(The only proper) PDO tutorial

There are many tutorials on PDO already, but unfortunately, most of them fail to explain the real benefits of PDO, or even promote rather bad practices. The only two exceptions are phptherightway.com (http://www.phptherightway.com/#pdo_extension) and hashphp.org (http://wiki.hashphp.org/PDO_Tutorial_for_MySQL_Developers), but they miss a lot of important information. As a result, half of PDO's features remain in obscurity and are almost never used by PHP developers, who, as a result, are constantly trying to reinvent the wheel which *already exists in PDO.*

Unlike those, this tutorial is written by someone who has used PDO for many years, dug through it, and answered thousands questions on Stack Overflow (the sole gold PDO badge bearer (http://stackoverflow.com/help/badges/4220/pdo)). Following the mission of this site (/), this article will disprove various delusions and bad practices, while showing the right way instead.

Although this tutorial is based on **mysql** driver, the information, in general, is applicable for any driver supported.

Why PDO?

First things first. Why PDO (PHP Data Objects) at all?

PDO is a Database Access Abstraction Layer (https://en.wikipedia.org/wiki/Database_abstraction_layer). The abstraction, however, is two-fold: one is widely known but less significant, while another is obscure but of most importance.

Everyone knows that PDO offers unified interface to access many different databases (http://php.net/manual/en/pdo.drivers.php). Although this feature is magnificent by itself, it doesn't make a big deal for the particular application, where only one database backend is used anyway. And, despite some rumors, it is impossible to switch database backends by changing a single line in PDO config - due to different SQL flavors (to do so, one needs to use an averaged query language like DQL (http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/reference/dql-doctrine-query-language.html)). Thus, for the average LAMP developer, this point is rather insignificant, and to him, PDO is just a more complicated version of familiar `mysql(i)_query()` function. However, it is not; it is much, much more.

PDO abstracts not only a database API, but also basic operations that otherwise have to be repeated hundreds of times in every application, making your code extremely WET (Write Everything Twice). Unlike *mysql* and *mysqli* (https://phpdelusions.net/pdo/mysqli_comparison), both of which are low level bare APIs not intended to be used directly (but only as a building material for some higher level abstraction layer), `PDO` *is* such an abstraction already. Still incomplete though, but at least usable.

The real PDO benefits are:

- **security** (*usable* prepared statements)
- **usability** (many helper functions to automate routine operations)
- **reusability** (unified API to access multitude of databases, from SQLite to Oracle)

Note that although PDO is the best out of native db drivers, for a modern web-application consider to use an ORM with a Query Builder, or any other higher level abstraction library, with only occasional fallback to vanilla PDO. Good ORMs are Doctrine, Eloquent, RedBean, and Yii::AR. Aura.SQL is a good example of a PDO wrapper with many additional features.

Either way, it's good to know the basic tools first. So, let's begin:

Connecting. DSN

PDO has a fancy connection method called DSN (https://en.wikipedia.org/wiki/Data_source_name). It's nothing complicated though - instead of one plain and simple list of options, PDO asks you to input different configuration directives in three different places:

- `database driver`, `host`, `db (schema) name` and `charset`, as well as less frequently used `port` and `unix_socket` go into DSN;
- `username` and `password` go to constructor;
- all other options go into options array.

where DSN is a semicolon-delimited string, consists of `param=value` pairs, that begins from the driver name and a colon:

```
       mysql:host=localhost;dbname=test;port=3306;charset=utf8mb4
driver^    ^ colon         ^param=value pair    ^semicolon
```

Note that it's important to follow the proper format - **no spaces or quotes or other decorations have to be used in DSN,** but only parameters, values and delimiters, as shown in the manual (http://php.net/manual/en/pdo.construct.php).

Here goes an example for mysql:

```
 $host = '127.0.0.1';
$db   = 'test';
$user = 'root';
$pass = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];
try {
     $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
     throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

With all aforementioned variables properly set, we will have proper PDO instance in `$pdo` variable.

Important notes for the late mysql extension users:

1. Unlike old `mysql_*` functions, which can be used anywhere in the code, `PDO` instance is stored in a regular variable, which means it can be inaccessible inside functions - so, one has to make it accessible, by means of passing it via function parameters or using more advanced techniques, such as IoC container.
2. The connection has to be made only once! No connects in every function. No connects in every class constructor. Otherwise, multiple connections will be created, which will eventually kill your database server. Thus, a sole PDO instance has to be created and then used through whole script execution.

3. It is very important to **set charset through DSN** - that's the only proper way because it tells PDO which charset is going to be used. Therefore forget about running `SET NAMES` query manually, either via `query()` or `PDO::MYSQL_ATTR_INIT_COMMAND`. Only if your PHP version is unacceptably outdated (namely below 5.3.6), you have to use `SET NAMES` query and always turn emulation mode off.

More details regarding Mysql can be found in the corresponding chapter, Connecting to MySQL (https://phpdelusions.net/pdo_examples/connect_to_mysql)

Running queries. PDO::query()

There are two ways to run a query in PDO. If no variables are going to be used in the query, you can use the PDO::query() (http://php.net/manual/en/pdo.query.php) method. It will run your query and return special object of PDOStatement class (http://php.net/manual/en/class.pdostatement.php) which can be roughly compared to a resource, returned by `mysql_query()`, especially in the way you can get actual rows out of it:

```
 $stmt = $pdo->query('SELECT name FROM users');
while ($row = $stmt->fetch())
{
    echo $row['name'] . "\n";
}
```

Also, the `query()` method allows us to use a neat method chaining for SELECT queries, which will be shown below.

Prepared statements. Protection from SQL injections

This is the main and the only important reason why you were deprived from your beloved `mysql_query()` function and thrown into the harsh world of Data Objects: PDO has prepared statements support out of the box. Prepared statement is **the only proper way to run a query**, if any variable is going to be used in it. The reason why it is so important is explained in detail in The Hitchhiker's Guide to SQL Injection prevention (/sql_injection).

So, for every query you run, if at least one variable is going to be used, you have to substitute it with a **placeholder**, then prepare your query, and then execute it, passing variables separately.

Long story short, it is not as hard as it seems. In most cases, you need only two functions - prepare() (http://php.net/manual/en/pdo.prepare.php) and execute() (http://php.net/manual/en/pdostatement.execute.php).

First of all, you have to alter your query, adding placeholders in place of variables. Say, a code like this

```
$sql = "SELECT * FROM users WHERE email = '$email' AND status='$status'";
```

will become

```
$sql = 'SELECT * FROM users WHERE email = ? AND status=?';
```

or

```
$sql = 'SELECT * FROM users WHERE email = :email AND status=:status';
```

Note that PDO supports positional (`?`) and named (`:email`) placeholders, the latter always begins from a colon and can be written using letters, digits and underscores only. Also note that **no quotes** have to be ever used around placeholders.

Having a query with placeholders, you have to prepare it, using the `PDO::prepare()` method. This function will return the same `PDOStatement` object we were talking about above, but *without any data attached to it*.

Finally, to get the query executed, you must run `execute()` method of this object, passing variables in it, in the form of array. And after that, you will be able to get the resulting data out of statement (if applicable):

```
 $stmt = $pdo->prepare('SELECT * FROM users WHERE email = ? AND status=?');
$stmt->execute([$email, $status]);
$user = $stmt->fetch();
// or
$stmt = $pdo->prepare('SELECT * FROM users WHERE email = :email AND status=:status');
$stmt->execute(['email' => $email, 'status' => $status]);
$user = $stmt->fetch();
```

As you can see, for the positional placeholders, you have to supply a regular array with values, while for the named placeholders, it has to be an associative array, where keys have to match the placeholder names in the query. You cannot mix positional and named placeholders in the same query.

Please note that positional placeholders let you write shorter code, but are sensitive to the order of arguments (which have to be exactly the same as the order of the corresponding placeholders in the query). While named placeholders make your code more verbose, they allow random binding order.

Also note that despite a widespread delusion, no " `:` " in the keys is required.

After the execution you may start getting your data, using all supported methods, as described down in this article.

More examples can be found in the respective article (https://phpdelusions.net/pdo_examples).

Binding methods

Passing data into `execute()` (like shown above) should be considered default and most convenient method. When this method is used, all values will be bound as **strings** (save for `NULL` values, that will be sent to the query as is, i.e. as SQL `NULL`), but most of time it's all right and won't cause any problem.

However, sometimes it's better to set the data type explicitly. Possible cases are:

- LIMIT clause (or any other SQL clause that just cannot accept a string operand) if emulation mode is turned ON.
- complex queries with non-trivial query plan that can be affected by a wrong operand type
- peculiar column types, like `BIGINT` or `BOOLEAN` that require an operand of exact type to be bound (note that in order to bind a BIGINT value with PDO::PARAM_INT you need a mysqlnd-based installation).

In such a case explicit binding have to be used, for which you have a choice of two functions, bindValue() (http://php.net/manual/en/pdostatement.bindvalue.php) and bindParam() (http://php.net/manual/en/pdostatement.bindparam.php). The former one has to be preferred, because, unlike `bindParam()` it has no side effects to deal with.

Query parts you can bind

It is very important to understand which query parts you can bind using prepared statements and which you cannot. In fact, the list is overwhelmingly short: only string and numeric literals can be bound. So you can tell that as long as your data can be represented in the query as a numeric or a quoted string literal - it can be bound. For all other cases you cannot use PDO prepared statements at all: neither an identifier, or a comma-separated list, or a part of a quoted string literal or whatever else arbitrary query part cannot be bound using a prepared statement.

Workarounds for the most frequent use cases can be found in the corresponding part of the article

Prepared statements. Multiple execution

Sometimes you can use prepared statements for the multiple execution of a prepared query. It is slightly faster than performing the same query again and again, as it does query parsing only once. This feature would have been more useful if it was possible to execute a statement prepared in another PHP instance. But alas - it is not. So, you are limited to repeating the same query only within the same instance, which is seldom needed in regular PHP scripts and which is limiting the use of this feature to repeated inserts or updates:

```
 $data = [
    1 => 1000,
    5 =>  300,
    9 =>  200,
];
$stmt = $pdo->prepare('UPDATE users SET bonus = bonus + ? WHERE id = ?');
foreach ($data as $id => $bonus)
{
    $stmt->execute([$bonus, $id]);
}
```

Note that this feature is a bit overrated. Not only it is needed too seldom to talk about, but the performance gain is not that big - query parsing is *real* fast these times.

Note that you can get this advantage only when emulation mode is turned **off**.

Running SELECT INSERT, UPDATE, or DELETE statements

Come on folks. There is absolutely nothing special in these queries. To PDO they all the same. It doesn't matter which query you are running.

Just like it was shown above, what you need is to prepare a query with placeholders, and then execute it, sending variables separately. Either for `DELETE` and `SELECT` query the process is essentially the same. The only difference is (as DML (Data Manipulation Language, INSERT, UPDATE and DELETE queries) queries do not return any data), that you can use the method chaining and thus call `execute()` right along with `prepare()`:

```
 $sql = "UPDATE users SET name = ? WHERE id = ?";
$pdo->prepare($sql)->execute([$name, $id]);
```

However, if you want to get the number of affected rows, the code will have to be the same boresome three lines:

```
 $stmt = $pdo->prepare("DELETE FROM goods WHERE category = ?");
$stmt->execute([$cat]);
$deleted = $stmt->rowCount();
```

More examples can be found in the respective article (https://phpdelusions.net/pdo_examples).

Getting data out of statement. foreach()

The most basic and direct way to get multiple rows from a statement would be `foreach()` loop. Thanks to Traversable (http://php.net/manual/en/class.traversable.php) interface, `PDOStatement` can be iterated over by using `foreach()` operator:

```
 $stmt = $pdo->query('SELECT name FROM users');
foreach ($stmt as $row)
{
    echo $row['name'] . "\n";
}
```

Note that this method is memory-friendly, as it doesn't load all the resulting rows in the memory but delivers them one by one (though keep in mind this issue).

Getting data out of statement. fetch()

We have seen this function already, but let's take a closer look. It fetches a single row from database, and moves the internal pointer in the result set, so consequent calls to this function will return all the resulting rows one by one. Which makes this method a rough analogue to `mysql_fetch_array()` but it works in a slightly different way: instead of many separate functions (`mysql_fetch_assoc()` , `mysql_fetch_row()` , etc), there is only one, but its behavior can be changed by a parameter. There are many fetch modes in PDO, and we will discuss them later, but here are few for starter:

- `PDO::FETCH_NUM` returns enumerated array
- `PDO::FETCH_ASSOC` returns associative array
- `PDO::FETCH_BOTH` - both of the above
- `PDO::FETCH_OBJ` returns object
- `PDO::FETCH_LAZY` allows all three (numeric associative and object) methods without memory overhead.

From the above you can tell that this function have to be used in two cases:

1. When only one row is expected - to get that only row. For example,

   ```
   $row = $stmt->fetch(PDO::FETCH_ASSOC);
   ```

   Will give you single row from the statement, in the form of associative array.

2. When we need to process the returned data somehow before use. In this case it have to be run through usual while loop, like one shown above.

Another useful mode is `PDO::FETCH_CLASS`, which can create an object of particular class

```
$news = $pdo->query('SELECT * FROM news')->fetchAll(PDO::FETCH_CLASS, 'News');
```

will produce an array filled with objects of News class, setting class properties from returned values. Note that in this mode

- properties are set *before* constructor call
- for all undefined properties `__set` magic method will be called
- if there is no `__set` method in the class, then new property will be created
- private properties will be filled as well, which is a bit unexpected but quite handy

Note that default mode is `PDO::FETCH_BOTH`, but you can change it using `PDO::ATTR_DEFAULT_FETCH_MODE` configuration option as shown in the connection example. Thus, once set, it can be omitted most of the time.

Return types.

Only when PDO is built upon mysqlnd *and* emulation mode is **off**, then PDO will return `int` and `float` values with respective types. Say, if we create a table

```
 create table typetest (string varchar(255), `int` int, `float` float, `null` int);
insert into typetest values('foo',1,1.1,NULL);
```

And then query it from mysqlnd-based PDO with emulation turned off, the output will be

```
 array(4) {
  ["string"] => string(3) "foo"
  ["int"]    => int(1)
  ["float"]  => float(1.1)
  ["null"]   => NULL
}
```

Otherwise the familiar `mysql_fetch_array()` behavior will be followed - all values returned as strings with only `NULL` returned as `NULL`.

If for some reason you don't like this behavior and prefer the old style with strings and NULLs only, then you can use the following configuration option to override it:

```
$pdo->setAttribute(PDO::ATTR_STRINGIFY_FETCHES, true);
```

Note that for the `DECIMAL` type the string is always returned, due to nature of this type intended to retain the precise value, unlike deliberately non-precise FLOAT and DOUBLE types.

Getting data out of statement. fetchColumn()

A neat helper function that returns value of the singe field of returned row. Very handy when we are selecting only one field:

```
 // Getting the name based on id
$stmt = $pdo->prepare("SELECT name FROM table WHERE id=?");
$stmt->execute([$id]);
$name = $stmt->fetchColumn();

// getting number of rows in the table utilizing method chaining
$count = $pdo->query("SELECT count(*) FROM table")->fetchColumn();
```

Getting data out of statement in dozens different formats. fetchAll()

That's most interesting function, with most astonishing features. Mostly thanks to its existence one can call PDO a wrapper, as this function can automate many operations otherwise performed manually.

`PDOStatement::fetchAll()` returns an array that consists of **all the rows** returned by the query. From this fact we can make two conclusions:

1. This function should not be used, if many rows has been selected. In such a case conventional while loop ave to be used, fetching rows one by one instead of getting them all into array at once. "Many" means more than it is suitable to be shown on the average web page.
2. This function is mostly useful in a modern web application that never outputs data right away during fetching, but rather passes it to template.

You'd be amazed, in how many different formats this function can return data in (and how little an average PHP user knows of them), all controlled by `PDO::FETCH_*` variables. Some of them are:

Getting a plain array.

By default, this function will return just simple enumerated array consists of all the returned rows. Row formatting constants, such as `PDO::FETCH_NUM`, `PDO::FETCH_ASSOC`, `PDO::FETCH_OBJ` etc can change the row format.

```
 $data = $pdo->query('SELECT name FROM users')->fetchAll(PDO::FETCH_ASSOC);
var_export($data);
/*
array (
  0 => array('John'),
  1 => array('Mike'),
  2 => array('Mary'),
  3 => array('Kathy'),
)*/
```

Getting a column.

It is often very handy to get plain one-dimensional array right out of the query, if only one column out of many rows being fetched. Here you go:

```
 $data = $pdo->query('SELECT name FROM users')->fetchAll(PDO::FETCH_COLUMN);
/* array (
  0 => 'John',
  1 => 'Mike',
  2 => 'Mary',
  3 => 'Kathy',
)*/
```

Getting key-value pairs.

Also extremely useful format, when we need to get the same column, but indexed not by numbers in order but by another field. Here goes `PDO::FETCH_KEY_PAIR` constant:

```
 $data = $pdo->query('SELECT id, name FROM users')->fetchAll(PDO::FETCH_KEY_PAIR);
/* array (
  104 => 'John',
  110 => 'Mike',
  120 => 'Mary',
  121 => 'Kathy',
)*/
```

Note that you have to select only two columns for this mode, first of which have to be unique.

Getting rows indexed by unique field

Same as above, but getting not one column but full row, yet indexed by an unique field, thanks to `PDO::FETCH_UNIQUE` constant:

```
 $data = $pdo->query('SELECT * FROM users')->fetchAll(PDO::FETCH_UNIQUE);
/* array (
  104 => array (
    'name' => 'John',
    'car' => 'Toyota',
  ),
  110 => array (
    'name' => 'Mike',
    'car' => 'Ford',
  ),
  120 => array (
    'name' => 'Mary',
    'car' => 'Mazda',
  ),
  121 => array (
    'name' => 'Kathy',
    'car' => 'Mazda',
  ),
)*/
```

Note that first column selected have to be unique (in this query it is assumed that first column is id, but to be sure better list it explicitly).

Getting rows grouped by some field

`PDO::FETCH_GROUP` will group rows into a nested array, where indexes will be unique values from the first columns, and values will be arrays similar to ones returned by regular `fetchAll()`. The following code, for example, will separate boys from girls and put them into different arrays:

```
 $data = $pdo->query('SELECT sex, name, car FROM users')->fetchAll(PDO::FETCH_GROUP);
array (
  'male' => array (
    0 => array (
      'name' => 'John',
      'car' => 'Toyota',
    ),
    1 => array (
      'name' => 'Mike',
      'car' => 'Ford',
    ),
  ),
  'female' => array (
    0 => array (
      'name' => 'Mary',
      'car' => 'Mazda',
    ),
    1 => array (
      'name' => 'Kathy',
      'car' => 'Mazda',
    ),
  ),
)
```

So, this is the ideal solution for such a popular demand like "group events by date" or "group goods by category". Some real life use cases:

- How to multiple query results in order to reduce the query number? (http://stackoverflow.com/q/35317555/285587)
- List records grouped by category name (http://stackoverflow.com/a/41263854/285587)

Other modes

Of course, there is a `PDO::FETCH_FUNC` for the functional programming fans.

More modes are coming soon.

Error handling. Exceptions

Although there are several error handling modes in PDO, the only proper one is `PDO::ERRMODE_EXCEPTION`. So, one ought to always set it this way, either by adding this line after creation of PDO instance,

```
$dbh->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
```

or as a connection option, as demonstrated in the example above. And this is *all* you need for the basic error reporting.

Reporting PDO errors

TL;DR:

Despite what all other tutorials say, **you don't need a `try..catch` operator to report PDO errors**. Catch an exception only if you have a handling scenario other than just reporting it. Otherwise just let it bubble up to a site-wide handler (note that you don't have to write one, there is a basic built-in handler in PHP, which is quite good).

The only exception (pun not intended) is the creation of the PDO instance, which in case of error might reveal the connection credentials (that would be the part of the stack trace). In order to hide them, we can wrap the connection code into a `try..catch` operator and then throw a new `ErrorException` that contains only the message but not the credentials.

A long rant on the matter:

Despite a widespread delusion, you should never catch errors to report them. A module (like a database layer) should not report its errors. This function has to be delegated to an application-wide handler. All we need is to raise an error (in the form of exception) - which we already did. That's all. Nor should you "*always wrap your PDO operations in a `try/catch`*" like the most popular tutorial from tutsplus recommends. Quite contrary, catching an exception should be rather an exceptional case (pun intended).

In fact, there is nothing special in PDO exceptions - they are errors all the same. Thus, you have to treat them exactly the same way as other errors. If you had an error handler before, you shouldn't create a dedicated one for PDO. If you didn't care - it's all right too, as PHP is good with basic error handling and will conduct PDO exceptions all right.

Exception handling is one of the problems with PDO tutorials. Being acquainted with exceptions for the first time when starting with PDO, authors consider exceptions dedicated to this library, and start diligently (but improperly) handling exceptions for PDO only. This is utter nonsense. If one paid no special attention to any exceptions before, they shouldn't have changed their habit for PDO. If one didn't use `try..catch` before, they should keep with that, eventually learning how to use exceptions and when it is suitable to catch them.

So now you can tell that the PHP manual is wrong, stating that (http://php.net/manual/en/pdo.connections.php)

> If your application does not catch the exception thrown from the PDO constructor, the default action taken by the zend engine is to terminate the script and display a back trace. This back trace will likely reveal the full database connection details, including the username and password.

However, **there is no such thing as "the displaying of a back trace"!** What zend engine *really* does is just convert an uncaught exception into a fatal error. And then this fatal error is treated **like any other error** - so it will be displayed only if appropriate `php.ini` directive is set. Thus, although you may or you may not catch an exception, it has absolutely nothing to do with displaying sensitive information, because it's **a totally different configuration setting** in response to this. So, do not catch PDO exceptions to report them. Instead, configure your server properly:

On a development server just turn displaying errors on:

```
 ini_set('display_errors', 1);
```

While on a production server turn displaying errors off while logging errors on:

```
 ini_set('display_errors', 0);
ini_set('log_errors', 1);
```

- keep in mind that *there are other errors that shouldn't be revealed to the user as well.*

Catching PDO exceptions

You may want to catch PDO errors only in two cases:

1. If you are writing a wrapper for PDO, and you want to augment the error info with some additional data, like query string. In this case, catch the exception, gather the required information, and **re-throw another Exception**.
2. If you have a **certain scenario** for handling errors in the particular part of code. Some examples are:
    - if the error can be bypassed, you can use try..catch for this. However, do not make it a habit. Empty catch in every aspect works as error suppression operator, and so equally evil it is (http://programmers.stackexchange.com/questions/219788/is-error-suppressing-bad-practice).
    - if there is an action that has to be taken in case of failure, i.e. transaction rollback.
    - if you are waiting for a particular error to handle. In this case, catch the exception, see if the error is one you're looking for, and then handle this one. Otherwise just throw it again - so it will bubble up to the handler in the usual way.

E.g.:

```
 try {
    $pdo->prepare("INSERT INTO users VALUES (NULL,?,?,?,?)")->execute($data);
} catch (PDOException $e) {
    $existingkey = "Integrity constraint violation: 1062 Duplicate entry";
    if (strpos($e->getMessage(), $existingkey) !== FALSE) {

        // Take some action if there is a key constraint violation, i.e. duplicate name
    } else {
        throw $e;
    }
}
```

However, in general, no dedicated treatment for PDO exceptions is ever needed. In short, to have PDO errors properly reported:

1. Set PDO in exception mode.
2. Do not use `try..catch` to report errors.
3. Configure PHP for proper error reporting
    - on a **live** site set `display_errors=off` and `log_errors=on`
    - on a **development** site, you may want to set `display_errors=on`
    - of course, `error_reporting` has to be set to E_ALL in both cases

As a result, you will be always notified of all database errors without a single line of extra code! Further reading (/try-catch).

Getting row count with PDO

You don't needed it.

Although PDO offers a function for returning the number of rows found by the query, `PDOstatement::rowCount()`, you scarcely need it. Really.

If you think it over, you will see that this is a most misused function in the web. Most of time it is used not to *count* anything, but as a mere flag - just to see if there was any data returned. But for such a case you have the data itself! Just get your data, using either `fetch()` or `fetchAll()` - and it will serve as such a flag all right! Say, to see if there is any user with such a name, just select a row:

```
 $stmt = $pdo->prepare("SELECT 1 FROM users WHERE name=?");
$stmt->execute([$name]);
$userExists = $stmt->fetchColumn();
```

Exactly the same thing with getting either a single row or an array with rows:

```
 $data = $pdo->query("SELECT * FROM table")->fetchAll();
if ($data) {
    // You have the data! No need for the rowCount() ever!
}
```

Remember that here you don't need the *count*, the actual number of rows, but rather a boolean flag. So you got it.

Not to mention that the second most popular use case for this function should never be used at all. One should never use the `rowCount()` to count rows in database! Instead, one has to ask a database to count them, and return the result in a **single** row:

