What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $\_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

<?php  
$cookie\_name = "user";  
$cookie\_value = "John Doe";  
setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/"); // 86400 = 1 day  
?>  
<html>  
<body>  
  
<?php  
if(!isset($\_COOKIE[$cookie\_name])) {  
    echo "Cookie named '" . $cookie\_name . "' is not set!";  
} else {  
    echo "Cookie '" . $cookie\_name . "' is set!<br>";  
    echo "Value is: " . $\_COOKIE[$cookie\_name];  
}  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_cookie1)

**Note:** The setcookie() function must appear BEFORE the <html> tag.

**Note:** The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

<?php  
$cookie\_name = "user";  
$cookie\_value = "Alex Porter";  
setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");  
?>  
<html>  
<body>  
  
<?php  
if(!isset($\_COOKIE[$cookie\_name])) {  
    echo "Cookie named '" . $cookie\_name . "' is not set!";  
} else {  
    echo "Cookie '" . $cookie\_name . "' is set!<br>";  
    echo "Value is: " . $\_COOKIE[$cookie\_name];  
}  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_cookie3)

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

<?php  
// set the expiration date to one hour ago  
setcookie("user", "", time() - 3600);  
?>  
<html>  
<body>  
  
<?php  
echo "Cookie 'user' is deleted.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_cookie4)

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $\_COOKIE array variable:

Example

<?php  
setcookie("test\_cookie", "test", time() + 3600, '/');  
?>  
<html>  
<body>  
  
<?php  
if(count($\_COOKIE) > 0) {  
    echo "Cookies are enabled.";  
} else {  
    echo "Cookies are disabled.";  
}  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_cookie5)

# PHP Sessions

[❮ Previous](https://www.w3schools.com/php/php_cookies.asp)[Next ❯](https://www.w3schools.com/php/php_filter.asp)

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

## What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

**Tip:** If you need a permanent storage, you may want to store the data in a [database](https://www.w3schools.com/php/php_mysql_intro.asp).

## Start a PHP Session

A session is started with the session\_start() function.

Session variables are set with the PHP global variable: $\_SESSION.

Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables:

### Example

<?php  
// Start the session  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Set session variables  
$\_SESSION["favcolor"] = "green";  
$\_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_session1)

**Note:** The session\_start() function must be the very first thing in your document. Before any HTML tags.

## Get PHP Session Variable Values

Next, we create another page called "demo\_session2.php". From this page, we will access the session information we set on the first page ("demo\_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()).

Also notice that all session variable values are stored in the global $\_SESSION variable:

### Example

<?php  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Echo session variables that were set on previous page  
echo "Favorite color is " . $\_SESSION["favcolor"] . ".<br>";  
echo "Favorite animal is " . $\_SESSION["favanimal"] . ".";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_session2)

Another way to show all the session variable values for a user session is to run the following code:

### Example

<?php  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
print\_r($\_SESSION);  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_session3)

**How does it work? How does it know it's me?**  
  
Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

## Modify a PHP Session Variable

To change a session variable, just overwrite it:

### Example

<?php  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// to change a session variable, just overwrite it  
$\_SESSION["favcolor"] = "yellow";  
print\_r($\_SESSION);  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_session4)

## Destroy a PHP Session

To remove all global session variables and destroy the session, use session\_unset() and session\_destroy():

### Example

<?php  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// remove all session variables  
session\_unset();  
  
// destroy the session  
session\_destroy();  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_session5)

# PHP Include Files

[❮ Previous](https://www.w3schools.com/php/php_date.asp)[Next ❯](https://www.w3schools.com/php/php_file.asp)

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

## PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

**The include and require statements are identical, except upon failure:**

* require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
* include will only produce a warning (E\_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

### Syntax

include '*filename*';  
  
or  
  
require '*filename*';

## PHP include Examples

### Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

<?php  
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";  
?>

To include the footer file in a page, use the include statement:

### Example

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<p>Some text.</p>  
<p>Some more text.</p>  
<?php include 'footer.php';?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include1)

### Example 2

Assume we have a standard menu file called "menu.php":

<?php  
echo '<a href="/default.asp">Home</a> -  
<a href="/html/default.asp">HTML Tutorial</a> -  
<a href="/css/default.asp">CSS Tutorial</a> -  
<a href="/js/default.asp">JavaScript Tutorial</a> -  
<a href="default.asp">PHP Tutorial</a>';  
?>

All pages in the Web site should use this menu file. Here is how it can be done (we are using a <div> element so that the menu easily can be styled with CSS later):

### Example

<html>  
<body>  
  
<div class="menu">  
<?php include 'menu.php';?>  
</div>  
  
<h1>Welcome to my home page!</h1>  
<p>Some text.</p>  
<p>Some more text.</p>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include2)

### Example 3

Assume we have a file called "vars.php", with some variables defined:

<?php  
$color='red';  
$car='BMW';  
?>

Then, if we include the "vars.php" file, the variables can be used in the calling file:

### Example

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php include 'vars.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include3)

## PHP include vs. require

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

### Example

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php include 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include4)

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

### Example

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php require 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include5)

### Example

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php require 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_include5)

Use require when the file is required by the application.

Use include when the file is not required and application should continue when file is not found.

