TRACE_HERE) <<<
Another meaningless error "Numeric value out of range"...

-> I was actually returning a date datatype (datetime or smalldatetime) "as is", that is, without converting it to varchar before including it in the result set... I don't know if PDO is responsible for converting it to a PHP datatype, but it doesn't. Convert it before it reaches PHP.

up
down
-2
djlopez at gmx dot de ¶
9 years ago

Note this:

Won't work:
```
$sth = $dbh->prepare('SELECT name, colour, calories FROM fruit WHERE ? < ?');
```

THIS WORKS!
```
$sth = $dbh->prepare('SELECT name, colour, calories FROM fruit WHERE calories < ?');
```

Parameters cannot be applied on column names!!

up
down
-1
ng4rrjanbiah at rediffmail dot com ¶
11 years ago
Some useful links on PDO:
1. PDO Wiki (http://wiki.cc/php/PDO)
2. Introducing PHP Data Objects (http://netevil.org/downloads/Introducing-PDO.ppt), [226 KB], Wez Furlong, 2004-09-24
3. The PHP 5 Data Object (PDO) Abstraction Layer and Oracle (http://www.oracle.com/technology/pub/articles/php_experts/otn_pdo_oracle5.html), [60.85 KB], Wez Furlong, 2004-07-28
4. PDO - Why it should not be part of core PHP! (http://www.akbkhome.com/blog.php/View/55/), Critical review, [38.63 KB], Alan Knowles, 2004-10-22

HTH,
R. Rajesh Jeba Anbiah
up
down
-2
www.navin.biz ¶
10 years ago
Below is an example of extending PDO & PDOStatement classes:

```
<?php

class Database extends PDO
{
    function __construct()
    {
        parent::__construct('mysql:dbname=test;host=localhost', 'root', '');
        $this->setAttribute(PDO::ATTR_STATEMENT_CLASS, array('DBStatement', array($this)));
    }
}

class DBStatement extends PDOStatement
{
    public $dbh;

    protected function __construct($dbh)
    {
        $this->dbh = $dbh;
        $this->setFetchMode(PDO::FETCH_OBJ);
    }

    public function foundRows()
    {
        $rows = $this->dbh->prepare('SELECT found_rows() AS rows', array(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY
=> TRUE));
        $rows->execute();
        $rowsCount = $rows->fetch(PDO::FETCH_OBJ)->rows;
        $rows->closeCursor();
        return $rowsCount;
    }
}

?>
```

up

down
-3
matthew at button-mashers dot net ¶
2 years ago
When using prepared statements there is no official PDO feature to show you the final query string that is submitted to a database complete with the parameters you passed.

Use this simple function for debugging. The values you are passing may not be what you expect.

```
<?php
//Sample query string
$query = "UPDATE users SET name = :user_name WHERE id = :user_id";

//Sample parameters
$params = [':user_name' => 'foobear', ':user_id' => 1001];

function build_pdo_query($string, $array) {
    //Get the key lengths for each of the array elements.
    $keys = array_map('strlen', array_keys($array));

    //Sort the array by string length so the longest strings are replaced first.
    array_multisort($keys, SORT_DESC, $array);

    foreach($array as $k => $v) {
        //Quote non-numeric values.
        $replacement = is_numeric($v) ? $v : "'{$v}'";

        //Replace the needle.
        $string = str_replace($k, $replacement, $string);
    }

    return $string;
}

echo build_pdo_query($query, $params);    //UPDATE users SET name = 'foobear' WHERE id = 1001
?>
```

up
down
-4
tomasz dot wasiluk at gmail dot com ¶
11 years ago
Watch out for putting spaces in the DSN
mysql:host=localhost;dbname=test works
mysql: host = localhost; dbname=test works
mysql: host = localhost; dbname = test doesn't work...

up
down
-3
rijnzael at gmail dot com ¶
8 years ago
If you plan on using prepared statements to issue a series of KILLs, think again. PDO apparently does not support this, and will tell you, assuming you don't put any parameters in the statement (a situation in which it would be pointless to use prepared statements anyway). If you do specify some ?-mark parameters, it will just spit out an error that doesn't help at all.

