**What is PHP? Write your first PHP Program**

**What is PHP?**

PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

PHP scripts can only be interpreted on a server that has PHP installed.

The client computers accessing the PHP scripts require a web browser only.

A PHP file contains PHP tags and ends with the extension ".php".

In this tutorial, you will learn-

* [What is a Scripting Language?](https://www.guru99.com/what-is-php-first-php-program.html#2)
* [Scripting VS Programming Language](https://www.guru99.com/what-is-php-first-php-program.html#1)
* [What does PHP stand for?](https://www.guru99.com/what-is-php-first-php-program.html#9)
* [Php Syntax](https://www.guru99.com/what-is-php-first-php-program.html#10)
* [Why use PHP?](https://www.guru99.com/what-is-php-first-php-program.html#3)
* [What is PHP used for & Market share](https://www.guru99.com/what-is-php-first-php-program.html#4)
* [PHP vs ASP.NET VS JSP VS CFML](https://www.guru99.com/what-is-php-first-php-program.html#5)
* [PHP File Extensions](https://www.guru99.com/what-is-php-first-php-program.html#6)
* [PHP Hello world](https://www.guru99.com/what-is-php-first-php-program.html#7)

**What is a Scripting Language?**

A script is a set of programming instructions that is interpreted at runtime.

A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments.

The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.

Server side scripts are interpreted on the server while client side scripts are interpreted by the client application.

PHP is a server side script that is interpreted on the server while[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.

**Programming Language Vs Scripting Language**

|  |  |
| --- | --- |
| **Programming language** | **Scripting language** |
| Has all the features needed to develop complete applications. | Mostly used for routine tasks |
| The code has to be compiled before it can be executed | The code is usually executed without compiling |
| Does not need to be embedded into other languages | Is usually embedded into other software environments. |

**What does PHP stand for?**

PHP means - **Personal Home Page**, but it now stands for the recursive backronym PHP: Hypertext Preprocessor.

PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks.

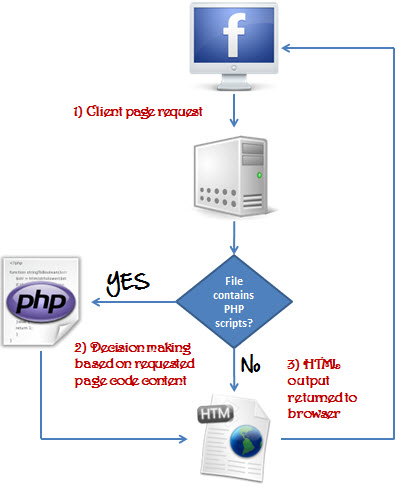
**Php Syntax**

[What is PHP? Write your first PHP Program](https://www.guru99.com/images/2013/04/php_code.png)

A PHP file can also contain tags such as HTML and client side scripts such as JavaScript.

* **HTML** **is an added advantage** when learning PHP Language. You can even learn PHP without knowing HTML but it’s recommended you at least know the basics of HTML.
* **Database management systems** DBMS for database powered applications.
* For more advanced topics such as interactive applications and web services, you will need **JavaScript and XML**.

The flowchart diagram shown below illustrates the basic architecture of a PHP web application and how the server handles the requests.

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

**Why use PHP?**

You have obviously heard of a number of programming languages out there; you may be wondering why we would want to use PHP as our poison for the web programming. Below are some of the compelling reasons.

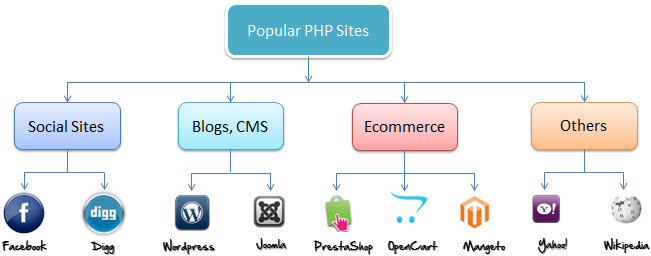
* PHP is **open source and free.**
* Short learning curve compared to other languages such as JSP, ASP etc.
* Large community document
* Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.
* PHP is regular updated to keep abreast with the latest technology trends.
* Other benefit that you get with PHP is that it’s a **server side scripting language**; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
* PHP has **in built support for working hand in hand with MySQL**; this doesn’t mean you can’t use PHP with other database management systems. You can still use PHP with
  + Postgres
  + Oracle
  + MS[SQL](https://www.guru99.com/sql.html)Server
  + ODBC etc.
* PHP is **cross platform;** this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.

**What is PHP used for & Market share**

In terms of market share, there are over 20 million websites and application on the internet developed using PHP scripting language.

This may be attributed to the points raised above;

The diagram below shows some of the popular sites that use PHP

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

**PHP vs Asp.Net VS JSP VS CFML**

[ASP](https://www.guru99.com/asp-net-tutorial.html) – Active Server Pages, [JSP](https://www.guru99.com/jsp-tutorial.html)– Java Server Pages, CFML – Cold Fusion Markup language The table below compares the various server side scripting languages with PHP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FEATURE** | **PHP** | **ASP** | **JSP** | **CFML** |
| Learning curve | short | Longer than PHP | Longer than PHP | Longer than PHP |
| Web hosting | Supported by almost all hosting servers | Needs dedicated server | Fairly supported | Needs dedicated server |
| Open source | Yes | No | Yes | Both commercial and open source |
| Web services support | Built in | Uses the .NET framework | Uses add on libraries | Built in |
| Integration with HTML | Easy | Fairly complex | Fairly complex | Easy |
| MySQL support | Native | Needs third party drivers | Needs third party drivers | Current version has native support. Older versions use ODBC |
| Easily extended by other languages | Yes | No | Extended using Java classes and libraries. | Yes |

**PHP File Extensions**

*File extension and Tags* In order for the **server** to **identify** our **PHP** **files** and **scripts**, we must **save** the **file** with the **“.php” extension**. Older PHP file extensions include

* .phtml
* .php3
* .php4
* .php5
* .phps

PHP was designed to work with HTML, and as such, it can be embedded into the HTML code.

[What is PHP? Write your first PHP Program](https://www.guru99.com/images/2013/04/php_html_code.jpg)

You can create PHP files without any html tags and that is called Pure PHP file .

The server interprets the PHP code and outputs the results as HTML code to the web browsers.

In order for the server to identify the PHP code from the HTML code, we must always enclose the PHP code in PHP tags.

A PHP tag starts with the less than symbol followed by the question mark and then the words “php”.

PHP is a case sensitive language, “VAR” is not the same as “var”.

The PHP tags themselves are not case-sensitive, but it is strongly recommended that we use lower case letter. The code below   illustrates the above point.

<?php … ?>

We will be referring to the PHP lines of code as statements. PHP statements end with a semi colon (;). If you only have one statement, you can omit the semi colon. If you have more than one statement, then you must end each line with a semi colon. For the sake of consistency, it is recommended that you always end your statement(s) with a semi colon.  PHP scripts are executed on the server. The output is returned in form of HTML.

**PHP Hello world**

The program shown below is a basic PHP application that outputs the words “Hello World!” When viewed in a web browser.

<?php

echo "Hello world";

?>

**Output:**

Hello world

**Summary**

* PHP stands for Hypertext pre-processor
* PHP is a server side scripting language. This means that it is executed on the server. The client applications do not need to have PHP installed.
* PHP files are saved with the ".php" file extension, and the PHP development code is enclosed in tags.
* PHP is open source and cross platform

**How to Download & Install XAMPP on Windows: PHP Tutorial**

**What is XAMPP?**

**XAMPP** is an open-source, cross-platform web server that consists of a web server, MySQL database engine, and PHP and[Perl](https://www.guru99.com/perl-tutorials.html)programming packages. It is compiled and maintained by Apache. It allows users to create WordPress websites online using a local web server on their computer. It supports Windows, Linux, and Mac.

It is compiled and maintained by apache. The acronym XAMPP stands for;

* X – [cross platform operating systems] meaning it can run on any OS  Mac OX , Windows ,[Linux](https://www.guru99.com/unix-linux-tutorial.html)etc.
* A –[Apache](https://www.guru99.com/apache.html)- this is the web server software.
* M – MySQL - Database.
* P – [PHP](https://www.guru99.com/php-tutorials.html)
* P – Perl – scripting language

**Why use XAMPP?**

XAMPP provides an easy-to-use control panel to manage Apache, MySQL, and other programs without using commands. To use PHP, we need to install Apache and MySQL. It’s not easy to install Apache and configure it as it needs to be set up and integrated with PHP and Perl, among other things. XAMPP deals with all the complexity to set up and integrate Apache with PHP and Perl.

Unlike[Java](https://www.guru99.com/java-tutorial.html)that runs with the Java SDK only, PHP requires a web-server to work.

In this XAMPP Tutorial, you will learn-

* [What is XAMPP?](https://www.guru99.com/xampp-netbeans.html#1)
* [Why use XAMPP?](https://www.guru99.com/xampp-netbeans.html#2)
* [How to Download and Install XAMPP](https://www.guru99.com/xampp-netbeans.html#3)
* [Basic XAMPP Web Server Configuration](https://www.guru99.com/xampp-netbeans.html#4)
* [XAMPP Control Panel](https://www.guru99.com/xampp-netbeans.html#5)
* [Configure XAMPP](https://www.guru99.com/xampp-netbeans.html#6)
* [What is the best PHP IDE?](https://www.guru99.com/xampp-netbeans.html#7)
* [Introduction to Netbeans IDE](https://www.guru99.com/xampp-netbeans.html#8)
* [Creating a new PHP project using the Netbeans IDE](https://www.guru99.com/xampp-netbeans.html#9)
* [Running your first PHP Example](https://www.guru99.com/xampp-netbeans.html#10)

**How to Install XAMPP**

We look into step by step process to install XAMPP for Windows. For Other Operating Systems, XAMPP installation steps are similar.

**Step 1) Download XAMPP**

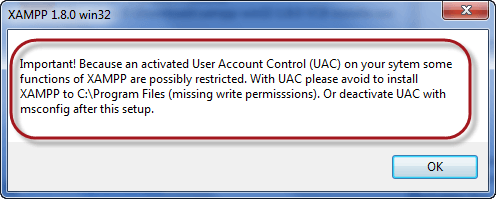
Click here to XAMPP download for Windows: <http://www.apachefriends.org/en/xampp-windows.html>

**Step 2) Start Installation**

XAMPP Installation is just like installing any other windows program. There are however, a few things that we must note.

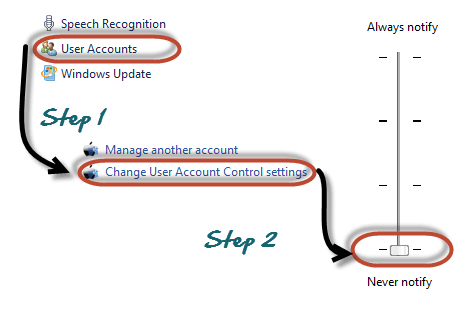
**Step 3) Run the Setup**

After you have downloaded XAMPP, run the setup. The warning message dialog window shown below appears.

[](https://www.guru99.com/images/2013/04/xampp_uac.png)

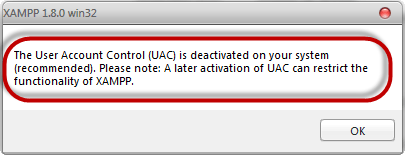
**Step 4) Change User Control Settings**

If you are using Windows Vista or Windows 7, make sure that you deactivate the User Account Control feature. To do this, Select Control Panel >  User Accounts >  Change User Access Control settings. The diagram below illustrates the main steps.