# PHP File Handling

[❮ Previous](https://www.w3schools.com/php/php_includes.asp)[Next ❯](https://www.w3schools.com/php/php_file_open.asp)

File handling is an important part of any web application. You often need to open and process a file for different tasks.

## PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

**Be careful when manipulating files!**

When you are manipulating files you must be very careful.

You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

## PHP readfile() Function

The readfile() function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and XML  
CSS = Cascading Style Sheets  
HTML = Hyper Text Markup Language  
PHP = PHP Hypertext Preprocessor  
SQL = Structured Query Language  
SVG = Scalable Vector Graphics  
XML = EXtensible Markup Language

The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

### Example

<?php  
echo readfile("webdictionary.txt");  
?>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_file_readfile)

The readfile() function is useful if all you want to do is open up a file and read its contents.

# PHP File Open/Read/Close

[❮ Previous](https://www.w3schools.com/php/php_file.asp)[Next ❯](https://www.w3schools.com/php/php_file_create.asp)

In this chapter we will teach you how to open, read, and close a file on the server.

## PHP Open File - fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

We will use the text file, "webdictionary.txt", during the lessons:

AJAX = Asynchronous JavaScript and XML  
CSS = Cascading Style Sheets  
HTML = Hyper Text Markup Language  
PHP = PHP Hypertext Preprocessor  
SQL = Structured Query Language  
SVG = Scalable Vector Graphics  
XML = EXtensible Markup Language

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

### Example

<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
echo fread($myfile,filesize("webdictionary.txt"));  
fclose($myfile);  
?>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_file_fopen)

**Tip:** The fread() and the fclose() functions will be explained below.

The file may be opened in one of the following modes:

|  |  |
| --- | --- |
| **Modes** | **Description** |
| r | **Open a file for read only**. File pointer starts at the beginning of the file |
| w | **Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a | **Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x | **Creates a new file for write only**. Returns FALSE and an error if file already exists |
| r+ | **Open a file for read/write**. File pointer starts at the beginning of the file |
| w+ | **Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a+ | **Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x+ | **Creates a new file for read/write**. Returns FALSE and an error if file already exists |

## PHP Read File - fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

fread($myfile,filesize("webdictionary.txt"));

## PHP Close File - fclose()

The fclose() function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

<?php  
$myfile = fopen("webdictionary.txt", "r");  
// some code to be executed....  
fclose($myfile);  
?>

## PHP Read Single Line - fgets()

The fgets() function is used to read a single line from a file.

The example below outputs the first line of the "webdictionary.txt" file:

### Example

<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
echo fgets($myfile);  
fclose($myfile);  
?>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_file_fgets)

**Note:** After a call to the fgets() function, the file pointer has moved to the next line.

## PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

### Example

<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
// Output one line until end-of-file  
while(!feof($myfile)) {  
  echo fgets($myfile) . "<br>";  
}  
fclose($myfile);  
?>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_file_feof)

## PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file.

The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

### Example

<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
// Output one character until end-of-file  
while(!feof($myfile)) {  
  echo fgetc($myfile);  
}  
fclose($myfile);  
?>

[Run example »](https://tryphp.w3schools.com/showphp.php?filename=demo_file_fgetc)

**Note:** After a call to the fgetc() function, the file pointer moves to the next character.

# PHP MySQL Database

[❮ Previous](https://www.w3schools.com/php/php_oop_static_properties.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_connect.asp)

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP.

## What is MySQL?

* MySQL is a database system used on the web
* MySQL is a database system that runs on a server
* MySQL is ideal for both small and large applications
* MySQL is very fast, reliable, and easy to use
* MySQL uses standard SQL
* MySQL compiles on a number of platforms
* MySQL is free to download and use
* MySQL is developed, distributed, and supported by Oracle Corporation
* MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

* Employees
* Products
* Customers
* Orders

## PHP + MySQL Database System

* PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

## Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

SELECT LastName FROM Employees

The query above selects all the data in the "LastName" column from the "Employees" table.

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

## Download MySQL Database

If you don't have a PHP server with a MySQL Database, you can download it for free here: [http://www.mysql.com](http://www.mysql.com/)

## Facts About MySQL Database

MySQL is the de-facto standard database system for web sites with HUGE volumes of both data and end-users (like Facebook, Twitter, and Wikipedia).

Another great thing about MySQL is that it can be scaled down to support embedded database applications.

Look at <http://www.mysql.com/customers/> for an overview of companies using MySQL.

# PHP Connect to MySQL

[❮ Previous](https://www.w3schools.com/php/php_mysql_intro.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_create.asp)

PHP 5 and later can work with a MySQL database using:

* **MySQLi extension** (the "i" stands for improved)
* **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

## Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

## MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

* MySQLi (object-oriented)
* MySQLi (procedural)
* PDO

## MySQLi Installation

For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.