up
down
-4
shaolin at adf dot nu ¶

10 years ago

If your having problems re-compiling PHP with PDO as shared module try this.

```
--enable-pdo=shared
--with-pdo-mysql=shared,/usr/local/mysql
--with-sqlite=shared
--with-pdo-sqlite=shared

1. If PDO is built as a shared modules, all PDO drivers must also be
built as shared modules.
2. If ext/pdo_sqlite is built as a shared module, ext/sqlite must also
be built as a shared module.
3. In the extensions entries, if ext/pdo_sqlite is built as a shared
module, php.ini must specify pdo_sqlite first, followed by sqlite.
```

up
down
-3

ob.php from daevel.fr ¶

9 years ago

Be careful with PDO extends : if you use the smileaf's example, PDO will close the connection only at the end of the script, because of the "array($this)" parameter used with the setAttribute() method.

Instead, I use only this :

```
    $this->setAttribute( PDO::ATTR_STATEMENT_CLASS, array( $this->statementClassName, array() ) );
```

And in prepare() and query() method you can populate the "dbh" if you really need it.

up
down
-3

anton dot clarke at sonikmedia dot com ¶

9 years ago

Not all PDO drivers return a LOB as a file stream; mysql 5 is one example. Therefore when streaming a mime typed object from the database you cannot use fpassthru.

The following is a modified example that works with a mysql database. (Tested FreeBSD v 6.2 with mysql 5.0.45 and php 5.2.3)

```
<?php
ob_start();
$db = new PDO('mysql:host=localhost;dbname=<SOMEDB>', '<USERNAME>', 'PASSWORD');
$stmt = $db->prepare("select contenttype, imagedata from images where id=?");
$stmt->execute(array($_GET['id']));
$stmt->bindColumn(1, $type, PDO::PARAM_STR, 256);
$stmt->bindColumn(2, $lob, PDO::PARAM_LOB);
$stmt->fetch(PDO::FETCH_BOUND);
ob_clean();
header("Content-Type: $type");
echo $lob; // fpassthru reports an error that $lob is not a stream so echo is used in place.
ob_end_flush();
?>
```

Please note the inclusion of buffer control. I only needed this when using 'include','include_once','require', or 'require_once' - my feeling is there is a subtle issue with those options as even an empty include file caused a buffer issue for me. === AND YES, I DID CHECK MY INCLUDE FILES DID NOT HAVE SPURIOUS WHITESPACE ETC OUTSIDE THE <?php ?> DELIMITERS! ===

up
down
-3

Dariusz Kielar ¶

10 years ago

I found a nice pdo modification written in php called Open Power Driver. It has identical API with the original, but allows you to cache query results: http://www.openpb.net/opd.php

up

down

-3

jose at thezcompany dot com ¶

6 years ago

I ran into a real annoying bug/feature when using PDO for SQL statements that use SQL user variables. I was working on some logic for a Geo Proximity Search for an events-venues system (sharing is caring so it's below) and it just wouldn't take and the errors returned were garbage. The SQL was sound as I verified it. So if you're having this issue, I hope this helps. What you need to do is break apart the query into two...

From:

```
<?php
$sql="set @latitude=:lat;
set @longitude=:lon;
set @radius=20;

set @lng_min = @longitude - @radius/abs(cos(radians(@latitude))*69);
set @lng_max = @longitude + @radius/abs(cos(radians(@latitude))*69);
set @lat_min = @latitude - (@radius/69);
set @lat_max = @latitude + (@radius/69);

SELECT *,
3956 * 2 * ASIN(SQRT(POWER(SIN((@latitude - ABS(venue_lat)) * PI()/180 / 2),2) + COS(@latitude * PI()/180) *
COS(ABS(venue_lat) * PI()/180) * POWER(SIN((@longitude - venue_lon) * PI()/180 / 2),2))) AS distance
FROM events LEFT JOIN venues ON venues.venue_id = events.venue_fk
WHERE (venue_lon BETWEEN @lng_min AND @lng_max)
AND (venue_lat BETWEEN @lat_min and @lat_max)
AND events.event_date >= CURDATE()
AND events.event_time >= CURTIME()
ORDER BY distance DESC;";

$stmt = $this->_db->prepare($sql);
$stmt->bindParam(':lat', $lat, PDO::PARAM_STR);
$stmt->bindParam(':lon', $lon, PDO::PARAM_STR);
$stmt->bindParam(':offset', $offset, PDO::PARAM_INT);
$stmt->bindParam(':max', $max, PDO::PARAM_INT);
$stmt->execute();
?>
```