```
 $count = $pdo->query("SELECT count(1) FROM t")->fetchColumn();
```

is the only proper way.

In essence:

- if you need to know how many rows in the table, use `SELECT COUNT(*)` query.
- if you need to know whether your query returned any data - check that data.
- if you still need to know how many rows has been returned by some query (though I hardly can imagine a case), then you can either use `rowCount()` or simply call `count()` on the array returned by `fetchAll()` (if applicable).

Thus you could tell that the top answer for this question on Stack Overflow (http://stackoverflow.com/a/883382/285587/) is essentially pointless and harmful - a call to `rowCount()` could be never substituted with `SELECT count(*)` query - their purpose is essentially different, while running an extra query only to get the number of rows returned by other query makes absolutely no sense.

Affected rows and insert id

PDO is using the same function for returning both number of rows returned by SELECT statement and number of rows affected by DML (Data Manipulation Language, INSERT, UPDATE and DELETE queries) queries - `PDOstatement::rowCount()`. Thus, to get the number of rows affected, just call this function after performing a query.

Another frequently asked question is caused by the fact that mysql won't update the row, if new value is the same as old one. Thus number of rows affected could differ from the number of rows matched by the WHERE clause. Sometimes it is required to know this latter number.

Although you can tell `rowCount()` to return the number of rows matched instead of rows affected by setting `PDO::MYSQL_ATTR_FOUND_ROWS` option to TRUE, but, as this is a connection-only option and thus you cannot change it's behavior during runtime, you will have to stick to only one mode for the application, which could be not very convenient.

Note that `PDO::MYSQL_ATTR_FOUND_ROWS` is not guaranteed to work, as it's described in the comment (https://phpdelusions.net/pdo#comment-276) below.

Unfortunately, there is no PDO counterpart for the `mysql(i)_info()` function which output can be easily parsed and desired number found. This is one of minor PDO drawbacks.

An auto-generated identifier from a sequence or auto_inclement field in mysql can be obtained from the PDO::lastInsertId (http://php.net/manual/en/pdo.lastinsertid.php) function. An answer to a frequently asked question, "whether this function is safe to use in concurrent environment?" is positive: yes, it is safe. Being just an interface to MySQL C API mysql_insert_id() (http://dev.mysql.com/doc/refman/5.7/en/mysql-insert-id.html) function it's perfectly safe.

Prepared statements and LIKE clause

Despite PDO's overall ease of use, there are some gotchas anyway, and I am going to explain some.

One of them is using placeholders with `LIKE` SQL clause. At first one would think that such a query will do:

```
$stmt = $pdo->prepare("SELECT * FROM table WHERE name LIKE '%?%'");
```

but soon they will learn that it will produce an error. To understand its nature one has to understand that, like it was said above, *a placeholder have to represent a complete data literal only* - a string or a number namely. And by no means can it represent either a part of a literal or some arbitrary SQL part. So, when working with LIKE, we have to prepare our **complete literal** first, and then send it to the query the usual way:

```
 $search = "%$search%";
$stmt  = $pdo->prepare("SELECT * FROM table WHERE name LIKE ?");
$stmt->execute([$search]);
$data = $stmt->fetchAll();
```

Prepared statements and IN clause

Just like it was said above, it is impossible to substitute an arbitrary query part with a placeholder. Any string you bind through a placeholder will be put into query as a single string literal. For example, a *string* `'1,2,3'` will be bound as a *string*, resulting in

```
SELECT * FROM table WHERE column IN ('1,2,3')
```

making SQL to search for just *one* value.

To make it right, one needs separated values, to make a query look like

```
SELECT * FROM table WHERE column IN ('1','2','3')
```

Thus, for the comma-separated values, like for `IN()` SQL operator, one must create a set of `?`s manually and put them into the query:

```
 $arr = [1,2,3];
$in  = str_repeat('?,', count($arr) - 1) . '?';
$sql = "SELECT * FROM table WHERE column IN ($in)";
$stm = $db->prepare($sql);
$stm->execute($arr);
$data = $stm->fetchAll();
```

Not very convenient, but compared to mysqli it's *amazingly* concise (https://phpdelusions.net/pdo/mysqli_comparison#in).

In case there are other placeholders in the query, you could use `array_merge()` function to join all the variables into a single array, adding your other variables in the form of arrays, in the order they appear in your query:

```
 $arr = [1,2,3];
$in  = str_repeat('?,', count($arr) - 1) . '?';
$sql = "SELECT * FROM table WHERE foo=? AND column IN ($in) AND bar=? AND baz=?";
$stm = $db->prepare($sql);
$params = array_merge([$foo], $arr, [$bar, $baz]);
$stm->execute($params);
$data = $stm->fetchAll();
```

In case you are using named placeholders, the code would be a little more complex, as you have to create a sequence of the named placeholders, e.g. `:id0,:id1,:id2`. So the code would be:

```
 // other parameters that are going into query
$params = ["foo" => "foo", "bar" => "bar"];

$ids = [1,2,3];
$in = "";
foreach ($ids as $i => $item)
{
    $key = ":id".$i;
    $in .= "$key,";
    $in_params[$key] = $item; // collecting values into key-value array
}
$in = rtrim($in,","); // :id0,:id1,:id2

$sql = "SELECT * FROM table WHERE foo=:foo AND id IN ($in) AND bar=:bar";
$stm = $db->prepare($sql);
$stm->execute(array_merge($params,$in_params)); // just merge two arrays
$data = $stm->fetchAll();
```

Luckily, for the named placeholders we don't have to follow the strict order, so we can merge our arrays in any order.

Protecting table and field names

On Stack Overflow I've seen overwhelming number of PHP users implementing the most fatal PDO code (/pdo/lame_update), thinking that only data values have to be protected. But of course it is not.

Unfortunately, PDO has no placeholder for identifiers (table and field names), so a developer must manually filter them out. Such a filter is often called a "white list" (where we only list allowed values) as opposed to a "black list" where we list disallowed values. Here is a brief example

```
 $orders  = ["name","price","qty"]; // the white list of allowed fierld names
$key     = array_search($_GET['sort'], $orders); // see if we have such a name
$orderby = $orders[$key]; //if not, first one will be set automatically. smart enuf :)
```

the same approach should be used for the direction, although the code would be a bit simpler

```
 $direction = _GET['direction'] == 'DESC' ? 'DESC' : 'ASC';
```

having gottent these two variables this way will make them 100% safe

```
 $query   = "SELECT * FROM `table` ORDER BY $orderby $direction"; // sound and safe
```

The same approach must be used every time a table of a field name is going to be used in the query.

A problem with LIMIT clause

Another problem is related to the SQL `LIMIT` clause. When in emulation mode (which is on by default), PDO substitutes placeholders with actual data, instead of sending it separately. And with "lazy" binding (using array in `execute()`), PDO treats every parameter as a string. As a result, the prepared `LIMIT ?,?` query becomes `LIMIT '10', '10'` which is invalid syntax that causes query to fail.

There are two solutions:

One is turning emulation off (as MySQL can sort all placeholders properly). To do so one can run this code:

```
$conn->setAttribute( PDO::ATTR_EMULATE_PREPARES, false );
```

And parameters can be kept in `execute()`:

```
$conn->setAttribute( PDO::ATTR_EMULATE_PREPARES, false );
$stmt = $pdo->prepare('SELECT * FROM table LIMIT ?, ?');
$stmt->execute([$offset, $limit]);
$data = $stmt->fetchAll();
```

Another way would be to bind these variables explicitly while setting the proper param type:

```
$stmt = $pdo->prepare('SELECT * FROM table LIMIT ?, ?');
$stmt->bindParam(1, $offset,PDO::PARAM_INT);
$stmt->bindParam(2, $limit,PDO::PARAM_INT);
$stmt->execute();
$data = $stmt->fetchAll();
```

One peculiar thing about `PDO::PARAM_INT`: for some reason it does not enforce the type casting. Thus, using it on a number that has a string type will cause the aforementioned error:

```
$stmt = $pdo->prepare("SELECT 1 LIMIT ?");
$stmt->bindValue(1, "1", PDO::PARAM_INT);
$stmt->execute();
```

But change `"1"` in the example to `1` - and everything will go smooth.

Transactions

To successfully run a transaction, you have to make sure that error mode is set to exceptions, and learn three canonical methods:

- `beginTransaction()` to start a transaction
- `commit()` to commit one
- `rollback()` to cancel all the changes you made since transaction start.

Exceptions are essential for transactions because they can be caught. So in case one of the queries failed, the execution will be stopped and moved straight to the catch block, where the whole transaction will be rolled back.

So a typical example would be like

```
 try {
    $pdo->beginTransaction();
    $stmt = $pdo->prepare("INSERT INTO users (name) VALUES (?)");
    foreach (['Joe','Ben'] as $name)
    {
        $stmt->execute([$name]);
    }
    $pdo->commit();
}catch (Exception $e){
    $pdo->rollback();
    throw $e;
}
```

Please note the following important things:

- PDO error reporting mode should be set to `PDO::ERRMODE_EXCEPTION`
- you have catch an `Exception`, not `PDOException`, as it doesn't matter what particular exception aborted the execution.
- you should re-throw an exception after rollback, to be notified of the problem the usual way.
- also make sure that a table engine supports transactions (i.e. for Mysql it should be InnoDB, not MyISAM)
- there are no Data definition language (DDL) statements that define or modify database schema among queries in your transaction, as such a query will cause an implicit commit

Calling stored procedures in PDO

There is one thing about stored procedures any programmer stumbles upon at first: every stored procedure always returns **one extra result set**: one (or many) results with actual data and one just empty. Which means if you try to call a procedure and then proceed to another query, then **"Cannot execute queries while other unbuffered queries are active"** error will occur, because you have to clear that extra empty result first. Thus, after calling a stored procedure that is intended to return only one result set, just call `PDOStatement::nextRowset()` (http://php.net/manual/en/pdostatement.nextrowset.php) once (of course after fetching all the returned data from statement, or it will be discarded):

```
 $stmt = $pdo->query("CALL bar()");
$data = $stmt->fetchAll();
$stmt->nextRowset();
```

While for the stored procedures returning many result sets the behavior will be the same as with multiple queries execution:

```
 $stmt = $pdo->prepare("CALL foo()");
$stmt->execute();
do {
    $data = $stmt->fetchAll();
    var_dump($data);
} while ($stmt->nextRowset() && $stmt->columnCount());
```

However, as you can see here is another trick have to be used: remember that extra result set? It is so essentially *empty* that even an attempt to fetch from it will produce an error. So, we cannot use just `while ($stmt->nextRowset())`. Instead, we have to check also for empty result. For which purpose

`PDOStatement::columnCount()` is just excellent.

This feature is one of essential differences between old mysql ext and modern libraries: after calling a stored procedure with `mysql_query()` there was no way to continue working with the same connection, because there is no `nextResult()` function for `mysql ext`. One had to close the connection and then open a new one again in order to run other queries after calling a stored procedure.

Calling a stored procedure is a rare case where `bindParam()` use is justified, as it's the only way to handle `OUT` and `INOUT` parameters. The example can be found in the corresponding manual chapter (http://php.net/manual/en/pdo.prepared-statements.php#example-1009). However, **for mysql it doesn't work.** You have to resort to an SQL variable and an extra call (http://stackoverflow.com/a/23749445/285587).

Note that for the different databases the syntax could be different as well. For example, to run a sored procedure against Microsoft SQL server, use the following format

```
$stmt = $pdo->prepare("EXEC stored_procedure ? ?");
```

where ? marks are placeholders. Note that no braces should be used in the call.

Running multiple queries with PDO

> Note there are no reasons to stuff multiple queries in a single call, and generally you don't need this functionality. Running queries one by one is equal in every way to running them in a batch. The only use case for this functionality I can think of is when you need to execute an existing SQL dump and check for the results.

When in emulation mode, PDO can run mutiple queries in the same statement, either via query() or `prepare()/execute()`. To access the result of consequent queries one has to use `PDOStatement::nextRowset()` (http://php.net/manual/en/pdostatement.nextrowset.php):

```
 $stmt = $pdo->prepare("SELECT ?;SELECT ?");
$stmt->execute([1,2]);
do {
    $data = $stmt->fetchAll();
    var_dump($data);
} while ($stmt->nextRowset());
```

Within this loop you'll be able to gather all the related information from the every query, like affected rows, auto-generated id or errors occurred.

It is important to understand that at the point of `execute()` PDO will report the error **for the first query only**. But if error occurred at any of consequent queries, to get that error one has to iterate over results. Despite some ignorant opinions (https://bugs.php.net/bug.php?id=61613), PDO can not and should not report all the errors at once. Some people just cannot grasp the problem at whole, and don't understand that error message is not the only outcome from the query. There could be a dataset returned, or some metadata like insert id. To get these, one has to iterate over resultsets, one by one. But to be able to throw an error immediately, PDO would have to iterate automatically, and thus **discard some results**. Which would be a clear nonsense.

Unlike `mysqli_multi_query()` PDO doesn't make an asynchronous call, so you can't "fire and forget" - send bulk of queries to mysql and close connection, PHP will wait until last query gets executed.

Emulation mode. PDO::ATTR_EMULATE_PREPARES

One of the most controversial PDO configuration options is `PDO::ATTR_EMULATE_PREPARES`. What does it do? PDO can run your queries in two ways:

1. It can use a **real** or native prepared statement:
   When prepare() is called, your query with placeholders gets sent to mysql as is, with all the question marks you put in (in case named placeholders are used, they are substituted with ?s as well), while actual data goes later, when execute() is called.
2. It can use **emulated** prepared statement, when your query is sent to mysql as proper SQL, with all the data in place, **properly formatted**. In this case only one roundtrip to database happens, with `execute()` call. For some drivers (including mysql) emulation mode is turned `ON` by default.

Both methods has their drawbacks and advantages but, and - I have to stress on it - both being **equally secure**, if used properly. Despite rather appealing tone of the popular article on Stack Overflow (http://stackoverflow.com/a/12202218/285587), in the end it says that **if you are using supported versions of PHP and MySQL properly, you are 100% safe**. All you have to do is to set encoding in the DSN, as it shown in the example above, and your emulated prepared statements will be as secure as real ones.

Note that when native mode is used, **the data is never appears in the query**, which is parsed by the engine as is, with all the placeholders in place. If you're looking into Mysql query log for your prepared query, you have to understand that it's just an artificial query that has been created solely for logging purpose, but not a real one that has been executed.

Other issues with emulation mode as follows:

When emulation mode is turned ON

one can use a handy feature of named prepared statements - a placeholder with same name could be used any number of times in the same query, while corresponding variable have to be bound only once. For some obscure reason this functionality is disabled when emulation mode is off:

```
 $stmt = $pdo->prepare("SELECT * FROM t WHERE foo LIKE :search OR bar LIKE :search");
$stmt->execute(['search'] => "%$search%");`
```

Also, when emulation is `ON`, PDO is able to run multiple queries in one prepared statement.

Also, as native prepared statements support only certain query types, you can run some queries with prepared statements only when emulation is `ON`. The following code will return table names in emulation mode and error otherwise:

```
 $stmt = $pdo->prepare("SHOW TABLES LIKE ?");
$stmt->execute(["%$name%"]);
var_dump($stmt->fetchAll());
```

When emulation mode is turned OFF

One could bother not with parameter types, as mysql will sort all the types properly. Thus, even string can be bound to LIMIT parameters, as it was noted in the corresponding chapter.

Also, this mode will allow to use the advantage of single prepare-multiple execute feature.

It's hard to decide which mode have to be preferred, but for usability sake I would rather turn it `OFF`, to avoid a hassle with `LIMIT` clause. Other issues could be considered negligible in comparison.

Mysqlnd and buffered queries. Huge datasets.

Recently all PHP extensions that work with mysql database were updated based on a low-level library called `mysqlnd`, which replaced old `libmysql` client. Thus some changes in the PDO behavior, mostly described above and one that follows:

There is one thing called buffered queries (http://php.net/manual/en/mysqlinfo.concepts.buffering.php). Although you probably didn't notice it, you were using them all the way. Unfortunately, here are bad news for you: unlike old PHP versions, where you were using buffered queries virtually for free, modern versions built upon mysqlnd driver (http://php.net/manual/en/book.mysqlnd.php) won't let you to do that anymore:

> When using libmysqlclient as library PHP's memory limit won't count the memory used for result sets unless the data is fetched into PHP variables. **With mysqlnd the memory accounted for will include the full result set.**

The whole thing is about a resultset, which stands for all the data found by the query.

When your SELECT query gets executed, there are two ways to deliver the results in your script: buffered and unbuffered one. When buffered method is used, all the data returned by the query **gets copied in the script's memory** at once. While in unbuffered mode a database server feeds the found rows one by one.

So you can tell that in buffered mode a resultset is always burdening up the memory on the server *even if fetching weren't started at all*. Which is why it is not advisable to select huge datasets if you don't need all the data from it.

Nonetheless, when old libmysql-based clients were used, this problem didn't bother PHP uers too much, because the memory consumed by the resultset didn't count in the the `memory_get_usage()` and `memory_limit`.

But with mysqlnd things got changed, and the resultset returned by the buffered query will be count towards both `memory_get_usage()` and `memory_limit`, no matter which way you choose to get the result:

```
 $pdo->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, FALSE);
$stmt = $pdo->query("SELECT * FROM Board");
$mem = memory_get_usage();
while($row = $stmt->fetch());
echo "Memory used: ".round((memory_get_usage() - $mem) / 1024 / 1024, 2)."M\n";

$pdo->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, TRUE);
$stmt = $pdo->query("SELECT * FROM Board");
$mem = memory_get_usage();
while($row = $stmt->fetch());
echo "Memory used: ".round((memory_get_usage() - $mem) / 1024 / 1024, 2)."M\n";
```

will give you (for my data)

```
 Memory used: 0.02M
Memory used: 2.39M
```

which means that with buffered query the memory is consumed **even if you're fetching rows one by one!**

So, keep in mind that if you are selecting a really huge amount of data, always set `PDO::MYSQL_ATTR_USE_BUFFERED_QUERY` to `FALSE`.

Of course, there are some drawbacks, two minor ones:

1. With unbuffered query you can't use `rowCount()` method (which is useless, as we learned above)
2. Moving (seeking) the current resultset internal pointer back and forth (which is useless as well).

And a rather important one:

1. While an unbufferered query is active, you cannot execute any other query using the same connection, so use this mode wisely.

RELATED ARTICLES:

> Simple yet efficient PDO wrapper (/pdo/pdo_wrapper)

> Usability problems of mysqli compared to PDO (/pdo/mysqli_comparison)

> PDO Fetch Modes (/pdo/fetch_modes)

> An SQL injection against which prepared statements won't help (/pdo/sql_injection_example)

> Your first database wrapper's childhood diseases (/pdo/common_mistakes)

> Fetching objects with PDO (/pdo/objects)

> MCVE or How to debug database interactions with PDO (/pdo/mcve)

> Authenticating a user using PDO and password_verify() (/pdo/password_hash)

> A cargo cult prepared statement (/pdo/cargo_cult_prepared_statement)

> Whitelisting helper function (/pdo/whitelisting_helper_function)

GOT A QUESTION?

I am the only person to hold a gold badge in pdo (http://stackoverflow.com/help/badges/4220/pdo), mysqli (http://stackoverflow.com/help/badges/6342/mysqli) and sql-injection (http://stackoverflow.com/help/badges/5981/sql-injection) on Stack Overflow and I am eager to show the right way for PHP developers.
Besides, your questions let me make my articles even better, so you are more than welcome to ask any question you got.

Click here to ask!

LATEST ARTICLE:
PHP Error Reporting (/articles/error_reporting)

SEE ALSO:

> Top 10 PHP delusions (/top)

> PDO Examples (/pdo_examples)

> Mysqli Examples (/mysqli_examples)

> Simple yet efficient PDO wrapper (/pdo/pdo_wrapper)

> Usability problems of mysqli compared to PDO (/pdo/mysqli_comparison)

> PDO Fetch Modes (/pdo/fetch_modes)

> An SQL injection against which prepared statements won't help (/pdo/sql_injection_example)

> Your first database wrapper's childhood diseases (/pdo/common_mistakes)

> Fetching objects with PDO (/pdo/objects)

> MCVE or How to debug database interactions with PDO (/pdo/mcve)

> Authenticating a user using PDO and password_verify() (/pdo/password_hash)

> A cargo cult prepared statement (/pdo/cargo_cult_prepared_statement)

> Whitelisting helper function (/pdo/whitelisting_helper_function)

LATEST COMMENTS:

18.04.20 00:46
Lonestar Jack for How to connect to MySQL using PDO: (/pdo_examples/connect_to_mysql#comment-846)
What is he best coding to use two queries on one page? One connection and then close it and do...
read more (/pdo_examples/connect_to_mysql#comment-846)

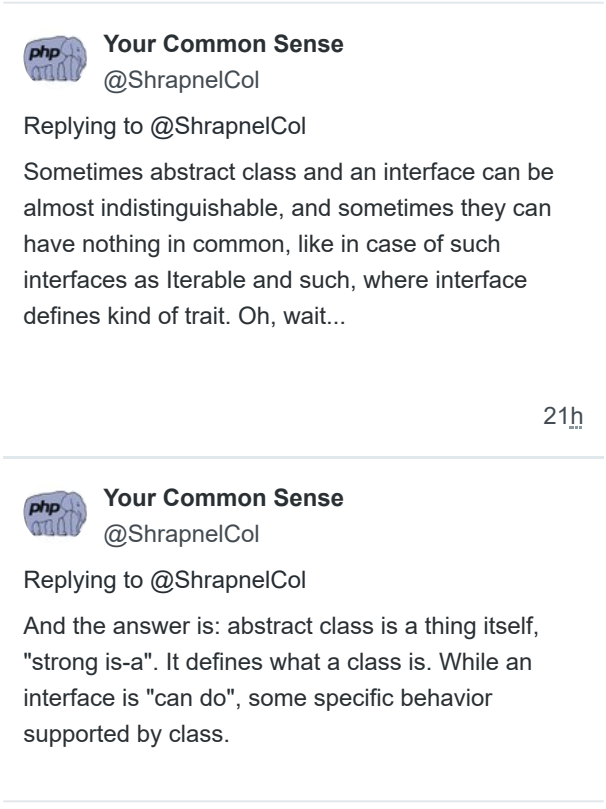
17.04.20 19:02
Gregory Stathes for How to use mysqli properly: (/mysqli#comment-845)

My issue is this, been trying to solve for days. And I am more than happy to pay for a solution,...
read more (/mysqli#comment-845)

16.04.20 20:47
Phil for How to use mysqli properly: (/mysqli#comment-844)
Which is better for performance mysqli or PDO?
read more (/mysqli#comment-844)

12.04.20 04:28
eljaydee for Mysqli examples: (/mysqli_examples#comment-843)
I cannot make this work: I have been told a million times to "review you query" but cannot find...
read more (/mysqli_examples#comment-843)

12.04.20 04:27
Elizabeth for PDO Examples: (/pdo_examples#comment-842)
Hi, my php programs which worked fine on my local server are now no longer passing through...
read more (/pdo_examples#comment-842)



Add a comment

Please refrain from sending spam or advertising of any sort.
Messages with hyperlinks will be pending for moderator's review.

Markdown is now supported:

- > before and an empty line after for a quote

- four spaces to mark a block of code

Your name

Are you a robot?

Message

Address

If you want to get a reply from admin, you may enter your E—mail address above

Send

Comments:

Wanze, 26.03.20 01:14

> How does inserting file into mysql table work from .csv file. I have a pdo db. Thanks.

AnrDaemon, 23.03.20 02:59

> @Ingus, did you run same query in mysql console by hands?

Ingus, 20.03.20 22:43

> Hello there! Could you help me understand why this query is not showing any row?