For installation details, go to: <http://php.net/manual/en/mysqli.installation.php>

## PDO Installation

For installation details, go to: <http://php.net/manual/en/pdo.installation.php>

## Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

### Example (MySQLi Object-Oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password);  
  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
echo "Connected successfully";  
?>

**Note on the object-oriented example above:**

$connect\_error was broken until PHP 5.2.9 and 5.3.0. If you need to ensure compatibility with PHP versions prior to 5.2.9 and 5.3.0, use the following code instead:  
  
// Check connection  
if (mysqli\_connect\_error()) {  
    die("Database connection failed: " . mysqli\_connect\_error());  
}

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password);  
  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
echo "Connected successfully";  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    echo "Connected successfully";  
    }  
catch(PDOException $e)  
    {  
    echo "Connection failed: " . $e->getMessage();  
    }  
?>

**Note:** In the PDO example above we have also **specified a database (myDB)**. PDO require a valid database to connect to. If no database is specified, an exception is thrown.

**Tip:** A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

## Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

### MySQLi Object-Oriented:

$conn->close();

### MySQLi Procedural:

mysqli\_close($conn);

### PDO:

$conn = null;

# PHP Create a MySQL Database

[❮ Previous](https://www.w3schools.com/php/php_mysql_connect.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_create_table.asp)

A database consists of one or more tables.

You will need special CREATE privileges to create or to delete a MySQL database.

## Create a MySQL Database Using MySQLi and PDO

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
// Create database  
$sql = "CREATE DATABASE myDB";  
if ($conn->query($sql) === TRUE) {  
    echo "Database created successfully";  
} else {  
    echo "Error creating database: " . $conn->error;  
}  
  
$conn->close();  
?>

**Note:** When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password).  
  
**Tip:** If you have to use a specific port, add an empty string for the database-name argument, like this: new mysqli("localhost", "username", "password", "", port)

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
// Create database  
$sql = "CREATE DATABASE myDB";  
if (mysqli\_query($conn, $sql)) {  
    echo "Database created successfully";  
} else {  
    echo "Error creating database: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

**Note:** The following PDO example create a database named "myDBPDO":

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
try {  
    $conn = new PDO("mysql:host=$servername", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $sql = "CREATE DATABASE myDBPDO";  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "Database created successfully<br>";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

**Tip:** A great benefit of PDO is that it has exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block. In the catch block above we echo the SQL statement and the generated error message.

# PHP MySQL Create Table

[❮ Previous](https://www.w3schools.com/php/php_mysql_create.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_insert.asp)

A database table has its own unique name and consists of columns and rows.

## Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg\_date":

CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
)

**Notes on the table above:**

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our [Data Types reference](https://www.w3schools.com/sql/sql_datatypes.asp).

After the data type, you can specify other optional attributes for each column:

* NOT NULL - Each row must contain a value for that column, null values are not allowed
* DEFAULT value - Set a default value that is added when no other value is passed
* UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
* AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
* PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO\_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
// sql to create table  
$sql = "CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
)";  
  
if ($conn->query($sql) === TRUE) {  
    echo "Table MyGuests created successfully";  
} else {  
    echo "Error creating table: " . $conn->error;  
}  
  
$conn->close();  
?>

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
// sql to create table  
$sql = "CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
)";  
  
if (mysqli\_query($conn, $sql)) {  
    echo "Table MyGuests created successfully";  
} else {  
    echo "Error creating table: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
    // sql to create table  
    $sql = "CREATE TABLE MyGuests (  
    id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
    firstname VARCHAR(30) NOT NULL,  
    lastname VARCHAR(30) NOT NULL,  
    email VARCHAR(50),  
    reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
    )";  
  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "Table MyGuests created successfully";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

# PHP MySQL Insert Data

[❮ Previous](https://www.w3schools.com/php/php_mysql_create_table.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_insert_lastid.asp)

## Insert Data Into MySQL Using MySQLi and PDO

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

* The SQL query must be quoted in PHP
* String values inside the SQL query must be quoted
* Numeric values must not be quoted
* The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

INSERT INTO table\_name (column1, column2, column3,...)  
VALUES (value1, value2, value3,...)

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

In the previous chapter we created an empty table named "MyGuests" with five columns: "id", "firstname", "lastname", "email" and "reg\_date". Now, let us fill the table with data.

**Note:** If a column is AUTO\_INCREMENT (like the "id" column) or TIMESTAMP with default update of current\_timesamp (like the "reg\_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the "MyGuests" table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if ($conn->query($sql) === TRUE) {  
    echo "New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
  
$conn->close();  
?>

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if (mysqli\_query($conn, $sql)) {  
    echo "New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('John', 'Doe', 'john@example.com')";  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "New record created successfully";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

# PHP MySQL Get Last Inserted ID

[❮ Previous](https://www.w3schools.com/php/php_mysql_insert.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_insert_multiple.asp)

## Get ID of The Last Inserted Record

If we perform an INSERT or UPDATE on a table with an AUTO\_INCREMENT field, we can get the ID of the last inserted/updated record immediately.