To:

```
<?php
$sql = "SET @latitude=:lat;
SET @longitude=:lon;
SET @radius=20;
SET @lng_min=@longitude - @radius/abs(cos(radians(@latitude))*69);
SET @lng_max=@longitude + @radius/abs(cos(radians(@latitude))*69);
SET @lat_min=@latitude - (@radius/69);
SET @lat_max=@latitude + (@radius/69);";

$this->_db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
$stmt = $this->_db->prepare($sql);
$stmt->bindParam(':lat', $lat, PDO::PARAM_STR);
$stmt->bindParam(':lon', $lon, PDO::PARAM_STR);
$stmt->execute();
```

```
$sql = "SELECT *,
(3956 * 2 * ASIN(SQRT(POWER(SIN((@latitude - ABS(venue_lat)) * PI()/180 / 2),2) + COS(@latitude * PI()/180) *
COS(ABS(venue_lat) * PI()/180) * POWER(SIN((@longitude - venue_lon) * PI()/180 / 2),2)))) AS distance
FROM events LEFT JOIN venues ON venues.venue_id = events.venue_fk
WHERE (venue_lon BETWEEN @lng_min AND @lng_max)
AND (venue_lat BETWEEN @lat_min and @lat_max)
AND events.event_date >= CURDATE()
AND events.event_time >= CURTIME()
ORDER BY distance DESC
LIMIT :offset,:max;";

$stmt = $this->_db->prepare($sql);
$stmt->bindParam(':offset', $offset, PDO::PARAM_INT);
$stmt->bindParam(':max', $max, PDO::PARAM_INT);
$stmt->execute();
?>
```

Hope this helps anyone out there!

up
down
-6
xorinox at vtxmail dot ch ¶
9 years ago

I have seen a lot of user struggling with calling mysql procedures with in/out/inout parameters using bindParam. There seems to be a bug or missing feature within the mysql C api. This at least I could find out after reading a lot of posts at different places...

At the moment I workaround it like below. $con is a PDO object:

```
<?php
//in
$proc = $con->prepare( "call proc_in( @i_param )" );
$con->query( "set @i_param = 'myValue'" );
$proc->execute();

//out
$proc = $con->prepare( "call proc_out( @o_param )" );
$proc->execute();
$o_param = $con->query( "select @o_param" )->fetchColumn();

//inout
$proc = $con->prepare( "call proc_inout( @io_param )" );
$con->query( "set @io_param = 'myValue'" );
$proc->execute();
$io_param = $con->query( "select @io_param" )->fetchColumn();
?>
```

up
down
-4
neonmandk at gmail dot com ¶
9 years ago

If you will make a OBJ row from PDO you can use this eg.

```
$resKampange = $dbc->prepare( "SELECT * FROM Table LIMIT 1" );
$resKampange->execute();
$rowKampange = $resKampange->fetch( PDO::FETCH_OB );
```

```
echo $rowKampange->felt1;

Good lock :0)
```

up
down
-4
metalim ¶
9 years ago

```
From Oracle example:
<?
$stmt->beginTransaction();
$stmt->execute();
$stmt->commit();
?>

PDOStatement has no beginTransaction(), nor commit(). Please fix documentation.
```

up
down
-6
cmcculloh ¶
8 years ago

```
It appears that PDO SQL statements can not make use of /* */ comments. Or at least, when I was trying to use
them with mine it was crashing without error and giving me a blank page (even though I surrounded with try
catch block).
```

up
down
-8
Ostap ¶
9 years ago

```
There is a book titled "Learning PHP Data Objects" written by Dennis Popel and published by Packt. There is a
post with further links (ordering, reviews) at author's blog: http://www.onphp5.com/article/58
```

add a note

- Abstraction Layers
  - DBA
  - dbx
  - ODBC
  - PDO

- My PHP.net
- Contact
- Other PHP.net sites
- Mirror sites
- Privacy policy