[](https://www.guru99.com/images/2013/04/deactivate_uac.png)

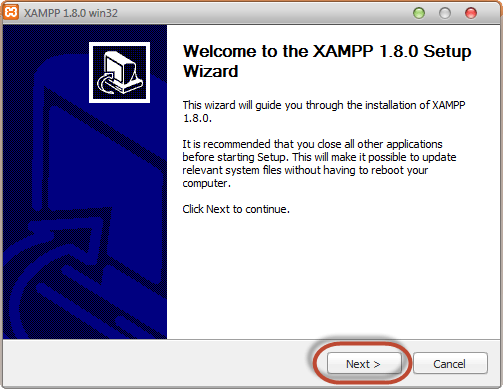
**Step 5) Save the settings**

* After you have deactivated the User Account Control, click on OK button on the warning message box.
* This time you get following message

[](https://www.guru99.com/images/2013/04/uac_deactivated.png)

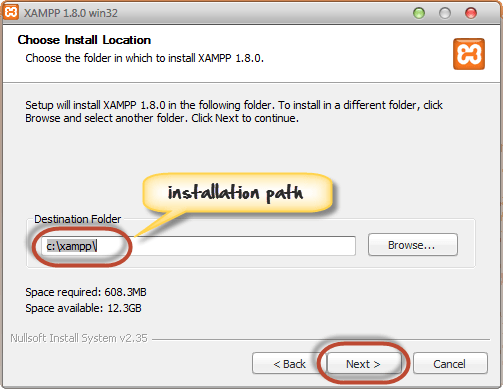
**Step 6) Click Next**

In the succeeding screen, click next

[](https://www.guru99.com/images/2013/04/xampp_install_step_1.png)

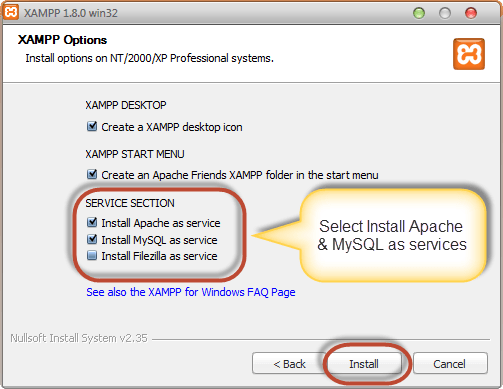
**Step 7) Choose the Insatllation path**

In the next screen, Change the installation path if required. Click Next

[](https://www.guru99.com/images/2013/04/xampp_install_step_2.png)

**Step 8) Check the necessary services**

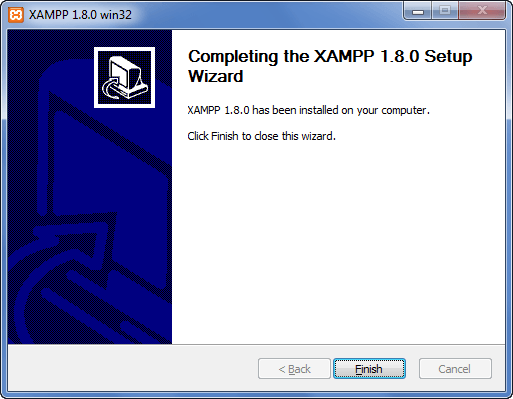
In the next screen select Apache and MySQL. You may optionally select FileZilla (FTP Client) if needed. Click Install

[](https://www.guru99.com/images/2013/04/xampp_install_step_3.png)

***Note****a service is a long-running program in windows that does not require user intervention. Services can be set to run automatically whenever the windows operating system is started.****For you to use Apache and MySQL, they are supposed to be running in the background****.****Installing them as services runs both Apache and MySQL automatically in the background whenever you power up your computer****. If you have not installed Apache and MySQL as services, then you have to manually start them every time that you want to use them. You will have to do this from the XAMPP control panel.PHP and*

**Step 9) Finish the installation**

On successful completion of installation, you will see following window

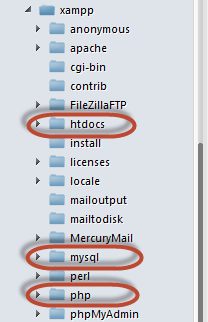
[](https://www.guru99.com/images/2013/04/xampp_install_step_4.png)

* Click on Finish button

Before we test our XAMPP installation, let’s first look at the basic directories that we will be working with.

**Basic XAMPP Web Server Configuration**

This XAMPP Tutorial assumes that you have **installed XAMPP on drive C in Windows using the steps mentioned above**. The following is a list of the basic directories that you are supposed to be aware of.

[](https://www.guru99.com/images/2013/04/xampp_directories.png)

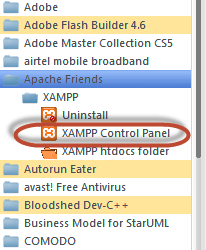
* **htdocs**; this is the web root directory. All of our PHP codes will be placed in this directory.
* **mysql** – this directory contains all the information related to MySQL database engine, by default it runs on port 3306.
* **php** – this directory contains PHP installation files. It contains an important file named php.ini. This directory is used to configure how PHP behaves on your server.

**By default**, the Apache web server runs on **port 80**. If port 80 is taken by another web server, you can use a different port number. For this tutorial we will assume we are using port 80. Note,If you use SKYPE , it uses the same port. Close Skype if you want to use XAMPP for PHP on port 80

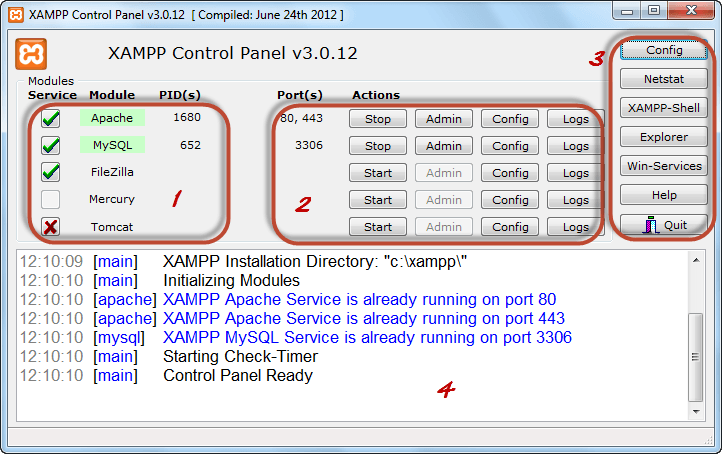
**XAMPP Control Panel**

The control panel is used to manage programs installed via XAMPP. To open the XAMPP Server control panel,

* Click on start menu
* Explore the programs directory and locate Apace Friends then XAMPP as shown in the diagram below

[](https://www.guru99.com/images/2013/04/xampp_control_panel.png)

* The diagram below shows the control panel.

[](https://www.guru99.com/images/2013/04/xampp_control_panel2.png)

**1)** This section lists the installed services, modules and the process IDs PID(s). A green tick means the module has been installed as a service. The red mark means it has not been installed as a service. To install a service, click on the red mark. If the button shows a green tick and you click on it, the control panel will ask you if you want to uninstall the system.

**2)**This section shows Port(s) associated with the modules. The actions section is for;

1. starting and stopping modules
2. Open the administrative windows for Apache and MySQL
3. Open configuration files for Apache, MySQL etc. to make changes
4. View log files for the modules

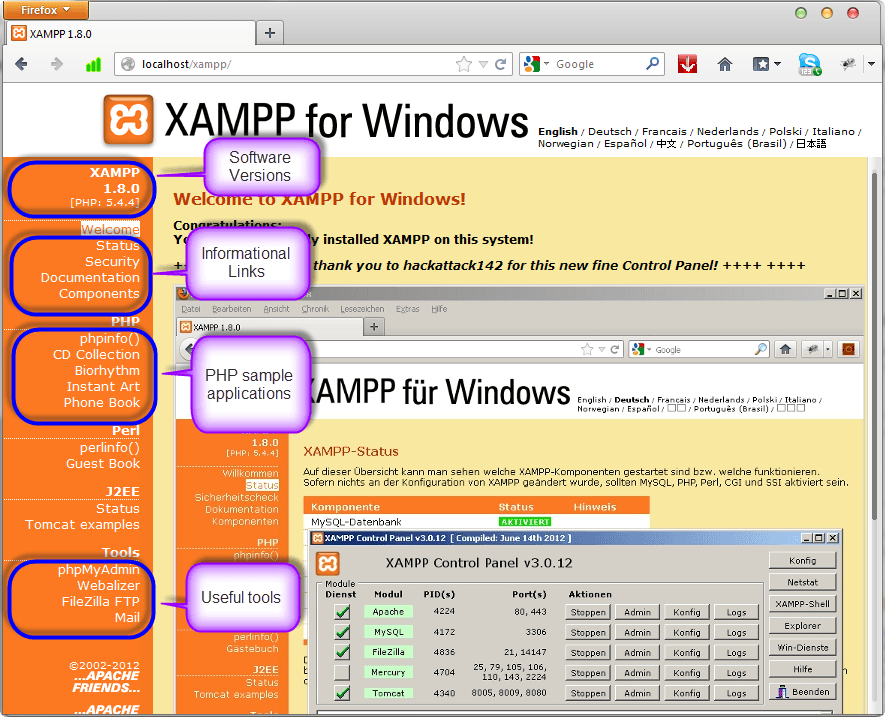
**3)** This section contains useful utilities such as Netsat, windows services short cuts etc.

**4)** This section displays status information on the modules. The control panel can be used to;

* Install and uninstall services such as Apache, MySQL etc. that are installed via XAMPP
* Start and stop services.
* Open configure files etc.

**Configure XAMPP**

Let’s now look at the basic configurations required before we start using our XAMPP installation for developing PHP powered web sites. Type the URL **http://localhost/xampp/** in your favorite browser. For this tutorial, we will be using Firefox as our web browser.

[](https://www.guru99.com/images/2013/04/configure_xampp.png)

If you are able to see the above screen then you have installed XAMPP successfully. The panel on the left hand side contains links to useful information such as;

* The version of PHP installed
* Security settings of XAMPP
* Access to utilities such as phpMyAdmin etc.

The PHP version shipped with XAMPP 1.8.0 is PHP 5.4.4

**What is the best PHP IDE?**

A PHP IDE is a program that allows you to easily write PHP codes. PHP IDEs are often equipped with syntax highlighting features and in some cases autocomplete features too. This means that if you write a PHP keyword that is known by the PHP interpreter, then the keyword will be highlighted a different color from the one used for regular statements. The autocomplete features automatically pops up known PHP keywords as you type them. Notepad can also be used to write and editor PHP codes. The disadvantage of using an editor such as Notepad is that debugging the scripts becomes difficult because it is not easy to spot errors such as misspelt keywords, unclosed braces etc. an IDE will highlight the statements with errors so it’s easy for you to spot them. The table shown below shows 5 popular PHP editors

| **Editor** | **License** | **Cross Platform** | **Brief description** |
| --- | --- | --- | --- |
| Netbeans IDE | Open Source | Yes | * Dedicated PHP coding environment with syntax highlighting and code completion for keywords and other known information. * Supports integration with PHP MVC frameworks i.e. Zend, * Code History that shows the changes made to a file * SFTP,FTP and SVN via plugins. |
| Dreamweaver | Commercial | Yes | * Supports HTML and PHP. * Syntax highlighting, code folding and completion for keywords and other known information. * Supports SFTP and FTP. |
| Zend studio | Commercial | Yes | * Integrated with Zend Server and  Zend PHP MVC framework, PHPUnit, phpDocumentor etc. * Has syntax highlighting, code folding, * Support for[Web services](https://www.guru99.com/web-services-tutorial.html)etc. |
| PHP Eclipse | Open Source | Yes | * Code formatter * Supports SVN, SHH/FTP |
| Notepad ++ | Freeware | Windows only | * Syntax highlighting * Supports SFTP and FTP via plugins. |

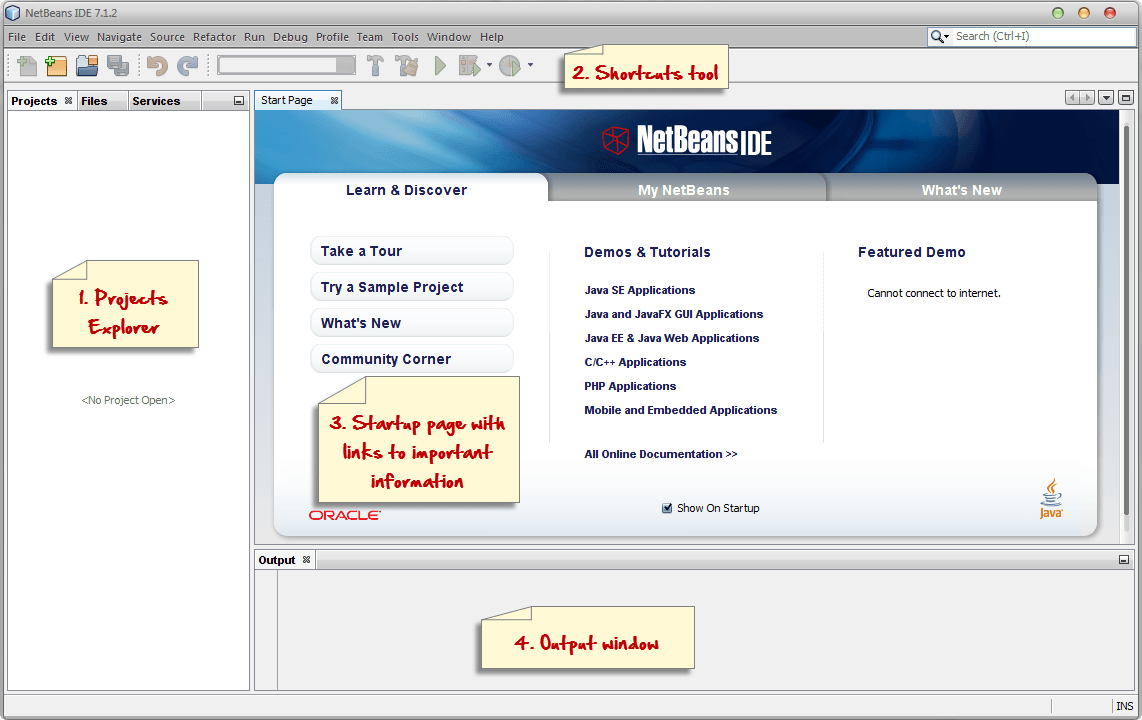
**Netbeans IDE PHP editor** As briefly highlighted in the above table, Netbeans IDE has powerful features that enhance the productive of PHP coders. The IDE can be freely downloaded from the <https://netbeans.org/downloads/index.html>