In the table "MyGuests", the "id" column is an AUTO\_INCREMENT field:

CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
)

The following examples are equal to the examples from the previous page ([PHP Insert Data Into MySQL](https://www.w3schools.com/php/php_mysql_insert.asp)), except that we have added one single line of code to retrieve the ID of the last inserted record. We also echo the last inserted ID:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if ($conn->query($sql) === TRUE) {  
    $last\_id = $conn->insert\_id;  
    echo "New record created successfully. Last inserted ID is: " . $last\_id;  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
  
$conn->close();  
?>

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if (mysqli\_query($conn, $sql)) {  
    $last\_id = mysqli\_insert\_id($conn);  
    echo "New record created successfully. Last inserted ID is: " . $last\_id;  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('John', 'Doe', 'john@example.com')";  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    $last\_id = $conn->lastInsertId();  
    echo "New record created successfully. Last inserted ID is: " . $last\_id;  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

# PHP MySQL Insert Multiple Records

[❮ Previous](https://www.w3schools.com/php/php_mysql_insert_lastid.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)

## Insert Multiple Records Into MySQL Using MySQLi and PDO

Multiple SQL statements must be executed with the mysqli\_multi\_query() function.

The following examples add three new records to the "MyGuests" table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com');";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Mary', 'Moe', 'mary@example.com');";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Julie', 'Dooley', 'julie@example.com')";  
  
if ($conn->multi\_query($sql) === TRUE) {  
    echo "New records created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . $conn->error;  
}  
  
$conn->close();  
?>

Note that each SQL statement must be separated by a semicolon.

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com');";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Mary', 'Moe', 'mary@example.com');";  
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('Julie', 'Dooley', 'julie@example.com')";  
  
if (mysqli\_multi\_query($conn, $sql)) {  
    echo "New records created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

The PDO way is a little bit different:

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
    // begin the transaction  
    $conn->beginTransaction();  
    // our SQL statements  
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('John', 'Doe', 'john@example.com')");  
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('Mary', 'Moe', 'mary@example.com')");  
    $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('Julie', 'Dooley', 'julie@example.com')");  
  
    // commit the transaction  
    $conn->commit();  
    echo "New records created successfully";  
    }  
catch(PDOException $e)  
    {  
    // roll back the transaction if something failed  
    $conn->rollback();  
    echo "Error: " . $e->getMessage();  
    }  
  
$conn = null;  
?>

# PHP MySQL Prepared Statements

[❮ Previous](https://www.w3schools.com/php/php_mysql_insert_multiple.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_select.asp)

Prepared statements are very useful against SQL injections.

## Prepared Statements and Bound Parameters

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Compared to executing SQL statements directly, prepared statements have three main advantages:

* Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
* Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
* Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

## Prepared Statements in MySQLi

The following example uses prepared statements and bound parameters in MySQLi:

### Example (MySQLi with Prepared Statements)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
// prepare and bind  
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");  
$stmt->bind\_param("sss", $firstname, $lastname, $email);  
  
// set parameters and execute  
$firstname = "John";  
$lastname = "Doe";  
$email = "john@example.com";  
$stmt->execute();  
  
$firstname = "Mary";  
$lastname = "Moe";  
$email = "mary@example.com";  
$stmt->execute();  
  
$firstname = "Julie";  
$lastname = "Dooley";  
$email = "julie@example.com";  
$stmt->execute();  
  
echo "New records created successfully";  
  
$stmt->close();  
$conn->close();  
?>

Code lines to explain from the example above:

"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"

In our SQL, we insert a question mark (?) where we want to substitute in an integer, string, double or blob value.

Then, have a look at the bind\_param() function:

$stmt->bind\_param("sss", $firstname, $lastname, $email);

This function binds the parameters to the SQL query and tells the database what the parameters are. The "sss" argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

The argument may be one of four types:

* i - integer
* d - double
* s - string
* b - BLOB

We must have one of these for each parameter.

By telling mysql what type of data to expect, we minimize the risk of SQL injections.

**Note:** If we want to insert any data from external sources (like user input), it is very important that the data is sanitized and validated.

## Prepared Statements in PDO

The following example uses prepared statements and bound parameters in PDO:

### Example (PDO with Prepared Statements)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
    // prepare sql and bind parameters  
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES (:firstname, :lastname, :email)");  
    $stmt->bindParam(':firstname', $firstname);  
    $stmt->bindParam(':lastname', $lastname);  
    $stmt->bindParam(':email', $email);  
  
    // insert a row  
    $firstname = "John";  
    $lastname = "Doe";  
    $email = "john@example.com";  
    $stmt->execute();  
  
    // insert another row  
    $firstname = "Mary";  
    $lastname = "Moe";  
    $email = "mary@example.com";  
    $stmt->execute();  
  
    // insert another row  
    $firstname = "Julie";  
    $lastname = "Dooley";  
    $email = "julie@example.com";  
    $stmt->execute();  
  
    echo "New records created successfully";  
    }  
catch(PDOException $e)  
    {  
    echo "Error: " . $e->getMessage();  
    }  
$conn = null;  
?>

# PHP MySQL Select Data

[❮ Previous](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_select_where.asp)

## Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

SELECT column\_name(s) FROM table\_name

or we can use the \* character to select ALL columns from a table:

SELECT \* FROM table\_name

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

## Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_oo)

Code lines to explain from the example above:

First, we set up an SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num\_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch\_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = mysqli\_query($conn, $sql);  
  
if (mysqli\_num\_rows($result) > 0) {  
    // output data of each row  
    while($row = mysqli\_fetch\_assoc($result)) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
  
mysqli\_close($conn);  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_proc)