```
 $Query = $main->con->prepare("SELECT * FROM earnings WHERE user = ? ORDER BY id DESC LIMI
T ?, ?");
$Query->execute([$_SESSION['UID'], $start, $limit]);
while($row = $Query->fetch()){
      ...
}
```

Michael, 18.03.20 21:10

Hi - Thank you for the excellent site and invaluable information! I'm following your advice in using prepared statements to lessen any risk on a site I'm building. I would like to use a prepared PDO statement for an INSERT...ON DUPLICATE KEY and an example would be very much appreciated. Thank you.

James Miller, 13.03.20 00:00

I see not examples of an UPDATE stored procedure in PDO. I have tried to make this work, but have failed. I have not problem with UPDATE PDO, but not calling a sproc for an update. Is this possible or is it nonsensical to try and use?

Coulston, 10.03.20 12:59

How can I create two forms on one page that submits form data in two different scripts; Let's say a login page and a registration page that submits data on two different scripts, one for login and another for registration

Noel, 02.03.20 13:01

Hey, great article, well done. Im not sure if i missed it but how do you close the connection and statement? In mysql you close the database connection and statement, is that something you do with PDO or did i miss something.

Sergio A. Kessler, 22.02.20 00:43

ok, what about this:

```
 // the list of allowed field names
$allowed = ["name","surname","email"];

$primary_key_name = 'id';
$primary_key_value = 48;

/**********************************************
 * update the record
 */
$set = implode('=?, ', array_keys(array_intersect_key($new_row, $allowed))) . '=?';

$sql = "update table1 set $set where $primary_key_name = ?";

$sql_params = array_values($new_row);
$sql_params[] = $primary_key_value;

$st = $db->prepare($sql);
$st->execute($sql_params);
/*
 * /update the record
 **********************************************/
```

Sergio A. Kessler, 19.02.20 16:17

WRT:

```
 $in  = str_repeat('?,', count($arr) - 1) . '?';
```

if $arr is empty, you still get a '?', I use:

```
 $in  = implode(',', array_fill(0, count($arr), '?'));
```

where if $arr is empty, $in is also empty

REPLY:

If array is empty then empty in() clause will cause an SQL error, so it has to be checked before the query execution.

Sergio A. Kessler, 19.02.20 03:13

I like to use:

```
  /*****************************************
 * update the record
 */
$set = implode('=?, ', array_keys($new_row)) . '=?';

$sql = "update centro_donacion set $set where $primary_key_name = ?";

$sql_params = array_values($new_row);
$sql_params[] = $primary_key_value;

$st = $db->prepare($sql);
$st->execute($sql_params);
/*
* /update the record
*****************************************/
```

REPLY:

Hello Sergio!

With this code you've got two problems:

- potential but **very real** security breach. Field names / array keys being added to the query **absolutely untreated**
- a possible formatting issue. the field name may happen to coincide with mysql reserved word and it will result in the query error.

I've got an article that covers both problems, please check it out:
https://phpdelusions.net/pdo_examples/dynamical_update
(https://phpdelusions.net/pdo_examples/dynamical_update)

GioMBG, 09.02.20 20:53

Hi Master, for the purpose of security, on a machine that connects to the dbs only locally, does it make sense to read the users dbs with a connection other than the one used to read other dbs ?? now therefore I make 2 types of different connections ($conn1 & $conn2) through 2 db.inc.php that I carry inside my functions through : global $conn1, $conn2... now I made some changes and managing 2 different types of connections became really problematic, and I think to understand the bad pratice of call the connections with global... so I was wondering if it made sense to keep them separate or go MUCH simpler keeping one for all without having problem recalling one connection only in my functions with global $conn... suggestions ? always thanks Gio

REPLY:

Hello Gio!

I don't see any reason to have an extra connections. As a rule, every website is using just a single connection. I have a feeling that anyone who recommends an extra connection for the extra security never used it in the real life.

John Thompson, 31.01.20 06:06

I'm trying to make my PDO code portable. With mysqli you can have you database, host, username/password in a db.inc or db.php then include it when you start your code. I am not finding that I'm able to do that. For example:

db.inc/db.php

```
 $host = "Server";
$database = "Cats";
$user = "Frank";
$password = "something";
```

On the php file to do the action:

```
include(db.php);
```

Then call your variables and run the mysqli connection then the query. If I try that in PDO I get errors. Is this just not done? Or there's a proper way to have 1 file with my DSN, username, password and database name?

Thank You in advance,

REPLY:

Hello John!

Of course it can be done.

The error is not some imminent but probably some mistake in your implementation. A few problems I can spot right away:

- there is no opening php tag in db.php. the first line should be <?php
- there are missing quotes when you call include. it should be include 'db.php';
- dubious path to db.php. Make sure you are providing a correct *absolute* path to a file. Check out the relevant article: https://phpdelusions.net/articles/paths (https://phpdelusions.net/articles/paths)

If you still encounter some errors, try to google them - usually there are many answers on Stack Overflow explainig the problem.

Charles Henderson, 20.01.20 21:53

Ok, first let me just say, I am not asking you to explain to me how to do this, just asking if it is possible. I asked if anyone could point me in the right direction on how to do it on Stack Exchange and was told that the answer to that question could fill books lol. So basically I just want to know if it is possible to do, and if it is, I will roll up my sleeves and do a lot of reading.

Basically we currently have a customer invoicing program made in Access, forms built in Access and using Access database. There are 2 main parts, the customer info and the invoice itself. It looks like there is a customer table and an invoice table. When an invoice is made it links to the customer by the incremental ID of the invoice. Basically I would like to make a web form that has all the information on one page. Top has customer info, and if it is a new customer the info can put inputted and submitted to database, if it is a return customer then the customer can be looked up by customer ID, license number, or lastname, and the rest of the info autofilled from the record. Then the invoice can be filled out and submitted, linking the invoice to the customer. In the end an invoice can be saved, emailed, or printed. Does that sound like something that is possible to make? If I did not do a good job explaining I can send you screenshots of what we are currently using and of what I would like the new one to look like. Thank you in advance for any help you can offer.

REPLY:

Hello Charles!

Well, everything is doable, the question is the amount of effort you are willing to spend.

The easiest way would be to ditch everything you have now and rebuild the system using PHP HTML and SQLite. For this stack the task is of low to moderate complexity, can be done in a month if you have no prior experience.

If you still need the connection to your existing Access database, it would be tricky to achieve and clumsy to maintain. And I've got no solution from top off my head right now.

Ingus, 10.01.20 13:14

Replaying to your last reply (Ingus, 10.01.20 11:16): Thanks for fetch() fix!

But what if i had like this:

```
 if(mysqli_num_rows($result)==0){ //do something }
if(mysqli_num_rows($result)==1){ //do something else }
if(mysqli_num_rows($result)==5){ //do something else }
```

How to get same result with PDO?

REPLY:

I can't even think where you would need such a code but you can fetch all rows into array using fetchAll() and then count them

```
 $rows = $result->fetchAll();
$count = count($rows);
if($count == 0){ //do something }
elseif($count == 1){ //do something else }
elseif($count == 5){ //do something else }
```

But wait. **If you need only count, but not the data itself** then select only count.

```
 $stmt = $pdo->prepare("SELECT count(*) FROM my_table WHERE id=? AND username=? AND email
=? ORDER BY id DESC");
$stmt->execute([$val1,$val2,$val3]);
$count = $stmt->fetchColumn();
if($count == 0){ //do something }
```

Ingus, 10.01.20 11:16

I m having issue converting mysqli to PDO

My code:

```
 $result = $pdo->prepare("SELECT status, date FROM my_table WHERE id=? AND username=? AND em
ail=? ORDER BY id DESC");
$result->execute([$val1,$val2,$val3]);
$row = $result->fetchColumn();
```

I had for mysqli:

```
 if ((mysqli_num_rows($result)==0)||(time()-strtotime($date)>120)){  //DO SOMETHING  }
```

The question is how to make same effect with count returned lines with PDO?

REPLY:

Hello Ingus!

I am not sure what are you doing here but at least I can tell that fetchColumn() is not suitable for such a query, because you are selecting two columns, not one.

Change the last line to

```
 $row = $result->fetch();
```

and then you can use this $row variable instead of mysqli_num_rows() because not empty $row means your query returned something:

```
  if ($row ||(time()-strtotime($date)>120)){  //DO SOMETHING  }
```

Here you can use status, $row['date'] and $row['status']. If $date above is the date from $row then just write $row['date'] instead of $date.

Ingus, 07.01.20 11:39

Hello, first of all thanks for such a good tutorial! It teached me allot!

I m about to make new script and i m going to use PDO! Clients that will use this script have different wishes about database type. .. One want SQL other want MySQL so i have question! Can i make script that allows both database structures and if database type would be needed to change all i had to do would be change connection string?

Thanks in advance!

REPLY:

Hello Ingus,

As a rule, it is not possible because of different SQL flavors. For example in SQL Server there is no LIMIT operator, but there is TOP instead and so on.

It means you have to test all your queries for all servers.

For a a true database-agnostic application look for some DBAL, such as Doctrine or Eloquent.

steve, 06.01.20 07:09

Just wanted to say THANK YOU for this! Still working through it, but the explanations for everything is so much better than YouTube videos and other tuts that just give code for one way that works. I feel like I'm actually learning this stuff (and the RIGHT way), and not just copying code.

Thanks again!

REPLY:

Hello Steve!

Thank you for your feedback. Such a response is a thing that keeps this site going.

Please keep in mind that best gratitude is questions. Do not hesitate to ask for clarification or an example. It will help me to make this site better.

grotto, 05.12.19 20:44

I'm having some issues with a particular file that I'm updating to pdo and I'm suspicious that it's related to the fact that some of the DB columns are bigint. Could you please elaborate on the bigint idiosyncrasies you mentioned in your main article and how to deal with them. I'm also looking for a place to get help with my pdo issues but I've gotten bad response from asking questions on SO and I'm not sure where else might be a good source for help. Any suggestions would be greatly appreciated!

Todd Loomis, 30.11.19 23:23

How does this work with fulltext searches? I'm using mysql and PHP, and I've been trying to create prepared statements, but cannot get it to work:

```
 if (((strpos($keywords,'+')) !== false) || ((strpos($keywords,'-')) !== false)) {
$booleaninsert = " IN BOOLEAN MODE";
} else {
$booleaninsert = " IN NATURAL LANGUAGE MODE";
}

$search = "%$keywords%";
// $stmt  = $pdo->prepare("SELECT * FROM table WHERE name LIKE ?");
$stmt  = $pdo->prepare("SELECT *, MATCH(modelnumber, mfg, releasedate, code) AGAINST (? " .
$booleaninsert . ") AS score FROM songs WHERE MATCH(modelnumber, mfg, releasedate, code) AGA
INST (? " . $booleaninsert . ")");
$stmt->execute([$search]);
$data = $stmt->fetchAll();
```

Thanks! Todd

Ted, 24.11.19 22:41

I'm extremely confused.

What the heck do you mean by "here is an example for mysql"? Do I need to use this code if I'm using a msql database? What does most of this code mean?

REPLY:

Hello Ted!

Sorry for the late answer.

I am not sure what do you mean with "msql". As far as I know, PDO doesn't support such a database. May be it's mysql or MS SQL server? For the latter you will need different connection code, see https://www.php.net/manual/en/ref.pdo-dblib.php

(https://www.php.net/manual/en/ref.pdo-dblib.php)

And what does this code mean is fully explained in another article,
https://phpdelusions.net/pdo_examples/connect_to_mysql
(https://phpdelusions.net/pdo_examples/connect_to_mysql)

Ricardo Miller, 14.11.19 00:22

Hiya thanks for your knowledge on the PDO its helping me massively understand PDO better but I am having a issue.

```
 private function __construct()
    {
        try {
            $params = [
                PDO::ATTR_EMULATE_PREPARES => false, // turn off emulation mode for "real" p
repared statements
               PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION, //turn on errors in the form o
f exceptions
               PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC, //make the default fetch b
e an associative array
           ];
        self::$_dbh = @new PDO(
            Config::DB_DRIVER . ':host=' . Config::DB_HOST . ';dbname=' . Config::DB_NAME .
';charset=' . Config::DB_CHARSET,
            Config::DB_USER,
            Config::DB_PASSWORD,
            /* PDO options */
            $params
        );
        //self::$_dbh->setAttribute( PDO::ATTR_EMULATE_PREPARES, false );

    } catch (Exception $e) {
        // If we can't established a database connection return and error
        DBWrapper::fatal("
            An error occurred while connecting to the database " .
            Config::DB_NAME . "The error reported by the server was:" . $e->getMessage()
        );
    }
    return self::$_dbh;
}
```

REPLY:

Hello Ricardo.

In order to have the code formatter you must pad it with 4 spaces from the left and an **empty line from the top**.

So what your issue is?

Justin, 08.11.19 21:40

First, thanks for your insight into PDO and the creation of a wrapper. My wrapper wasn't too far off but was able to make some adjustments thanks to the information you have shared. I have also gone through the delusions article as well and it has also helped me reexamine parts of my code.

Secondly, I have a site running on PHP 5.4.45 with a custom created PDO wrapper as well as other custom classes I created to fulfill my needs. It has been running near flawlessly (except for some of hand PHP errors) for the past 4 or so years this way. Now however is the time to upgrade the server. After upgrading the server to 5.6 I started to experience major issues right away with MySQL. It seems that there was a major shift in code handling by PHP from 5.4 to 5.6 with the way it is handling DB connections. The server is now hating my code and opening way too many connections to MySQL and crashing the server. Do you know what change they made between these two versions that I may be overlooking?

Thanks in advance.

REPLY:

Hello Justin!

I cannot think of any reason related to PHP version. It should work exactly as before.

However, there are two most frequent reasons for too many connections.

First of all make sure you are no using a persistent connection.
Then make sure your wrapper connects only once during the script executions. Most likely it's the case.

James Miller, 01.11.19 01:22

Not a question, but a thank you and praise for your work. I cannot tell you how much simpler your code examples have made my attempts at being a programmer. You must have a paypal account or a manner to make contributions. If not you should. Once agains thank you.

Mohamed, 24.10.19 20:01

Please advise whether below method is correct, if I don't set to PDO::ATTR_PERSISTENT => TRUE, I don't know why it's not rolling back.

```
class Database {
   private $host = DB_HOST;
   private $user = DB_USER;
   private $pass = DB_PASS;
   private $dbname = DB_NAME;
   private $charset = 'utf8mb4';

   private $dbh;
   private $stmt;
   private $error;
   // public $table = NULL;
   /*-------------------------------------------------------------------------------*/

   public function __construct(){
       // Set DSN
       $dsn = 'mysql:host='.$this->host.';dbname='.$this->dbname;$this->charset;
       $options = array(
           PDO::ATTR_PERSISTENT => TRUE,
           PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
           // PDO::ATTR_EMULATE_PREPARES => false
       );

       // Create PDO instance
       try{
           $this->dbh = new PDO($dsn, $this->user, $this->pass, $options );
           } catch (PDOException $e){
           $this->error = $e->getMessage();
           // echo $this->error;
           die("Database Connection Failed " . $this->error);
       }
   }
```

Mohamed, 24.10.19 19:19

Hi, Thanks for your replay.

Transactions is not rolling back on mysql query failure, if PDO::ATTR_PERSISTENT => TRUE is not used.

Please advise the solution.

REPLY:

Hello Mohamed

The PDO::ATTR_PERSISTENT is not relevant to transactions. Most likely your transaction is not rolled back because the table in the database does not support them. Make sure the table is created with InnoDB engine. And also check out other prerequisites noted in the corresponding section: https://phpdelusions.net/pdo#transactions (https://phpdelusions.net/pdo#transactions)

Wim de Bok, 18.10.19 15:32

Finding valuable sources on the internet is never easy but I am glad I found this one. Great article on PHP and PDO, thank you!

Mohamed, 18.10.19 09:56

Please advise on usage on below, whether to use or not in transanctions

PDO::ATTR_PERSISTENT => TRUE,

REPLY:

Hello Mohamed!

The answer is very simple: just **never** use PDO::ATTR_PERSISTENT => TRUE unless you have a very skilled dedicated server admin who would tell you this option is necessary and who would then configure your database server, HTTP server and PHP properly.

In any other case this option will do you much more harm than good. Especially with transactions, as such an option will result in transactions extended between different scripts execution with unpredictable results.

Elizabeth, 17.10.19 12:11

Hi, I managed to solve my issue sent earlier today by replacing the while loop with [$user = $stmt->fetch();] which i grabbed from your site - thank you :)

REPLY:

Hello Elizabeth,

Thank you very much, such feedback means a lot!

Daniel Collin, 04.10.19 15:46

Hi, very good article! Thanx

Lynne K., 12.09.19 12:11

Hi, I'd like to use PDO to access a reviews database with mysql. I'm completely new to PDO and am upgrading from PHP5 to PHP7. The query I am having trouble with is:

```
$query = "select * from rsd_reviews order by new_date limit $offset, $limit";
```

Could you assist?

REPLY:

Hello Lynne!

Three things:

1. Always use **placeholders** in place of actual variables in youe query, just like it explained in the article.
2. Make sure $offset and $limit have correct values.
3. Make sure you are connecting to PDO properly (https://phpdelusions.net/pdo_examples/connect_to_mysql) and have PHP configured to report errors

In case it doesn't help, please get back with a more detailed description what particular trouble you are having.

Christopher, 06.09.19 01:05

What does PDO stand for? PHP Data Objects? Always give a definition for an acronym, if it is one.

pink, 21.08.19 06:31

> How is it different from your method?

```
 <?php
class DB {
private $pdo = null;
private $stmt = null;

function __construct(){
try {
  $this->pdo = new PDO(
    "mysql:host=localhost;dbname=test;charset=utf8",
    "root", "", [
      PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
      PDO::ATTR_EMULATE_PREPARES => false,
    ]
  );
} catch (Exception $ex) { die($ex->getMessage()); }
```

}

```
 function __destruct(){
if ($this->stmt!==null) { $this->stmt = null; }
if ($this->pdo!==null) { $this->pdo = null; }
```

}

function select($sql, $cond=null){ $result = false; try { $this->stmt = $this->pdo->prepare($sql); $this->stmt->execute($cond); $result = $this->stmt->fetchAll(); } catch (Exception $ex) { die($ex->getMessage()); } $this->stmt = null; return $result; } } ?>

Tachi, 14.08.19 16:34

You are great! I'm quite new to PHP and I'm renewing old code of a former colleague with PDO and I found your example with the IN clause very helpful. Many many thanks Here I have an example where the placeholder after the IN clause could be a single value or a range of values. I wanted to share. I hope it is useful for others.

```
 if ( is_array($groupID) ) {
    $in =  str_repeat("?,", count($groupID) - 1) . "?";
    $arr = $groupID;
} else {
    $in = "?";
    $arr = [$groupID];
}
$sql = "SELECT parent_groupID, child_groupID FROM table WHERE parent_groupID IN ($in) ORDE
R BY ordinal ASC";
$stmt = $dsn->prepare($sql);
$stmt->execute($arr);
$r = $stmt->fetchAll();
```

Chuck Robinson, 01.07.19 22:35

I noticed something weird about pdo ssl, if you misspell the key name values such as PDO::MYSQL_ATTR_SSL_KEY => 'mispeled key name' it still works which is kinda strange.

Joe, 08.06.19 14:28

I have issue with single quote ' I can't see the word like Don't or won't instead of don't I can see "don" missing apostrophe and t how do I fix it? Thank you

```
 <?php
try {
    $conn = new PDO("mysql:host=localhost; dbname=test", "root", "");
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$stmt = $conn->query(" SELECT * FROM users ");
while($row = $stmt->fetch())
{
echo "
<form action='update.php' method='POST'>
<input type='hidden' name= 'id' value='$row[id]'>
<td> <input type='text' name= 'firstname' value='$row[firstname]' >
<input type='text' name='lastname' value='$row[lastname]'>
<input type='text' name='email' value='$row[email]'>
<input type='submit' name='insert' value='Update'>
</form>
}
?>
```

REPLY:

Hello Joe,

This is a very common problem. Just look at the HTML source of your page and you'll find why. When yo have "don't" in the value, it becomes like this

```
 <input type='text' name='lastname' value='don't'>
```

see - it closes the quote.

To avoid this, you must apply `htmlspecialchars()` with `ENT_QUOTES` parameter to all your variables.

ymtaufik, 30.04.19 01:10

Hello Expert

To prevent my apps from sql injection, I use str_replace for any input parameter ($kiki) as below :

```
 foreach ($_POST as $varname => $varvalue) {
    $arr[$i] = $_POST[$varname];
    $i = $i +1;
}

if ($i >=1 ) {$kiki = $arr[0] . "^";}
if ($i >=2 ) {$kiki = $kiki . $arr[1] . "^";}
if ($i >=3 ) {$kiki = $kiki . $arr[2] . "^";}
if ($i >=4 ) {$kiki = $kiki . $arr[3] . "^";}
if ($i >=5 ) {$kiki = $kiki . $arr[4] . "^";}

$vowels = array(";",":","#","+","*",".","|","=","%","'","/","$","(","{","[","&","!","@","?",
",");
$kiki = str_replace($vowels, "-", $kiki);

$val = explode("^", $kiki);
```

then :

```
 $sql = "SELECT * FROM zuser WHERE nameu='".$val[1]."' AND passw='".$val[2]."'";
```

is it still vulnerable?, can you help how to inject with above sanitation ? regards

REPLY:

Hello.

Yes it is vulnerable. You can find many answers on Stack Overflow that explain how to inject with above sanitation.

You should use prepared statements instead of this.

Shahrulhadi, 17.04.19 06:32

Hi, i am currently using php framework Yii 2. Yii 3 is in development, and i am planning to use that when it is release. Can you verify if they use proper PDO? If Yii doesnt use proper PDO, what php framework do you suggest?

Dave Coates, 10.04.19 17:15

Hi. This is an excellent web site - thanks!
I have read page after page about retrieving multiple rows and multiple row sets, but I can't find anywhere which explains the mechanics of how multiple row sets work. My question is... When calling a stored procedure via PDO which returns multiple row sets, are all the row sets returned to PDO at once, or is there subsequent communication between the server and PDO every time I make a call to GetNextRowSet? Thanks in advance. Dave.

REPLY:

Hello Dave!

That's a very good question, and the answer is the option 2: subsequent communication between the server and PDO.

That's actually the whole point, as each resultset is completely separated from others, bearing its own metadata such as error information number if returned rows, etc.

After all, you can tell that other resultsets are waiting on the server by trying to execute another query while not finishing with getting them all - the error message will say that you cannot execute any queries while other results are waiting.

Hope it is clearer for you now! Feel free to to ask if you still have questions though.

tamir, 01.04.19 21:38

hi, i have problem with prepare query and the asc/desc parameter, i cant bind in the execute, i tried to catcatinate it before the query and dont manage to do it. what is the right way?

Gavin, 01.04.19 16:27

Please NEVER take this website down. It is EXTREMELY useful and concise. Thankyou

Nedum, 30.03.19 14:33

Please Sir, While running a query, this error message came up: Unknown Character Set! While building the DSN, I added charset=$charset to the string. And $charset = "utf8mb4" . What did I do wrong here Please?

And, when I do search and retrieve pages of data, to format display pagination, is it wrong to use PDO::rowCount() in this case too?

REPLY:

Hello Nedum!

Most likely your database server is rather old and doean't support this rather new encoding. Use utf8 instead.

As of PDO::rowCount() - yes, it is gravely wrong to use it for pagination. Use SELECT COUNT(*) instead.

Ali, 18.03.19 23:34

Great article with great details. I spent years and years on using PDO and I learnt a lot with pain and blood, but this article still had new things for me. I wish I had read that many many years ago. Thank you!

However I found a paragraph in this article that somehow I think that can be misleading for beginners:

"An answer to a frequently asked question, "whether this function is safe to use in concurrent environment?" is positive: yes, it is safe. Being just an interface to MySQL C API mysql_insert_id() function it's perfectly safe."

So for Beginners, I want to rephrase this paragraph:

PDO::lastInsertId IS GUARANTEED to return correct ID of last inserted row in the script, even on concurrent environments, but it CAN BE harmfully dangerous and unsafe with seeded IDs (e.g. mysql's auto-increment or most pseudo randoms like rand() function) on concurrent environments IF you use it to calculate next row's ID. One should never calculate next row's ID based on previous row's ID in a concurrent environment. If you need next ID, just insert a new row in the table, and use PDO::lastInsertID to get that ID. That is because with seeded row IDs on concurrent environments like the web, collision IDs WILL happen and you end up in tears.

So, just to repeat it again:

1. At first, insert a row
2. Then use PDO::lastInsertID if you need the ID of the inserted row
3. If you need the ID of next inserting row, don't calculate it from last inserted row, just go back to #1

REPLY:

Hello Ali!

Thank you very much for such a thoughtful comment!
I wholeheartedly agree with this warning, but I feel that it is a bit off topic, as it is not directly related to PDO, but to the abuse of the auto-generated identifiers in general.

Ho be honest, given right now am not thinking of reducing the size of the article rather than enlarging it, I am reluctant to adding any additional information not directly related to PDO. Either way this comment will remain here to warn all the readers, so thank you again for it!

KK, 15.03.19 05:24

How can you execute trigger queries with pdo.

I am using easydb and I get the following error. I have tried with and without delimiter statements and both fail.

Fatal error: Uncaught PDOException: SQLSTATE[HY000]: General error: 2014 Cannot execute queries while other unbuffered queries are active. Consider using PDOStatement::fetchAll(). Alternatively, if your code is only ever going to run against mysql, you may enable query buffering by setting the PDO::MYSQL_ATTR_USE_BUFFERED_QUERY attribute.

