The mysql extension is deprecated and will be removed in the future: use mysqli or PDO instead [duplicate]

Asked 12 years ago Modified 8 years, 6 months ago Viewed 398k times Part of PHP Collective

177

This question already has answers here:

Why shouldn't I use mysql_* functions in PHP? (14 answers)

Closed 10 years ago.

When I attempt to connect to a MySQL server from PHP, I see the following error:

> Deprecated: The mysql extension is deprecated and will be removed in the future: use mysqli or PDO instead in /path/to/filename.php on line 123

The code on the referenced line is:

```
mysql_connect($server, $username, $password);
```

I am certain that the arguments are correct, and this exact code has been working for years without problem. Indeed, I obtained it from a well-sourced tutorial on PHP.

1. Why is this happening?
2. How can I fix it?
3. I understand that it's possible to suppress deprecation errors by setting `error_reporting` in `php.ini` to exclude `E_DEPRECATED`:

   ```
   error_reporting = E_ALL ^ E_DEPRECATED
   ```

   What will happen if I do that?

PHP mysql deprecated php

Share Improve this question Follow

asked Dec 19, 2012 at 3:11

eggyal

126k 18 216 242

1 There is a source scanning tool for conversion mysql to mysqli here: Converting to MySQLi (Dec 2011; by Keith Larson; Oracle Mysql Wiki) – hakre Sep 27, 2013 at 9:50

5 Use `pdo_query()`. It's the simplest upgrade path from mysql_, and uses PDO; which is the friendlier of the two alternatives. MYSQLI is really just a stop-gap API; sounds tempting to newcomers at first due to name similarity, but requires more rewriting due to the shifted function signatures, and then even makes parameterized queries more effort. – mario Dec 29, 2013 at 19:46

As to answer the question title - PDO is more general, and ***generally*** it is a better solution. @DennisDegryse => though I prefer object-oriented PDO, MySQLi also HAS an object-oriented way to use it (search for "mysqli class"). More on comparsion between PDO|MySQLi on code.tutsplus.com/tutorials/... and php.net/manual/en/mysqli.overview.php – jave.web Aug 19, 2015 at 11:19

Nothing found at Oracle Mysql Wiki (weblink commented above) . – David Augustus Jan 22, 2016 at 4:26

1 @hakre Thanks for the link to the tool. Just as a general comment, please be aware that you cannot simply convert mysql_ to mysqli_ for some commands, as they do function differently. – user1239087 Jan 19, 2017 at 3:45

1 Answer

Sorted by: Highest score (default)

188

1. Why is this happening?

The entire `ext/mysql` PHP extension, which provides all functions named with the prefix `mysql_`, was officially deprecated in PHP v5.5.0 and removed in PHP v7.

It was originally introduced in PHP v2.0 (November 1997) for MySQL v3.20, and no new features have been added since 2006. Coupled with the lack of new features are difficulties in maintaining such old code amidst complex security vulnerabilities.

The manual has contained warnings against its use in new code since June 2011.

2. How can I fix it?

As the error message suggests, there are two other MySQL extensions that you can consider: MySQLi and PDO_MySQL, either of which can be used instead of `ext/mysql`. Both have been in PHP core since v5.0, so if you're using a version that is throwing these deprecation errors then you can almost certainly just start using them right away—i.e. without any installation effort.

They differ slightly, but offer a number of advantages over the old extension including API support for transactions, stored procedures and prepared statements (thereby providing the best way to defeat SQL injection attacks). PHP developer Ulf Wendel has written a thorough comparison of the features.

Hashphp.org has an excellent tutorial on migrating from `ext/mysql` to PDO.

3. > I understand that it's possible to suppress deprecation errors by setting `error_reporting` in `php.ini` to exclude `E_DEPRECATED`:
   >
   > ```
   > error_reporting = E_ALL ^ E_DEPRECATED
   > ```
   >
   > What will happen if I do that?

   Yes, it is possible to suppress such error messages and continue using the old `ext/mysql` extension for the time being. But **you really shouldn't do this**—this is a final warning from the developers that the extension may not be bundled with future versions of PHP (indeed, as already mentioned, it has been removed from PHP v7). Instead, you should take this opportunity to migrate your application *now*, before it's too late.

