php

- Downloads
- Documentation
- Get Involved
- Help

Search

PHPKonf: Istanbul PHP Conference 2017

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

PDO::query

(PHP 5 >= 5.1.0, PHP 7, PECL pdo >= 0.2.0)

PDO::query — Executes an SQL statement, returning a result set as a PDOStatement object

Description ¶

public PDOStatement **PDO::query** (string `$statement`)
public PDOStatement **PDO::query** (string `$statement` , int `$PDO::FETCH_COLUMN` , int `$colno`)
public PDOStatement **PDO::query** (string `$statement` , int `$PDO::FETCH_CLASS` , string `$classname` , array `$ctorargs`)
public PDOStatement **PDO::query** (string `$statement` , int `$PDO::FETCH_INTO` , object `$object`)

PDO::query() executes an SQL statement in a single function call, returning the result set (if any) returned by the statement as a PDOStatement object.

For a query that you need to issue multiple times, you will realize better performance if you prepare a PDOStatement object using PDO::prepare() and issue the statement with multiple calls to PDOStatement::execute().

If you do not fetch all of the data in a result set before issuing your next call to **PDO::query()**, your call may fail. Call PDOStatement::closeCursor() to release the database resources associated with the PDOStatement object before issuing your next call to **PDO::query()**.

> **Note**:
>
> Although this function is only documented as having a single parameter, you may pass additional arguments to this function. They will be treated as though you called PDOStatement::setFetchMode() on the resultant statement object.

Parameters ¶

`statement`

The SQL statement to prepare and execute.

Data inside the query should be properly escaped.

Return Values ¶

PDO::query() returns a PDOStatement object, or **`FALSE`** on failure.

Examples ¶

Example #1 Demonstrate PDO::query

A nice feature of **PDO::query()** is that it enables you to iterate over the rowset returned by a successfully executed SELECT statement.

```php
<?php
function getFruit($conn) {
    $sql = 'SELECT name, color, calories FROM fruit ORDER BY name';
    foreach ($conn->query($sql) as $row) {
        print $row['name'] . "\t";
        print $row['color'] . "\t";
        print $row['calories'] . "\n";
    }
}
?>
```

The above example will output:

```
apple   red     150
banana  yellow  250
kiwi    brown   75
lemon   yellow  25
orange  orange  300
pear    green   150
watermelon      pink    90
```

See Also ¶

- PDO::exec() - Execute an SQL statement and return the number of affected rows
- PDO::prepare() - Prepares a statement for execution and returns a statement object
- PDOStatement::execute() - Executes a prepared statement

add a note

User Contributed Notes 9 notes

up
down
36
fredrik at NOSPAM dot rambris dot com ¶

9 years ago

```
The handling of errors by this function is controlled by the attribute PDO::ATTR_ERRMODE.

Use the following to make it throw an exception:
<?php
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
?>
```

up
down
10

stefano[dot]bertoli [at] gmail[dot]com ¶

2 years ago

```
Trying to pass like second argument PDO::FETCH_ASSOC it still work.

So passing FETCH TYPE like argument seems work.

This save you from something like:

<?php
$result = $stmt->setFetchMode(PDO::FETCH_NUM);
?>

Example:
<?php
$res = $db->query('SELECT * FROM `mytable` WHERE true', PDO::FETCH_ASSOC);

?>
```

up
down
5

paolo at dellunto dot net ¶

3 years ago

```
If you are using PDO to create an SQLite dbfile that will be used by an Android application, you can
set common values via the $dbh->query("PRAGMA ...") statement;

a tipical example would be the user_version of the database or the page_size
<?php
...
$dbh = new PDO($PDO_DSN, null, null, null);
$dbh->query("PRAGMA page_size = 4096"); //Android match page size
$dbh->query("PRAGMA user_version = 2"); //This match super(context, DB_NAME, null, DB_VERSION) of the
DatabaseOpenHelper
....
?>
```