```
                    $triggerquery = " CREATE TRIGGER `$triggername` $triggertime ON `$tabl
ename`
                      FOR EACH ROW
                      BEGIN
                          IF(NEW.`$triggercol` IS NULL)
                          THEN
                              SET NEW.`$triggercol` = UNIX_TIMESTAMP();
                          END IF;
                      END;
                  ";
```

KK, 12.03.19 12:21

I am using paragonies easydb pdo wrapper. but for connection errors, it displays username and pwd. i have tried using silent mode. but, no luck. is there any other way to ensure username and pwd are not used. fyi: it uses the try catch mechanism.

REPLY:

Hello!

That's a very good question.

The leak is coming from the way Factory::create() method is called.

The simplest solution would be to wrap the Factory::create() call in a try-catch, as it is used for vanilla PDO in this article:

```
 try {
    $db = \ParagonIE\EasyDB\Factory::create(
        'mysql:host=localhost;dbname=something',
        'username',
        'putastrongpasswordhere'
    );
} catch(Throwable $e) {
     throw new \Exception($e->getMessage());
}
```

It will take the credential details out of the error message.

This method is not very clean but it's quick and it works.

The better solution would be to change the way Factory::create() is called, so I will reach Paragonie to discuss the matter.

Silver, 21.02.19 12:23

Hey, Thanks alot for the reply! I looked into the CSRF token and I'm definetly going to be using that for my forms. Also about the third point you made, did you mean I shouldn't be giving users the ability to input/check some box in order to include some php file? or did you have something else in mind? Also thanks for the file upload part, I'll definetly go by that advice next time I am going to be working with files.

Silver

REPLY:

Choosing a file to include based on the user input is perfectly fine. I meant you should never directly put the user ipout in include (or any other function dealing with files).

Silver, 19.02.19 18:50

Hey I'm building a web application with php/sql and also js on the user's side. I have a question regarding security, specifically about input/output sanitization. Right now I'm using PDO with prepared statements to contact my DB (SQL) and also I am using Regex for input validation both on the server side and user side. Is there anything else I should be doing to user inputs? Secondly on the output side i understand it is needed to sanitize output with htmlspecialchars() so people can't add in "evil html code/and js" is there anything else I should look into there or is that enough for a small userbase web application? I can also add that some of the data I'm working with falls under delicate user data according to GDPR. Sorry for the long question! :)

REPLY:

Hello Silver!

First of all, security is not a simple matter that can be covered in a comment. There are a lot of areas to look for.

But speaking of SQL injection and XSS you are taking it right.

Other areas to look for are (you can google up the terms)

- Consider adding CSRF token to your form
- make sure you don't include any file on your site based on user input.
- make sure you are not using eval()
- if you're allowing file uploads, filter uploaded files based on their extensions and rename files to your own names.

that's of course not the full list, just to elaborate a bit on your question. But again, it it not a simple matter that can be covered in a comment.

Dennis, 15.02.19 12:23

Hello! Thank you for this site! Although a lot is very clear, there is something I can't work out. So, my problem is: How can I connect my database class with my other class which has a function insert() to start a transaction with prepared statement to start a transaction for multiple inserts... Can you show me a example? My Database connection class:

```
 <?php

class DatabaseController
{
private $servername;
private $username;
private $password;
private $dbname;
private $charset;
protected $conn;

public function connect()
{
    $this->servername = "localhost";
    $this->username = "root";
    $this->password = "";
    $this->dbname = "scores";
    $this->charset = "utf8mb4";
    $options = [
        PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES   => false,
    ];
    try {
        $dsn = "mysql:host=" . $this->servername . ";dbname=" . $this->dbname;
        $conn = new PDO($dsn, $this->username, $this->password, $this->charset, $options);

        return $conn;
    } catch (Exception $e) {
        echo "Connection failed: " . $e->getMessage();

    }

   }
}
```

REPLY:

Hello Dennis!

Using the database connection is a tricky subject. There are several approaches discussed in the adjacent chapter: https://phpdelusions.net/pdo/pdo_wrapper#availability (https://phpdelusions.net/pdo/pdo_wrapper#availability) It's hard to tell with one will suit you best, but at least you should make sure that there is only one connection opened and used for all database interactions during a single script runtime. Feel free to reach back if something is still unclear.

Also, a bit irrelevant, but I noticed some issues with your DatabaseController class. I have an article on the common issues with database classes: https://phpdelusions.net/pdo/common_mistakes (https://phpdelusions.net/pdo/common_mistakes) you may find it interesting.

Jerome Betareu, 12.02.19 11:46

hello, the table contains over a million records. 1 record per minute for several years. No wonder the treatment times were getting longer and longer! In short, I removed almost the entire beginning of the code to process a week of data and the processing time is now 1.24 seconds :) Thank you very much for opening my eyes to the problem.

Jerome Betareu, 11.02.19 20:46

You are absolutely right ! I focused on the query itself but it is actually the 2 min and max queries that are not good! Obviously the whole table is treated by the WHERE clause since $timestampmin is the timestamp of the first record of the table and $timestampmax is the last! I will look at this tomorrow because it starts to be late (and my eyes do not see much ...) thank you again for your precious help.

Jerome Betareu, 11.02.19 11:09

Here is the request

```
 $connection = sql_pdo_connexion ($server, $base, $login, $pass);
$timestampmin = $ connection -> query ("SELECT MIN (timestamp) FROM` $table_type` ") -> fetc
hColumn ();
$timestampmax = $connection -> query ("SELECT MAX (timestamp) FROM` $table_type` ") -> fetch
Column ();
$datetimedebut = date ('Y-m-d', $timestampmin);
$datetimefin = date ('Y-m-d', $timestampmax);
$datedebut = date ('Y-m-d', $timestampdebut);
$query = "SELECT timestamp, Tmax, Tmin, TCon, Ton, Teur, flag
FROM `$table_type`
WHERE timestamp BETWEEN $timestampdebut AND $timestampfin
ORDER BY timestamp ";

$sql = sql_pdo_query ($connection, $query);
```

REPLY:

Hello Jerome

To me, the above code makes no sense. It looks like all the records from table are selected. You are selecting all rows between min and max value in some column - so it is effectively all the rows. What's the point of it? Why bother with WHERE condition then?

In case this query returns too much rows, then indeed in will take a lot of time. Any chance you could reduce the number of rows selected?

Jerome Betareu, 11.02.19 11:02

the code

```
 $d1  = array();
$d2  = array();
$d3  = array();
$d5  = array();
$array_ON = array();
$array_OFF = array();
while ($row = $sql->fetch(PDO::FETCH_ASSOC)) {
    //   Execution in 11 seconds ! ! ! !  -- very BAD !
    $t = $row['timestamp'] * 1000;
    array_push ($d1, array($t, floatval($row["Tmax"])));
    array_push ($d2, array($t, floatval($row["Tmin"])));
    array_push ($d3, array($t, floatval($row["TCon"])));
    array_push ($d5, array($t, floatval($row["Teur"])));
    if (floatval($row["flag"]) == 0) { // Chauffage OFF
        array_push ($array_OFF , array($t, floatval($row["Ton"])));
        array_push ($array_ON , array($t, null));
    } else {
        array_push ($array_ON , array($t, floatval($row["Ton"])));
        array_push ($array_OFF , array($t, null));
    }
}
$sql->closeCursor();
```

Thank you again for your help

Jerome Betareu, 11.02.19 11:00

Hello, thank you for responding quickly. After analysis of some requests it turns out that it is the extraction of the data which is extremely long. For example, the next query runs in about 1 second while data retrieval takes 11 seconds! I know I'm abusing (a bit) and optimizing queries is not part of your post but if you could give me a solution it would be really nice. An article above would be welcome and I think that newbies like me would find very valuable information.

Jerome Betareu, 08.02.19 13:05

Hello, first of all I apologize for my english (I'm French). I am a student and I am currently working for a company that would like to take over a web site developed by another student in the past ... in short, my knowledge of PHP are rudimentary and your site has taught me a lot but I ask myself a few questions about the obtimization of performance in the writing of PDO requests. For example, is it possible to optimize this query in terms of CPU utilization and RAM. The site being sullied on a Raspberry PI v2, the speed of display is also an important criterium ...

```
 $query="SELECT qdate, DATE_FORMAT(qdate, '$dateformatsql') AS 'periode' ,
ROUND( ((MAX(`type_1`) - MIN(`type_1`)) / 1000) ,1 ) AS 'type_1',
ROUND( ((MAX(`type_2`) - MIN(`type_2`)) / 1000) ,1 ) AS 'type_2'
FROM `$table_type`
WHERE timestamp > '$timedebut'
GROUP BY periode
ORDER BY qdate" ;
```

thank you for your reply

REPLY:

Hello Jerome!

This question is a bit out of the scope of this article but I'll be glad to help. I think you should make sure there is an index on the timestamp field. It's hard to tell any more without knowing the details. But you could always try to check the performance by adding EXPLAIN keyword before SELECT in your query and studying its output.

William Entriken, 01.02.19 01:49

Some of the " in example code are unnecessary and could be '

Walter, 25.01.19 01:38

Please help with SQL application.

I want to `Select * from table Quotes` , where one or more items exist in table `Work Orders` where `Quotes.quote = Work Orders.quote` and `Work Orders.stat='OPEN'`

I know I could do a query on table `Work Orders` , fetch column, and then feed the result into a query on table `Quotes,` but I feel like the database can do this. What is the SQL to have the database make this comparison?

REPLY:

Hello Walter!

Yes you are right, a database can do it. The query is almost one you wrote already, you only need to add a JOIN clause

```
 SELECT Quotes.* FROM Quotes
JOIN `Work Orders` ON Quotes.quote = `Work Orders`.quote
WHERE `Work Orders`.stat='OPEN'
```

Stack Overflow ans Wikipedia has many great articles explaining JOIN, you may want to learn about joins there.

Khangi, 24.01.19 23:29

Hello, I do not see what's wrong in my request! Can you give me the solution please?

```
 $stat = 'off';
$req = $pdo->prepare('UPDATE bus SET onoff = :onoff');
 $req->execute(array('onoff' => $stat));
```

thank you so much

REPLY:

Hello Khangi!

I don't see it either. Just to be sure, don't you forget to add a WHERE clause?

In any case, any problem with a PDO query would raise an error that should help you to find the problem

Kris, 18.01.19 13:44

Great article. Helped me a lot. Thanks!

John Pierce, 17.01.19 20:13

thank you very much for your super fast response! Actually, I had thought to do like that but I thought wrong that as I had no data fields in the table I could not do it ... error!

it is unfortunate that no command like BOTTOM or TOP is available under SQL, it would facilitate many things ... :) Thx

REPLY:

Hello John!

I must warn you in this regard again, because if you think that using BOTTOM or TOP would facilitate many things you are probably doing something wrong.

And your UPDATE query is the example. If you want to update the "last record" than you definitely doing it wrong. To update a record you must know for certain which particular record you want to update, not just a relative position. If you can explain why do you need this update I will tell you ow to do it right

John Pierce, 17.01.19 19:10

I will have another question about UPDATE. How to formulate the query to modify only the last record of the table? Of course, there are no ID fields in this table, only a primary key. thx

REPLY:

Hello John.

At least for mysql, there is no way to tell "the last" record, unless you have a field designated for this purpose. So I would recommend either an auto_incremented id, or at least a datetime field that would hold the time when the row was added.

Besides, "the last" is too clumsy a description, you must know the exact key for the row you need to update. Other wise a race condition may ruin your operation

John Pierce, 17.01.19 19:05

Hello, by doing some research I found your site which unfortunately did not answer to my problem. I would like to UPDATE a table looking for a particular record using WHERE but obviously my syntax should not be correct. Can you help me please ?

```
 $data = "pierce";
$req = $con->prepare ("UPDATE contact SET
bic =: bic,
orson =: orson,
country =: country,
code =: code,
wash =: wash
WHERE name = $data");

 $req-> execute (array (
 'bic' => $bic,
 'orson' => $orson,
 'country' => $country,
 'code' => $code,
 'wash' => $wash));
```

The request is executed but the table is not modified! Can you tell me how to implement this please? WHERE name = $data

REPLY:

Hello John!

Well, the answer is written on this very page, although I admit it is not that explicit as should be:

> So, for every query you run, if at least one variable is going to be used, you have to substitute it with a placeholder, then prepare your query, and then execute it, passing variables separately.

It meand that **every single variable** should be substituted with a placeholder and then go into execute:

```
 $data = "pierce";
$req = $con->prepare ("UPDATE contact SET
bic = :bic,
orson = :orson,
country = :country,
code = :code,
wash = :wash
WHERE name = :data");

 $req-> execute (array (
 'bic' => $bic,
 'orson' => $orson,
 'country' => $country,
 'code' => $code,
 'wash' => $wash,
 'data' => $data,
));
```

In you current form the query is not executed but rather produces an error, due to that $data veriable i the query and also a space between a colon and a placeholder name (it should be :bic, not :bic).

I fixed these issues in your code and now it should work.

Haseeb khan, 01.01.19 12:37

i want a to select data between two dates using pdo here is the column value (12/31/2018 6:25:21 PM)

momo, 24.12.18 09:54

Can you teach me how to search for my question?

My question is: I did this:

```
$statement = $pdo->prepare('select * from mytodo.todos');
```

However, only by luck did I try to qualify my todos table with its database mytodo and see that when I

```
var_dump($statement->fetchAll());
```

there is anything but an empty array.

I was adding to this table, and on my sql gui I could see they were adding, and same with the sql terminal, yet the fetchAll array was empty when I neglected the `mytodo.` part.

If I didn't have luck, what search terms could I have entered on search engines to tell me that I need to do that prepending of the database name to my column?

I'm following these beginner PHP videos on laracast, and quite a few times copying the exact thing in the video doesn't seem to work (I'm on PHP 7.2 with MySQL8, that itself was not smooth to set up with all the troubleshooting the "caching_sha2_password" stuff), so I've been searching a lot from the beginning. Couldn't search for the right term for this problem I had and I'm scared once I progress to bigger errors, I wouldn't be able to find my way out!

Thanks

REPLY:

Hello momo

I don't know if it's your case, but only prepare() is not enough. You must always execute a prepared query to get the results.

As of searching the answer, I would probably search for the basic PDO select example and follow it attentively.

However, if it works only if you add mytodo. part, it simply means that you are connection to some other database and natuarally you have to denote the database yoiu want to work with in the query.

Searching would be straightforward, something like my query works only if I add database to the table name

Gio, 11.12.18 03:06

Hi, any compliment is superfluous. A question fi possible, I have some difficulties to run a query LIKE :

```
SELECT id FROM table WHERE id = :id AND catalog LIKE ?
```

suggestions ?

REPLY:

Hello Gio!

You cannot use positional and named placeholders in the same query

So it should be either

```
SELECT id FROM table WHERE id = :id AND catalog LIKE :text
```

or

```
SELECT id FROM table WHERE id = ? AND catalog LIKE ?
```

gratefulHacker, 09.12.18 16:33

Just want to say thank you for your articles. They have been incredibly informative and helpful.

Chris S, 23.11.18 20:32

I'm not sure if it's possible to reply to a reply here for us regular peons, but thank you for your quick and very thorough reply (and I apologize for starting a new comment thread if it's possible to reply to my own). That was an extremely helpful response.

To be honest, I actually always saw the backticks used elsewhere and had no idea the purpose... "what an unnecessary, and ugly-looking thing to do", I thought. I come from a professional Oracle background, so MySQL is "familiar", but not second nature. It wasn't until reading this (fantastic) article that I learned what those backticks are for. Thank you!

I'll go update my code to use the backticks, just in case. And yeah, sorry, "user input" wasn't a very descriptive term to use, but I can say that yes, everything is hardcoded. I definitely would not be using any reserved words in table names, but I like the idea of just following good standards, and doing that early on enough in development (which is the case here) and sticking to them throughout any coding I do. So, off I go to fix it all up!

Thanks again. I can't stress just how much I learned from this article already.

Chris S, 22.11.18 22:45

I'm a little confused on whether I need to worry about escaping/backticking my table names in querying/DDL statements.

My table names are all constants, stored in a constants php file so that if I ever change things like schema name or something like that, I have one central place to update. So for example, I have a constant defined like so:

```
DEFINE('NS_TBL_PLAYERS', 'NS_PLAYERS');
```

So in my code I may do something like:

```
 $sql = "SELECT column1 from ".NS_TBL_PLAYERS." WHERE id = :player_id";
stmt = $pdo->prepare($sql);
$stmt->bindValue(":id", $player_id, PDO::PARAM_INT);
if ($stmt->execute()) {.. yada yada..
```

My syntax above might be a little off.. part of that is by memory, but that's not the point of my qustion. , I'm just wondering if I open myself to SQL injection by doing the table dynamically like above, without any backticking/escaping or anything. A user has no input whatsoever into what "NS_TBL_PLAYERS" is... that all comes from "constants.php" file I have on the server. So I assume that unless they somehow gain access to that file to update it (in which case I probably have bigger worries) the above code is safe?

Thanks for your time and advice, and this article ((The only proper) PDO tutorial) is incredibly well done.

REPLY:

Hello Chris!

That's a really good question.

From the SQL injection point of view, rules are simple:

1. All data must be supplied though a placeholder.
2. All other parts of the query must be hard-coded in your script.

So as far as I can see, your code follows the rules, therefore there is no problem.

However, from the correct SQL syntax point of view there could be a problem. Your table name could be an SQL reserved word or an incorrect syntax (contain a space or a hyphen for example) so it's much better to enclose it in backticks, just in case. So the only difference I would make is

```
$sql = "SELECT column1 from `".NS_TBL_PLAYERS."` WHERE id = :player_id";
```

And it should be OK.

Whereas you need to worry about escaping/backticking only if your query parts are coming from the user side (field names for example). In this case you must choose them from a whitelist - so effectively it will make them hardcoded and thus immune to injection.

I would also avoid the "user input" term as being quite vague and uncertain. A "hardcoded value" is much better to me, as you can always be certain, whether it's written in your code or not. If not - then ot hould be added to the query through a placeholder.

Hope this long and winding rant would be of any value for you! :)

Harachi, 12.11.18 20:09

Hello, thank you for this excellent articles that filled a lot of my shortcomings. unfortunately there is still a lot and if you want to fill another ... :)

I am stuck in my development because I can not use the min and max functions in a select with a limit. Visibly, 'limit' is incompatible with these 2 functions but there may be a trick ... here is my code:

```
 $query = 'SELECT MAX(price) AS Pmax , MIN(price) AS Pmin FROM book ORDER BY timestamp DES
C LIMIT 0,10';
$sql = sql_pdo_query($connexion, $query);
$row = $sql->fetch(PDO::FETCH_ASSOC);
$Pmax = $row['Pmax'];
$Pmin = $row['Pmin'];
echo $Pmin.' '. $Pmax;
```

I also tested with this code:

```
 $Pmax = (float)current($connexion->query("SELECT price FROM book ORDER BY timestamp DESC LI
MIT 0,10")->fetch());
$Pmin = (float)current($connexion->query("SELECT price FROM book ORDER BY timestamp ASC LIMI
T 0,10")->fetch());
echo 'Pmax = '.$Pmax .' '. 'Pmin = '.$Pmin;
```

with this code I would like the min and the max of the last 10 records of my database but obviously as soon as 'limit' is used the returned values are not good! Do you have a way of doing that works please? Thank you for your help

REPLY:

Hello Harachi!

Thank you for the good question.

Your queries don't work because LIMIT is indeed incompatible with the way you are using min and max. SQL operators work from left to right, so min and max are taken from the entire table ordered by timestamp, and then *only one row returned* gets limited to 10 rows.

The simplest solution would be to add a sub-query (not tested):

```
 SELECT MAX(price) AS Pmax , MIN(price) AS Pmin
FROM (SELECT price FROM book ORDER BY timestamp DESC LIMIT 0,10) as t;
```

Also note that the way you are taking a single value from the query is rather peculiar, given PDO has a method for that already:

```
$Pmax = (float)$connexion->query("SELECT ...")->fetchColumn();
```

Zeef75, 06.11.18 23:37

Hello, certainly off topic but I start ... Already, very good article on PDO, I learned a lot. My problem is that I have to resume a development written in C CLI that I have to adapt to the PDO. Of course I can not find any documentation. Would you be so kind as to indicate a URL or can I find this information?

for example, how to program this function?

```
 int readconfig ()
{
MYSQL mysql;
    mysql_init (& mysql);
    mysql_options (& mysql, MYSQL_READ_DEFAULT_GROUP, "option");
    if (mysql_real_connect (& mysql, mysql_host, mysql_login, mysql_pwd, mysql_db, 0, NULL,
0)) {
        mysql_query (& mysql, "SELECT * FROM Netwho");
        MYSQL_RES * result = NULL;
        MYSQL_ROW row;
        result = mysql_use_result (& mysql);
        while ((row = mysql_fetch_row (result))) {
            Tconsign_p = (float) atof (row [0]);
            Tconsign_c = (float) atof (row [1]);
        }
        mysql_free_result (result);
        mysql_close (& mysql);
    } else {
        syslog (LOG_INFO, "ERROR");
    }
    return 0;
}
```

With my thanks

REPLY:

Hello Zeef!

That's a standard Mysql C API. To rewrite it to PHP/PDO, you basically need this:

```
 function readconfig ($pdo)
{
    global $Tconsign_p, $Tconsign_c;

    $stmt = $pdo->query("SELECT * FROM Netwho");
    while (($row = $stmt->fetch(PDO::FETCH_NUM))) {
        $Tconsign_p = $row [0];
        $Tconsign_c = $row[1];
    }
}
```

note that the function is called with a parameter $pdo, which should be a valid PDO instance (https://phpdelusions.net/pdo_examples/connect_to_mysql) created before.

The function is written to the best of my C knowledge (which is very limited), so it could be wrong. Feel free to ask any questions, I'll be happy to help.

Ezeribe Adolf, 30.10.18 02:51

Please accept my gratitude for your kindness and willingness to take questions and sacrifice your time for me and a host of others. My question: In the Catch Block, following the $pdo instantiation, a backslash preceded the PDOException. What role is the slash performing there?

REPLY:

Hello Adolf!

Thank you for your kind words!

A backslash here denotes a namespace (http://php.net/manual/en/language.namespaces.php) - a relatively new concept in PHP.

It does not affect your code if you don't use namespaces but prevents an error/confusion if you do. Basically all PHP built-in classes are in the root namespace, so just a single backslash means - the root namespace.

Madapakas, 11.10.18 18:56

Hello Thanks for PDO Examples :D Very Helpful <3

Najam, 28.09.18 15:19

That is one fine article about PDO. Thanks a million.

moose, 14.09.18 01:03

$opt does not equal $options, see below.

```
 $opt = [
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
PDO::ATTR_EMULATE_PREPARES => false,
```

];

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

REPLY:

Hello moose!

Thank you very much, fixed!

Joseph Milumbe, 02.09.18 23:46

Can you make a PDF version of this walk-through?

Jeff, 14.08.18 04:22

You're doing god's work.

Stephen Austin, 07.08.18 18:28

I am a novice. I want to build a scoring application for a 'cricket' sports match which will involve a lot of real time inserting recalculation and updating and outputting. I've set up WampServer, I know I can build the Database and tables. Because of all the good things I've heard about PDO and because I will need to be connected to Mysql for about six hours straight, My question is, Do you think that this is the best or a good way to approach such a project. (Software wise)?

REPLY:

Hello Stephen!

Yes, of course, PDO is the best API a novice could choose.

Please note that unless you are using websockets, there won't be a constant connection to a database. By default, PHP processes are atomic, a request initiated from a browser makes a php script run, return the requested data, and die. So it connects to a database every time. A very good and simple model.

Jeff Hawkins, 02.08.18 01:59

Great stuff, but... I've posted a question at stackexchange for "PHP 7.2.8 PDO Fails to Connect to MySQL 8.0.12" that you may be interested in: https://serverfault.com/questions/924367/php-7-2-8-pdo-fails-to-connect-to-mysql-8-0-12-ga-in-aws-ec2-lemp-stack (https://serverfault.com/questions/924367/php-7-2-8-pdo-fails-to-connect-to-mysql-8-0-12-ga-in-aws-ec2-lemp-stack)

In short, I used your advice above to write code more than a year ago, that has worked great with PHP 7.0 and MySQL 5.7. But though all other connections that I can think of to this MySQL work, PHP will not connect. Have tried many things so far. Wondering if you will have better ideas.

Thanks much.

REPLY:

Hello Jeff!

Glad you got your case resolved. I had no idea of this issue and had nothing to offer but now I see it was resolved.

Benedict, 23.07.18 19:07

This is tutorial is superb. I would advice that you compile this into a book.

Henry, 22.06.18 20:47

When is it OK NOT to use parameter binding, and when should you REALLY use it? Use when there is user input, or data being passed from one page to another (PHP)? Don't need to use when getting data within the same code page without any input except for the code itself? Would appreciate our opinion. Thanks!

REPLY:

Hello Henry!

Thank you for the good question. There is indeed too much misunderstanding in the matter. So let's make it straight:

In the article it is explicitly stated as:

> **for every query you run, if at least one variable is going to be used, you have to substitute it with a placeholder, then prepare your query, and then execute it, passing variables separately**

Notice that there is **not a single condition** you mentioned in your comment like "user input", "data within the same code page" and such. Simply because all these terms are essentially vague and uncertain. There is no certain definition for "user input". Different people take it differently, with catastrophic results. The moment you start separating the seep from the goat, the "clean" data from "unclean", the moment you are making your first step to disaster.

Remember, every application evolves over time. And something that has been hardcoded for ten years, one day becomes dynamically changed and dangerous. Why take chances? Why base your decision empty musings instead of following one simple rule - any variable that goes into query should be substituted with a placeholder, regardless of the source.

Erik, 20.06.18 14:10

user interface for upload a photo

```
 <div class="col-md-3">
<form class="form-group" method="POST" enctype="multipart/data-form">
<label for="avatar">Upload Avatar</label>
<input type="file" name="avatar" class="form-control" required>
<p class="help-block">we accepted 2 mb file only!!</p>
<input type="submit" name="upload" value="Upload Avatar">
```

for transaction

```
 if(isset($_POST['upload'])){
    $file = $_FILES["avatar"]["tmp_name"];
    $dir = "upload/";
    $path = $dir.$file;
    if(move_uploaded_file($path, $dir)){;
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $user_id = (int)$_SESSION['namauser'];
    $insert_photo = $db->query("SELECT * FROM member WHERE namauser = '" . $_SESSION['namaus
er'] . "'");
    $sql = $db->query("INSERT INTO avatar VALUES ('0', '$path', '$user_id', NOW())");
}else{
    echo '<script type="text/javascript">alert("Sorry, the data cannot be processed");</scri
pt>';
}
}
</form>
```

// this part of user profile picture preview

```
 <div class="col-md-6">
<?php
    $ambilphoto = $db->query("SELECT * FROM avatar");
    $photo = $ambilphoto->fetch(PDO::FETCH_ASSOC);
?>
<h2 class="text-center text-muted">Apakah ini photo anda?</h2>
    <img src="<?=$photo['avatar']?>" class="img-responsive" style="width:5
    0%; height:auto;">
</div>?>
```

I have already tried googling anywhere but has no result, when i remember i have bookmarked your page, then I open it. I cannot upload my photo with this.

REPLY:

Hello Erik!

At least I an see that enctype="multipart/data-form" part is wrong. It should be enctype="multipart/form-data".

There could be other issues as well. But it makes very little sense in posting a code and asking what is wrong with it. Humans are not intended to run a code mentally, it's a job for a computer. So you have to run your code, make sure that all possible error reporting is on and start debugging. To tell you truth, debugging is the main occupations for any programmer. So it is not writing a program that takes most time but finding out why doesn't it work.

Erik, 17.06.18 17:55