* Syntax highlighting and auto-complete features enhances your **productivity**
* It has native support for database systems like MySQL. **You don’t need to use two programs to code and develop your database**.
* The IDE can be used in a **collaborative environment**. This comes in handy when you have to work with other developers as a team.
* The IDE has **support for other languages** such as;
  + Java SE
  + Java EE
  + C
  + C++

**The current version of the Netbeans IDE as of this writing is version 7.3**

**Introduction to Netbeans IDE**

After you have successfully installed the Netbeans IDE PHP editor, run the program just like any other windows program. The window shown below appears

[](https://www.guru99.com/images/2013/04/netbeans_ide_walk_through.png)

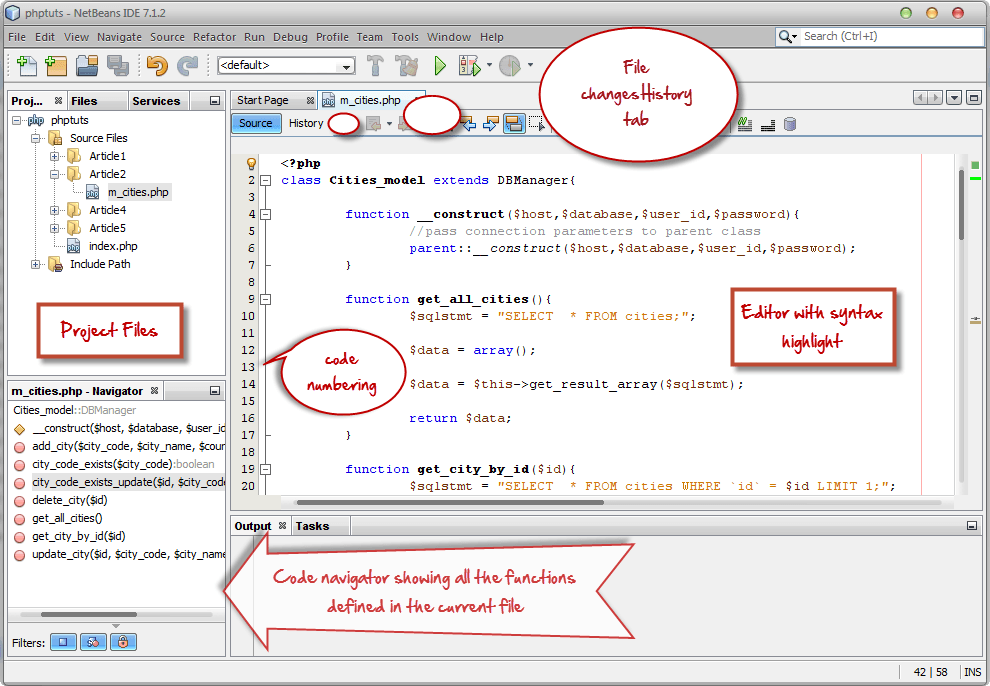
**1. Project explorer** – this panel is used to display all the opened projects. The projects are listed in a tree view.

**2. Shortcuts tool bar** – this toolbar contains shortcuts to frequently performed tasks such as creating a new project, opening an existing project, undo and redo actions etc.

**3. Startup page** – this page contains 3 tabs namely- Learn & Discover, My Netbeans and What’s New.

* The first tab [Learn and Discover] introduces you to the features of the Netbeans IDE, showcases some demos and tutorials that can be developed in the Netbeans IDE.
* The second tab [My Netbeans] lists the recently opened projects, allows you to install plugins and activate features of the IDE.

**4. Output window** – it is used to display output from programs such as Java console applications. It is also used to display log and debug information. The screenshot below shows the IDE with a project open.

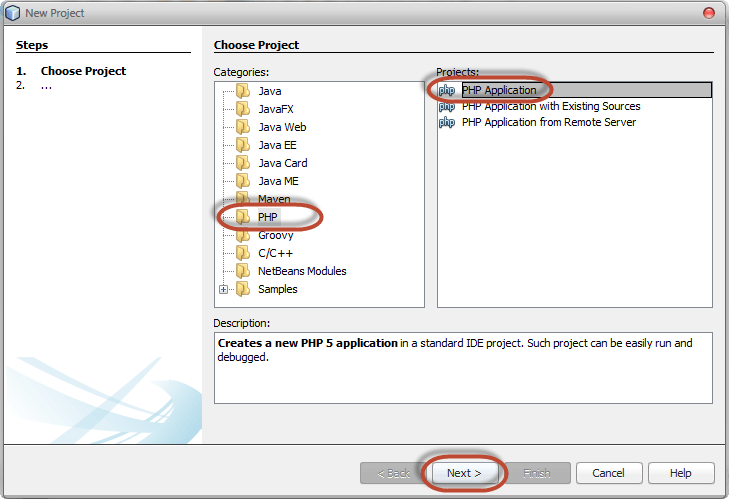
[](https://www.guru99.com/images/2013/04/netbeans_ide_features.png)

**Creating a new PHP project using the Netbeans IDE**

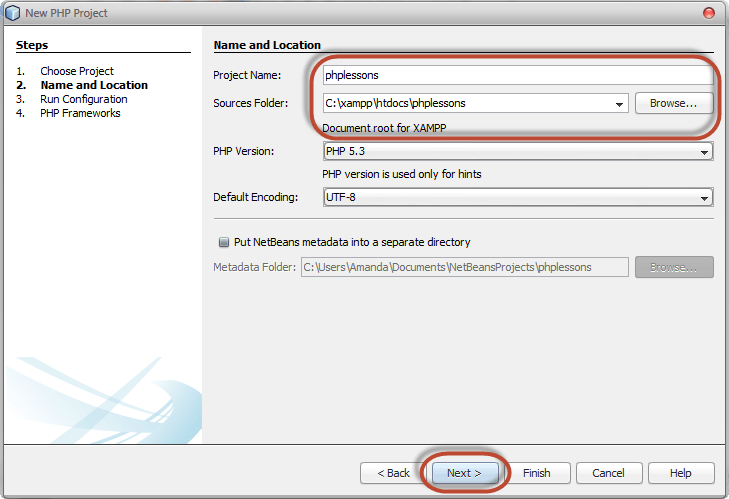
* Click on the create new project button on the tool bar as shown below

[](https://www.guru99.com/images/2013/04/create_new_project.png)

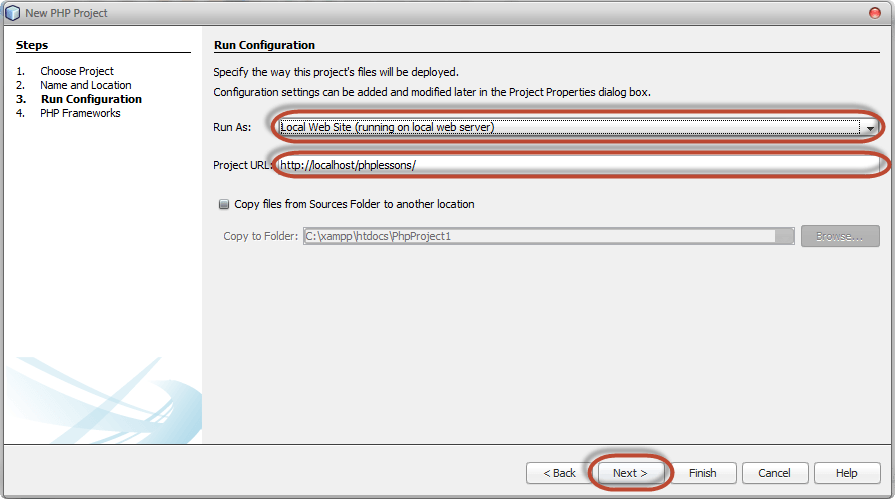
* If you downloaded all the bundles available in the XAMPP NetBeans IDE, make sure you choose PHP under project category, PHP Application under Projects then click on Next button.

[](https://www.guru99.com/images/2013/04/new_project_wizard_1.png)

* Enter the project name as shown below.

[](https://www.guru99.com/images/2013/04/new_project_wizard_2.png)

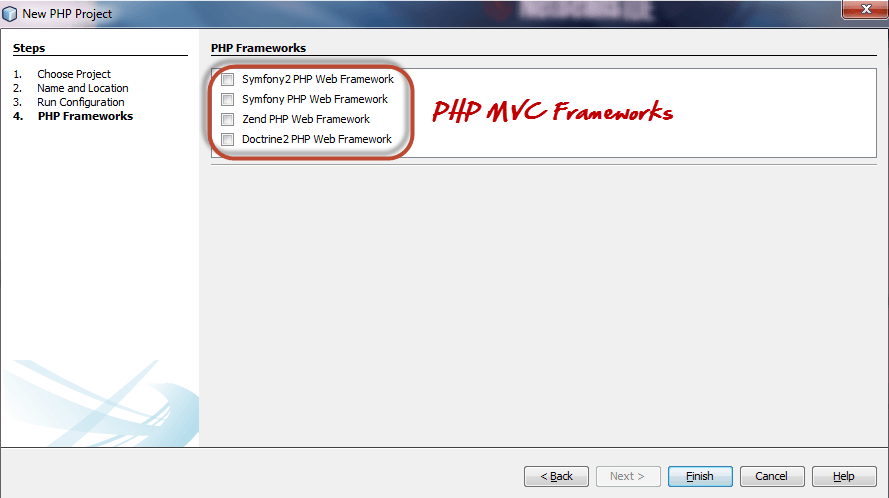
* Make sure the folder is saved in the XAMPP PHP installation directory as shown above.
* Click on next button when done.

[](https://www.guru99.com/images/2013/04/new_project_wizard_3.png)

* **Make sure Run as: is set to Local Web Site(running on local web server)**
  + The Project URL: is set to **http://localhost/phplessons/**

***Note****the above settings will be set for you by default. You don’t have to change anything unless you are an expert*

* Click on Next button

[](https://www.guru99.com/images/2013/04/new_project_wizard_4.png)

* The Netbeans PHP editor allows for integration with PHP MVC frameworks such as Symfony and Zend. For now we will not select any MVC framework. Click on Finish button.
* Your newly created project will be displayed in the project browser and an index.php page created for you.
* The newly create page contains some html code. Replace it with the following code shown below.

**Running your first PHP Example**

<?php

echo "Hello World!";

?>

* Click on the run button on the toolbar as shown below

[Download and Install XAMPP & Netbeans](https://www.guru99.com/images/2013/04/run_project.png)

* Your default browser will be opened with the URL **http://localhost/phplessons/index.php** . The output “Hello World!” will be displayed in your browser.

**Summary**

* The XAMPP full form is X-cross platform, Apache, MySQL, PHP and Perl
* A PHP editor is a program that allows you to write PHP code within the shortest possible time and allows you to debug your syntax errors at design time.
* Netbeans PHP editor is a cross platform open source editor that enhances the productivity of PHP developers.

# PHP Data Types, Variables, Constant, Operators Tutorial

In this tutorial, you will learn-

* [PHP Data Types](https://www.guru99.com/data-types-variables-and-operators.html#1)
* [PHP Variable](https://www.guru99.com/data-types-variables-and-operators.html#2)
* [Use of variables](https://www.guru99.com/data-types-variables-and-operators.html#3)
* [Variable type casting](https://www.guru99.com/data-types-variables-and-operators.html#4)
* [PHP Constant](https://www.guru99.com/data-types-variables-and-operators.html#5)
* [PHP Operators](https://www.guru99.com/data-types-variables-and-operators.html#6)
* [Arithmetic operators](https://www.guru99.com/data-types-variables-and-operators.html#7)
* [Assignment Operators](https://www.guru99.com/data-types-variables-and-operators.html#8)
* [Comparison operators](https://www.guru99.com/data-types-variables-and-operators.html#9)
* [Logical operators](https://www.guru99.com/data-types-variables-and-operators.html#10)

## PHP Data Types

A Data type is the classification of data into a category according to its attributes;

* Alphanumeric characters are classified as strings
* Whole numbers are classified integers
* Numbers with decimal points are classified as floating points.
* True or false values are classified as Boolean.

PHP is a loosely typed language; it does not have explicit defined data types. PHP determines the data types by analyzing the attributes of data supplied. PHP implicitly supports the following data types

* Integer – whole numbers e.g. -3, 0, 69. The maximum value of an integer is platform-dependent. On a 32 bit machine, it’s usually around 2 billion. 64 bit machines usually have larger values. The constant PHP\_INT\_MAX is used to determine the maximum value.

<?php

echo PHP\_INT\_MAX;

?>

**Output:**

9223372036854775807

* Floating point number – decimal numbers e.g. 3.14. they are also known as double or real numbers.  The maximum value of a float is platform-dependent. Floating point numbers are larger than integers.
* Character string – e.g. Hello World
* Boolean – e.g. True or false.

Before we go into more details discussing PHP data types, let’s first discuss variables.

## PHP Variable

A variable is a name given to a memory location that stores data at runtime.