You can also put the result in an HTML table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    echo "<table><tr><th>ID</th><th>Name</th></tr>";  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".$row["lastname"]."</td></tr>";  
    }  
    echo "</table>";  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_oo_table)

## Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table and displays it in an HTML table:

### Example (PDO)

<?php  
echo "<table style='border: solid 1px black;'>";  
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";  
  
class TableRows extends RecursiveIteratorIterator {  
    function \_\_construct($it) {  
        parent::\_\_construct($it, self::LEAVES\_ONLY);  
    }  
  
    function current() {  
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";  
    }  
  
    function beginChildren() {  
        echo "<tr>";  
    }  
  
    function endChildren() {  
        echo "</tr>" . "\n";  
    }  
}  
  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests");  
    $stmt->execute();  
  
    // set the resulting array to associative  
    $result = $stmt->setFetchMode(PDO::FETCH\_ASSOC);  
    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {  
        echo $v;  
    }  
}  
catch(PDOException $e) {  
    echo "Error: " . $e->getMessage();  
}  
$conn = null;  
echo "</table>";  
?>

# PHP MySQL Use The WHERE Clause

[❮ Previous](https://www.w3schools.com/php/php_mysql_select.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_select_orderby.asp)

## Select and Filter Data From a MySQL Database

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

SELECT column\_name(s) FROM table\_name WHERE column\_name operator value

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

## Select and Filter Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it on the page:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_where_oo)

Code lines to explain from the example above:

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe". The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num\_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch\_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";  
$result = mysqli\_query($conn, $sql);  
  
if (mysqli\_num\_rows($result) > 0) {  
    // output data of each row  
    while($row = mysqli\_fetch\_assoc($result)) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
  
mysqli\_close($conn);  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_where_proc)

You can also put the result in an HTML table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    echo "<table><tr><th>ID</th><th>Name</th></tr>";  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".$row["lastname"]."</td></tr>";  
    }  
    echo "</table>";  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_where_oo_table)

## Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

It selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it in an HTML table:

### Example (PDO)

<?php  
echo "<table style='border: solid 1px black;'>";  
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";  
  
class TableRows extends RecursiveIteratorIterator {  
    function \_\_construct($it) {  
        parent::\_\_construct($it, self::LEAVES\_ONLY);  
    }  
  
    function current() {  
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";  
    }  
  
    function beginChildren() {  
        echo "<tr>";  
    }  
  
    function endChildren() {  
        echo "</tr>" . "\n";  
    }  
}  
  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'");  
    $stmt->execute();  
  
    // set the resulting array to associative  
    $result = $stmt->setFetchMode(PDO::FETCH\_ASSOC);  
    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {  
        echo $v;  
    }  
}  
catch(PDOException $e) {  
    echo "Error: " . $e->getMessage();  
}  
$conn = null;  
echo "</table>";  
?>

# PHP MySQL Use The ORDER BY Clause

[❮ Previous](https://www.w3schools.com/php/php_mysql_select_where.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_delete.asp)

## Select and Order Data From a MySQL Database

The ORDER BY clause is used to sort the result-set in ascending or descending order.

The ORDER BY clause sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

SELECT column\_name(s) FROM table\_name ORDER BY column\_name(s) ASC|DESC

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

## Select and Order Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_orderby_oo)

Code lines to explain from the example above:

First, we set up the SQL query that selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, the function num\_rows() checks if there are more than zero rows returned.

If there are more than zero rows returned, the function fetch\_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

The following example shows the same as the example above, in the MySQLi procedural way:

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";  
$result = mysqli\_query($conn, $sql);  
  
if (mysqli\_num\_rows($result) > 0) {  
    // output data of each row  
    while($row = mysqli\_fetch\_assoc($result)) {  
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
    }  
} else {  
    echo "0 results";  
}  
  
mysqli\_close($conn);  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_orderby_proc)

You can also put the result in an HTML table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
    echo "<table><tr><th>ID</th><th>Name</th></tr>";  
    // output data of each row  
    while($row = $result->fetch\_assoc()) {  
        echo "<tr><td>".$row["id"]."</td><td>".$row["firstname"]." ".$row["lastname"]."</td></tr>";  
    }  
    echo "</table>";  
} else {  
    echo "0 results";  
}  
$conn->close();  
?>

[Run example »](https://tryphp.w3schools.com/showphpfile.php?filename=demo_db_select_orderby_oo_table)

## Select Data With PDO (+ Prepared Statements)

The following example uses prepared statements.