```
             $id_user =(int)$_SESSION['id'];
            $userdata = $db->prepare("SELECT * FROM member WHERE id = ?");
            $userdata->execute([$_SESSION['id']]);
            $usr = $userdata->fetch(PDO::FETCH_ASSOC);

        <tr>
            <td><?=$usr['id']?></td>
            <td><?=$usr['usernama']?></td>
            <td><?=$usr['nomorhp']?></td>
            <td><?=$usr['email']?></td>
            <td><?=$usr['password']?></td>
            <td><?=$usr['kelas']?></td>
            <td><?=$usr['jurusan']?></td>
            <td><?=$usr['kabupaten']?></td>
        </tr>?>
```

I've used to your code given, but I do beg your pardon, I cannot see the data I need, I just see white screen whit table head data.

REPLY:

This code should work, given there is a closing PHP tag ?> between PHP and HTML

Erik, 17.06.18 11:42

code:

```
     <?php
        $id_user =(int)$_SESSION['id'];
        $userdata = $db->prepare("SELECT * FROM member WHERE id = ?",'$id_user');
        $userdata->binparam(':$_SESSION['id']',?);
        $userdata->execute();
        $usr = $userdata->fetch(PDO::FETCH_ASSOC);

    ?>
        <tr>
            <td>
                <?=$usr['user data'] //how to fetch whole user (login) data here?'?>
            </td>
        </tr>
    </tbody>
</table>
</div>?>
```

REPLY:

Hello Erik!

You have problem in your code. it should be

```
         $id_user =(int)$_SESSION['id'];
        $userdata = $db->prepare("SELECT * FROM member WHERE id = ?");
        $userdata->execute([$_SESSION['id']]);
        $usr = $userdata->fetch(PDO::FETCH_ASSOC);
```

Then you can display all the user data that is already fetched. You need to know the column names in order to display them though.

For example, if there is a name column, you can display it using $usr['name'] and so on.

Sebastian Wallace, 12.06.18 23:23

What a fantastic article. Very helpful, packed full of information! Many thanks.

Developer, 07.06.18 21:17

Excelent article! Answered some of my questions right up !

Johnny B, 18.04.18 21:18

Hi, I was wondering if you could do a simple client/server pdo api example? I have been unable to find an examples/tutorials for a simple one. This should be geared towards having an application log data and then submit it either php or json/js to a remote DB site. I have made it work without an api, and with a simple php api . However I fail to be able to do this with PDO. It is not as simple as one would think.

REPLY:

Hello Johnny!

You are probably mistaking PDO with something else. PDO is simply a database API, it knows nothing of json/js or application log data. It belongs to that "remote DB site" only. How this site is called is of no PDO's concern. In short, PDO is just a function to run your SQL query against a database server, no more.

Such a client/server pdo api example simply doesn't exist. The only server PDO talks to is a database server. Whereas any API is always implemented in PHP, which, in turn, uses PDO to query a database.

Script47, 15.04.18 08:07

This resource you have come up with is extremely helpful, especially the parts where you explain where prepared statements can be used.

Thank you for all the effort you put in to it.

In the transaction section when you make mention of the rollback function, is it not meant to be in camel case as shown in the documentation?

(http://php.net/manual/en/pdo.rollback.php (http://php.net/manual/en/pdo.rollback.php))

Have you considered writing articles on design patterns and or generic articles regarding writing smarter code in general?

REPLY:

Hello!

In PHP, function and method names are case-insensitive, so technically it could be anything.

Although it could make sense to use a camel case to separate different words in the method name, "rollback" is the established term by itself in a database word and could be count as a whole word, thus making it unnecessary to use a camel case in it.

Marvin, 11.04.18 17:14

Does this statement prevent sql injection or does it turn emulation mode off??

setAttribute(PDO::ATTR_EMULATE_PREPARES, false);

REPLY:

Hello Marvin!

It does turn emulation mode off

Greg, 10.04.18 21:17

Regarding the following error when using PDO fetch():

> General error: 2014 Cannot execute queries while other unbuffered queries are active. Consider using PDOStatement::fetchAll(). Alternatively, if your code is only ever going to run against mysql, you may enable query buffering by setting the PDO::MYSQL_ATTR_USE_BUFFERED_QUERY attribute.

It seems as though there are multiple ways to fix this, however I'm really curious what YOU would recommend. There are so many differing opinions online, however yours is the only one I actually trust, so I thought I would ask you directly.

The possible fixes seem to be:

- Changing all fetch() instances to fetchAll() (which would require some additional code re-writing)
- If only using MySQL, enable MYSQL_ATTR_USE_BUFFERED_QUERY (it seems like this fix might not work 100% of the time)
- Using closeCursor after fetch() calls

Are there any of these methods that you prefer? Or NOT prefer?

REPLY:

Hello Greg!

This error can be caused by different problems. Therefore different solutions.

Most of time you should never see such an error at all. The only plausible case could be caused by a stored procedure, and should be treated as explained in the corresponding chapter.

If you have a certain scenario in mind, please share, I'll try to sort it out

eFko, 04.04.18 22:54

$stmt = $pdo->prepare("SELECT value FROM table WHERE id=?"); $stmt->execute([$_GET['id']]); $data = $stmt->fetchColumn(); if ($data){ if ($data==0){ // do something } else { // do something else } } else { // do default action }

If value is 0, this code always do default action. How to select only one column a check it when this column is 0?

REPLY:

Hello eFko!

If I get you right, instead of `if($data)` you can test for FALSE value with strict comparison in the first condition:

```
if ($data !== false){
```

should do something when $data = 0

tharring, 27.03.18 20:04

That was it! I closed the <input> tag too soon. I also needed to remove id= for every input field, including the submit button, which I changed from type="button" to type="submit". Now everything works! I as able to insert into select fields in the table, not all fields. Thanks so much for your help!

tharring, 27.03.18 05:52

You are not showing an INSERT example and I really need help using a FORM and getting the info from the form into the db. AMPPS says I'm using php7 but my exports say it's 5.6.37. Here's my info and using php tags:

```
 include "localpdo.php";
if (isset($_POST['insert'])) {
    // Prepare the SQL query
    $stmt = $conn->prepare("INSERT INTO timesheet (vol_id, flag) VALUES(:vol_id,:flag)");
    $stmt->bindParam('vol_id', $_POST['vol_id'], PDO::PARAM_INT);
    $stmt->bindParam('flag', $_POST['flag'], PDO::PARAM_STR);
    $stmt->execute();
} else { // }
- - -
<form name="form1" method="post" action="">
    <input type="text" name="vol_id" id="vol_id">
        <?php if (isset($_POST['vol_id']) === true) { echo 'value="', $_POST['vol_id'], '"'
; } ?>
    <input type="text" name="flag" id="flag">
        <?php if (isset($_POST['flag']) === true) { echo 'value="', $_POST['flag'], '"'; } ?
>
    <input name="insert" id="insert" type="button" value="Start Time">
</form>?>
```

I have two TIMESTAMP fields for start/end times and an AI id field which auto-fill with each new record so I don't insert into them. Can you see what I'm doing wrong and help me? Thanks.

REPLY:

Hello1 tharring!

There is nothing wrong with your insert. Its HTML form which is at least wrong. You are closing an HTML tag too early should be like

```
 <form name="form1" method="post" action="">
    <input type="text" name="vol_id" id="vol_id" value="<?php if (isset($_POST['vol_i
d'])) { ?><?= _POST['vol_id'] ?><?php } ?>">
    <input type="text" name="flag" id="flag" <?php if (isset($_POST['flag'])) { ?><?= $_POST
['flag'] ?><?php } ?>">
    <input name="insert" id="insert" type="button" value="Start Time">
</form>?>
```

James, 25.03.18 04:33

Hi there, firstly thanks a lot for the article. I have a question of my own.

Is there an efficient way of doing an `UPDATE..SET..WHERE`

with large dataset? For example, instead of multiple SET `queue_order=$queue_order WHERE queue_id=$queue_id;` , is there a way to bundle them in a single PDO statement without using a for loop?

Thanks.

REPLY:

Hello James!

Although there are ways to do that, I would advise against. There is no point in stuffing as much queries in one statement as possible. **Prepared statements** are exactly for that purpose. Just prepare your UPDATE statement once and then execute it in a loop, like it shown in this article: https://phpdelusions.net/pdo#multiexec (https://phpdelusions.net/pdo#multiexec)

It will be as fast as a single statement.

If your updates will run unexpectedly slow, then it's your database settings to blame. To fix that, wrap your updates in a single transaction.

armel, 24.03.18 06:35

Hello I wish to resume coding after more than 2 years without practice. now i want to realize my personal projects with php mvc in a first so I want to understand how to start! thank you!

REPLY:

Hello Armel!

My strong recommendation wouild be to try a well-established framework, Symfony preferably. It will give you the best understanding of MVC.

armel, 24.03.18 06:31

Bonjour Je souhaiterai reprendre ? coder apr?s plus de 2 ans sans pratique. maintenant je veux r?aliser mes projets perso avec php mvc dans un premier tant je veux comprendre comment d? buter! merci!

Michael, 01.03.18 22:48

Thanks for your reply. I followed your guide for beginners and implemented the logging-errors advice. So far, the best approach :-)

Santi, 10.02.18 13:55

$query = "SELECT tbl_product.*, tbl_cart.id as cart_id,tbl_cart.quantity FROM tbl_product, tbl_cart WHERE tbl_product.id = tbl_cart.product_id AND tbl_cart.member_id = ?";

what does this queery mean?

Victor Alvarado, 03.02.18 22:54

Hi men, good post, VERY GOOD, i have a question about procedures.

How can i use procedures with FOREACH, how can i replace the while with foreach, implementing the empty result prevention, or is more efficiente use while instead?, because i like work with foreach

REPLY:

Hello Victor!

It is not very clear what you are asking, but in general there is no difference between using foreach and while.

In both cases you can just iterate over the statement. In case there is no result, then there simply will be no output:

```
 foreach ($stmt as $row) {
    echo $row['name'];
}
```

In case you want to know if there is any result beforehand, just fetch the resulting dataset using fetchAll():

```
 $data = $stmt->fetchAll();
if ($data){
    foreach ($data as $row) {
        echo $row['name'];
    }
} else {
    echo "No data";
}
```

Hope it is clear now, but if not, please ask

Michael, 03.02.18 17:18

Hi, Very good tutorial, thanks for sharing. My production database server went down and when the app tried to connect to it, the error report threw back the database connection details. Not nice. How would you recommend to handle this error? Thx

REPLY:

Hello Michael!

This is a very important question! However, it is not actually related to PDO or database, but rather to the whole PHP configuration. Your PHP sees to be misconfigured, leaking every error message outside, which shouldn't be. Actually, every error message is a breach in the security, so the solution should be a general one.

On a live server PHP must be configured to log errors instead of displaying them, so it would never happen again to leak your database credentials outside. You may find useful my article on the PHP error reporting configuration: https://phpdelusions.net/articles/error_reporting (https://phpdelusions.net/articles/error_reporting)

Feel free to drop a line if something is still unclear or you disagree with the proposed solution

GioMBG, 31.01.18 02:37

first thx for your tutorial. injection paranoia : prepare = safe ?

```
 $Q = ('SELECT COUNT(Artist) AS cnt_artists FROM artists WHERE Artist LIKE ?');
$q = $conn->prepare($Q);
$q->execute(array($letter.'%'));
```

it is safe ? thx

REPLY:

Hello Gio!

Yes, that's perfectly safe. PDO witll either correctly format a parameter for you (if emulation mode is turned on) or even send it **completely separated** from the query, and so there would be no way for it to interfere.

Tony L, 19.01.18 13:26

Beware that bitfields in MySQL suffer from the same problem that terms in the LIMIT clause do when using prepared statements, i.e. in emulation mode, strings are not correctly interpreted. So if you are converting code like mysql_query("create table foo (bits bit(48))"); mysql_query("insert into foo (bits) values ($val)"; into $pdo->prepare("insert into foo (bits) values (?)")->execute([64]); you will actually get the value 13876 inserted (0011 0110 0011 0100) which is the bytes of the ASCII characters '6' and '4'.

To fix this, either turn off emulation, or use bindParam as described in the section on LIMIT, or change your code to $pdo->prepare("insert into foo (bits) values (cast(? as decimal))")->execute([64]); Something for the next version of this page perhaps?

Brinth, 09.01.18 09:27

hi i am getting such an error- " Please make the application/config/database.php file writable".

how to make a php file writable using chd in remote server?

jeff van fleet, 08.01.18 18:53

i am brand new to pdo and i have a question about database connections.

on one page you say:

> constructor parameter is the most robust method in case your code is OOP. For the every class that needs a database connection, make PDO a constructor parameter

but on another page you say:

> The connection has to be made only once! No connects in every function. No connects in every class constructor. Otherwise, multiple connections will be created, which will eventually kill your database server. Thus, a sole PDO instance has to be created and then used through whole script execution

these statements seem contradictory to me. can you please explain how to properly connect to a database with pdo? I am using oop php.

thanks, jeff

REPLY:

Hello Jeff!

Thank you for the good question!

When you assign an object to a variable, there remains just a single object, whereas the second variable being just a *reference* to it. So when you are sending a PDO instance into a class' constructor, then it's actually a reference, that points to the same PDO instance. So the initial PDO object is not gets duplicated but remains essentially a single instance.

So you can tell that the former statement doesn't violate the rule from the latter but rater gets along with it

Hope it is clear now but feel free to ask if not!

T G, 26.12.17 00:33

Great Column!

I was wondering on the following code snippet:

```
 $data = ['name' => 'foo', 'submit' => 'submit']; // data for insert
$allowed = ["name", "surname", "email"]; // allowed fields
$values = [];
$set = ""; foreach ($allowed as $field) {
if (isset($data[$field])) {
$set.="".str_replace("", "`", $field)."". "=:$field, ";
$values[$field] = $data[$field];
}
}
$set = substr($set, 0, -2);
```

Why are you escaping the field names. Should the $allowed whitelist already handle any SQL injection risk?

REPLY:

Hello T G!

Yeah, that's an awkward moment. On the one hand, you don't have to escape a whitelisted value as nobody in their right mind would use a backtick in the field name. On the other hand, however, if you take the formatting routine as an abstract one, unaware of the data source, it would be natural to do the full processing, both quoting and escaping - just like any good quoting function should do.

I think it would be better to move the full quoting routine into a distinct function that will be doing the correct formatting regardless. I suppose it will arise less questions. What do you think?

jayenn, 23.12.17 23:49

Hi and thanks for this great information. I have learnt a lot about PDO and - at least as useful - about a consistent approach to error handling! I also like your style of writing. Thanks mucho!

REPLY:

Hello Jayenn!

Thank you for your kind words and especially for mentioning the error handling stuff! I am nor sort of crusade in changing the current state of error handling in PHP.

Feel free to check back if you have any problem though - any real life use case can make my articles batter for everyone.

AA, 21.12.17 09:53

I have a pdo search engine using MYSQL database. How can I get the number of search results per "MYSQL field"?

REPLY:

Hello!

Just run a

```
SELECT COUNT(*) FROM your_table WHERE  field = ?
```

and select that number using fetchColumn()

Jason Franklin, 07.12.17 00:09

It appears that adding a code block does not work. I just tried submitting another comment with code, but it did not appear as I expected.

Jason Franklin, 06.12.17 19:27

I submitted a comment a few days ago about escaping operands for the LIKE operator. Do you think it may have been sent to a spam folder?

I figured it would have made it through by now...

REPLY:

Hello Jason!

There is no spam folder, but it could have been an error on the site.

Escaping special characters used for the LIKE operator is not specific to PDO, it's rather related to the particular database. But probably I should add a mention at least for MySQL.

What was your question anyway?

vincent, 05.12.17 14:49

I was talking with another programmer and hes was complaining about SQL query not working (He has not the use to test his queries on the database before coding them on PHP). And he said :

-- It would be usefull to be able to directely see the query made by PDO to the database.
-- you mean with the values of binded ?
-- Yes !
-- For debugging ?
-- Of course, what kind of programmer do you take me for ?

I searched I stumbled on you (Your articles are wonderfully readable by the way).

REPLY:

Hello Vincent!

Although it is often useful to have a query from PDO with all the data interpolated, I would rather use it for the profiling.

Whereas when a query is not working, all you need as an error message from a database, that will tell you precisely, what is the problem. Note that you should be using prepared statements to avoid silly syntax errors caused by the data - it goes without saying.

That said, there are several code snippets and packages that can get you raw SQL from a prepared query, just google for "pdo interpolated query" or "PDO raw query".

Mike McPherson, 30.11.17 14:31

I have a simple loop that runs a small number of update statements if there's only one in the loop it works fine more than one and only the first works, with no errors reported I'm confused!!

```
 for ($i=0; $i<$count; $i++) {
$SQL = "UPDATE Candidate C SET engineer = '" .$engineer. "'
            WHERE ...";
    $result = $db->query($SQL);
}
```

REPLY:

Hello Mike!

Dunno why did you cut off the part after WHERE but it's impossible to tell anything without seeing the actual code, sorry.

Most likely you are just updating the same row.

Jorge, 27.11.17 06:57

> Unfortunately, there is some confusion. PDO::quote() creates a data literal, not an identifier. For example, for MySQL the returned value will be enclosed in quotes, not backticks, which will make it unsuitable to be used as a table name.

How frustrating. I just tested with PostgreSQL, but it uses single quotes so it worked there. One wonders why in so many years, such a basic and necessary feature is not there (found a thread from 2008 with people complaining about this, sayng it's years overdue). I just migrated from MDB2 because of it's unfinished state and difficult install method and chose PDO. One of my tables matches one of MySQL's many reserved words, so I must quote it and I was using quoteidentifier from MDB2. I see Zend Engine has it, but that means a large dependency for a single quoting function. Seems I'll have to roll up my sleeves. Thanks for the heads up, it could have gone unnoticed otherwise!

Jorge, 26.11.17 03:29

Nice rundown, much better than overly complex tutorials. However, identifier quoting does exist in PDO (didn't back then?).

$pdo->quote('mytablename'); will take care of any strange quotes. Keep in mind only MySQL uses backticks, so using this is better if you want to keep it a little more DB-agnostic.

REPLY:

Hello Jorge!

Thank you for your feedback.

Unfortunately, there is some confusion. PDO::quote() creates a data literal, not an identifier. For example, for MySQL the returned value will be enclosed in quotes, not backticks, which will make it unsuitable to be used as a table name.

Emma, 14.11.17 09:22

Dear Colonel, I greatly appreciate this website and the common sense you share on Stack Overflow. May I ask you, please, if you have time, to take a look at a conundrum I have posted on SO? (https://stackoverflow.com/questions/47278348/php-output-echoed-out-of-order (https://stackoverflow.com/questions/47278348/php-output-echoed-out-of-order)). I understand if that's not possible for you. Best wishes for the future

REPLY:

Hello Emma!

I see it has been happily resolved already. Glad you got it working and thank you for your kind words.

Feel free to come back if you have any other questions :)

Tyler, 13.11.17 09:25

Oh I typed in my second thought before I saw that you had responded to my question. Looks like we came to the same conclusion,

Awesome Thanks for your response

Tyler, 13.11.17 09:21

I suppose I could run a sub-query on COUNT(*)

Tyler, 13.11.17 08:02

Hello, I am running into a problem with pagination. As you recommend I use COUNT(*). But when using with GROUP BY I have a problem. With just one GROUP BY I can use COUNT(DISTINCT column). But now I have a query that requires 2 GROUP BY's and am

having a problem and SQL_CALC_FOUND_ROWS is not a solution for my case. It's looking like 'then you can either use rowCount() or simply call count() on the array returned by fetchAll()" are my only solutions.

Do you have any suggestions?

P.S. Thanks for your great PDO reference site.

Ty

REPLY:

Hello Tyler!

Thank you for the good question. I should have added it to the article. And especially thank you for asking. It's always better to ask than devise something of your own.

Yes, there is a way to ask a database to count on its side in this case as well. Just wrap your query into another query, like

```
SELECT count(*) FROM (SELECT 1 FROM ... here goes the rest of your query)
```

The idea is to wrap your query into another, and use it to count results. For this purpose we are selecting just a constant value "1" instead of the actual fields in your main query, just to make a database to do less job.

Feel free to ask if you got any other questions!

John Hind, 09.11.17 14:45

In testing I notice that if you use sql statements containing LIKE in combination with $_GET it is possible to inject % into the statement and retrieve all the data from a table. e.g. if I do not escape the % this retrieves all users. http://example.php?first_name (http://example.php?first_name)=%

The issue is referenced here http://php.net/manual/en/pdostatement.bindvalue.php (http://php.net/manual/en/pdostatement.bindvalue.php)

but I can't find any mention of it in your more extensive documentation. I have to say that I did not understand the code example you use to explain how to use 'like' with pdo.

Thanks, John

REPLY:

Hello John.

first of all, be advised to distinguish an *official* manual page for *user-contributed comments*. The latter can be left by any unsuspecting fellow and should be taken with a pinch of salt.

As of your concern, it's really simple.

1. if your code inside example.php is not intended for the substring search, then you should NOT use LIKE in the first place.
2. if your code inside example.php is intended for the substring search, then it means that the entire table contents is *already* exposed and you have nothing to complain of. for example, if you are using something like `'%'.$_GET['first_name'].'%', then I will have only run 28 requests with a different letter each to dump your "entire database" without any special character used.

M, 06.11.17 12:04

Greetings. I have enjoyed reading your content, especially the part about the PDO wrapper. Your site has become a bit of a reference on the topic. Helps people not to reinvent the wheel every day. As a personal wish, if I may, I'd like to get your insights on 'cache invalidation' an PDO. A bit off topic perhaps - it's just something I am working on these days. An ambitious idea of having a "caching system" underlying the app, i.e. transparent to the app who whouldn't know if data is coming from Memcache or the database. Queries would be handled by the PDO wrapper (-ish) and cached (using memcache), and also invalidated when 'interested' rows are updated/deleted. Not an easy task to develop something like that. Also, it'd be nice to have the User comments about "nested transactions" (see http://php.net/manual/en/pdo.begintransaction.php (http://php.net/manual/en/pdo.begintransaction.php)) incorporated in your articles, would probably be useful for a bunch of people. Great work.

REPLY:

Hello M!

Yes this question a bit off topic, but from my experience I would rather warn you against such a generic all-embracing cache solution. In my opinion, your regular queries should be fast by themselves, without the need of any cache. And it's only a few exceptional cases should be handled using a cache, but such cases should be carefully handpicked, considered using a regular optimization, and only then optimized using a cache, with a carefully considered invalidation.

Regarding transactions, thank you for the suggestion, I will add a link to that comment.

Zohra Pardesi, 02.11.17 16:57

Hi, I again stuck at some point, Now I am trying to edit product details which I already have added. I also have added image field when I click on edit button and then when I add only some feature after clicking on update button I find Picture which I did not touch have gone.... Now after analysing I have understood the Problem but I am unable to resolve the issue if you can help me in this please. my code is:

```
if(isset($_POST['update_product'])){
        $product_name = $_POST['product_name'];
        $cat_name=$_POST['cat_name'];
        $sub_cat_name=$_POST['sub_cat_name'];

        $product_img1=$_FILES['product_img1']['name'];
        move_uploaded_file($product_img1_tmp, "images/products_images/$product_img1");
```

See in above When we use move_uploaded_file then what happen actually is our database only get the path of image while image actually move or resides into the folder we have mentioned into path. So, when I try to edit/update page when code approached to database field product_img1 it only find path but not actual image. I think there must be a way like move_uploaded_file which I don't know :(

Please could you help me in this? how to fix this issue that if I don't want to update image, image should remain there?????

Zohra Pardesi, 22.10.17 22:39

Hi David, Yeah, you are right, I am mad that since hours I was checking my code madly, then took some rest and when again checked, I found my mistakes, both foreign keys I had placed wrongly; I corrected in the way below and my code is now working perfectly: into insert statement (cat_id, sub_cat_id) and into values($cat_name, $sub_cat_name)

I also removed function NOW() and mysql is taking time itself. i might don't need to do anything. I was so happy and wanted to let you know, when came here I found your reply. I really appreciate you for a quick response, you are doing such a great job by helping students. Great work! I will try to bound foreign keys as well as suggested. Thanks a lot and Regards,

Zohra Pardesi, 22.10.17 16:54

Hi, I am trying to learn PDO but stuck as receiving error message says: "PDOStatement::execute(): SQLSTATE[HY093]: Invalid parameter number: parameter was not defined" here is code:

```
if(isset($_POST['add_product'])) {
   $product_name = $_POST['product_name'];
   $cat_id=$_POST['cat_name'];    //Foreign Key from main_cat table
   $sub_cat_id=$_POST['sub_cat_name']; //Foreign Key from sub_cat table
   $product_price=$_POST['product_price'];
   $product_warranty=$_POST['product_warranty'];
```

cat_id and sub_cat_id are foreign keys while I also want to get date and time into Products table by function Now() Prepare statement and bindParam is as below:

```
 $add_product = $conn->prepare("insert into products( product_name, cat_name, sub_cat_nam
e, product_price, product_warranty, product_add_date) v
alues (:product_name, $cat_id, $sub_cat_id, :product_price, :product_warranty, NOW())");
$add_product->bindParam(":product_name", $product_name);
$add_product->bindParam(":product_price", $product_price);
$add_product->bindParam(":product_warranty", $product_warranty);
```

Call to function has been made into form where I need this to be executed.

It looks like I have set bindParam wrongly, I don't understand how to make it perfectly for cat_id, sub_cat_id and function NOW(); your help will be appreciated. regards,

REPLY:

Hello Zohra!

All you need is to brace yourself and write a consistent code. Either in your editor or here in comments.

I took your code and run it on my local server and there is no "Invalid parameter number". Therefore I suppose that you are just doing some mistakes when running your code and don't verify it, hence the error.

So just double check your code, and it will work!

Note that you should use prepared statements for ALL variables. $cat_id, $sub_cat_id should be bound as well.

David E, 21.10.17 14:37

Can you help me please. I am trying to perform the following PDO. All the values are here are present. However I get an error message " Call to a member function prepare() on null in "etc.