The scope of a variable determines its visibility.

A Php global variable is accessible to all the scripts in an application.

A local variable is only accessible to the script that it was defined in.

Think of a variable as a glass containing water. You can add water into the glass, drink all of it, refill it again etc.

The same applies for variables. Variables are used to store data and provide stored data when needed. Just like in other programming languages, PHP supports variables too. Let’s now look at the rules followed when creating variables in PHP.

* All variable names must start with the dollar sign e.g.
* [Learn about PHP Variable, Operators and Data types](https://www.guru99.com/images/2013/04/php_variables_rules_1.png)
* Variable names are case sensitive; this means $my\_var is different from $MY\_VAR
* [Learn about PHP Variable, Operators and Data types](https://www.guru99.com/images/2013/04/php_variables_rules_2.png)
* All variables names must start with a letter follow other characters e.g. $my\_var1. $1my\_var is not a legal variable name.
* [Learn about PHP Variable, Operators and Data types](https://www.guru99.com/images/2013/04/php_variables_rules_3.png)
* Variable names must not contain any spaces, “$first name” is not a legal variable name. You can instead use an underscore in place of the space e.g. $first\_name. You cant use characters such as the dollar or minus sign to separate variable names.
* [Learn about PHP Variable, Operators and Data types](https://www.guru99.com/images/2013/04/php_variables_rules_4.png)

  Let’s now look at how PHP determines the data type depending on the attributes of the supplied data.

<?php

$my\_var = 1;

echo $my\_var;

?>

Output:

1

**Floating point numbers**

<?php

$my\_var = 3.14;

echo $my\_var;

?>

Output:

3.14

**Character strings**

<?php

$my\_var ="Hypertext Pre Processor";

echo $my\_var;

?>

Output:

Hypertext Pre Processor

## Use of Variables

Variables help separate data from the program algorithms.

The same algorithm can be used for different input data values.

For example, suppose that you are developing a calculator program that adds up two numbers, you can create two variables that accept the numbers then you use the variables names in the expression that does the addition.

## Variable Type Casting

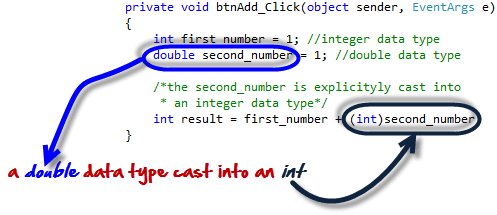
Performing arithmetic computations using variables in a language such as[C#](https://www.guru99.com/c-sharp-tutorial.html)requires the variables to be of the same data type.

Type casting is converting a variable or value into a desired data type.

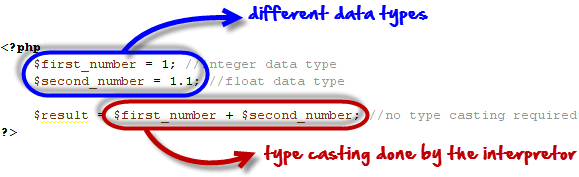
This is very useful when performing arithmetic computations that require variables to be of the same data type.

Type casting in PHP is done by the interpreter.

In other languages such as C#, you have to cast the variables. The code below shows type casting in C#.

[](https://www.guru99.com/images/2013/04/c-sharp_data_casting.png)

The diagram below shows PHP implementing the above example.

[](https://www.guru99.com/images/2013/04/php_data_casting.png)

PHP also allows you to cast the data type. This is known as explicit casting. The code below demonstrates explicit type casting.

<?php

$a = 1;

$b = 1.5;

$c = $a + $b;

$c = $a + (int) $b;

echo $c;

?>

**Output:**

2

Above Code Output 2 The var\_dump function is used to determine the data type. The code below demonstrates how to use the var\_dump function.

<?php

$a = 1;

var\_dump($a);

$b = 1.5;

var\_dump($b);

$c = "I Love PHP";

var\_dump($c);

$d = true;

var\_dump($d);

?>

**Output:**

int(1) float(1.5) string(10) "I Love PHP" bool(true)

## PHP Constant

**Define constant**- A constant is a variable whose value cannot be changed at runtime.

Suppose we are developing a program that uses the value of PI 3.14, we can use a constant to store its value.

Let’s now look at an example that defines a constant. define('PI',3.14); //creates a constant with a value of 3.14 Once you define PI as 3.14 , writing a code like below will generate an error PI = 4; //PI has been defined as a constant therefore assigning a value is not permissible.

## PHP Operators

### Arithmetic operators

Arithmetic operators are used to perform arithmetic operations on numeric data. The concatenate operator works on strings values too. PHP supports the following operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| + | Addition | Summation of x and y | 1 + 1; | 2 |
| - | Subtraction | Difference between x and y | 1 – 1; | 0 |
| \* | Multiplication | Multiplies x and y | 3 \* 7; | 21 |
| / | Division | Quotient of x and y | 45 / 5; | 9 |
| % | Php Modulus | Gives reminder of diving x and y | 10 % 3; | 1 |
| -n | Negation | Turns n into a negative number | -(-5); | 5 |
| x . y | Concatenation | Puts together x and y | "PHP" . " ROCKS";10 . 3; | PHP ROCKS103 |

### Assignment Operators

Assignment operators are used to assign values to variables. They can also be used together with arithmetic operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| x = ? | assignment | Assigns the value of x to ? | $x = 5; | 5 |
| x += ? | addition | Increments the value of x by ? | $x = 2;$x += 1; | 3 |
| X -= ? | subtraction | Subtracts ? from the value of x | $x = 3;$x -= 2; | 1 |
| X \*=? | multiplication | Multiplies the value of x ? times | $x = 0;$x \*=9; | 0 |
| X /=? | division | Quotient of x and ? | $x = 6;$x /=3; | 2 |
| X %=? | modulus | The reminder of dividing x by? | $x = 3;$x %= 2; | 1 |
| X .=? | concatenate | Puts together items | " $x = ‘Pretty’;$x .= ‘ Cool!’;" | Pretty Cool! |

### Comparison operators

Comparison operators are used to compare values and data types.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X == y | Equal | Compares x and y then returns true if they are equal | 1 == "1"; | True or 1 |
| X === y | identical | Compares both values and data types. | 1 === "1"; | False or 0. Since 1 is integer and “1” is string |
| X != y, x <> y | PHP Not equal | Compares values of x and y. returns true if the values are not equal | 2 != 1; | True or 1 |
| X > y | Greater than | Compares values of x and y. returns true if x is greater than y | 3 > 1; | True or 1 |
| X < y | Less than | Compares values of x and y. returns true if x is less than y | 2 < 1; | False or 0 |
| X >= y | Greater than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 1 >=1 | True or 1 |
| X <= y | Less than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 8 <= 6 | False or 0 |

### Logical operators

When working with logical operators, any number greater than or less than zero (0) evaluates to true. Zero (0) evaluates to false.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X and y, x && y | And | Returns true if both x and y are equal | 1 and 4;True&& False; | True or 1False or 0 |
| X or y, x || y | Or | Returns true if either x or y is true | 6 or 9;0 || 0; | True or 1False or 0 |
| X xor y | Exclusive or, xor | Returns true if only x is true or only y is true | 1 xor 1;1 xor 0; | False or 0True or 1 |
| !x | Not | Returns true if x is false and false if x is true | !0; | True or 1 |

## Summary

* PHP is a loosely typed language.
* Variables are memory locations used to store data
* The value of constants cannot be changed at runtime
* Type casting is used to convert a value or variable into a desired data type
* Arithmetic operators are used to manipulate numeric data
* Assignment operators are used to assign data to variables
* Comparison operators are used to compare variables or values
* Logical operators are used to compare conditions or values

**PHP Comments, Include/Include\_once, Require/Require\_once**

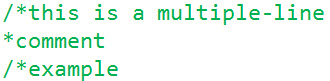
**Why use Comments?**

* If you don’t work on the source code for some time, it’s easy to forget what the code does. Commenting the source code helps remember what the code does.
* Commenting source code is also very important when multiple developers have to work on the same project. The changes made by one developer can be easily understood by other developers by simply reading the comments.
* As the best practice, you must have 3 lines of comments for every 10 lines of code

**In this tutorial, you will learn-**

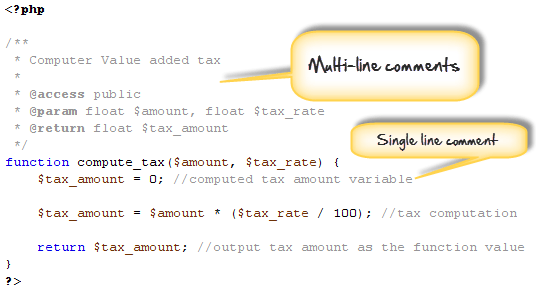
* [PHP Comments](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#1)
* [PHP Include & PHP Include\_once](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#3)
* [Example : Include / Include\_once](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#4)
* [PHP Require & PHP require\_once](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#5)
* [Example : Require](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#6)
* [PHP include vs require](https://www.guru99.com/comments-includeinclude-once-requirerequire-once.html#7)

**PHP Comments**

* Comments help us to understand the code
* Comments are explanations that we include in our source code. These comments are for human understanding.
* Single line comments start with double forward slashes // and they end in the same line.
* [PHP Include, Require & Comments](https://www.guru99.com/images/2013/04/php_single_comment.jpg)
* Multiple line comments start with a forward slash followed by the asterisk /\* and end with the asterisk followed by the forward slash \*/.
* [](https://www.guru99.com/images/2013/04/php_multi_line_comments.jpg)

The diagram below shows a PHP file with both multiple line and single line comments

PHP Example

[](https://www.guru99.com/images/2013/04/php_comments.png)

**PHP Include & PHP Include\_once**

The “include” php statement is used to include other files into a PHP file.

It has two variations, include and include\_once.  Include\_once is ignored by the PHP interpreter if the file to be included.

The include statement has the following syntax

<?php

include 'file\_name';

?>

The include\_once statement has the following syntax

<?php

include\_once 'file\_name';

?>

HERE,

* “Include/include\_once” is the statement that includes file
* “'file\_name'” is the name of the file to be included.

**Example : Include / Include\_once**

Suppose you are developing a website that contains the same navigation menu across all the pages.

You can create a common header then include it in every page using the include statement Let’s see how this can be done.

* We will create 2 files names
* header.php, index.php

Below are the codes for; *header.php*

<a href="/index.php">Home</a>

<a href="/aboutus.php">About us</a>

<a href="/services.php">Services</a>

<a href="/contactus.php">Contact Us</a>

*index.php*

<?php

include 'header.php';

?>

The header page above will output

**PHP Require & PHP require\_once**

The require statement has two variations, require and require\_once.

The require/require\_once statement is used to include file.

Require\_once is ignored if the required file has already been added by any of the four include statements.

It has the following syntax

<?php

require 'file\_name';

?>

<?php

require\_once 'file\_name';

?>

HERE,

* “require/require\_once” is the statement that includes file
* “'file\_name'” is the name of the file to be included.

**Example : Require**

Suppose we are developing a database powered application.

We can create a configuration file that we can include in all pages that connect to the database using the require statement. config.php

<?php

$config['host'] = 'localhost';

$config['db'] = 'my\_database';

$config['uid'] = 'root';

$config['password'] = '';

?>

Let’s now look at the sample code that requires the config file. *Pages\_model.php*

<?php

require 'config.php'; //require the config file

//other code for connecting to the database

?>

**Php include vs require**

The difference between include / require

|  |  |
| --- | --- |
| **Include** | **Require** |
| Issues a warning when an error occurs | Does not issue a warning |
| Execution of the script continues when an error occurs | Execution of the script stops when an error occurs. |

  Generally, it’s recommended using the include statement so that when an error occurs, execution of the script continues to display the webmaster email address or the contact us page.

The require statement should be used if the entire script cannot run without the requested file.

The “include” and “require” statements can be used at any line in the source codes where you want the code to appear.

**Summary**

* Single HTML code such as headers, footers, side bars etc. can be shared across many pages. This makes it easy to update the website by just updating a single file.
* PHP code such as database configuration settings, custom functions etc. can be shared across many pages ensuring the website/application uses the same settings.
* Comments are used to help understand source code. They are for human understanding
* Single line comment statements start with double forward slashes //.
* Multi-line comment statements are enclosed between /\* statements \*/.
* The “include, include\_once, require and require\_once” statements are used to include files.
* Include\_once/require\_once is ignored if the requested file has already been included using any of the four statements.
* The “include” statement issues a warning and continues with the execution if the requested file has not been found.
* The require statement raises a fatal error and stops the script execution.
* The “include” statement should be in most cases except in situations where without the requested file to be include, the entire script cannot run.

# PHP Array: Associative, Multidimensional

## What is a PHP Array?

A PHP array is a variable that stores more than one piece of related data in a single variable.

Think of an array as a box of chocolates with slots inside.

The box represents the array itself while the spaces containing chocolates represent the values stored in the arrays.

The diagram below illustrates the above syntax.  
 