Here we select the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column, and it will be displayed in an HTML table:

### Example (PDO)

<?php  
echo "<table style='border: solid 1px black;'>";  
echo "<tr><th>Id</th><th>Firstname</th><th>Lastname</th></tr>";  
  
class TableRows extends RecursiveIteratorIterator {  
    function \_\_construct($it) {  
        parent::\_\_construct($it, self::LEAVES\_ONLY);  
    }  
  
    function current() {  
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";  
    }  
  
    function beginChildren() {  
        echo "<tr>";  
    }  
  
    function endChildren() {  
        echo "</tr>" . "\n";  
    }  
}  
  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $stmt = $conn->prepare("SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname");  
    $stmt->execute();  
  
    // set the resulting array to associative  
    $result = $stmt->setFetchMode(PDO::FETCH\_ASSOC);  
    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {  
        echo $v;  
    }  
}  
catch(PDOException $e) {  
    echo "Error: " . $e->getMessage();  
}  
$conn = null;  
echo "</table>";  
?>

# PHP MySQL Delete Data

[❮ Previous](https://www.w3schools.com/php/php_mysql_select_orderby.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_update.asp)

## Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

DELETE FROM table\_name  
WHERE some\_column = some\_value

**Notice the WHERE clause in the DELETE syntax:** The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

Let's look at the "MyGuests" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **firstname** | **lastname** | **email** | **reg\_date** |
| 1 | John | Doe | john@example.com | 2014-10-22 14:26:15 |
| 2 | Mary | Moe | mary@example.com | 2014-10-23 10:22:30 |
| 3 | Julie | Dooley | julie@example.com | 2014-10-26 10:48:23 |

The following examples delete the record with id=3 in the "MyGuests" table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
// sql to delete a record  
$sql = "DELETE FROM MyGuests WHERE id=3";  
  
if ($conn->query($sql) === TRUE) {  
    echo "Record deleted successfully";  
} else {  
    echo "Error deleting record: " . $conn->error;  
}  
  
$conn->close();  
?>

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
// sql to delete a record  
$sql = "DELETE FROM MyGuests WHERE id=3";  
  
if (mysqli\_query($conn, $sql)) {  
    echo "Record deleted successfully";  
} else {  
    echo "Error deleting record: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
    // sql to delete a record  
    $sql = "DELETE FROM MyGuests WHERE id=3";  
  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "Record deleted successfully";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

# PHP MySQL Update Data

[❮ Previous](https://www.w3schools.com/php/php_mysql_delete.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_select_limit.asp)

## Update Data In a MySQL Table Using MySQLi and PDO

The UPDATE statement is used to update existing records in a table:

UPDATE table\_name  
SET column1=value, column2=value2,...  
WHERE some\_column=some\_value

**Notice the WHERE clause in the UPDATE syntax:** The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

To learn more about SQL, please visit our [SQL tutorial](https://www.w3schools.com/sql/default.asp).

Let's look at the "MyGuests" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **firstname** | **lastname** | **email** | **reg\_date** |
| 1 | John | Doe | john@example.com | 2014-10-22 14:26:15 |
| 2 | Mary | Moe | mary@example.com | 2014-10-23 10:22:30 |

The following examples update the record with id=2 in the "MyGuests" table:

### Example (MySQLi Object-oriented)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
    die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
if ($conn->query($sql) === TRUE) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . $conn->error;  
}  
  
$conn->close();  
?>

### Example (MySQLi Procedural)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
if (mysqli\_query($conn, $sql)) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

### Example (PDO)

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
    $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
    // Prepare statement  
    $stmt = $conn->prepare($sql);  
  
    // execute the query  
    $stmt->execute();  
  
    // echo a message to say the UPDATE succeeded  
    echo $stmt->rowCount() . " records UPDATED successfully";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }  
  
$conn = null;  
?>

After the record is updated, the table will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **firstname** | **lastname** | **email** | **reg\_date** |
| 1 | John | Doe | john@example.com | 2014-10-22 14:26:15 |
| 2 | Mary | Doe | mary@example.com | 2014-10-23 10:22:30 |

[❮ Previous](https://www.w3schools.com/php/php_mysql_delete.asp)[Next ❯](https://www.w3schools.com/php/php_mysql_select_limit.asp)

# PHP MySQL Limit Data Selections

[❮ Previous](https://www.w3schools.com/php/php_mysql_update.asp)[Next ❯](https://www.w3schools.com/php/php_xml_parsers.asp)

## Limit Data Selections From a MySQL Database

MySQL provides a LIMIT clause that is used to specify the number of records to return.

The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance.

Assume we wish to select all records from 1 - 30 (inclusive) from a table called "Orders". The SQL query would then look like this:

$sql = "SELECT \* FROM Orders LIMIT 30";

When the SQL query above is run, it will return the first 30 records.

What if we want to select records 16 - 25 (inclusive)?

Mysql also provides a way to handle this: by using OFFSET.

The SQL query below says "return only 10 records, start on record 16 (OFFSET 15)":

$sql = "SELECT \* FROM Orders LIMIT 10 OFFSET 15";

You could also use a shorter syntax to achieve the same result:

$sql = "SELECT \* FROM Orders LIMIT 15, 10";

Notice that the numbers are reversed when you use a comma.

What is XML?

XML is a mark-up language to share the data across the web, XML is for both human read-able and machine read-able. Example of share-able xmls are RSS Feeds. XML parsers are useful to read and update the data by using web browsers.

Types of XML

* Tree based
* Event based

XML Parse Extensions

XML parse Extensions are works based on libxml. The following xml parsers are available in the php core.

* Simple XML parser
* DO XML parser
* XML parser
* XML Reader

Simple XML parser

The Simple XML parser also called as tree based XML parser and it will parse the simple XML file. Simple XML parse will call simplexml\_load\_file() method to get access to the xml from specific path.

DOM parser

DOM Parser also called as a complex node parser, Which is used to parse highly complex XML file. It is used as interface to modify the XML file. DOM parser has encoded with UTF-8 character encoding.