```
 $mbrwks = "UPDATE friends1 SET Wks = ?, Yno = ?, Mno = ?, Regdat = ?,
           Amount = ?,Cheque = ?,Status = ? WHERE Friendship = ?";
$pdo->prepare($mbrwks)->execute([$weeks,$yrs,$mm,$rdat,$fee,$cn,$c,$rn]);
```

REPLY:

Hello David!

This error means that you are calling $pdo inside a function, where variable scope is different. You need to make your PDO instance accessible. A short list of variants is listed here: https://phpdelusions.net/pdo_examples/connect_to_mysql#access (https://phpdelusions.net/pdo_examples/connect_to_mysql#access)

Please feel free to comment back if you need any help with any particular method

Ed, 17.10.17 18:43

Thanks so much for your quick response. I think I just need to take a step back and do some more reading. I think this code wont work anyway because the scope of the $bdo variable is outside of the function. I need to work that one out next!

REPLY:

Hello Ed!

Whoops, I didn't notice the issue with variable scope, shame on me.

There are several ways you could get a PDO instance. You may check this page for some suggestions: PDO wrapper (https://phpdelusions.net/pdo/pdo_wrapper)

Ed, 17.10.17 13:49

Thanks for the great and extensive instructions. I havent got very far but I am having a problem with passing the result of a function into a conditional statement. The database connects ok but I think the problem is here:

```
 $result = $stmt->fetch();
return ($result ==1)?true:false);
```

Any ideas?

```
 $host = 'localhost';
$db = 'mydb';
$user = 'dba';
$pass = '';
##$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db";
##var_dump($dsn);

$opt = [
PDO::ATTR_ERRMODE            =>        PDO::ERRMODE_EXCEPTION,
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
PDO::ATTR_EMULATE_PREPARES   => false,
];

try{
$pdo = new PDO($dsn, $user, $pass, $opt);
##var_dump($pdo);
}catch(Exception $e){
echo 'Message: ' .$e->getMessage();
exit;
}

function user_exists($username) {
$stmt = $pdo->prepare('SELECT COUNT(uid) FROM users WHERE username = :username');
$stmt->execute(['username' => $username]);
$result = $stmt->fetch();
return ($result ==1)?true:false);
}

if(user_exists('admin') === true) {
echo "exists";
} else {
echo "doesnt exist";
}
```

REPLY:

Hello Ed!

Thank you for the perfectly formatted question!

Yes, you spotted the problem perfectly. The value returned by fetch() method is not a number but array, and therefore comparing it with a number won't give you any good result.

If you want to fetch the number, you should use fetchColumn() instead. So your code would be

```
 function user_exists($username) {
    $stmt = $pdo->prepare('SELECT COUNT(uid) FROM users WHERE username = :username');
    $stmt->execute(['username' => $username]);
    $result = $stmt->fetchColumn();
    return ($result == 1);
}
```

Besides, the result of == operator is already a boolean, no need to add another condition, you can return it right away.

Also note that echong the error message right away is not advised. Consider the information from the section related to PHP error reporting (https://phpdelusions.net/articles/error_reporting)

Feel free to ask if you have any other questions!

Konrad, 15.10.17 01:15

Hi.

I have a small suggestion - if there are more than e.g. 10 comments, they could be hidden under "show more comments" link, or something. IMHO it doesn't have to be ajax, just a hidden which you show upon clicking the link.

Right now people look at the scrollbar and get scared, because they think the article is soooo long. While it is quite long, and comprehensive, it is not *that* huge :P

Plus, next to the 'address' field in the 'comments' section, you could add "not required" and "will not be published"

Thank you very much for this article, I like pointing here ppl who want to start with PDO :P

REPLY:

Hello Konrad!

Thank you for the suggestions! I'll definitely implement both of them soon.

Bryte, 12.10.17 12:46

Hi, please am trying to get a value from a field in a datatable, the scenario is like this: I have a table with data about four columns, at a click of a button which is an action in the table, a modal appears with four input boxes, I made the first input box disabled so as to pick a value from a field in the table. But i can't seem to implement it. Please can you help?

David, 05.10.17 18:45

Thank you for your reply. I am a little confused though, but will try to solve my lack of knowledge. Regards. David.

David Entwistle, 05.10.17 18:09

I would appreciate your help with a PDO issue. I am calling a PDO select function which works except I only get one piece of data from a file that contains two pieces. the code is

The field "Dn" should return "Firstname Surname" separted by a space. All I get returned is the Firstname. Please can you help me ? Regards David.

REPLY:

Hello David!

Most likely your issue is not with PDO but with HTML. Check the HRML source of your page and find your last name all right.

Then follow the HTML syntax and wrap Dn field value in quotes :)

Tyler, 04.10.17 06:03

Hey, I was linked to your site, and it seems to have some nice recommendations. That said, I was looking at the section on rowCount, and it seems like you take quite the position on 'it shouldn't be necessary'. I've run into this quite a bit when performing pagination of results, and I'm curious as to what your recommendation is to do in this case instead of getting a row count?

REPLY:

Hello Tyler!

I should have probably emphasized it better, but using anything like row count for pagination is not something "not necessary" but a big mistake.

For paginafion you need 2 queries: one that gets actual data for a particular page, for this query you don't need no count at all.

Another query, which is used to get the total number of rows, should never be using anything like row count. Instead of selecting actual rows and then discarding them, only to get the count, one have to get just the count already, by using a select count(*) query. Otherwise it will defeat the very purpose of pagination: selecting only a smaller amount of data that will be actually processed. It's not a big deal for a personal home page, but selecting all the big data only to count it may kill a mature site.

MF, 29.09.17 14:57

Hi! Whoever you are. I just wanted to let you know, that all your tips and best practices are helping me a lot to code better and more efficient. I can't thank you enough!

K, 28.09.17 22:48

I just wanted to say thank you - your solution worked perfectly!

JamesP, 27.09.17 11:50

So I understand PDO Prepared Statements should protect from SQL injection and ' escapes. But when I attempted the following...

```
 $id = "2' AND name='Entry2";
$someinfo = "updated";
...DB Stuff...
$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $dbpassword);
$stmt = $conn->prepare("UPDATE testdb SET info=:someinfo WHERE id=:id");
$stmt->bindParam(':id', $id);
$stmt->bindParam(':someinfo', $someinfo);
$stmt->execute();
```

Then the row with id=2 and name=entry2 would be updated. I should note, I only saw this behavior within the "WHERE ..." part of the query. It did not escape in the "SET...WHERE" part of the query, and I wasn't able to add any additional queries in either. I observed the same behavior when I tried with mysqli as well.

That said, I didn't find anyone pointing this out anywhere, and when I asked on Stack Overflow, no one believed me. I could probably just manually replace ' instances with something else for storage and replace again for output, but the fact that I'm the only one who seems to observe this has me seriously confused. Any insight would be greatly appreciated.

REPLY:

Hello James!

It is quite strange that on Stack Overflow nobody believed you, as the behavior is quite expected, given the way mysql (and PHP as well) treat input values. Well, nothing strange on the other hand, given the average level of expertise there.

So, the issue is not actually an injection, neither it has anything to do with PDO. It's an integral part of MySQL. By default, in the older versions of Mysql (pre 5.7. if I am not mistaken), when casting input value to an integer, it takes first numeric characters, cutting off everything else. You can see it with such a simple example:

```
 mysql> select '25 ponies' + 0;
+-----------------+
| '25 ponies' + 0 |
+-----------------+
|              25 |
+-----------------+
```

As you can see, the value become just 25 when cast to an integer to perform an arithmetic operation.

Exactly the same happened in your example. Value "2" has been taken out of the "2' AND name='Entry2" string which made a legitimate query and thus the record got updated.

To prevent this behavior you can set mysql in strict mode, like

```
set SQL_MODE='STRICT_TRANS_TABLES';
```

it will start throwing errors for the malformed numeric values instead of silently casting them.

I've reopened your question on Stack Overflow, you can now write an answer for it and link to this comment, https://phpdelusions.net/pdo#comment-277 (https://phpdelusions.net/pdo#comment-277) as a proof

Your Friendly PHP Neighbour, 26.09.17 19:10

Beware that nothing will alert you about the lack of support for the `PDO::MYSQL_ATTR_FOUND_ROWS` attribute.

Only way to check if your PHP is going to ignore this attribute is to call `$pdo->getAttribute(PDO::MYSQL_ATTR_FOUND_ROWS)`. Not sure why this isn't done when setting the attribute (https://bugs.php.net/bug.php?id=62782). Go figure the reasoning behind this.

To reiterate:

1. The PDO class constant `PDO::MYSQL_ATTR_FOUND_ROWS` doesn't always work, even if the constant is defined (https://bugs.php.net/bug.php?id=63781).
2. To test if it will work, you must use `getAttribute()`.
3. If the connection option doesn't work, you're out of luck (short of recompiling your PDO driver).
4. Don't trust the manual. (https://bugs.php.net/bug.php?id=44135).

REPLY:

Hello Neighbour!

I thank you for your valuable info! Looks like an important case. I added the link to this comment to the article.

Though I can answer one of your concerns. Given that `MYSQL_ATTR_FOUND_ROWS` is a *connection* option for MySQL, it's impossible to use it anywhere else, and thus there is no way to change this parameter on the fly, within the same connection, unless Mysqli C API will be changed.

K, 20.09.17 22:45

Hi,

I'm sorry if my previous question was unclear, but I'd love an answer, even as to whether this is possible, if you have the time.

I'd like to do something like this:

```
$query = $dbc->prepare("UPDATE
   customer_info
   SET
   customer_name = :customer_name,
   billing_email = :billing_email
   WHERE customer_id = :customer_id");
```

where customer_name and customer_email come from (sanitized) user inputs. Is it possible to format this query so that if I wanted to update ONLY the customer name or ONLY the customer email it would not overwrite the item that I'm NOT updating? So if the user inputs a customer name and leaves the email blank, it will update the name but preserve the email that is already in the database.

In order to do this would I have to write a separate query for each item? In my actual code I have many other items that I'd like to be able to update without overwriting the information in all of the database columns at once. It seems like using positional parameters might be a solution, but I haven't been able to get it to work. I was hoping you could tell me if this is possible, thanks!

REPLY:

Hello!

Sorry for the delayed answer, by the time you asked for the first time, I've been on vacations and missed your comment.

What I would suggest is to create a query dynamically, from the array provided, as it's explained in this article (https://phpdelusions.net/pdo/sql_injection_example#solution)

So it should be like

```
 // create an array that contains fields you want to update
$data = ["customer_name"] = "something";

// create a whitelist to be sure
$allowed = ["customer_name","billing_email"];

// create a SET clause dynamically
$params = [];
$setStr = "";
foreach ($allowed as $key)
{
    if (isset($data[$key]) && $key != "id")
    {
        $setStr .= "`".str_replace("`", "``", $key)."` = :".$key.",";
        $params[$key] = $_POST[$key];
    }
}
$setStr = rtrim($setStr, ",");
$params['id'] = $id;
$pdo->prepare("UPDATE customer_info SET $setStr WHERE customer_id = :id")->execute($params);
```

Feel free to ask if you have any other questions

Jeremy, 15.09.17 11:04

Thank you for your reply. I will give more detail into my reasoning.

I have an online store i have built and i am afraid that a customer may be left on a blank page or an abruptly ended script without reason during the checkout process, or post checkout process receiving their order confirmation. I've encountered such an error myself on another website from having an intermittent internet connection and being left without any notification it leaves fear of having paid for something without the order having being processed properly on their end.

I've been trying to handle every single type of error imaginable so my code is full of try... catches as well as a billion if statements. I'm desperately trying make my coding much more efficient.

My use of the try... catch operator would also be that if there was an error on a product page I would have liked it to just keep the script running and display the 'Sorry we could not find your product' situation, as a kind of diversion until i can fix what is wrong.

I will read you article on error reporting in further depth.

Thank you

REPLY:

Hello Jeremy!

I am eager to hear what are your thoughts after reading the article? Do you agree that centralized error reporting is better than catching every exception separately?

At the very least you an create a global try..catch wrapping your whole code from start to end. But To me, using a handler is more elegant solution

David Entwistle, 11.09.17 18:05

Please can you help me resolve an issue? I have a user input box which starts a search of a PDO MySql table :- using this code snippet.<input type="text" name = "t1" id="t1" placeholder="Enter an RD number"> <input type="button" name="button1" value="GO" onClick="aa();"> </td> <td colspan="2" align="center"> </td> via this code- <script type="text/javascript"> function aa() { var xmlhttp=new XMLHttpRequest(); xmlhttp.open("GET","getrd.php?Regno="+document.getElementById("t1").value,false); xmlhttp.send(null); document.getElementById("d1").innerHTML=xmlhttp.responseText;<script> this returns 3 data items. All this works. My issue is how can I put these 3 data items into <input type="text" name = "Rd" id="ret1" placeholder="Rd No" > <input type="text" name = "Dn" id="ret2" placeholder="DN"> <input type="text" name = "St" id="ret3" placeholder="Status"> I have tried a number of variations of the above code without success, I am sure there is a way to do it but I am unable to get there yet. Many thanks. David.

Jeremy, 08.09.17 01:01

So i understand that i shouldn't use try .. catch for error reporting. But is it fine to use for displaying a custom notification to the user that an error has occurred?

Furthermore, if omitting rowCount as a boolean flag and using the $stmt variable to see if any data was returned;

```
 $data = $pdo->query("SELECT * FROM table")->fetchAll();
if ($data) {
    // You have the data! No need for the rowCount() ever!
}
```

If an exception has been thrown and caught the $data variable remains set and giving a true boolean value even though there is no data present, and it doesn't seem possible to globally unset($data) inside the catch as it's a function.

I would like to catch an error to notify the use 'Something went wrong' and to return a false boolean flag to handle the rest of the script outside of the try... catch. How would you recommend the best way i can achieve this?

I'm still learning programming.

REPLY:

Hello Jeremy!

Sorry for the delayed answer, I've been on vacation.

Regarding the first question, it is *extremely seldom* when you need to provide a custom error message. Given you are still learning, it is possible that your understanding of the error reporting is rather wrong. Kindly read this article, https://phpdelusions.net/articles/error_reporting (https://phpdelusions.net/articles/error_reporting) and I hope you'll change your mind.

The second question is simpler to answer. If an exception has been thrown and caught, the $data variable is **not set**, so there is no trouble with unsetting it whatsoever.

If you want to continue the script execution even in case of error, it's a **fair use** of the try..catch operator. But still, I cannot imagine the practical case. Could you please comment back with a little more detailed description of the scenario you have in mind?

Adri, 03.09.17 18:25

Thank you for that. However, I'm now confused even more. I get: "Fatal error: Call to a member function query() on null" and then the path to the document... Alternatively, I get a blank page

REPLY:

Well, this one is simple. All you need is to read the article above, Particularly the error reporting (https://phpdelusions.net/pdo#errors) part. You didn't tell a database to report mysql errors and thus getting this cryptic one. Just make it the proper way and see.

Adri, 03.09.17 15:54

Thanks very much (hope formatting is correct). Here it is. I created one class for connecting to my database:

```
 <?php
class Database {
private static $instance = null;
private function __construct(){}
     public static function getInstance() {
     if(is_null(self::$instance)){
     self::$instance = new PDO("mysql:host=localhost;dbname=elektroakustika;", "root", "");
     }
     return self::$instance;
      }

     }
```

And now I'm making one that is supposed to handle queries (called it ActiveRecord as that is the pattern I'm trying to implement):

```
 abstract class ActiveRecord {
    private static $conn;
    public static function getAll(){
        $conn = Database::getInstance();
        $table = static::$table;
        $q = $conn->query("select * from {$table}");
        $res = $q->fetchAll();
        return $res;
    }
}
```

Finally, I plan to create a class for each table in my db, that will extend the previous class and use one of the methods I'll create in ActiveRecord. Something like this:

```
 class Autor extends ActiveRecord {
    public static $table = "autor";
}
$conn = Database::getInstance();
$allAutors = Autor::getAll();
print_r($allAutors);
```

When I run the last code, I get the error, so obviously, my query is wrong. I realize this is quite basic, but I can't seem to solve the problem. Thank you!

REPLY:

Hello Adri!

Sorry I forgot to mention that a code should be padded by empty lines as well. But nevermind, I already added them.

First and foremost: the "error" you are getting is not just a red card telling you that you failed something. It usually **explains** what specific error you made. So it's always worth to read what the error says or at least provide to a person you are asking for help. There could be thousands errors, and its no use to stare at the code looking for them if PHP can tell you that already. Especially because it can be caused not by the code itself but by some other issue - a database, or server config or whatever.

As of your code, I tried it and it works.

Adri, 02.09.17 16:49

Hi! I'm trying to implement PDO in my code (I'm a complete noob, by the way) and I have a "why won't this work" type of issue. Would you mind checking it out (I'm not sure if that is OK, which is why I'm not sharing the code now)? Thanks for your explanations, by the way, they are incredibly useful!

REPLY:

Hello Adri!

Yes it's all right to ask a question here. Just make sure that your code is indented by 4 spaces, so it will be nicely formatted

Valdas, 27.08.17 23:18

Thanks a lot, PDO is FTW :)))

K, 25.08.17 17:50

Hi, sorry if this is a double post! I wasn't sure if my other comment was in moderation or disappeared.

Thanks for the great tutorial, it's one of the best I've come across for learning to use PDO.

I was wondering if you could give an example of how to create a PDO update query where you can have a variable number of inputs. So for example if I have a form where I can update a customer's name, email, or phone number. But sometimes I want to update only one at a time without overwriting the information already in the database.

Tomi, 14.08.17 20:17

Awesome! I'm just learning PHP and this was the first tutorial that made any sense. Most have code without going to the details which are the most confusing part at the start

REPLY:

Hello Tomi!

Thank you for your kind words!

Can you do me a favor? I am going to add a section called "PDO examples" with practical examples of PDO usage. If you can think of any example case you'd like you see, please drop a comment.

abinax, 11.08.17 22:27

Thank You

Trevor S, 26.07.17 21:27

This is an awesome guide, thanks so much for this.

Christian, 24.07.17 18:05

So a normal use case would be like this.

```
 $table = “myTable”;
$table = "`".str_replace("`","``",$table)."`";
```

Now $table is:

```
 $table = `myTable`;
```

Then I use this for query:

```
 $sql = “SELECT * from $table”;
```

or use

```
 $sql = “SELECT * from `$table`”;
```

?

Thanks again.

REPLY:

Hello Christian!

If backticks are already added to the identifier, then there is no point to add them again. So it should be your first variant.

Christian, 24.07.17 16:37

Thank you for articles and your responses to questions. Can you please clarify the section ‘Prepared statements and table names’?

```
 $table = "`".str_replace("`","``",$table)."`"
```

I read that MySQL an identifier needs a (`) backtick if it contains special characters or is a reserved word. In your example we concatenate (`) backticks on both side of the identifier and replace single backticks with double backticks on original $table variable. So the new $table variable would be ``identifier`` because extra backticks are like an escape character(\) and that would remove one backtick? How does this sanitize $table variable ? Does this have anything to do with PHP Execution Operators? Thank you for your help!

REPLY:

Hello Christian!

Thank you for the good question. Look at this from the point of view of the regular string formatting. To format a string we need to enclose it into delimiters and then escape those delimiters that could happen to be inside. Exactly the same we are doing here: We are adding delimiters (single backticks around) and then escaping these that are could be already inside of the table name. But we don't double the surrounding backticks, neither removing any. The above statement could be split into two:

```
 $table = str_replace("`","``",$table); // escape possible delimiters inside
$table = "`$table)`"; // adding backticks
```

Imagine we have a table name coming from the user input and a hacker put a backtick inside. Without doubling, with only surrounding backticks we'll get an injection:

```
 $table = "table` UNION ... --"; // coming from user
$sql = "SELECT FROM `$table` LIMIT 10";
```

will result with malicious query

```
 $sql = "SELECT FROM `table` UNION ... --` LIMIT 10";
```

whereas with doubled backticks inside the whole statement will make a query like this

```
 $sql = "SELECT FROM `table`` UNION ... --` LIMIT 10";
```

Where

```
 `table`` UNION ... --`
```

is a single identifier (a-non existent of course, which will produce an error, but an error is better than injection).

Hope it is clear now. If not, please ask :)

Christian, 24.07.17 16:32

Thank you for articles and your responses to questions. Can you please clarify the section ‘Prepared statements and table names’?

```
$table = "`".str_replace("`","``",$table)."`"
```

I read that MySQL an identifier needs a ( ) backtick if it contains special characters or is a reserved word. In your example we concatenate ( ) backticks on both side of the identifier and replace single backticks with double backticks on original $table variable. So the new $table variable would be `identifier` because extra backticks are like an escape character() and that would remove one backtick? How does this sanitize $table variable ? Does this have anything to do with PHP Execution Operators? Thank you for your help!

Bill Hayden, 13.07.17 13:44

There seems to be a bug in the code sample that describes the foreach functionality:

```
 $stmt = $pdo->query('SELECT name FROM users');
foreach ($stmt as $row)
{
    echo $row['name'] . "\n";
}
```

This should be:

```
 $stmt = $pdo->query('SELECT name FROM users');
foreach ($stmt->fetchAll() as $row)
{
    echo $row['name'] . "\n";
}
```

Alternatively, you could put ->fetchAll() at the end of the $stmt assignment instead of in the foreach.

REPLY:

Hello, Bill!

There is no error and there is an explanation why it is so.

Let me suggest you to try the first code snippet yourself, see it actually works and then follow the link from the article explaining the trick :)

Christian, 12.07.17 22:17

Does Sanitizing like the following make sense?

```
 $pdo = new PDO('sqlite:/path/db/users.db');
$stmt = $pdo->prepare('SELECT name FROM users WHERE id = :id');
// filter your data first , especially important for INSERT, UPDATE, etc.
$id = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
// Automatically sanitized for SQL by PDO
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->execute();
```

REPLY:

Hello Christian!

If you mean `filter_input()` , it's superfluous for this code snippet. Using prepared statement is secure enough. So, an extra sanitization won't make too much sense.

Personally, I would make such a snippet quite shorter

```
 $pdo = new PDO('sqlite:/path/db/users.db');
$stmt = $pdo->prepare('SELECT name FROM users WHERE id = ?');
$stmt->execute([$_GET['id']]);
```

but it's only a matter of taste and you can keep with manual binding and named parameters.

David Entwistle, 10.07.17 16:22

Thanks for your reply. When a user has logged in they presented with a menu screen to choose where on the site they want to work. Then a menu second allows them to select a sub area to work in according to their user rights. its function is to open MySql tables via a PDO connection. this is the problem screen. it is taking too long to appear. Thanks David E.

REPLY:

It's hard to tell what could be a problem. But at least try to change "localhost" to 127.0.0.1 in the DSN.

If it's not the case, you need to investigate more, what particular operator in your code takes all the time.

David Entwistle, 10.07.17 15:01

Many thanks for your prompt reply. I was trying to avoid the "waiting for" messages. Your answer is very much appreciated. Thanks again. David E.

REPLY:

Can you please elaborate on these "waiting for" messages? May be I can offer some solution, but for the moment I am not quite getting what is it.

David Entwistle, 10.07.17 14:45

A question of good practice. Is it appropriate to create database connection at login time and then not close the connection until logging out ?

REPLY:

Hello David!

thank you for the good question.

It depends on what you're calling "login" and "logout".
In case you're talking of a user visiting your site, it would be impossible. The nature of PHP's life-cycle is atomic. It starts every time a user visits another page, generates the page to be shown, and then dies, a fraction of second after starting. And so with every new page a new PDO connection inevitably have to be established.

In case you are talking of a single script execution - yes, like it said in the article, the connection should be established only once.

You're welcome to ask if something is still unclear for you!

Sujata, 10.07.17 14:14

> Thanks for the great article!

David Entwistle, 06.07.17 18:06

Many thanks for the updated code, it is working perfectly now. Very much appreciated. Kind regards. David E.

David Entwistle, 05.07.17 14:55

Many thanks for rapid response to my PDO query and thanks for the conversion coding. I have this line in my connection code already, but I did add your line to the new script. PDO::ATTR_EMULATE_PREPARES => false, However I am still getting error message SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'LIKE ? ORDER BY Friendship LIMIT ?,?' at line 1. I am stuck now. Hope you can help me. Many thanks.

REPLY:

Hello David!

First, I must confess that I made a terrible mistake. I overlooked the severe vulnerability in your code. If $option is coming from the browser, then it's a disaster. You should sanitize this variable, checking it against an array of predefined values. And most likely this is the source of the error you are getting.

So the final code would be

```
 $options = ["name","price","qty"]; // set your own values
$key     = array_search($option, $options);
$option  = $options[$key];

$query = "SELECT * FROM friends1 WHERE `$option` LIKE ?
          ORDER BY Friendship LIMIT ?,?";
$stmt = $pdo->prepare($query);
$stmt->execute(["%$srch%",$start, $records_per_page]);
$data = $stmt->fetchAll();
```

The first three lines are responsible for the $option santitization.

William Entriken, 30.06.17 18:55

s/count(/COUNT(/g

William Entriken, 30.06.17 18:34

"Good ORMs are Doctrine, Eloquent, RedBean, and Yii::AR. Aura.SQL is a good example of a PDO wrapper with many additional features."

Please provide a review of these or maybe a separate post. It is a huge qualifier when you say "for real applications you'll need to use one of these bunch of libraries but for now let me tell you all about PDO..."

REPLY:

Hello William!

Thank you for a good suggestion. Indeed it's a very good idea to outline at least essential features of these libraries and to show why they actually to be preferred. I cannot promise it will appear soon, as it' pretty much an undertaking, but I'll definitely will start composing such an article

David Entwistle, 30.06.17 12:23

How may I convert this mysql query to PDO query.