**In this tutorial, you will learn-**

* [Numeric Arrays](https://www.guru99.com/arrays.html#1)
* [PHP Associative Array](https://www.guru99.com/arrays.html#2)
* [PHP Multi-dimensional arrays](https://www.guru99.com/arrays.html#3)
* [PHP Array operators](https://www.guru99.com/arrays.html#4)

## Numeric Arrays

Numeric arrays use number as access keys.

An access key is a reference to a memory slot in an array variable.

The access key is used whenever we want to read or assign a new value an array element.

Below is the syntax for creating numeric array in php.

Array Example

<?php

$variable\_name[n] = value;

?>

Or

<?php

$variable\_name = array(n => value, …);

?>

HERE,

* “$variable\_name…” is the name of the variable
* “[n]” is the access index number of the element
* “value” is the value assigned to the array element.

Let’s now look at an example of a numeric array.

Suppose we have 5 movies that we want to store in array variables.

We can use the example shown below to do that.

<?php

$movie[0] = 'Shaolin Monk';

$movie[1] = 'Drunken Master';

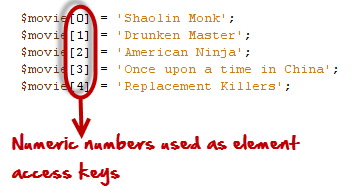
$movie[2] = 'American Ninja';

$movie[3] = 'Once upon a time in China';

$movie[4] = 'Replacement Killers';

?>

Here,

[](https://www.guru99.com/images/2013/04/Array.png)

Each movie is given an index number that is used to retrieve or modify its value. Observe the following code-

<?php

$movie[0]="Shaolin Monk";

$movie[1]="Drunken Master";

$movie[2]="American Ninja";

$movie[3]="Once upon a time in China";

$movie[4]="Replacement Killers";

echo $movie[3];

$movie[3] = " Eastern Condors";

echo $movie[3];

?>

**Output:**

Once upon a time in China Eastern Condors

As you can see from the above examples, working with arrays in PHP when dealing with multiple values of the same nature is very easy and flexible.

Alternatively, the above array variables can also be created using the following code.

<?php

$movie = array(0 => "Shaolin Monk",

1 => "Drunken Master",

2 => "American Ninja",

3 => "Once upon a time in China",

4 =>"Replacement Killers" );

echo $movie[4];

?>

**Output:**

Replacement Killers

## PHP Associative Array

Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

Below is the syntax for creating associative array in php.

<?php

$variable\_name['key\_name'] = value;

$variable\_name = array('keyname' => value);

?>

HERE,

* “$variable\_name…” is the name of the variable
* “['key\_name']” is the access index number of the element
* “value” is the value assigned to the array element.

  Let’s suppose that we have a group of persons, and we want to assign the gender of each person against their names.

We can use an associative array to do that.The code below helps us to do that.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

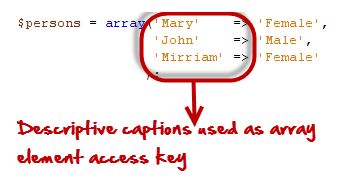
print\_r($persons);

echo "";

echo "Mary is a " . $persons["Mary"];

?>

**HERE,**

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

**Output:**

Array ( [Mary] => Female [John] => Male [Mirriam] => Female ) Mary is a Female

Associative array are also very useful when retrieving data from the database.

The field names are used as id keys.

## PHP Multi-dimensional arrays

These are arrays that contain other nested arrays.

The advantage of multidimensional arrays is that they allow us to group related data together.

Let’s now look at a practical example that implements a php multidimensional array.

The table below shows a list of movies by category.

| **Movie title** | **Category** |
| --- | --- |
| Pink Panther | Comedy |
| John English | Comedy |
| Die Hard | Action |
| Expendables | Action |
| The Lord of the rings | Epic |
| Romeo and Juliet | Romance |
| See no evil hear no evil | Comedy |

The above information can be represented as a multidimensional array. The code below shows the implementation.

<?php

$movies =array(

"comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),

"action" => array("Die Hard", "Expendables"),

"epic" => array("The Lord of the rings"),

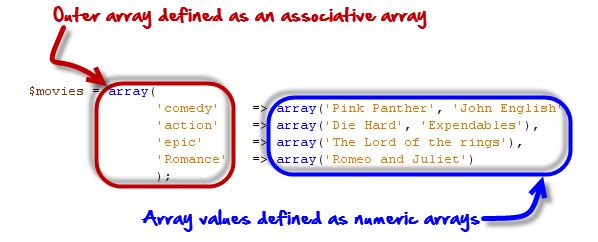
"Romance" => array("Romeo and Juliet")

);

print\_r($movies);

?>

  HERE,

[](https://www.guru99.com/images/2013/04/Capture1.jpg)

**Output:**

Array ( [comedy] => Array ( [0] => Pink Panther [1] => John English [2] => See no evil hear no evil ) [action] => Array ( [0] => Die Hard [1] => Expendables ) [epic] => Array ( [0] => The Lord of the rings ) [Romance] => Array ( [0] => Romeo and Juliet ) )

Another way to define the same array is as follows

<?php

$film=array(

                "comedy" => array(

                                0 => "Pink Panther",

                                1 => "john English",

                                2 => "See no evil hear no evil"

                                ),

                "action" => array (

                                0 => "Die Hard",

                                1 => "Expendables"

                                ),

                "epic" => array (

                                0 => "The Lord of the rings"

                                ),

                "Romance" => array

                                (

                                0 => "Romeo and Juliet"

                                )

);

echo $film["comedy"][0];

?>

**Output:**

Pink Panther

  Note: the movies numeric array has been nested inside the categories associative array

## PHP Arrays: Operators

| **Operator** | **Name** | **Description** | **How to do it** | **Output** |
| --- | --- | --- | --- | --- |
| x + y | Union | Combines elements from both arrays | <?php  $x = array('id' => 1);  $y = array('value' => 10);  $z = $x + $y;  ?> | Array([id] => 1 [value] => 10) |
| X == y | Equal | Compares two arrays if they are equal and returns true if yes. | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x == $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | True or 1 |
| X === y | Identical | Compares both the values and data types | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x === $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | False or 0 |
| X != y, x <> y | Not equal |  | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x != $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | False or 0 |
| X !== y | Non identical |  | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x !== $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | True or 1 |

## PHP Array Functions

### Count function

The count function is used to count the number of elements that an php array contains. The code below shows the implementation.

<?php

$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");

echo count($lecturers);

?>

**Output:**

3

### is\_array function

The is\_array function is used to determine if a variable is an array or not. Let’s now look at an example that implements the is\_array functions.

<?php

$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");

echo is\_array($lecturers);

?>

**Output:**

1

### Sort

This function is used to sort arrays by the values.

If the values are alphanumeric, it sorts them in alphabetical order.

If the values are numeric, it sorts them in ascending order.

It removes the existing access keys and add new numeric keys.

The output of this function is a numeric array

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

sort($persons);

print\_r($persons);

?>

**Output:**

Array ( [0] => Female [1] => Female [2] => Male )

### ksort

This function is used to sort the array using the key. The following example illustrates its usage.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

ksort($persons);

print\_r($persons);

?>

**Output:**

Array ( [John] => Male [Mary] => Female [Mirriam] => Female )

### asort

This function is used to sort the array using the values. The following example illustrates its usage.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

asort($persons);

print\_r($persons);

?>

**Output:**

Array ( [Mary] => Female [Mirriam] => Female [John] => Male )

### Why use arrays?

* Contents of Arrays can be stretched,
* Arrays easily help group related information such as server login details together
* Arrays help write cleaner code.

## Summary

* Arrays are special variables with the capacity to store multi values.
* Arrays are flexibility and can be easily stretched to accommodate more values
* Numeric arrays use numbers for the array keys
* PHP Associative array use descriptive names for array keys
* Multidimensional arrays contain other arrays inside them.
* The count function is used to get the number of items that have been stored in an array
* The is\_array function is used to determine whether a variable is a valid array or not.
* Other array functions include sort, ksort, assort etc.

**PHP Control Structures: If else, Switch Case**

**What is a control structure?**

Code execution can be grouped into categories as shown below

* **Sequential**– this one involves executing all the codes in the order in which they have been written.
* **Decision** – this one involves making a choice given a number of options. The code executed depends on the value of the condition.

A control structure is a block of code that decides the execution path of a program depending on the value of the set condition.

Let’s now look at some of the control structures that PHP supports.

**PHP IF Else**

If… then... else is the **simplest** **control** **structure**. It evaluates the conditions using Boolean logic  
**When to use if… then… else**

* You have a block of code that should be executed only if a certain condition is true
* You have two options, and you have to select one.
* If… then… else if… is used when you have to select more than two options and you have to select one or more

**Syntax** The syntax for if… then… else is;

<?php

if (condition is true) {

block one

else

block two

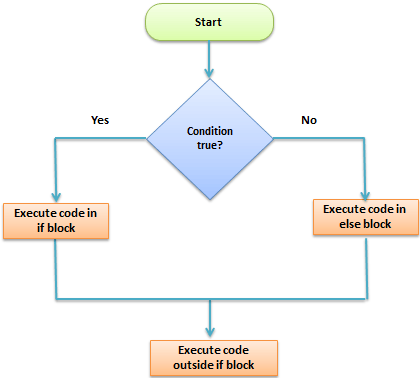
}

?>

**HERE,**

* “**if (condition is true)”** is the control structure
* “**block one**” is the code to be executed if the condition is true
* **{…else…}**is the fallback if the condition is false
* “**block two**” is the block of code executed if the condition is false

**How it works** The flow chart shown below illustrates how the if then… else control structure works

[](https://www.guru99.com/images/2013/04/if_then_flowchart.png)

**Let’s see this in action** The code below uses “if… then… else” to determine the larger value between two numbers.

<?php

$first\_number = 7;

$second\_number = 21;

if ($first\_number > $second\_number){

echo "$first\_number is greater than $second\_number";

}else{

echo "$second\_number is greater than $first\_number";

}

?>

**Output:**

21 is greater than 7

**PHP Switch Case**

**Switch… case** is similar to the **if then… else** control structure.

It only **executes** a single block of code depending on the **value** of the condition.

If no condition has been met then the default block of code is executed.

It has the following basic syntax.

<?php

switch(condition){

case value:

//block of code to be executed

break;

case value2:

//block of code to be executed

break;

default:

//default block code

break;

}

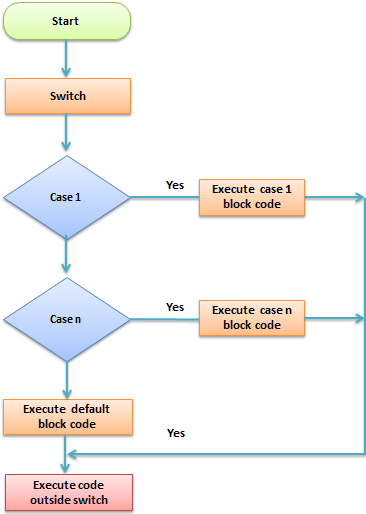
?>

**HERE,**

* **“switch(…){…}”** is the control structure block code
* **“case value: case…”** are the blocks of code to be executed depending on the value of the condition
* **“default:”** is the block of code to be executed when no value matches with the condition

**How it works**

The flow chart shown below illustrates how the switch control structure works

[](https://www.guru99.com/images/2013/04/switch_flowchart.png)

**Practical example**

The code below uses the switch control structure to display a message depending on the day of the week.

<?php

$today = "wednesday";

switch($today){

case "sunday":

echo "pray for us sinners.";

break;

case "wednesday":

echo "ladies night, take her out for dinner";

break;

case "saturday":

echo "take care as you go out tonight.";

break;

default:

echo "have a nice day at work";

break;

}

?>

**Output:**

ladies night, take her out for dinner

**Summary**

* Control structures are used to control the execution of the program
* The if then... else is when you have more than route block of code to execute depending on the value of the condition
* Switch… case is used to when you have a number of block codes, and you only have to execute one of them depending on the value of the set case.

# PHP Loop: For, ForEach, While, Do While [Example]

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

### PHP For Loop

The above code outputs “21 is greater than 7” For loops For... loops execute the block of code a specifiednumber of times. There are basically two types of for loops;

* for
* for… each.

Let’s now look at them separately. **For loop** It has the following basic **syntax**

<?php

for (initialize; condition; increment){

//code to be executed

}

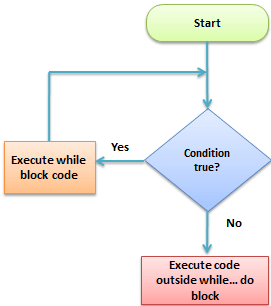
?>

**HERE,**

* **“for…{…}”** is the loop block
* “**initialize**” usually an integer; it is used to set the counter’s initial value.
* **“condition”** the condition that is evaluated for each php execution. If it evaluates to true then execution of the for... loop is terminated. If it evaluates to false, the execution of the for... loop continues.
* **“increment”** is used to increment the initial value of counter integer.