XML parse

XML parsing is based on SAX parse. It is more faster the all above parsers. It will create the XML file and parse the XML. XML parser has encoded by ISO-8859-1, US-ASCII and UTF-8 character encoding.

XML Reader

XML Reader parse also called as Pull XML parse. It is used to read the XML file in a faster way. It works with high complex XML document with XML Validation.

## The simple XML parser

The simple XML parser is used to parse Name, attributes and textual content.

The simple XML functions are shown below −

## simplexml\_load\_file()

This function accepts file path as a first parameter and it is mandatory.

simplexml\_load\_file(($fileName,$class,$options,$ns,$is\_prefix)

## simplexml\_load\_string()

This function accepts the string instead of file reference.

simplexml\_load\_string($XMLData,$class,$options,$ns,$is\_prefix)

## simplexml\_import\_dom()

This function accepts DOM formatted XML content and it converts into simple XML.

simplexml\_load\_string($DOMNode,$class)

The following example shows, How to parse a xml data file.

<?php

$data = "<?xml version = '1.0' encoding = 'UTF-8'?>

<note>

<Course>Android</Course>

<Subject>Android</Subject>

<Company>TutorialsPoint</Company>

<Price>$10</Price>

</note>";

$xml = simplexml\_load\_string($data) or die("Error: Cannot create object");

?>

<html>

<head>

<body>

<?php

print\_r($xml);

?>

</body>

</head>

</html>

It will produce the following result −

SimpleXMLElement Object ( [Course] => Android [Subject] => Android [Company] => TutorialsPoint [Price] => $10 )

We can also call a xml data file as shown below and it produces the same result as shown above −

<?php

$xml = simplexml\_load\_file("data") or die("Error: Cannot create object");

print\_r($xml);

?>

## What is XML?

XML is the acronym for Extensible Markup Language.

XML is used to structure, store and transport data from one system to another.

XML is similar to HTML.

It uses opening and closing tags.

Unlike HTML, XML allows users to define their own tags.

In this tutorial, you will learn-

* [What is DOM?](https://www.guru99.com/php-and-xml.html#2)
* [XML Parsers](https://www.guru99.com/php-and-xml.html#3)
* [Why use XML?](https://www.guru99.com/php-and-xml.html#4)
* [XML Document Example](https://www.guru99.com/php-and-xml.html#5)
* [How to Read XML using PHP](https://www.guru99.com/php-and-xml.html#6)
* [How to Create an XML document using PHP](https://www.guru99.com/php-and-xml.html#8)

## What is DOM?

DOM is the acronym for Document Object Model.

It’s a cross platform and language neutral standard that defines how to access and manipulate data in;

* HTML
* XHTML
* XML

DOM XML is used to access and manipulate XML documents. It views the XML document as a tree-structure.

## XML Parsers

An XML parser is a program that translates the XML document into an XML Document Object Model (DOM) Object.

The XML DOM Object can then be manipulated using JavaScript, Python, and PHP etc.

The keyword CDATA which is the acronym for (Unparsed) Character Data is used to ignore special characters such as “<,>” when parsing an XML document.

## Why use XML?

* Web services such as SOAP and REST use XML format to exchange information. Learning what XML is and how it works will get you competitive advantage as a developer since modern applications make heavy use of web services.
* XML documents can be used to store configuration settings of an application
* It allows you to create your own custom tags which make it more flexible.

## XML Document example

Let’s suppose that you are developing an application that gets data from a web service in XML format.

Below is the sample of how the XML document looks like.

<?xml version="1.0" encoding="utf-8"?>

<employees status = "ok">

        <record man\_no = "101">

            <name>Joe Paul</name>

            <position>CEO</position>

        </record>

        <record man\_no = "102">

            <name>Tasha Smith</name>

            <position>Finance Manager</position>

        </record>

</employees>

  HERE,

* “<?xml version="1.0" encoding="utf-8"?>” specifies the xml version to be used and encoding
* “<employees status = "ok">” is the root element.
* “<record…>…</record>” are the child elements of administration and sales respectively.

## How to Read XML using PHP

Let’s now write the code that will read the employees XML document and display the results in a web browser. *Index.php*

<?php

$xml = simplexml\_load\_file('employees.xml');

echo '<h2>Employees Listing</h2>';

$list = $xml->record;

for ($i = 0; $i < count($list); $i++) {

    echo '<b>Man no:</b> ' . $list[$i]->attributes()->man\_no . '<br>';

    echo 'Name: ' . $list[$i]->name . '<br>';

    echo 'Position: ' . $list[$i]->position . '<br><br>';

