



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)

[PHPKonf: Istanbul PHP Conference 2017](#)

## [Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

## [Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Errors](#)

[Exceptions](#)

[Generators](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

## [Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[Using Register Globals](#)

[User Submitted Data](#)

[Magic Quotes](#)

[Hiding PHP](#)

[Keeping Current](#)

## [Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)  
[Handling file uploads](#)  
[Using remote files](#)  
[Connection handling](#)  
[Persistent Database Connections](#)  
[Safe Mode](#)  
[Command line usage](#)  
[Garbage Collection](#)  
[DTrace Dynamic Tracing](#)

## [Function Reference](#)

[Affecting PHP's Behaviour](#)  
[Audio Formats Manipulation](#)  
[Authentication Services](#)  
[Command Line Specific Extensions](#)  
[Compression and Archive Extensions](#)  
[Credit Card Processing](#)  
[Cryptography Extensions](#)  
[Database Extensions](#)  
[Date and Time Related Extensions](#)  
[File System Related Extensions](#)  
[Human Language and Character Encoding Support](#)  
[Image Processing and Generation](#)  
[Mail Related Extensions](#)  
[Mathematical Extensions](#)  
[Non-Text MIME Output](#)  
[Process Control Extensions](#)  
[Other Basic Extensions](#)  
[Other Services](#)  
[Search Engine Extensions](#)  
[Server Specific Extensions](#)  
[Session Extensions](#)  
[Text Processing](#)  
[Variable and Type Related Extensions](#)  
[Web Services](#)  
[Windows Only Extensions](#)  
[XML Manipulation](#)  
[GUI Extensions](#)

## Keyboard Shortcuts

? This help  
j Next menu item  
k Previous menu item  
g p Previous man page  
g n Next man page  
G Scroll to bottom  
g g Scroll to top

g h Goto homepage  
g s Goto search  
(current page)  
/ Focus search box

[PDOStatement::fetchObject »](#)

[« PDOStatement::fetchAll](#)

- [PHP Manual](#)
- [Function Reference](#)
- [Database Extensions](#)
- [Abstraction Layers](#)
- [PDO](#)
- [PDOStatement](#)

Change language:

[Edit](#) [Report a Bug](#)

## PDOStatement::fetchColumn

(PHP 5 >= 5.1.0, PHP 7, PECL pdo >= 0.9.0)

PDOStatement::fetchColumn — Returns a single column from the next row of a result set

### Description ¶

public [mixed](#) PDOStatement::fetchColumn ([ int \$column\_number = 0 ] )

Returns a single column from the next row of a result set or **FALSE** if there are no more rows.

#### Note:

**PDOStatement::fetchColumn()** should not be used to retrieve boolean columns, as it is impossible to distinguish a value of **FALSE** from there being no more rows to retrieve. Use [PDOStatement::fetch\(\)](#) instead.

### Parameters ¶

column\_number

0-indexed number of the column you wish to retrieve from the row. If no value is supplied, **PDOStatement::fetchColumn()** fetches the first column.

### Return Values ¶

**PDOStatement::fetchColumn()** returns a single column from the next row of a result set or **FALSE** if there are no more rows.

### Warning

There is no way to return another column from the same row if you use **PDOStatement::fetchColumn()** to retrieve data.

## Examples ¶

### Example #1 Return first column of the next row

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

print("Fetch the first column from the first row in the result set:\n");
$result = $sth->fetchColumn();
print("name = $result\n");

print("Fetch the second column from the second row in the result set:\n");
$result = $sth->fetchColumn(1);
print("colour = $result\n");
?>
```

The above example will output:

```
Fetch the first column from the first row in the result set:
name = lemon
Fetch the second column from the second row in the result set:
colour = red
```

## See Also ¶

- [PDO::query\(\)](#) - Executes an SQL statement, returning a result set as a PDOStatement object
- [PDOStatement::fetch\(\)](#) - Fetches the next row from a result set
- [PDOStatement::fetchAll\(\)](#) - Returns an array containing all of the result set rows
- [PDO::prepare\(\)](#) - Prepares a statement for execution and returns a statement object
- [PDOStatement::setFetchMode\(\)](#) - Set the default fetch mode for this statement

 [add a note](#)

## User Contributed Notes 2 notes

[up](#)  
[down](#)  
 42

[PhoneixSegovia at GOOGLE\\_MAIL\\_SERVER dot com ¶](#)

6 years ago

fetchColumn return boolean false when a row not is found or don't had more rows.

[up](#)  
[down](#)  
 9

[seanferd at assmasterdonkeyranch dot com ¶](#)

9 years ago

This is an excellent method for returning a column count. For example:

```
<?php
$db = new PDO('mysql:host=localhost;dbname=pictures','user','password');
```

```
$pics = $db->query('SELECT COUNT(id) FROM pics');  
$this->totalpics = $pics->fetchColumn();  
$db = null;  
?>
```

In my case `$pics->fetchColumn()` returns 641 because that is how many pictures I have in my db.

 [add a note](#)

- [PDOStatement](#)
  - [bindColumn](#)
  - [bindParam](#)
  - [bindValue](#)
  - [closeCursor](#)
  - [columnCount](#)
  - [debugDumpParams](#)
  - [errorCode](#)
  - [errorInfo](#)
  - [execute](#)
  - [fetch](#)
  - [fetchAll](#)
  - [fetchColumn](#)
  - [fetchObject](#)
  - [getAttribute](#)
  - [getColumnMeta](#)
  - [nextRowset](#)
  - [rowCount](#)
  - [setAttribute](#)
  - [setFetchMode](#)
- [Copyright © 2001-2017 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Mirror sites](#)
- [Privacy policy](#)