**How it works**

The flowchart shown below illustrates how for loop in php works

[](https://www.guru99.com/images/2013/04/loop.png)

**How to code**

The code below uses the “for… loop” to print values of multiplying 10 by 0 through to 10

<?php

for ($i = 0; $i < 10; $i++){

$product = 10 \* $i;

echo "The product of 10 \* $i is $product <br/>";

}

?>

**Output:**

The product of 10 x 0 is 0

The product of 10 x 1 is 10

The product of 10 x 2 is 20

The product of 10 x 3 is 30

The product of 10 x 4 is 40

The product of 10 x 5 is 50

The product of 10 x 6 is 60

The product of 10 x 7 is 70

The product of 10 x 8 is 80

The product of 10 x 9 is 90

### PHP For Each loop

The php foreach loop is used to iterate through array values. It has the following basic syntax

<?php

foreach($array\_variable  as $array\_values){

block of code to be executed

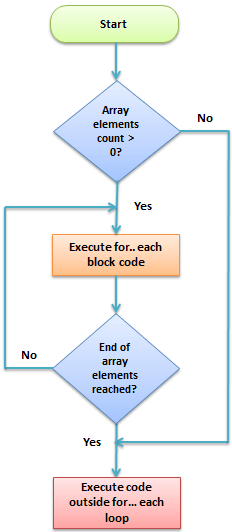
}

?>

**HERE,**

* **“foreach(…){…}”** is the foreach php loop block code
* **“$array\_data”** is the array variable to be looped through
* **“$array\_value “** is the temporary variable that holds the current array item values.
* “block of code…” is the piece of code that operates on the array values

**How it works** The flowchart shown below illustrates how the for… each… loop works

[](https://www.guru99.com/images/2013/04/for_loop_flowchart.png)

**Practical examples**

The code below uses for… each loop to read and print the elements of an array.

<?php

$animals\_list = array("Lion","Wolf","Dog","Leopard","Tiger");

foreach($animals\_list as $array\_values){

echo $array\_values . "<br>";

}

?>

**Output:**

Lion

Wolf

Dog

Leopard

Tiger

Let’s look at another example that loops through an **associative array**.

An associative array uses alphanumeric words for access keys.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

foreach($persons as $key => $value){

echo "$key is $value"."<br>";

}

?>

  The names have been used as array keys and gender as the values.

**Output:**

Mary is Female

John is Male

Mirriam is Female

## While Loop

### PHP While loop

They are used to execute a block of code a repeatedly until the set condition gets satisfied

**When to use while loops**

* While loops are used to execute a block of code until a certain condition becomes true.
* You can use a while loop to read records returned from a database query.

**Types of while loops**

* **Do… while** - executes the block of code at least once before evaluating the condition
* **While…** - checks the condition first. If it evaluates to true, the block of code is executed as long as the condition is true. If it evaluates to false, the execution of the while loop is terminated.

**While loop**

It has the following syntax

<?php

while (condition){

block of code to be executed;

}

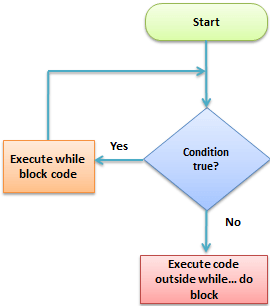
?>

**HERE,**

* **“while(…){…}”** is the while loop block code
* **“condition”** is the condition to be evaluated by the while loop
* **“block of code…”** is the code to be executed if the condition gets satisfied

**How it works**

The flow chart shown below illustrates how the while… loop works

[](https://www.guru99.com/images/2013/04/while_loop_flowchart1.png)

**Practical example**

The code below uses the while… loop to print numbers 1 to 5.

<?php

$i = 0;

while ($i < 5){

echo $i + 1 . "<br>";

$i++;

}

?>

**Output:**

1

2

3

4

5

### PHP Do While

The difference between While… loop and Do… while loop is do… while is executed at-least once before the condition is evaluated.

Let’s now look at the basic syntax of a do… while loop

<?php

do{

block of code to be executed

}

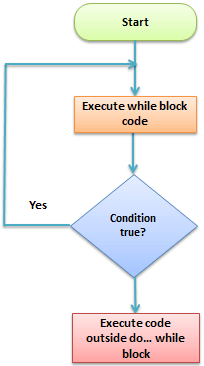
?>

  while(condition);  
  
**HERE,**

* **“do{…} while(…)”** is the do… while loop block code
* **“condition”** is the condition to be evaluated by the while loop
* **“block of code…”** is the code that is executed at least once by the do… while loop

**How it works**

The flow chart shown below illustrates how the while… loop works

[](https://www.guru99.com/images/2013/04/do_while_loop.png)

**Practical example**

We are now going to modify the while… loop example and implement it using the do… while loop and set the counter initial value to 9.

The code below implements the above modified example

<?php

$i = 9;

do{

    echo "$i is"." <br>";

}

while($i < 9);

?>

**The above code outputs:**

9

***Note****the above example outputs 9 only.*

*This is because the do… while loop is executed at least once even if the set condition evaluates to false.*

## Summary

* The for… loop is used to execute a block of a specified number of times
* The foreach… loop is used to loop through arrays
* While… loop is used to execute a block of code as long as the set condition is made to be false
* The do… while loop is used to execute the block of code at least once then the rest of the execution is dependent on the evaluation of the set condition

**PHP Strings: PHP String Functions Explained with Examples**

**What is String in PHP?**

A string is a collection of characters. String is one of the data types supported by PHP.

The string variables can contain alphanumeric characters. Strings are created when;

* You declare variable and assign string characters to it
* You can directly use PHP Strings with echo statement.
* PHP String functions are language construct, it helps capture words.
* Learning how strings work in PHP and how to manipulate them will make you a very effective and productive developer.

**In this**[**PHP Tutorial**](https://www.guru99.com/php-tutorials.html)**, you will learn-**

* [What is String in PHP?](https://www.guru99.com/php-strings.html#1)
* [PHP Create Strings Using Single quotes with Example](https://www.guru99.com/php-strings.html#2)
* [PHP Create Strings Using Double quotes with Example](https://www.guru99.com/php-strings.html#3)
* [PHP Heredoc with Example](https://www.guru99.com/php-strings.html#4)
* [PHP Nowdoc with Example](https://www.guru99.com/php-strings.html#5)
* [PHP String Function Examples](https://www.guru99.com/php-strings.html#6)

**PHP Create Strings Using Single quotes with Example**

Let’s now look at the four different ways of creating PHP string functions and string manipulation in PHP.

Creating PHP Strings Using Single quotes: The simplest way to create a string is to use single quotes.

Let’s look at an example that creates a simple string in PHP.

<?php

    var\_dump('You need to be logged in to view this page');

?>

**Output:**

string(42) "You need to be logged in to view this page"

If the single quote is part of the string value, it can be escaped using the backslash.

The code below illustrates how to escape a single quote.

<?php

echo 'I \'ll be back after 20 minutes';

?>

**Output:**

I'll be back after 20 minutes

**PHP Create Strings Using Double quotes with Example**

The double quotes are used to create relatively complex strings compared to single quotes.

Variable names can be used inside double quotes and their values will be displayed.

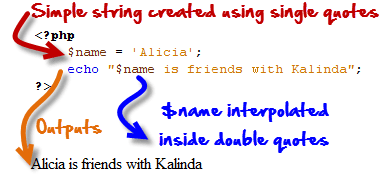
Let’s look at an example.

<?php

$name='Alicia';

echo "$name is friends with kalinda";

?>

[](https://www.guru99.com/images/2013/04/php_variable_interporation.png)

HERE,

* The above example creates a simple string with the value of Alicia.
* The variable name is then used in the string created using double quotes and its value is interpolated at run time.

**Output:**

Alicia is friends with kalinda

In addition to variable interpolations, the double quote string can also escape more special characters such as “\n for a linefeed, \$ dollar for the dollar sign” etc.

More examples Let’s suppose that we have the following code

<?php $pwd = "pas$word"; echo $pwd; ?>

**Output:**

NOTICE : Undefined variable

pas

executing the above codes issues a notice “Notice: Undefined variable”.

This is because $word is treated as a variable.

If we want the dollar sign to be treated as a literal value, we have to escape it.

<?php

$word="word";

$pwd = "pas\$word";

echo $pwd; ?>

**Output:**

pas$word

**PHP Heredoc with Example**

This heredoc methodology is used to create fairly complex strings as compared to double quotes.

The heredoc supports all the features of double quotes and allows creating string values with more than one line without PHP string concatenation.

Using double quotes to create strings that have multiple lines generates an error.

You can also use double quotes inside without escaping them.

The example below illustrates how the Heredoc method is used to create string values.

<?php

$baby\_name = "Shalon";

echo <<<EOT

    When $baby\_name was a baby,

    She used to look like a "boy".

EOT;

?>

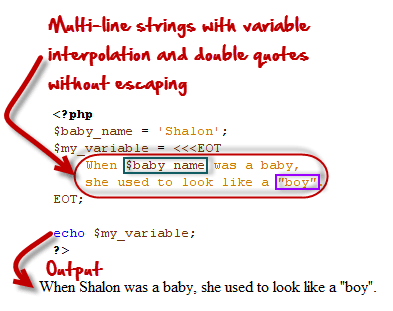
  HERE,

**<<<EOT** is the string delimiter.

EOT is the acronym for end of text.

It should be defined in its on line at the beginning of the string and at the end.

Note: you can use anything you like in place of EOT

[](https://www.guru99.com/images/2013/04/php_eot.png)

**Output:**

When Shalon was a baby, She used to look like a "boy".

**PHP Nowdoc with Example**

The Nowdoc string creation method is similar to the heredoc method but works like the way single quotes work.

No parsing takes place inside the Nowdoc.

Nowdoc is ideal when working with raw data that do not need to be parsed.

The code below shows the Nowdoc implementation

<?php

$baby\_name = "Shalon";

$my\_variable = <<<'EOT'

    When $baby\_name was a baby,

    She used to look like a "boy".

EOT;

echo $my\_variable;

?>

**Output:**

When $baby\_name was a baby, She used to look like a "boy".

**PHP String Function Examples**

String functions in PHP are used to manipulate string values.

We are now going to look at some of the commonly used string functions in PHP

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| strtolower | Used to convert all string characters to lower case letters | echo strtolower( 'Benjamin'); | outputs benjamin |
| strtoupper | Used to convert all string characters to upper case letters | echo strtoupper('george w bush'); | outputs GEORGE W BUSH |
| strlen | The string length function is used to count the number of character in a string. Spaces in between characters are also counted | echo strlen('united states of america'); | 24 |
| explode | Used to convert strings into an array variable | $settings = explode(';', "host=localhost; db=sales; uid=root; pwd=demo"); print\_r($settings); | Array ( [0] => host=localhost [1] => db=sales [2] => uid=root [3] => pwd=demo ) |
| substr | Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned. | $my\_var = 'This is a really long sentence that I wish to cut short';echo substr($my\_var,0, 12).'...'; | This is a re... |
| str\_replace | Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is the replacement text and the third argument is the text that is analyzed. | echo str\_replace ('the', 'that', 'the laptop is very expensive'); | that laptop is very expensive |
| strpos | Used to locate the and return the position of a character(s) within a string. This function accepts two arguments | echo strpos('PHP Programing','Pro'); | 4 |
| sha1 | Used to calculate the SHA-1 hash of a string value | echo sha1('password'); | 5baa61e4c 9b93f3f0 682250b6cf8331b 7ee68fd8 |
| md5 | Used to calculate the md5 hash of a string value | echo md5('password'); | 9f961034ee 4de758 baf4de09ceeb1a75 |
| str\_word\_count | Used to count the number of words in a string. | echo str\_word\_count ('This is a really long sentence that I wish to cut short'); | 12 |
| ucfirst | Make the first character of a string value upper case | echo ucfirst('respect'); | Outputs Respect |
| lcfirst | Make the first character of a string value lower case | echo lcfirst('RESPECT'); | Outputs rESPECT |

For a complete list of PHP strings, check [https://php.net/manual/en/ref.strings.php](https://www.php.net/manual/en/ref.strings.php)

**Summary**

* Define string in PHP: A string function in PHP is a set of characters
* Explain string function in PHP: Strings are created when you declare a variable and assign string characters to it.
* Single quotes are used to specify simple strings in PHP
* Double quotes are used to create fairly complex strings in PHP
* heredoc is used to create complex strings
* Nowdoc is used to create strings that cannot be parsed.

**PHP Function: How to Define? Built in | String | User Defined**

**What is a Function in PHP?**

A **Function in PHP** is a reusable piece or block of code that performs a specific action. It takes input from the user in the form of parameters, performs certain actions, and gives the output. Functions can either return values when called or can simply perform an operation without returning any value.

PHP has over 700 functions built in that perform different tasks.

**In this tutorial, you will learn-**

* [Why use Functions?](https://www.guru99.com/functions-in-php.html#1)
* [Built in Functions](https://www.guru99.com/functions-in-php.html#2)
* [String Functions](https://www.guru99.com/functions-in-php.html#3)
* [Numeric Functions](https://www.guru99.com/functions-in-php.html#4)
* [Date Function](https://www.guru99.com/functions-in-php.html#5)
* [Why use User Defined Functions?](https://www.guru99.com/functions-in-php.html#6)

**Why use Functions?**

* Better code organization – PHP functions allow us to group blocks of related code that perform a specific task together.
* Reusability – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
* Easy maintenance- updates to the system only need to be made in one place.