   Note also that this technique will suppress *all* `E_DEPRECATED` messages, not just those to do with the `ext/mysql` extension: therefore you may be unaware of other upcoming changes to PHP that would affect your application code. It is, of course, possible to only suppress errors that arise on the expression at issue by using PHP's error control operator—i.e. prepending the relevant line with `@`—however this will suppress *all* errors raised by that expression, not just `E_DEPRECATED` ones.

What should you do?

- **You are starting a new project.**

  There is *absolutely no reason* to use `ext/mysql`—choose one of the other, more modern, extensions instead and reap the rewards of the benefits they offer.

- **You have (your own) legacy codebase that currently depends upon `ext/mysql`.**

  It would be wise to perform regression testing: you really shouldn't be changing *anything* (especially upgrading PHP) until you have identified all of the potential areas of impact, planned around each of them and then thoroughly tested your solution in a staging environment.

  - **Following good coding practice, your application was developed in a loosely integrated/modular fashion and the database access methods are all self-contained in one place that can easily be swapped out for one of the new extensions.**

    Spend half an hour rewriting this module to use one of the other, more modern, extensions; test thoroughly. You can later introduce further refinements to reap the rewards of the benefits they offer.

  - **The database access methods are scattered all over the place and cannot easily be swapped out for one of the new extensions.**

    Consider whether you really need to upgrade to PHP v5.5 at this time.

You should begin planning to replace `ext/mysql` with one of the other, more modern, extensions in order that you can reap the rewards of the benefits they offer; you might also use it as an opportunity to refactor your database access methods into a more modular structure.

However, if you have an *urgent* need to upgrade PHP right away, you might consider suppressing deprecation errors for the time being: but first be sure to identify any other deprecation errors that are also being thrown.

- **You are using a third party project that depends upon `ext/mysql`.**

  Consider whether you really need to upgrade to PHP v5.5 at this time.

  Check whether the developer has released any fixes, workarounds or guidance in relation to this specific issue; or, if not, pressure them to do so by bringing this matter to their attention. If you have an *urgent* need to upgrade PHP right away, you might consider suppressing deprecation errors for the time being: but first be sure to identify any other deprecation errors that are also being thrown.

  It is absolutely essential to perform regression testing.

Share Improve this answer

Follow

edited May 23, 2017 at 10:31

Community Bot

1 ● 1

answered Dec 19, 2012 at 3:11

eggyal

126k ● 18 ● 216 ● 242

4 please also recommended/suggest to use prepared statement ,,, many time i saw that user using pdo or mysqli query in the same way as mysql even they are not escaping single quote which is rather more dangerous – NullPointer Dec 19, 2012 at 4:25

4 @NullPointer: It already says "*they...offer...prepared statements (thereby providing the best way to defeat SQL injection attacks)*". I don't really want to give examples of parameterised queries in this answer, as it's not really relevant to the question at hand; how do you think it could be more clear? – eggyal Dec 19, 2012 at 9:19

3 If you want a quick and dirty fix, just put @ before mysql_connect to suppress it. eg. @mysql_connect(...); By this way, you do not have to change any other configurations. Using rest of mysql_ functions are ok. Only mysql_connect() gives this message. – Bimal Poudel May 4, 2015 at 13:03

@BimalPoudel: I already say that in my answer—"It is, of course, possible to only suppress errors that arise on the expression at issue by using PHP's error control operator—i.e. prepending the relevant line with `@`—however this will suppress all errors raised by that expression, not just `E_DEPRECATED` ones." – eggyal May 4, 2015 at 20:43

2 @FranKee: Where do you store these user-generated headlines, if not in a database? What is the "speak after me"-attack to which you refer? Using `mysql_real_escape_string()` without a MySQL database is ***absolutely definitely not*** the correct way to defeat anything (since it escapes strings according to the character set of your MySQL database connection, which you don't have!). If you're trying to prevent XSS attacks, you should be using `htmlentities()`. For any other attack, please elaborate on the exact threat. – eggyal Oct 6, 2015 at 15:38

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.