```
$query = "SELECT * FROM friends1 WHERE $option LIKE '%$srch%'
         ORDER BY Friendship LIMIT $start,$records_per_page";
```

`$option` is user selected from dropdown list. `$srch` is user input search criteria.

REPLY:

Hello David!

There are two issues to keep in mind when converting such a query, both mentioned in this article:

1. First, you have to make sure that the whole search string is set to a variable, including wildcard characters
2. Second, make sure that emulation is turned off, to make LIMIT parameters binding easier

So the final code would be

```
 $query = "SELECT * FROM friends1 WHERE $option LIKE ?
         ORDER BY Friendship LIMIT ?,?";
$stmt = $pdo->prepare($query);
$stmt->execute(["%$srch%",$start, $records_per_page]);
$data = $stmt->fetchAll();
```

Hope it helps!

Jan Stekl, 20.06.17 03:37

Very, very good website for someone starting with PDO. Thank You for making it!

Ganesh, 17.06.17 09:25

Hi,

Trying to get last 25 row in ascending order but getting error any idea what could be error

```
 $sql = '(SELECT `id`,`msg`,`time`,`sender_id`,`mid` from msgs WHERE mid > 0 and `grp_id
` = :grp_id2 ORDER BY id DESC limit 25) ORDER BY id ASC';
$sql_array = array(
    'grp_id2' => $grp_id
);
$stmt = $data['pdo']->prepare($sql);
$stmt->execute($sql_array);
```

output

> Warning: PDO::prepare(): Unknown type 17 sent by the server. Please send a report to the developers in msgs.php on line 174 Fatal error: Call to a member function execute() on boolean in msgs.php on line 176

REPLY:

Hello Ganesh!

Thank you for your question. I never seen such an error message before. Looks like you already posted a bug on the PHP site, https://bugs.php.net/bug.php?id=74769 (https://bugs.php.net/bug.php?id=74769)

Looks like it's all you can do.

Ganesh Kandu, 15.06.17 14:38

Hi, Thanks for Awesome Documentation and easiest on

what should i care about when i am migrating to postgresql from mysql ???

Ganesh Kandu, 15.06.17 14:15

Thanks

Ganesh Kandu, 14.06.17 10:20

I Want to use dbprefix

```
 CREATE TABLE `%DBPREFIX%_poll_vote` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `ip` int(11) DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

in select,update and insert also should i use :dbprefix but not working

REPLY:

Hello Ganesh!

Unfortunately, PDO doesn't have a placeholder for the table name. So So all you can do is to use a PHP variable and make sure it is not coming from user input:

```
 CREATE TABLE `{$DBPREFIX}_poll_vote` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `ip` int(11) DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

bifa, 09.06.17 11:50

Very instructives Thanks.

Thapelo, 07.06.17 13:43

Thank you for your advice. I would like you to be my mentor. I have still going through your site and I am happy to tell you that I am really learning a lot.

Thank you very much.

Thapelo, 06.06.17 20:14

Hi, I found out that your site is very useful to me as i am a beginner to coding. I wanted to know whether to use mysqli or pdo. I searched and came to a point which i decided to pay more attention to using pdo than mysqli. My question is, am i on the right track with my decision or do i have to lern php with mysqli first?

REPLY:

Hello!

PDO is easier to use, therefore it is easier to learn as well. So I would recommend to keep up with PDO.

Greg, 02.06.17 18:24

"*In such a case explicit binding have to be used, for which you have a choice of two functions, bindValue() and bindParam(). The former one has to be preferred, because, unlike bindParam() it has no side effects to deal with.*"

You mention that bindParam has side effects, but there's no explanation as to what they are. PHP The Right Way and some other sites recommend bindParam, so I'm really curious why you recommend bindValue.

daniel phillis, 30.05.17 22:12

Thank you for providing a great resource. I want to reference your site in my undergrad paper to Flidners University South Australia, on serving my first database with WAMP, can i obtain your name ?

Mark, 25.05.17 20:13

I'm trying to run a stored procedure on a Sybase 11.0.1.2596 database in a PHP file using PDO and dblib as the driver. I can call a procedure with no parameters with no problems using something like:

```
 call custom.show_clocked_in_employees
```

However, if the procedure takes parameters, I get an error. If I have a procedure like this that takes 2 parameters:

```
 create procedure custom.custom_sp_R_cons_rvc_time_prd_ttls(in business_date_start timestamp
,in business_date_end timestamp)
result(start_business_date timestamp,end_business_date timestamp,store_number OBJ_NUM,store_
id SEQ_NUM,...)
begin
  declare @start_business_date timestamp;
  declare @end_business_date timestamp;
  ...
end
```

I have tried calling it many ways and received different errors including this:

```
 SQL Anywhere Error -188: Not enough values for host variables [13638] (severity 16) [(null
)]
```

etc. If I run this procedure in a SQL client like RazorSQL using:

```
call custom.custom_sp_R_cons_rvc_time_prd_ttls('2017-05-02', '2017-05-02')
```

it works perfectly. The exact way I'm trying to call it is like this:

```
 $dbh = new PDO('dblib:host=<host_ip>', $user, $password, null);
$stmt = $dbh->prepare('call custom.custom_sp_R_cons_rvc_time_prd_ttls(:start_date, :end_dat
e)');
$new_params = array(':start_date' => '2017-05-02', ':end_date' => '2017-05-02');
$stmt->execute($new_params);
```

John, 25.05.17 15:09

I would like to know more, is there any pdf's available about this tutorial I'm eager to learn

Alex, 23.05.17 11:11

Hi! Another brief question: I was expecting to receive the exact same results using this approaches, but it seems I got different results (results in terms of the output results/data of the query)

```
 $stmt->bindParam(':Lim', $L, PDO::PARAM_INT);
$stmt->bindParam(':Lim2', $L2, PDO::PARAM_INT);
$stmt->bindParam(':Name', $Name, PDO::PARAM_STR);
$stmt->bindParam(':Name2', $Name2, PDO::PARAM_STR);
$stmt->execute();
```

And then this one: $stmt->execute(["Lim" => $L, "Lim2" => $L2, "Name" => $Name, "Name2" => $Name2]);

I'm I missing something? Shouldn't have been the exact results?

Thanks!

REPLY:

Nope, as long as this code works, there should be no difference. The only case I can think of is when you are comparing a numeric value with a string field, something like `name=1` in SQL. In this case results are indeed unpredictable.

Do you mind to provide more details on the difference you are talking about?

Alex, 23.05.17 10:53

Hello, I am following closely your tutorial and I am really grateful for the useful information. Now I need your help again:

As you said, there is a problem with the Limit clause.. and I'm stuck in the middle.

On one hand, I need the emulation mode turned ON as I have to query for multiple fields having the same name, as in your example:

```
 $stmt = $pdo->prepare("SELECT * FROM t WHERE foo LIKE :search OR bar LIKE :search");
$stmt->execute(['search'] => "%$search%");`

The query is even more complex, type of:

$stmt = $pdo->prepare("SELECT * FROM t WHERE (((foo LIKE :search OR bar LIKE :search) AND (b
oo LIKE :search2")) OR ((foo2 LIKE :search OR bar2 LIKE :search)  AND (boo LIKE: search
2)) ... and so on )  LIMIT :Limit1, Limit2
```

And on the other hand I need the Limit parameters, for which the emulation needs to be OFF in order to function.

Otherwise, Is there any way to bind only the values for Limit parameters and continue executing: $stm ->execute(array_merge($params,${"in_params{$x}"}));

Thanks a lot!

REPLY:

Hello Alex!

Thank you for a good question!

Indeed there is no way to combine the execute and bindvalue methods. So in general you just have to write all the binding by hand. I suppose that for an exceptional case it is not a big deal.

However, I think there is a workaround, which is quite close to your requirements: you can write a loop, binding all other parameters from array, and then make two distinct bindvalue calls, something like this:

```
 for each($params as $key => $value)
{
    $stmt->bindValue($key, $value);
}
$stmt->bindValue('Limit1', $limit1);
$stmt->bindValue('Limit2', $limit2);
```

This is all I can offer. Hope it helps!

Boz, 14.05.17 01:30

Hi,

I have a piece of hardware that collects data and operates a machine and then you can log into it to get statistics. A second operation it does (not very well) is send out data to a web database which stores what it sends. Every tutorial i find with api for php uses mysqli connect or other bad practices. Do you have to do a tutorial with the right pdo framework?

Friend, 04.05.17 17:07

There's a misspelling in the Transactions section:

"to cancel all the changes you made sihce transaction start"

sihce should be since

REPLY:

Thank you, fixed.

MgAnt, 24.04.17 19:37

It's c. 2017, seven years after mysql 5.5.3 was published, which supports utf8mb4 - don't you think you should suggest this as encoding instead of utf-8?

Dave, 21.04.17 23:17

> you can use the method chaining and thus call execute() right along with prepare():

```
 $sql = "UPDATE users SET name = ? WHERE id = ?";
$pdo->prepare($sql)->execute([$name, $id]);
```

> However, if you want to get the number of affected rows, the code will have to be the same boresome three lines:

```
 $stmt = $pdo->prepare("DELETE FROM goods WHERE category = ?");
$stmt->execute([$cat]);
$deleted = $stmt->rowCount();
```

Have you considered

```
 ($stmt = $pdo->prepare("DELETE FROM goods WHERE category = ?"))->execute([$cat]);
$deleted = $stmt->rowCount();
```

REPLY:

Hello Dave!

Everyone is tempted to do that at first, but soon realizes it's impossible :)

The method chaining returns the return value of the *last method* in the chain, `execute()` here, and it's a boolean value. So it won't work.

To overcome this inconvenience I propose to extend PDO a bit, as it shown in the other article, https://phpdelusions.net/pdo/pdo_wrapper (https://phpdelusions.net/pdo/pdo_wrapper)

So we can make it

```
 $deleted = $pdo->run("DELETE FROM goods WHERE category = ?",[$cat])->rowCount();
```

Phil Wilkinson, 21.04.17 01:29

Got caught out last night when upgrading our production server's PHP7 version and modules (most likely php7-pdo?). PDO couldn't parse this previously working DSN:

```
mysql:host=localhost:13304;dbname=test;charset=utf8
```

Fixed with this (note the port has moved in the DSN):

```
mysql:host=localhost;dbname=test;port=13304;charset=utf8
```

Better to stay safe and follow the manual gents.

Priya Telvekar, 19.04.17 18:50

Hi, really good stuff to know. Helped me!! Thanks!

Jeff, 18.04.17 23:56

I'm having a hard time gaining an understanding of best practices of real world implementation of PDO. Especially file structuring - as in where my PDO connection should be located and efficiently used throughout a php app. If this is not within the scope of this site perhaps you can direct me to a good reference. Thanks!

REPLY:

Hello Jeff!

Thank you for the good question, it's perfectly within the scope of this site.

In general, you just create a separate file where PDO instance is created, like `$pdo = new PDO ...` . And then just include/require that file into every script that needs a PDO connection.

The main question here is how to use that instance. And this answer depends on your coding style. If your code is procedural, then just use that `$pdo` variable. To make it accessible inside functions, either make it `global` or use a static wrapper like one is shown on the next page: https://phpdelusions.net/pdo/pdo_wrapper (https://phpdelusions.net/pdo/pdo_wrapper)

If your code is OOP, that it's a different tory, too big to cover in a comment. You may google for the Dependency injection and IoC containers.

If you still have some questions, I'll be glad to answer.

Alex, 21.02.17 16:24

So the following query seems to not work:

```
 $sql = "SELECT u.*, AVG(Rating) As AverageRating FROM users u
        LEFT JOIN RatingsTable r WHERE r.UserID=u.id
        WHERE Active = 1";
$users = $pdo->query($sql)->fetchAll();
```

I don't know if is due to the fact that there are 2 WHERE clauses (I've removed one). Also as logic I don't know if it is intended to return what I expect.

- So I need to retrieve all the details from Users (from those users that are active);
- then, for the users that have Ratings (and the ratings are also active), I need to see the ratings, otherwise I'll just stick with the user's info and mention that there is no rating for him.

Isn't the JOIN supposed to return only those that have ratings?

Thanx!

REPLY:

Hi Alex,

Sorry it was a copy paste error. There should be ON instead of first WHERE. Regarding JOIN, the the original one indeed will get you only users with ratings. But LEFT JOIN is a special join to give you all users regardless.

As of your other question, remember those u and r used in the query? they are called aliases and intended to distinguish field names from different tables. So the final query would be

```
 $sql = "SELECT u.*, AVG(Rating) As AverageRating FROM users u
    LEFT JOIN RatingsTable r ON r.UserID=u.id AND r.Active = 1
    WHERE u.Active = 1";
$users = $pdo->query($sql)->fetchAll();
```

And, regarding your last comment, PHP is all right.

Alex, 21.02.17 16:02

Hi again, (it seems that my first comment wasn't posted - It was submitted several minutes before this one: Alex, 21.02.17 01:36)

OK, thanks a lot for suggestions, I'll give it another shot during this night. Only one more question here:

- as my both selects contain "WHERE ACTIVE = 1" it refers one time at User being active and the second time it refers the Rating being Active = 1 (so we have to make sure that users is activated and that his rating has been approved). Using your suggested code, could I make something like this? $sql = "SELECT u.*, AVG(Rating) As AverageRating FROM users u WHERE Active = 1 (## Meaning User to be active ##) LEFT JOIN RatingsTable r WHERE r.UserID=u.id WHERE Active = 1"; (## Meaning the Rating for the corresponding User - to be active as well ##)

Thanks a lot!

P.S. My first comment (really first comment that I cannot see it here / maybe it wasn't even submitted successfully) was about your suggestion between PHP / Java / Python for a good, robust, stable but flexible web application, based on your experience. Maybe also suggesting a set of "knowledge toolkit".

Kind regards, Alex

Alex, 21.02.17 01:36

Hello again, How do you see a better implementation for this case, as migrating it from simple mySql to PDO is seems a lot slower: I have multiple users to retrieve from DB and show their information, but some information is stored in another tables, so then I have to run another query in the while loop, for each User ID

Example:

```
 $stmt = $pdo->prepare("SELECT * FROM users WHERE  Active = 1 ORDER BY RAND ()");
$stmt->execute();
$i = 0;
foreach ($stmt as $row) {
    $Fname[$i] = $row['Fname'];
.... and so on

    $getRatings = $pdo->prepare("SELECT AVG(Rating) As AverageRating FROM RatingsTable WHER
E UserID= '$UserID[$j]' AND Active = '1'");
     $getRatings->execute();

     $i++;
}
```

so the point here is that I'm assigning the details from DB to arrays of First Names, Last Names, User ID's, then Ratings and some other information that I need to retrieve from other tables based on the specific UserID. Is there a better, cleaner, faster way to do it, maybe not using so many arrays and using the $row['anythingHere'] variables and using it / displaying in the loop (not just storing / assigning in the loop and then use it later based on the array indexes).

Thanks a lot!

REPLY:

Hello Alex!

That's a really good question. Indeed, there are means for the improvement, both for the performance and cleaner code.

From what I see, there are three main points

- Take out ORDER BY RAND() part. It doesn't seem to do any good, but can slow down things considerably.
- Take out the nested query as well, and use LEFT JOIN instead. You can join as many tables as you wish
- Get all records at once in a single array using fetchAll() method

So it should be something like this

```
 $sql = "SELECT u.*, AVG(Rating) As AverageRating FROM users u
        LEFT JOIN RatingsTable r WHERE r.UserID=u.id
        WHERE Active = 1";
$users = $pdo->query($sql)->fetchAll();
shuffle($users); // if you need them to be random
```

Now you can loop over single $users array and get any column. Fname, for example:

```
 foreach($users as $row) {
    echo $row['Fname'];
}
```

If the code still be slow, make sure that you have an index for the UserID field in AverageRating table (and in the all other linked tables as well).

Feel free to ask if you have any other questions!

quraisah, 20.02.17 17:22

Thanks it help me.thanks again if i had any inquiry regarding pdo i will back to this web again.

quraisah, 20.02.17 10:38

how suppose im write using pdo style for user redirect their respective webpage after login. example admin login will re direct to admin page and normal user redirect to user page after login. please show some simple example.

REPLY:

Hello!

this question doesn't really belong to PDO as you have to deal with the information already fetched from the database. Assuming you have a variable `$row` with user data where level `field` is responsible for the user level, the code would be

```
if($row['level'] == 'admin') header("Location: /admin.php")
```

You can add as many such condition as you like. Hope it helps!

Relexk, 18.02.17 01:19

how do i make something like this with pdo

$userquery = mysql_query("SELECT fname, lname, email FROM register WHERE email ='$login_session'") or die("the query could be fale please try again"); if(mysql_num_rows($userquery) != 1){ die("that username could not be found!"); } while($row = mysql_fetch_array($userquery, MYSQL_ASSOC)){

```
 $dbfname = $row["fname"];
$dblname = $row["lname"];
$dbemail = $row["email"];
}
if($login_session != $dbemail){
        }
```

thanks i advance

REPLY:

Hello!

With PDO this code will be much simpler:

```
 $stmt = $pdo->prepare("SELECT fname, lname, email FROM register WHERE email =?");
$stmt->execute([$login_session]);
$row = $stmt->fetch();
if($row){
    die("that username could not be found!");
}
$dbfname = $row["fname"];
$dblname = $row["lname"];
$dbemail = $row["email"];
```

Pablo, 14.02.17 00:30

Thank you! Great job. You helped me to understand better what I was just assuming as best practice.

Andy, 08.02.17 16:56

How can I use PDO for this kind of statement:

```
 $sql1 = "INSERT INTO dates (d_insertdate, d_date, d_time, d_name";
$sql2 =  ") VALUES (SYSDATE(),'" . $d_date . "', '" . $d_time . "', '" . $d_name . "'";

if (strlen($d_admission) > 0) {$sql1 = $sql1 . ", d_admission"; $sql2 = $sql2 . ", '$d_admis
sion'";}
if (strlen($d_artists) > 0)   {$sql1 = $sql1 . ", d_artists";   $sql2 = $sql2 . ", '$d_artis
ts'";}
if (strlen($d_infotext) > 0)  {$sql1 = $sql1 . ", d_infotext";  $sql2 = $sql2 . ", '$d_infot
ext'";}
```

Thx!

REPLY:

Hello Andy!

Thank you for a real good question.

For PDO there is no simple solution. But it can be done with a code like this

```
 $sql = "INSERT INTO dates (d_insertdate, d_date, d_time, d_name";

$parameters = [$d_date, $d_time, $d_name];

if (strlen($d_admission) > 0) {$sql .= ", d_admission"; $parameters[] = $d_admission;}
if (strlen($d_artists) > 0)   {$sql .= ", d_artists";   $parameters[] = $d_artists;}
if (strlen($d_infotext) > 0)  {$sql .= ", d_infotext";  $parameters[] = $d_infotext;}

$sql .= ") VALUES (SYSDATE()".str_repeat(", ?", count($parameters)).")";

$pdo->prepare($sql)->execute($parameters);
```

Here we are using just a single $sql variable which is getting field names based on your conditions. Also, we are adding all our variables into `$parameters` array.

Finally, we are adding VALUES part to the query, creating as many `?` marks in the query as many items in the `$parameters` array.
And all we need now is to prepare a query and execute it with parameters.

Feel free to ask if you have any further questions!

raNor, 03.02.17 01:25

Thanks a lot for this. As a beginner with PDO this taught me a lot.

I will definitely share your work with my classmates.

Cheers,

Steve R, 28.01.17 12:19

Thank you for this extraordinary work. I found your concise, well formed presentation of material an easy read and the explanations, examples and insights on technique to be extremely helpful. My gratitude to you for your work on and in, presenting this document.

Greg, 26.01.17 04:12

Lots of useful details on PDO! I use wizzyweb as it automates the code creation for data entry and reports, even prepared statements for safer code.

Albert221, 11.01.17 01:10

Hello!

I was at the beginning of writing a complex tutorial on how to use PDO in polish, because there's not even one **good** tutorial about it and while I was researching for some resources other than PHP manual I encountered your The only proper PDO tutorial and I got an idea to translate it into polish, of course give you a proper credit and publish it on my blog (http://nastoletni.pl (http://nastoletni.pl)).

I'd be very grateful and I think polish beginners would be really grateful too for such a quality tutorial in such an important and trivial thing which is connecting to database in PHP.

Don't mind sending an email if you prefer it.

Best regards, Albert Wolszon

REPLY:

Hello Albert!

Thank you for giving this article such a credit! That would be a honor for me if you translate it to your native language. Surely, I am giving you the permission to translate.
Please send me a link when it will be done!

Also, feel free to ask for clarifications, whenever you find it necessary.

Kalule, 08.01.17 10:53

Thanks for the the articles. Very educative even for experienced programmers. Keep up the good work bro.

REPLY:

Hello Kalule!

Thanks a lot for your feedback! May I ask, what you find especially interesting in this article? and also, may be there are some parts that are not very clear or detailed? Please share your opinion.

Danilcha, 04.01.17 05:29

In this example: $stmt = $pdo->prepare("DELETE FROM goods WHERE category = ?"); $stmt->execute([$cat]); $deleted = $stmt->fetchColumn();

Shouldn't it be rowCount instead of fetchColumn?

REPLY:

Hello Danilcha!

Sure, that's my fault. Fixed. Thanks for pointing out!

Viktor, 03.01.17 17:42

Hi,

You seem to be very a skilled PHP developer, could you link me a good PHP tutorial so I can start learning the language the right way? Right now I have a website that I wrote from scratch using HTML, CSS of course and a bit of copy-pasted, slightly edited JS. The only PHP I'm using are include statements so that my pages don't become too hard to maintain. I'd like to be able to load images, table content and text from a database, dynamically creating a page the same way it's also done on forums and webshops. I understand that PHP 7 is relatively new so most of the tutorials out there are written for PHP 5.6 or older. Is this a problem or is the conversion to 7 easy?

Thanks,

Viktor

REPLY:

Hello Viktor! Thank you for the good question.

The best resources to learn Php are http://www.phptherightway.com/ (http://www.phptherightway.com/) and http://www.laracasts.com/ (http://www.laracasts.com/)

As of the version, just don't worry: php7 is 99.9% the same as 5.6.

Feel free to ask other questions!

Tyler, 02.01.17 09:39

Just a note, if you are using an IN clause, and you need to ->execute more values (besides those in the IN clause), this seems like a clean way to do it:

```
$query->execute(array_merge(
   array ($value1, $value2),
   array_merge ($IN_CLAUSE_ARRAY,
       array ($another_value, $another_value))));
```

REPLY:

Hello Tyler!

Thank you for the suggestion!

Indeed, last time I've answered the similar question on SO, I tought I should add such an example but got busy and forgot it.

Yet I think that you made your example a bit overcomplicated. You need only one `array_merge()` call:

```
$query->execute(array_merge(
   array($value1, $value2),$IN_CLAUSE_ARRAY, array ($another1, $another2)));
```

Please check the updated section: https://phpdelusions.net/pdo#in (https://phpdelusions.net/pdo#in)

M, 31.12.16 05:39

Question(s) regarding "Mysqlnd and buffered queries. Huge datasets.".

With the mysqlnd driver, what does that mean for fetch vs fetchall?

- Will the database be "hit" multiple times using fetch and only once using fetchall?

Thanks.

Joe, 26.12.16 22:49

Hi, If I wanted to save a single row from my database as a variable for later use, would this be the correct way to do it ?