**Built in Functions**

Built in functions are predefined functions in PHP that exist in the installation package.

These PHP inbuilt functions are what make PHP a very efficient and productive scripting language.

The built in functions of PHP can be classified into many categories. Below is the list of the categories.

**String Functions**

These are functions that manipulate string data, refer to the article on strings for implementation examples of string functions

**Numeric Functions**

Numeric functions in PHP are the functions that return numeric results.

Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.

**The table below shows the common PHP numeric functions**

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| is\_number | Accepts an argument and returns true if its numeric and false if it’s not | <?php  if(is\_numeric("guru"))  {  echo "true";  }  else  {  echo "false";  }  ?> | false |
| <?php  if(is\_numeric (123))  {  echo "true";  }  else  {  echo "false";  }  ?> | true |
| number\_format | Used to formats a numeric value using digit separators and decimal points | <?php  echo number\_format(2509663);  ?> | 2,509,663 |
| rand | Used to generate a random number. | <?php  echo rand();  ?> | Random number |
| round | Round off a number with decimal points to the nearest whole number. | <?php  echo round(3.49);  ?> | 3 |
| sqrt | Returns the square root of a number | <?php  echo sqrt(100);  ?> | 10 |
| cos | Returns the cosine | <?php  echo cos(45);  ?> | 0.52532198881773 |
| sin | Returns the sine | <?php  echo sin(45);  ?> | 0.85090352453412 |
| tan | Returns the tangent | <?php  echo tan(45);  ?> | 1.6197751905439 |
| pi | Constant that returns the value of PI | <?php  echo pi();  ?> | 3.1415926535898 |

**Date Function**

The date function is used to format[Unix](https://www.guru99.com/unix-linux-tutorial.html)date and time to human readable format.

Check the article on PHP date functions for more details. Other functions

These include;

* Arrays – see the article on arrays for examples
* Files – see the article on files for examples
* Database functions – see the article on [MySQL PHP and other database access methods](https://www.guru99.com/mysql-php-and-other-database-access-methods.html) v2

**Why use User Defined Functions?**

User defined functions come in handy when;

* you have routine tasks in your application such as adding data to the database
* performing validation checks on the data
* Authenticating users in the system etc.

These activities will be spread across a number of pages.

Creating a function that all these pages can be calling is one of the features that make PHP a powerful scripting language.

Before we create our first user defined function, let’s look at the rules that we must follow when creating our own functions.

* Function names must start with a letter or an underscore but not a number
* The function name must be unique
* The function name must not contain spaces
* It is considered a good practice to use descriptive function names.
* Functions can optionally accept parameters and return values too.

Let’s now create our first function. We will create a very basic function that illustrates the major components of a function in PHP.

<?php

//define a function that displays hello function

function add\_numbers(){

echo 1 + 2;

}

add\_numbers ();

?>

**Output:**

3

  HERE,

* “function…(){…}”  is the function block that tells PHP that you are defining a custom function
* “add\_numbers” is the function name that will be called when using the function.
* “()” can be used to pass parameters to the function.
* “echo 'Hello function!';” is the function block of code that is executed. It could be any code other than the one used in the above example.

Let’s now look at a fairly complex example that accepts a parameter and display a message just like the above function.

Suppose we want to write a function that prints the user name on the screen, we can write a custom function that accepts the user name and displays it on the screen.

**The code below shows the implementation.**

<?php

function display\_name($name)

{

echo "Hello " . $name;

}

display\_name("Martin Luther King");

?>

**Output:**

Hello Martin Luther King

  HERE,

* “…($name){…” is the function parameter called name and is initialized to nameless. If no parameter is passed to the function, nameless will be displayed as the name. This comes in handy if not supplying any parameter to the function can result in unexpected errors.

Let’s now look at a function that accepts a parameter and then returns a value. We will create a function that converts kilometers to miles. The kilometers will be passed as a parameter. The function will return the miles equivalent to the passed kilometers. The code below shows the implementation.

<?php

function kilometers\_to\_miles($kilometers = 0)

{

$miles\_scale = 0.62;

return $kilometers \* $miles\_scale;

}

echo kilometers\_to\_miles(100);

?>

**Output:**

62

**Summary**

* Define function in PHP: Function is a block of code that performs specific task.
* Built in function in PHP is a function that is shipped with PHP
* PHP has over 700 built in functions
* String functions manipulate string data
* Numeric functions manipulate numeric data
* Date functions manipulate date data
* Other functions such as is\_array, fopen etc. are used to manipulate arrays and files respectively
* User defined functions are functions that you can create yourself to enhance PHP

**PHP Session & PHP Cookies with Example**

**What is Cookie?**

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer.

Once a cookie has been set, all page requests that follow return the cookie name and value.

A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain [www.guru99.com](https://www.guru99.com/) can not be read from the domain [career.guru99.com](https://career.guru99.com/).

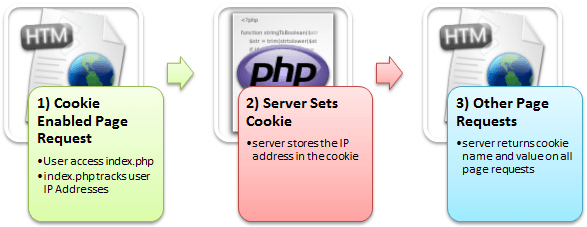
Most of the websites on the internet display elements from other domains such as advertising. The domains serving these elements can also set their own cookies. These are known as third party cookies.

A cookie created by a user can only be visible to them. Other users cannot see its value.

Most web browsers have options for disabling cookies, third party cookies or both.

If this is the case then PHP responds by passing the cookie token in the URL.

**The diagram shown below illustrates how cookies work.**

[](https://www.guru99.com/images/2013/04/how_cookies_work.png)

Here,

1) A user requests for a page that stores cookies

2) The server sets the cookie on the user’s computer

3) Other page requests from the user will return the cookie name and value

**In this tutorial, you will learn-**

* [Why and when to use Cookies?](https://www.guru99.com/cookies-and-sessions.html#3)
* [Creating Cookies](https://www.guru99.com/cookies-and-sessions.html#4)
* [Retrieving the Cookie value](https://www.guru99.com/cookies-and-sessions.html#5)
* [Delete Cookies](https://www.guru99.com/cookies-and-sessions.html#6)
* [What is a Session?](https://www.guru99.com/cookies-and-sessions.html#7)
* [Why and when to use Sessions?](https://www.guru99.com/cookies-and-sessions.html#8)
* [Creating a Session](https://www.guru99.com/cookies-and-sessions.html#9)
* [Destroying Session Variables](https://www.guru99.com/cookies-and-sessions.html#10)

**Why and when to use Cookies?**

* Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user’s computer.

The path were the cookies are stored depends on the browser.

Internet Explorer usually stores them in Temporal Internet Files folder.

* Personalizing the user experience – this is achieved by allowing users to select their preferences.

The page requested that follow are personalized based on the set preferences in the cookies.

* Tracking the pages visited by a user

**Creating Cookies**

Let’s now look at the basic syntax used to create a cookie.

<?php

setcookie(cookie\_name, cookie\_value, [expiry\_time], [cookie\_path], [domain], [secure], [httponly]);

?>

HERE,

* Php“setcookie” is the PHP function used to create the cookie.
* “cookie\_name” is the name of the cookie that the server will use when retrieving its value from the $\_COOKIE array variable. It’s mandatory.
* “cookie\_value” is the value of the cookie and its mandatory
* “[expiry\_time]” is optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour.
* “[cookie\_path]” is optional; it can be used to set the cookie path on the server. The forward slash “/” means that the cookie will be made available on the entire domain. Sub directories limit the cookie access to the subdomain.
* “[domain]” is optional, it can be used to define the cookie access hierarchy i.e. [www.cookiedomain](http://www.cookiedomain/).com means entire domain while [www.sub.cookiedomain.com](http://www.sub.cookiedomain.com/) limits the cookie access to [www.sub.cookiedomain.com](http://www.sub.cookiedomain.com/) and its sub domains. *Note it’s possible to have a subdomain of a subdomain as long as the total characters do not exceed 253 characters.*
* “[secure]” is optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.
* “[Httponly]” is optional. If it is set to true, then only client side scripting languages i.e.[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)cannot access them.

*Note: the php set cookie function must be executed before the HTML opening tag.*

Let’s now look at an example that uses cookies.

We will create a basic program that allows us to store the user name in a cookie that expires after  ten seconds.

The code below shows the implementation of the above example “cookies.php”.

<?php

setcookie("user\_name", "Guru99", time()+ 60,'/'); // expires after 60 seconds

echo 'the cookie has been set for 60 seconds';

?>

**Output:**

the cookie has been set for 60 seconds

**Retrieving the Cookie value**

Create another file named “cookies\_read.php” with the following code.

<?php

print\_r($\_COOKIE); //output the contents of the cookie array variable

?>

**Output:**

Array ( [PHPSESSID] => h5onbf7pctbr0t68adugdp2611 [user\_name] => Guru99 )

  Note: $\_COOKIE is a PHP built in super global variable.

It contains the names and values of all the set cookies.

The number of values that the

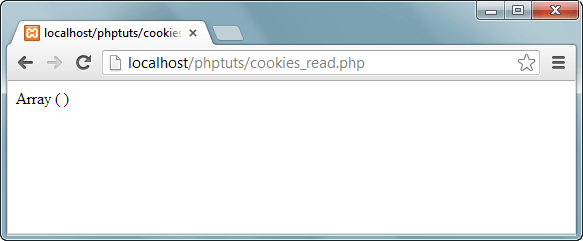
$\_COOKIE array can contain depends on the memory size set in php.ini.

The default value is 1GB.

Testing our application.

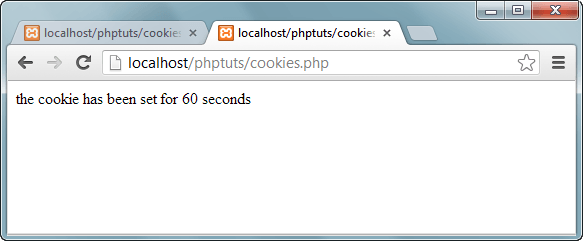
Let’s assume you have saved your PHP files in phptus folder.

* Step 1 – open your web browser and enter the URL **http://localhost/phptuts/cookies\_read.php**

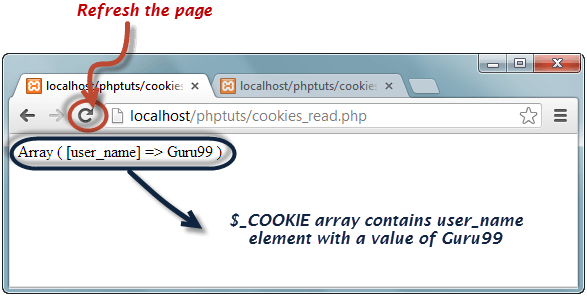
[](https://www.guru99.com/images/2013/04/cookie_empty_array.png)

*Note: Only an empty array has been displayed*

* Step 2 – Browser to the URL **http://localhost/phptuts/cookies.php**

[](https://www.guru99.com/images/2013/04/cookie_60_seconds.png)

* Step 3 – Switch back to the first tab then click on refresh button

[](https://www.guru99.com/images/2013/04/cookie_array.png)

Wait for a minute then click on refresh button again. What results did you get?

**Delete Cookies**

* If you want to destroy a cookie before its expiry time, then you set the expiry time to a time that has already passed.
* Create a new filed named cookie\_destroy.php with the following code

<?php

setcookie("user\_name", "Guru99", time() - 360,'/');

?>

* Repeat steps 1 through to 3 from the above section on retrieving cookie values.
* Open the URL **http://localhost/phptuts/cookie\_destroy.php**
* Switch to the URL **http://localhost/phptuts/cookies\_read.php** what results does it display?

**What is a Session?**

* A session is a global variable stored on the server.
* Each session is assigned a unique id which is used to retrieve stored values.
* Whenever a session is created, a cookie containing the unique session id is stored on the user’s computer and returned with every request to the server.  If the client browser does not support cookies, the unique php session id is displayed in the URL
* Sessions have the capacity to store relatively large data compared to cookies.
* The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
* Just like the $\_COOKIE array variable, session variables are stored in the $\_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

**Why and when to use Sessions?**

* You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
* You want to pass values from one page to another.
* You want the alternative to cookies on browsers that do not support cookies.
* You want to store global variables in an efficient and more secure way compared to passing them in the URL
* You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

**Creating a Session**

In order to  create a session, you must first call the PHP session\_start function and then store your values in the $\_SESSION array variable.

 Let’s suppose we want to know the number of times that a page has been loaded, we can use a session to do that.

The code below shows how to create and retrieve values from sessions

<?php

session\_start(); //start the PHP\_session function

if(isset($\_SESSION['page\_count']))

{

$\_SESSION['page\_count'] += 1;

}

else

{

$\_SESSION['page\_count'] = 1;

}

echo 'You are visitor number ' . $\_SESSION['page\_count'];

?>

**Output:**

You are visitor number 1

**Destroying Session Variables**

The session\_destroy() function is used to destroy the whole Php session variables.