}

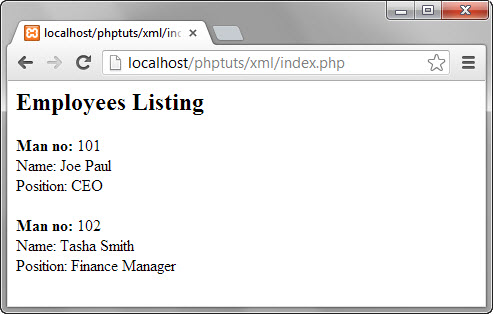
?>

  HERE,

* “$xml = simplexml\_load\_file('employees.xml');” uses the simplexml\_load\_file function to load the file name employees.xml and assign the contents to the array variable $xml.
* “$list = $xml->record;” gets the contents of the record node.
* “for ($i = 0; $i < count(…)…” is the for loop that reads the numeric array and outputs the results
* “$list[$i]->attributes()->man\_no;” reads the man\_no attribute of the element
* “$list[$i]->name;” reads the value of the name child element
* “$list[$i]->position;” reads the value of the position child element

## Testing our application

Assuming you saved the file index.php in phptus/xml folder, browse to the URL**http://localhost/phptuts/xml/index.php**

[](https://www.guru99.com/images/2013/04/employees_listing.jpg)

## How to Create an XML document using PHP

We will now look at how to create an XML document using PHP.

We will use the example above in the DOM tree diagram.

The following code uses the PHP built in class DOMDocument to create an XML document.

<?php

$dom = new DOMDocument();

$dom->encoding = 'utf-8';

$dom->xmlVersion = '1.0';

$dom->formatOutput = true;

$xml\_file\_name = 'movies\_list.xml';

$root = $dom->createElement('Movies');

$movie\_node = $dom->createElement('movie');

$attr\_movie\_id = new DOMAttr('movie\_id', '5467');

$movie\_node->setAttributeNode($attr\_movie\_id);

$child\_node\_title = $dom->createElement('Title', 'The Campaign');

$movie\_node->appendChild($child\_node\_title);

$child\_node\_year = $dom->createElement('Year', 2012);

$movie\_node->appendChild($child\_node\_year);

$child\_node\_genre = $dom->createElement('Genre', 'The Campaign');

$movie\_node->appendChild($child\_node\_genre);

$child\_node\_ratings = $dom->createElement('Ratings', 6.2);

$movie\_node->appendChild($child\_node\_ratings);

$root->appendChild($movie\_node);

$dom->appendChild($root);

$dom->save($xml\_file\_name);

echo "$xml\_file\_name has been successfully created";

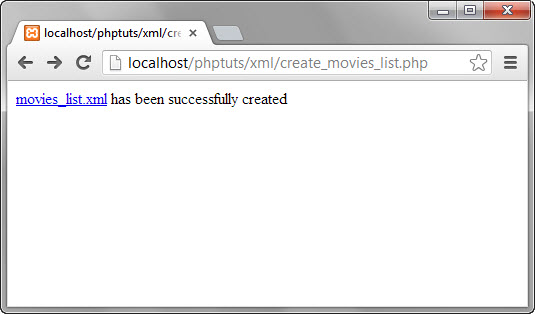
?>

  HERE,

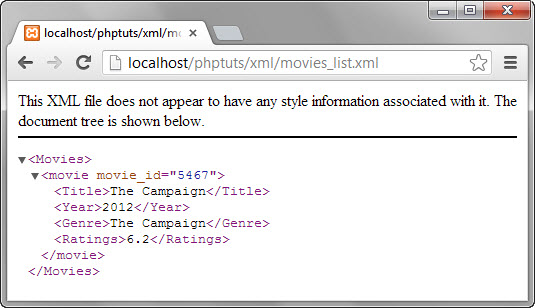
* “$dom = new DOMDocument();” creates an instance of DOMDocument class.
* “$dom->encoding = 'utf-8';” sets the document encoding to utf-8
* “$dom->xmlVersion = '1.0';” specifies the version number 1.0
* “$dom->formatOutput = true;” ensures that the output is well formatted
* “$root = $dom->createElement('Movies');” creates the root node named Movies
* “$attr\_movie\_id = new DOMAttr('movie\_id', '5467');” defines the movie id attribute of Movies node
* “$child\_node\_element\_name = $dom->createElement('ElementName', 'ElementValue')” creates the child node of Movies node. ElementName specifies the name of the element e.g. Title. ElementValue sets the child node value e.g. The Campaign.
* “$root->appendChild($movie\_node);” appends the movie\_node elements to the root node Movies
* “$dom->appendChild($root);” appends the root node to the XML document.
* “$dom->save($xml\_file\_name);” saves the XML file in the root directory of the web server.
* “echo '<a href= "'.$xml\_file\_name.'">' . $xml\_file\_name . '</a> has been successfully created';” creates the link to the XML file.

## Testing our application

Assuming you saved the file create\_movies\_list in phptuts/xml folder, browse to the URL **http://localhost/phptuts/xml/create\_movies\_list.php**

[](https://www.guru99.com/images/2013/04/create_xml.jpg)

Click on movies\_list\_xml link

[](https://www.guru99.com/images/2013/04/view_xml.jpg)

## Summary

* XML is the acronym for Extensible Markup Language
* XML can be used for exchanging information between systems or store configuration settings of an application etc.
* DOM is the acronym for Document Object Model. XML DOM views the XML document as a tree-structure
* An XML Parser is a program that translates XML an XML document into a DOM tree-structure like document.
* CDATA is used to ignore special characters when parsing XML documents.
* PHP uses the simplexml\_load\_file to read XML documents and return the results as a numeric array
* PHP DOMDocument class to create XML files.