```
 $stmt = $pdo->prepare("SELECT count     FROM punch WHERE id=1 LIMIT 1");
$stmt->execute();
$row = $stmt->fetch();
$counter = $row['count'];
echo $counter;
```

Thanks :)

REPLY:

Hi Joe!

You just have to read this tutorial a little further (https://phpdelusions.net/pdo#fetchcolumn) ;)

Depends on whether you're using a variable in your query or not, there will be either one line or three lines, not five. Just check the code by the link and choose the variant that suits you best.

Joe, 26.12.16 20:46

Hi, I am following your code, however I am getting the following error.

> Parse error: syntax error, unexpected '[' in connect.php on line 9

REPLY:

Hello Joe!

This one is very easy to fix. 5 years ago a new syntax for arrays was introduced in PHP - square braces instead of `array()`. You can change your code as follows. But better consider upgrading your PHP to a more recent version

```
 $host = '127.0.0.1';
$db   = 'test';
$user = 'root';
$pass = '';
$charset = 'utf8';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$opt = array(
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
);
$pdo = new PDO($dsn, $user, $pass, $opt);
```

Hope you get the idea

HashHashPHPNinjaCat, 21.12.16 03:03

Hi everyone!

If you you wish to be able to full Unicode in MySQL database, I would read here on how to do that utf8mb4

Make your code appear like this:

```
 $host = '127.0.0.1';
$db   = 'test';
$user = 'root';
$pass = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$opt = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];
$pdo = new PDO($dsn, $user, $pass, $opt);
```

Notice the "utf8mb4" instead of "utf8". 😸

Then you will be able to store these into your database.

REPLY:

Hello PHP Ninja Cat!

Thank you for your suggestion!

Ian, 12.12.16 22:26

[Thank you for your reply dated 12 Dec 16] Hello again. After using the code above:-

```
 $host = '127.0.0.1';
$db   = 'test';
$user = 'root';
$pass = '';
$charset = 'utf8';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$opt = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];
$pdo = new PDO($dsn, $user, $pass, $opt);
```

and inserting the server ip address, my database user name, password, database name. I'm finding it don't work. Am I going wrong?

Would you write a sample code which will, Connect and open the database, create a table and finally insert data and close? Thank you.

REPLY:

Hello Ian!

Well, this this is the exact code that should work, given you provides the correct database credentials. To avoid a confusion, please keep in mind the following:

- when something doesn't work, always provide the actual outcome - the error message, the information shown, etc. You know it's hard to tell anything if you can't see the actual result
- this code have to be used with PDO only. whatever else functions like mysql_query or mysqli_query won't work with it.
- Taken by itself, this very code is not supposed to "work". It just connects to a database, that's all. What is your expectations for this code, how it's supposed to work?

Ian, 12.12.16 02:07

Hello. Just a quick query, I'm currently teaching myself php, and started looking at database. I'd just like to ask, how do I insert using mysqli as the data is been displayed on the screen only, and not been inserted into the database? Secondly is pdo better, if so, how do I get started with it?

Thank You

REPLY:

Hello Ian!

Thank you for your questions.

For the first question it's hard to tell, without a code. All I can tell is that you have to use a prepared statement for your query because it helps you to avoid many problems.

As of the PDO - you just landed on the exact page you need: this is a very good PDO tutorial. Feel free to ask any question you have.

Hannes, 09.12.16 09:58

I really like your page and whole blog :) everybody should use code like you explain. If you ever release a book i'll buy it :)

Sam Judge, 07.12.16 21:57

Oh wow didn't know about the trailing comma in the arrays, learn something new every day :D Great tutorial, very comprehensive and useful !

REPLY:

You're welcome!

Sam Judge, 07.12.16 12:47

You have an error in one of your first snippets :

```
 $opt = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];
```

You have a comma after your last key-value pair! I'm sure that's not right :)

REPLY:

Hello Sam!

Thank you for taking care, but there is no error:)
Unlike JS, in PHP it's pretty legitimate (http://php.net/manual/en/language.types.array.php) and hugely useful too, allowing an uniform syntax for all array members.

grnadpa, 05.12.16 04:20

in the try-catch you allow, the

```
 try {
    $pdo->prepare("INSERT INTO users VALUES (NULL,?,?,?,?)")->execute($data);
} catch (PDOException $e) {
    if ($e->getCode() == 1062) {
```

did not work for me as the value was 23000 or some such on a duplicate. What did work for me was

```
 if ($e->errorInfo[1] == 1062) {
```

I'm running on a dell Inspiron 15 laptop under windows 10 using Chrome Version 54.0.2840.99 . PHP version: 5.6.21 MySQL 5.7

REPLY:

Thank you for the report!

Although it's just to illustrate the handling process, I have to dig into these codes deeper. I had the same issue before but no time to investigate and develop the universal solution. Will add to my TODO list.

Ian Shepherd, 23.11.16 19:05

This is a great article, and I have bookmarked the site for future use. Having seen your comment about inviting corrections to your English (which is excellent, by the way), I have one that will improve readability in this article: in several places, you say "one have to", which should be "one has to". The verb "to have" is conjugated like so: I have; you (singular) have; he/she/it/one has; we have; you (plural) have; they have. Hope this helps :)

REPLY:

Hello Ian!

Thank you for your kind words and a suggestion! Been busy recently, but finally got time to fix that.

My name, 19.11.16 15:55

That code is vulnerable to sql injection. you obviously havent tested it thoroughly.

Alan Cameron, 24.10.16 22:21

I would like more on how to INSERT INTO a MySQL database. I am having trouble understanding some code I have gathered from the internet and the insert function of the DB.php after assembling the data keys and values as fields and placeholders calls the query function of the DB.php to do the insert as a Query. It then checks to see if an error occurred using

```
 if (!$this->query($sql, $fields)->error())
{
    return true;
}
```

This is returning without any error but no data is inserted. What is the correct way using PDO to check for successful completion of the insert?

REPLY:

Hello Alan!

Thank you for the good question. Although the example you posted is not a genuine PDO (there is no error() method), and thus I cannot comment on it, I can answer the question regarding error handling in general.

Most of time, you don't have to check for the success. Just write your code out of the assumption that everything goes well. So in your case you can just return true, unconditionally.

As of the possible failure, you don't have to write any dedicated handling code either. Let me explain. You see, an error message like "Sorry insert failed" exists only in schoolbooks. While for the real life sites it is agreed upon, that on any unexpected error your program should just halt. Given PHP can halt on error by itself, you don't have to write any handling code by hand. However, this halting process consists of several steps. I am still in the process of writing a whole article on the matter but in short:

- on a live site, a complete error message should go into error log. Just to let you know what went wrong.
- while in the browser a generic error page have to be shown.

Both actions (logging and showing an excuse page) can be set up in a centralized way, written only once, somewhere else in your code. Means for the moment you can keep your code as is, while by the time it goes live, you will be able to add whatever centralized error handling code you wish.

To let PDO halt on errors, just set it in exception mode is it's shown in the article above, and avoid catching exceptions right in place.

Please feel free to ask if something is still unclear to you or you have some particular scenario in mind you have to handle.

Sameer sakhare, 07.10.16 14:55

Thanks for the reply and the solution.

craz-ola, 06.10.16 06:24

I really don't see the benefit of PDO. I've written almost 100 e-stores from the ground up with all SQL flavors. This just seems like a big headache and annoying. It's easy to prepare SQL statements and send them to a server. I personally don't think 'oh wow its gonna help my queries so much' ... I usually output the query in text to test it in a shell window.... and PDO is the type of thing that causes sites to crash , be slow, and be dumb.. because the developer put too much faith into it..! it just makes it harder to debug later....

REPLY:

PDO doesn't cause sites to crash or be slow. You confused it with something else.

robot, 05.10.16 18:40

pdo o mysqli?

REPLY:

Hello robot! :)

Here is an article for you, https://phpdelusions.net/pdo/mysqli_comparison (https://phpdelusions.net/pdo/mysqli_comparison)

Ionut, 05.10.16 10:19

Interesting and logical points of view on this site, I like it.

I have a question regarding exception handling though: ok, it makes sense to let exceptions bubble up the call stack and use an application-wide handler, but what do I do if I need to show an error message in a certain place on the page?

For instance, I have a page that displays a menu at the top, and underneath that menu is the place where I want success/error messages to appear. If the user tries to perform an operation in that page, and the operation fails because of the db, how can I display an error message in the proper place on the page if I use a generic exception handler?

REPLY:

Hello Ionut!

Thank you for a very good question.

Speaking of database errors, you don't actually show error messages for them. For the success message it's all right; For the validation errors it's all right; But for the system errors it is not.

For such a system-wide error like a database failure (or whatever else PHP error), it's better to show a static 500 error message. Because if your database failed, then most likely you won't get your page properly anyway - there are a lot of other queries that won't fire too. Just look at all these big boys - they don't show you anything like that on page reload.

The only case when such an error message can be shown is AJAX call. But again, for a server it's all the same: on failure it have to send an error HTTP status code and a static error page. Then either the latter will be shown (if requested directly) or the AJAX handler will consider the response status code (which is not 200) and show an error message.

If you still have any doubts, please provide a particular error example you are thinking of. It will make my answer more focused.

Sameer sakhare, 04.10.16 22:47

I was waiting for my comment to be approved by moderator as it has link of image file : explaining my Question in detail.

I really need answer for the question, can you please look into it.

REPLY:

Sorry, but there was no comment with an image. Can you please try to re-post it? Or you can reply to the notification email directly, with your image attached.

Jouni "rautamiekka" J?rvinen, 04.10.16 18:05

"if you need to know how many rows in the table, use SELECT COUNT(*) query."

Partially false. First, don't take every column into the counting, it's worthless wasting. Second, make use of indexes, especially the primary key (or closest unique key of same purpose if no primary exists) like this:

```
SELECT COUNT(`primarykeycolumn`) FROM `table` USE INDEX (`PRIMARY`)
```

REPLY:

Hello Joni!

Thank you for your contribution. Unfortunately, both your statements are not true.

The first one is just irrelevant to the question. count(*) has nothing to do with selecting columns. When counting, your database actually doesn't select anything, it just counts.

The second statement is wrong too, just because your database is smart enough to pick up the appropriate index by itself. You can test it yourself, by running query

```
EXPLAIN SELECT count(*) FROM table
```

and checking the KEY section. There will be a key, if you have one defined in your table.

Hope it is all clear now :)

Stefan, 03.10.16 06:28

You can't seriously call this a "proper PDO tutorial" when you don't ever say what "PDO" means, not even once. I had to google it to find out that it means "PHP Data Objects", which doesn't appear anywhere on this page (except now in my comment).

REPLY:

Funny, but you are right!

Dunno what practical use you can make from knowing that acronym meaning, or whether "PHP Data Objects" makes any better sense for you, but indeed you are right - I didn't bother to put it here. I will fix it, special for you!

I am glad it was the only factual error you found :)

Sameer sakhare, 02.10.16 17:23

Your post covers every thing about PDO and its execution, but i have following doubts:

1. What about sub-queries in where clause

eg :

```
Select * from tblUser Where nUserTypeID=(
   SELECT nTypeID from tblTypeMaster where vTypeName = 'admin')
```

sorry if query is not well formated

2. What about JOINS

REPLY:

The kind of SQL query is irrelevant to PDO: you can run any kind of query with PDO. You can run joins, sub-queries, stored procedures; set variables - whatever. PDO has nothing to do with your SQL - it will just relay it to database server.

Hope it's clear now.

Leslie, 27.09.16 15:29

You have a typo in the INSERT/UPDATE example:

```
$values[$field] = $source[$field];
```

should be

```
$values[$field] = $data[$field];
```

REPLY:

Thanks a lot! Fixed.

Danny, 17.09.16 11:42

I am not a robot >.>

REPLY:

Neither I am :)

Danny, 17.09.16 11:41

Hello

Bob, 24.08.16 14:23

You say "This is the main and the only important reason why you were deprived from your beloved mysql_query() function and thrown into the harsh world of Data Objects: PDO has prepared statements support out of the box"

But $mysqli->prepare supports prepared statements. So prepared statements do not seem to be a reason to use PDO instead of mysqli.

REPLY:

Thank you for catching that. Indeed the phrasing is not that clear. However, mysqli is not mentioned not implied here. This article is not about mysqli vs PDO comparison. It's solely about PDO, and it is expected that the reader is interested in PDO and this is why it mentioned here.

If you're looking for the article explaining why one should prefer PDO over mysqli, here is one: https://phpdelusions.net/pdo/mysqli_comparison (https://phpdelusions.net/pdo/mysqli_comparison)

Hope it is clear now.

Will, 24.08.16 14:19

As requested, a correction to the last paragraph on your landing page. You can delete this comment after using it.

Text with improved grammar: Disclaimer: I am not a native English speaker and have never even attended an English class, so there are tons of mistakes in the text for sure - I beg your pardon for them. Please, feel free to leave comments with corrections.

REPLY:

Thanks a lot! Corrected.

George Butter, 22.08.16 06:07

This is the best PDO tutorial I have found yet and I have SCOURED the internet. Your explanations are so much more logical than anyone else's and you have covered so much. Thank You.

Tyler, 07.08.16 00:54

If one should never use the rowCount() to count rows in database, what is the best way to handle pagination?

REPLY:

Hello Tyler!

Thank you for the good question. Pagination is exactly the case when one should never ever use the `rowCount()`.

By using this function you will ask a database to send into your PHP script all the rows found. It is highly inefficient and even may lead your server to failure.

Instead, just like it is said in the article, you have to ask a database to count the number of rows, by sending a query

```
SELECT count(*) FROM ...
```

and then reading that single result.

Feel free to ask if you have any further questions!

jamsden, 03.08.16 05:14

This site has taught me more than 100 others. Thanks for the clear explanation of these confusing points.

One comment, it might be helpful to absolute beginners like myself, if you showed both email and status in your placeholders example. It was a little confusing for me to make the leap from SQL to named placeholder substitution.

REPLY:

Hi there!

Thank you a lot for the kind words, and - especially - for the suggestion. Indeed it was a stupid for me to omit status in the examples. Fixed now!

You're welcome to share any other suggestions or confusions - it will help me to make the site better for other people!

Merlin, 30.07.16 19:49

Merlin rolls a +8 fireball against the robot.

robot, 26.07.16 22:02

iRobot

Cliff, 01.07.16 04:17

I have tried everything to get a piece of code to work. Can I pay you to give me a solution that works, as well as review other pieces of code I have written. If you reply to my email I will send the code. Unfortunately the time I have spent trying to learn this stuff has been sucking up all my time when I should be doing other things that make money.

Bell, 26.06.16 17:19

"Recently all PHP extensions that work with mysql database were updated based on a low-level library called mysqlnd, which replaced old libmysql client"

This means PHP 5.3+. The word "recently" should be replaced.

REPLY:

Thank you for your suggestion.

I will try to re-phrase it, but the problem is that replacement is not permanent. There are distributions, where libmysql is used by default. Means one could still encounter with outdated setup. So, for the end user the change is still recent, in the meaning one cannot rely on mysqlnd for sure.

gemozlobin, 25.06.16 12:56

It did not work, beacause PDO::FETCH_UNIQUE does not work with PDO statement::setFetchMode, it affect only if use in fetchAll method :(

REPLY:

Thank you for shring!

I will not delete your comments because it may help someone else. We all learn from our mistakes and sharing a mistake is as good as sharing an achievement.

I will add a clarification regarding setFetchMode() to the article.

Thank you for this finding again!

gemozlobin, 25.06.16 12:42

About previous comment Sorry it was my mistake. I use ORM over PDO and it did not work, but clean PDO is working. You can delete this and my previous comment

gemozlobin, 25.06.16 00:51

Thanks for great job But unfortunely PDO::FETCH_UNIQUE in mysql is not work as wrote in article. This options tell PDO return unique value not row indexed by key :(

REPLY:

Such a use case (to get only unique values) would make a very little sense, as any database can return you unique values already.

Besides, all PDO fetch modes are irrelevant to database backend, working exactly the same way for all databases including mysql.

I tested this code many times and always get the result as shown in the example.

What is your setup? Do you have an unique value listed first in the field list in your query?

Gerrit, 16.06.16 15:03

Hi, thanks for the excellent article. It gives some really helpful advice even for a little advanced developer.

REPLY:

Thank you for your kind feedback! This kind of response helps to keep the thing going. Check back for updates!

Julien, 06.06.16 22:28

Great article. It convinced me to switch from mysqli to PDO. There is just one detail that bother me, about the prepared statements and LIKE clauses. I may be wrong, but I'm pretty sure this don't work as intended:

```
 $search2 = "%$search%";
$stmt  = $pdo->prepare("SELECT * FROM table WHERE name LIKE ?");
$stmt->execute([$search2]);
$data = $stmt->fetchAll();
```

PDO will escape the two extra '%' in $search2 as it will escape every other '%' that may have been already in $search. It's the same thing for this:

```
 $stmt = $pdo->prepare("SHOW TABLES LIKE ?");
$stmt->execute(["%$name%"]);
var_dump($stmt->fetchAll());
```

A proper way to handle this might be:

```
 $stmt  = $pdo->prepare("SELECT * FROM table WHERE name LIKE CONCAT('%', ?, '%')");
$stmt->execute([$search]);
$data = $stmt->fetchAll();
```

REPLY:

This confusion is very easy to clear: pdo won't escape % symbol and has not a single reason to do so. Therefore all the above codes would work.

Your variant is perfectly legitimate too, but to me it's just too lengthy.

test, 30.05.16 14:54

This is a test to see if i am a robot works

REPLY:

Passed, of course. There is no armor to break.

AucT, 23.05.16 11:02

I mean what do you think about using for simple task smth like http://stackoverflow.com/a/27826114/1767461 (http://stackoverflow.com/a/27826114/1767461)

REPLY:

Thank you for the suggestion.

There is a serious flaw in the answer by the link provided, as it doesn't sanitize the field names.

You may find a correct solution in this article, in the following section: https://phpdelusions.net/pdo#identifiers (https://phpdelusions.net/pdo#identifiers)

Please ask if you find anything unclear or if you have any further questions.

AucT, 23.05.16 10:54

Hi! I would like you to put some simple active record function, to update and insert assoc array

wmm, 15.05.16 21:07

sorry, i wasn't clear. i was asking about what actually gets sent to mysql/maria (hereafter, myria). your comment about the artificial query made me curious about what PDO actually sends. i used wireshark to expose the conversation (WS is your friend :) and it appears the completely prepared query is sent to myria.

so, now, i'm a bit more confused - can you illucidate? (i'm really NOT trying to make a point here, it's just that inquiring minds need to know :)

REPLY:

Got you. It's ok with your confusion. I understand the feeling. You just need to put all the tiles together. Either way, I don't mind.

If you are using PDO with default settings, then emulation mode is used, when PDO is sending a regular SQL with all the values put i the query. It occurs when execute() method is called.

While with setting PDO::ATTR_EMULATE_PREPARES to FALSE, the behavior changes: PDO sends a query with placeholders, when prepare() is called. So my comment on the artificial query is only applicable to this mode.

If you are already turned emulation off and still observing SQL with data substituted - then it is something unusual.

wmm, 14.05.16 00:52

last question (promise) - you say that the query in mysql log is "an artificial query that has been created solely for logging purpose, but not a real one that has been executed" - exactly what is sent to mysql?

REPLY:

Well, first of all please promise to ask more. When asking, you are helping me to make this article better.

When in native mode (i.e. when emulation is turned off) the exact query with placeholders is sent. I.e. if you wrote

SELECT * FROM t WHERE id=?

then exactly the same query is sent to mysql and executed. This is how native prepared statements are intended to work. You may read my answer with explanation here: http://stackoverflow.com/a/8265319/285587 (http://stackoverflow.com/a/8265319/285587)

wmm, 13.05.16 23:58

thanks - in practice, never. but i ran across this catch-22 behavior experimenting with PDO in phpsh quite by accident. i thought there might be some way to recover, but, at the time, i had to exit phpsh then re-connect.

wmm, 13.05.16 22:13

i take it that, short of using buffered queries (or script exit), using PDO::query('SELECT something') will get you into a position where you can't recover, since there's no PDOStatement on which to do a fetch/fetchAll(). or did i miss something? i assume the situation clears on script exit and/or connection loss.

tnx for the write up, both here and in SO - i refer to your thoughts frequently.

REPLY:

Yes, you are right. Everything that was associated with the connection (unfetched resultsets, prepared statements, transactions etc.) will be cleared up on the connection close (and PHP will close a connection on exit).

However, I am not quite getting your point, why would you run a query without assigning the result to a variable.

Socrat, 11.05.16 10:48

You have an error in text: for all undefined properties **get magic method will be called if there is no** get method in the class, then new property will be created

PDO doesn't execute **get, instead it calls** set to set values. Cheers ;)

REPLY:

Thanks a ton! Fixed!

Trevor, 30.04.16 20:57

Thank you so much for this brilliant document. It really covers most of the things in an inspiring way.

REPLY:

Thank you for your feedback!

If you have any questions, I'll be glad to answer, as answering questions lets me make the article more clear and useful.

idk15, 26.04.16 06:45

One other downside to unbuffered queries is you can't run any other queries until the current query has returned all of its results. That's why it is more convenient for the user to have PHP retrieve all the rows of a dataset before iterating over them - doing so allows other queries to be run inside the while loop.

REPLY:

Thanks a lot for the reminder. Surely it's a really important issue. Added it to the text.

ydk2, 15.04.16 10:34

Yes great and simple.

brokenOval, 12.04.16 22:05

Great article - love the gotchas at the end. I would love to see an addition dealing with the INSERT... ON DUPLICATE UPDATE procedure (which I'm currently exploring) and how the single statement would take x number of bound variables on the INSERT with additional variables if the UPDATE procedure is sent.

REPLY:

Hello!

I am in doubts whether to add on duplicate case or not, as it's not directly related to PDO. Either way the answer is sumple.

This kind of query has a neat feature exactly for your case. You can refer to an insert part value using values() function (which is different from regular values()):

Insert into t (foo, bar) values (?,?) On duplicate key update bar= values(bar);

So you have to bind your variables only once. Hope you get the idea.

Either way feel free to ask any questions!

len, 09.04.16 17:53

This tutorial is a hackers wet dream. The statement $pdo = new PDO($dsn, $user, $pass, $opt);

could possibly dump your complete login details (server, user, pw, etc) if there is any issue with connecting to the db (timeout or whatever). This info is contained in the exception. I stopped reading after that statement, I don't care if it is mentioned later on, YOU SHOULD NEVER send that command naked. I know what you are going to say later on but what I wrote is 100% true and I tested it.

No wonder php has such a bad reputation.

REPLY:

PHP has its reputation due to users. Who is ignorant about most basic PHP settings. display_errors for instance. Which have to be turned OFF on a live site - a PROPER way for hiding error messages.

ALL messages, mind you, not only one you've accidentally became aware of.

Salagir, 01.04.16 16:30

Thanks for the great article! I've been modifiying my code already!

A few things:

> "but most of time it's all right and won't cause any problem. E.g." There is no example after that.
>
> $deleted = $stmt->fetchRow(); I didn't find this function in the manual. Are you thinking rowCount() ?
>
> Although you can tell rowCount() Font change for ")" stops too soon ;p

Geist, 10.03.16 04:51

Hi, I'm kinda new to PDO and would like some advice.

Right now I have three different tables in three different & separate databases (2 PGSQL, 1 MSSQL) that I have to INSERT into for user registration. This means I have 3 PDO connections, now all three INSERT are a bit different:

SQL 1: INSERT into user (Name, Password, Age) SQL 2: INSERT into users ([LOGINNAME], [PASS], [OCCUPATION]) SQL 3: INSERT into users (Name, Address, Position)

Now what I want to do is to somehow rollback the INSERTs if one or more fails to insert, and commit only if all three can be inserted? Can I rollback or commit outside of the try catch? or do I put all of them in one huge try catch block? or is there a better way that I can approach this?

REPLY:

Hi!

I am afraid I can't offer anything better.

It's impossible to have a proper transaction for the separate connections. If you commit outside of try catch, there could be an error during commit itself - yet you'll be unable to rollback.

But that's better than nothing - so, I think that you have to commit outside of try catch. You may also want to add another try catch to wrap commits, and try to recover successful inserts somehow.

Redeemer Pace, 25.02.16 01:02

Well done tutorial especially for who is new to PDO.

I use it as OOP mode on base class, way easier that way :P

Bad Habit, 24.02.16 21:06

Please tell about PDO::FETCH_CLASSTYPE, it comes handy when you store objects of different subclasses in a single table

Krash, 29.01.16 19:44

Not covered topic with inserting batches. E.g. INSERT INTO MyTable (col) VALUES (1),(2),(3); AND INSERT INTO MyTable (col) VALUES (1),(2),(3),(4); Is two separate prepared needed? $q1 = $pdo->prepare("INSERT INTO MyTable (col) VALUES (?),(?),(?),(?)"); $q1 = $pdo->prepare("INSERT INTO MyTable (col) VALUES (?),(?),(?)");

boogers, 25.01.16 09:27

nice job ripping the design from SO

REPLY:

You meant bootstrap and PHP.NET? ;)

Hugo, 20.01.16 16:13

@Ry nothing wrong with doing so, if you actually handle the error. If you are just echo'ing out the error message, you are better off not catching anything at all, you will get more information that way. And you can then turn off display_errors in production.

Ry, 14.12.15 02:08

great article, but what's wrong with wrapping pdo code by try - catch? I thought it was the way one handles exceptions in PHP?

REPLY:

I completely rewrote this section. Hope now it's clear.

David, 21.11.15 16:05

Thank you

Yaqoob, 16.11.15 17:57

This article pretty much clarified the proper useage of PDO, but sometimes it got over my head. old habits from mysqli_*

Sunny, 23.10.15 17:52

I'm trying to go from old, raw mysql to abstraction. PDO looks like a good path. Thanks !

Porky, 14.10.15 22:01

I am robot

mac ll, 29.09.15 04:28

good