up
down
2

tgrl5000 ¶

1 year ago

```
Connecting
==============================
<?php
try{
$db = new PDO("dbtype:host=yourhost;dbname=yourdbname;charset=utf8","username","password");
/*Other Codes*/
}catch(PDOException  $e ){
echo "Error: ".$e;
}
?>
Excute query with secure data
==============================
<?php
try{
$db = new PDO("dbtype:host=yourhost;dbname=yourdbname;charset=utf8","username","password");
$mysecuredata=14;
$db->query("Select * from table where id=".$mysecuredata);
}catch(PDOException  $e ){
echo "Error: ".$e;
}
?>
Excute query with insecure data
==============================
<?php
try{
$db = new PDO("dbtype:host=yourhost;dbname=yourdbname;charset=utf8","username","password");
$myinsecuredata=$_GET["id"];
$query=$db->prepare("Select * from table where id=?");
$query->excute(array($myinsecuredata));
}catch(PDOException  $e ){
echo "Error: ".$e;
}
?>
Getting Data in database
==============================
<?php
try{
$db = new PDO("dbtype:host=yourhost;dbname=yourdbname;charset=utf8","username","password");
$myinsecuredata=$_GET["table"];
$query=$db->prepare("Select * from ?");
$query->excute(array($myinsecuredata));
while($row=$query->fetch(PDO::FETCH_OBJ)) {
/*its getting data in line.And its an object*/
        echo $row->yourcolumnname;
    }
}catch(PDOException  $e ){
echo "Error: ".$e;
}
?>
Reference
===============================
```

http://gencbilgin.net/pdo-kullanimi-php-de-veritabani-islemleri.html

up
down
5
dozoyousan at gmail dot com ¶
10 years ago

```
> When query() fails, the boolean false is returned.
```

```
I think that is "Silent Mode".
If that set attribute ErrorMode "Exception Mode"
then that throw PDOException.
$pdoObj = new PDO( $dsn, $user, $pass );
$pdoObj->setAttribute("PDO::ATTR_ERRMODE", PDO::ERRMODE_EXCEPTION);
```

up
down
1
andrea at bhweb dot it ¶
8 years ago

If someone is suffering of the "MySQL server has gone away" problem after executing multiple queries, this is a solution that solved it for me. It's similar to the one needed for the exact same problem in mysqli.

```
<?php
$stmt=$db->prepare($query);
$stmt->execute();
do { $stmt->fetch(); $stmt->closeCursor(); ++$line; } while($stmt-
>nextRowset());
?>
```

I found this only works using prepare and execute this way, not if you directly execute the query with query().
up
down
0
marcos at marcosregis dot com ¶
8 years ago

After a lot of hours working with DataLink on Oracle->MySQL and PDO we (me and Adriano Rodrigues, that solve it) discover that PDO (and oci too) need the attribute AUTOCOMMIT set to FALSE to work correctly with.
There's 3 ways to set autocommit to false: On constructor, setting the atribute after construct and before query data or initiating a Transaction (that turns off autocommit mode)

The examples:

```
<?php
// First way - On PDO Constructor
$options = array(PDO::ATTR_AUTOCOMMIT=>FALSE);

$pdo = new PDO($dsn,$user,$pass,$options);

// now we are ready to query DataLinks

?>

<?php
// Second Way - Before create statements
$pdo = new PDO($dsn,$user,$pass);

$pdo->setAttribute(PDO::ATTR_AUTOCOMMIT,FALSE);
// or
$pdo->beginTransaction();

// now we are ready to query DataLinks
```

```
?>

To use DataLinks on oci just use OCI_DEFAULT on oci_execute() function;
```

up
down
-9
nicobn at gmail dot com ¶
9 years ago

```
Please note that when Query() fails, it does not return a PDOStatement object . It simply returns
false.
```

up
down
-58
NUNTIUS ¶
8 years ago

```
I found this method extremely useful for getting the iteration count. Note the usage of "for" instead
of "while" or "foreach". Just place the "$row = $query->fetch()" as the second condition of your for
loop (which is do until). This is the best of both worlds IMHO. Criticism welcome.

try {
    $hostname = "servername";
    $dbname = "dbname";
    $username = "username";
    $pw = "password";
    $pdo = new PDO ("mssql:host=$hostname;dbname=$dbname","$username","$pw");
  } catch (PDOException $e) {
    echo "Failed to get DB handle: " . $e->getMessage() . "\n";
    exit;
  }
      $query = $pdo->prepare("select name FROM tbl_name");
      $query->execute();

      for($i=0; $row = $query->fetch(); $i++){
        echo $i." - ".$row['name']."<br/>";
      }

      unset($pdo);
      unset($query);
```

add a note

- PDO
  - beginTransaction
  - commit
  - __construct
  - errorCode
  - errorInfo
  - exec
  - getAttribute
  - getAvailableDrivers
  - inTransaction
  - lastInsertId
  - prepare
  - query
  - quote

- rollBack
- setAttribute