If you want to destroy only a session single item, you use the unset() function.

The code below illustrates how to use both methods.

<?php

session\_destroy(); //destroy entire session

?>

<?php

unset($\_SESSION['product']); //destroy product session item

?>

  Session\_destroy removes all the session data including cookies associated with the session.

Unset only frees the individual session variables.

Other data remains intact.

**Summary**

* Cookies are small files saved on the user’s computer
* Cookies can only be read from the issuing domain
* Cookies can have an expiry time, if it is not set, then the cookie expires when the browser is closed
* Sessions are like global variables stored on the server
* Each session is given a unique identification id that is used to track the variables for a user.
* Both cookies and sessions must be started before any HTML tags have been sent to the browser.

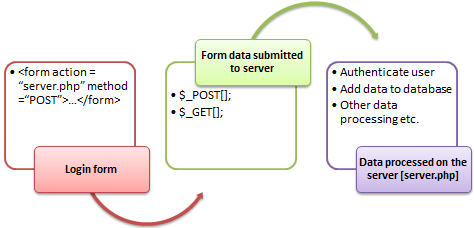
# PHP Registration Form using GET, POST Methods with Example

## What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

 The diagram below illustrates the form handling process.

[](https://www.guru99.com/images/2013/04/php_forms.png)

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

**In this tutorial, you will learn-**

* [When and why we are using forms?](https://www.guru99.com/php-forms-handling.html#2)
* [Create a form](https://www.guru99.com/php-forms-handling.html#3)
* [POST method](https://www.guru99.com/php-forms-handling.html#5)
* [GET method](https://www.guru99.com/php-forms-handling.html#6)
* [GET vs POST Methods](https://www.guru99.com/php-forms-handling.html#7)
* [Processing the registration form data](https://www.guru99.com/php-forms-handling.html#8)
* [More examples](https://www.guru99.com/php-forms-handling.html#9)

## When and why we are using forms?

* Forms come in handy when developing flexible and dynamic applications that accept user input.
* Forms can be used to edit already existing data from the database

## Create a form

We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

* Opening and closing form tags <form>…</form>
* Form submission type POST or GET
* Submission URL that will process the submitted data
* Input fields such as input boxes, text areas, buttons,checkboxes etc.

**The code below creates a simple registration form**

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST"> First name:

<input type="text" name="firstname"> <br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

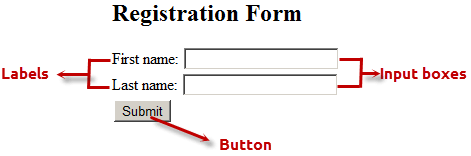
<input type="submit" value="Submit">

</form>

</body>

</html>

Viewing the above code in a web browser displays the following form.

[](https://www.guru99.com/images/2013/04/html_form.png)

HERE,

* <form…>…</form> are the opening and closing form tags
* action="registration\_form.php" method="POST"> specifies the destination URL and the submission type.
* First/Last name: are labels for the input boxes
* <input type=”text”…> are input box tags
* <br> is the new line tag
* <input type="hidden" name="form\_submitted" value="1"/> is a hidden value that is used to check whether the form has been submitted or not
* <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

## Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

### PHP POST method

* This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
* The array variable can be accessed from any script in the program; it has a global scope.
* This method is ideal when you do not want to display the form post values in the URL.
* A good example of using post method is when submitting login details to the server.

**It has the following syntax.**

<?php

$\_POST['variable\_name'];

?>

  HERE,

* “$\_POST[…]” is the PHP array
* “'variable\_name'” is the URL variable name.

### PHP GET method

* This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
* The array variable can be accessed from any script in the program; it has a global scope.
* This method displays the form values in the URL.
* It’s ideal for search engine forms as it allows the users to book mark the results.

**It has the following syntax.**

<?php

$\_GET['variable\_name'];

?>

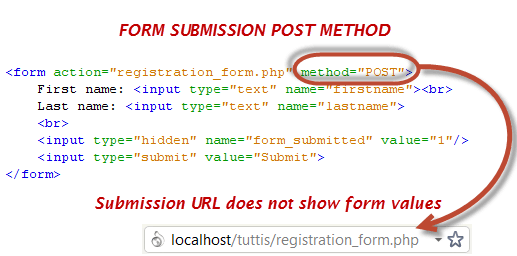
  HERE,

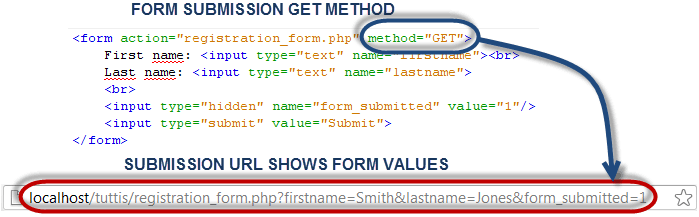
* “$\_GET[…]” is the PHP array
* “'variable\_name'” is the URL variable name.

## GET vs POST Methods

|  |  |
| --- | --- |
| **POST** | **GET** |
| Values not visible in the URL | Values visible in the URL |
| Has not limitation of the length of the values since they are submitted via the body of HTTP | Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser. |
| Has lower performance compared to Php\_GET method due to time spent encapsulation the Php\_POST values in the HTTP body | Has high performance compared to POST method dues to the simple nature of appending the values in the URL. |
| Supports many different data types such as string, numeric, binary etc. | Supports only string data types because the values are displayed in the URL |
| Results cannot be book marked | Results can be book marked due to the visibility of the values in the URL |

**The below diagram shows the difference between get and post**

[](https://www.guru99.com/images/2013/04/post_form_submission.png)

[](https://www.guru99.com/images/2013/04/get_form_submission.png)

## Processing the registration form data

The registration form submits data to itself as specified in the action attribute of the form.

 When a form has been submitted, the values are populated in the $\_POST super global array.

We will use the PHP isset function to check if the form values have been filled in the $\_POST array and process the data.

We will modify the registration form to include the PHP code that processes the data. Below is the modified code

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<?php if (isset($\_POST['form\_submitted'])): ?> //this code is executed when the form is submitted

<h2>Thank You <?php echo $\_POST['firstname']; ?> </h2>

<p>You have been registered as

<?php echo $\_POST['firstname'] . ' ' . $\_POST['lastname']; ?>

</p>

<p>Go <a href="/registration\_form.php">back</a> to the form</p>

<?php else: ?>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST">

First name:

<input type="text" name="firstname">

<br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

<?php endif; ? >

</body>

</html>

  HERE,

* <?php if (isset($\_POST['form\_submitted'])): ?> checks if the form\_submitted hidden field has been filled in the $\_POST[] array and display a thank you and first name message.

If the form\_fobmitted field hasn’t been filled in the $\_POST[] array, the form is displayed.

## More examples

### Simple search engine

We will design a simple search engine that uses the PHP\_GET method as the form submission type.

For simplicity’s sake, we will use a PHP If statement to determine the output.

We will use the same HTML code for the registration form above and make minimal modifications to it.

<html>

<head>

<title>Simple Search Engine</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<?php if (isset($\_GET['form\_submitted'])): ?>

<h2>Search Results For <?php echo $\_GET['search\_term']; ?> </h2>

<?php if ($\_GET['search\_term'] == "GET"): ?>

<p>The GET method displays its values in the URL</p>

<?php else: ?>

<p>Sorry, no matches found for your search term</p>

<?php endif; ?>

<p>Go <a href="/search\_engine.php">back</a> to the form</p>

<?php else: ?>

<h2>Simple Search Engine - Type in GET </h2>

<form action="search\_engine.php" method="GET">

Search Term:

<input type="text" name="search\_term">

<br>

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

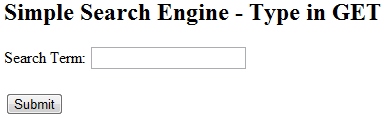
<?php endif; ?>

</body>

</html>

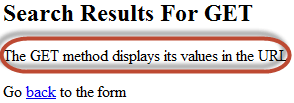
View the above page in a web browser

**The following form will be shown**

[](https://www.guru99.com/images/2013/04/simple_search_engine.png)

Type GET in upper case letter then click on submit button.

The following will be shown

[](https://www.guru99.com/images/2013/04/search_engine_results.png)

The diagram below shows the URL for the above results

[PHP Form](https://www.guru99.com/images/2013/04/search_engine_url.png)

*Note the URL has displayed the value of search\_term and form\_submitted.* Try to enter anything different from GET then click on submit button and see what results you will get.

### Working with check boxes, radio buttons

If the user does not select a check box or radio button, no value is submitted, if the user selects a check box or radio button, the value one (1) or true is submitted.

We will modify the registration form code and include a check button that allows the user to agree to the terms of service.

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<?php if (isset($\_POST['form\_submitted'])): ?>

<?php if (!isset($\_POST['agree'])): ?>

<p>You have not accepted our terms of service</p>

<?php else: ?>

<h2>Thank You <?php echo $\_POST['firstname']; ?></h2>

<p>You have been registered as

<?php echo $\_POST['firstname'] . ' ' . $\_POST['lastname']; ?>

</p>

<p> Go <a href="/registration\_form2.php">back</a> to the form</p>

<?php endif; ?>

<?php else: ?>

<h2>Registration Form</h2>

<form action="registration\_form2.php" method="POST">

First name:

<input type="text" name="firstname">

<br> Last name:

<input type="text" name="lastname">

<br> Agree to Terms of Service:

<input type="checkbox" name="agree">

<br>

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

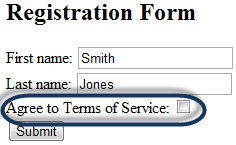
</form>

<?php endif; ?>

</body>

</html>

**View the above form in a browser**

[](https://www.guru99.com/images/2013/04/registration_form.png)

Fill in the first and last names

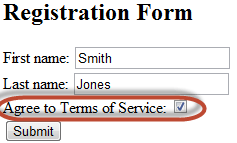
*Note the Agree to Terms of Service checkbox has not been selected.*

Click on submit button

**You will get the following results**

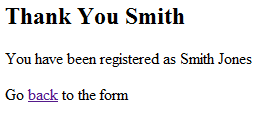
[PHP Form](https://www.guru99.com/images/2013/04/radio_button_false.png)

Click on back to the form link and then select the checkbox

[](https://www.guru99.com/images/2013/04/registration_form2.png)

Click on submit button

**You will get the following results**

[](https://www.guru99.com/images/2013/04/radio_button_true.png)

## Summary

* Forms are used to get data from the users
* Forms are created using HTML tags
* Forms can be submitted to the server for processing using either POST or GET method
* Form values submitted via the POST method are encapsulated in the HTTP body.
* Form values submitted via the GET method are appended and displayed in the URL.

**PHP File() Function: File\_exists, Fopen, Fwrite, Fclose, Fgets, copy, unlink**

**What is a File?**

A file is simply a resource for storing information on a computer.

Files are usually used to store information such as;

* Configuration settings of a program
* Simple data such as contact names against the phone numbers.
* Images, Pictures, Photos, etc.

**In this tutorial, you will learn-**

* [PHP File Formats Support](https://www.guru99.com/php-file-processing.html#1)
* [PHP files Functions](https://www.guru99.com/php-file-processing.html#2)
* [PHP File\_exists Function](https://www.guru99.com/php-file-processing.html#3)
* [PHP Fopen Function](https://www.guru99.com/php-file-processing.html#4)
* [PHP Fwrite Function](https://www.guru99.com/php-file-processing.html#5)
* [PHP Fclose Function](https://www.guru99.com/php-file-processing.html#6)
* [PHP Fgets Function](https://www.guru99.com/php-file-processing.html#7)
* [PHP Copy Function](https://www.guru99.com/php-file-processing.html#8)
* [Deleting a file](https://www.guru99.com/php-file-processing.html#9)
* [PHP File\_get\_contents Function](https://www.guru99.com/php-file-processing.html#10)

**PHP File Formats Support**

PHP file functions support a wide range of file formats that include;

* File.txt
* File.log
* File.custom\_extension i.e. file.xyz
* File.csv
* File.gif, file.jpg etc
* Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
* You want to store simple data such as server logs for later retrieval and analysis
* You want to store program settings i.e. program.ini

**PHP files Functions**

PHP provides a convenient way of working with files via its rich collection of built in functions.

Operating systems such as Windows and MAC OS are not case sensitive while[Linux](https://www.guru99.com/unix-linux-tutorial.html)or[Unix](https://www.guru99.com/unix-linux-tutorial.html)operating systems are case sensitive.

Adopting a naming conversion such as lower case letters only for file naming is a good practice that ensures maximum cross platform compatibility.

Let’s now look at some of the most commonly used PHP file functions.

**PHP File\_exists Function**

This function is used to determine whether a file exists or not.

* It comes in handy when we want to know if a file exists or not before processing it.
* You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file\_exist function has the following syntax.

<?php

file\_exists($filename);

?>

HERE,

* “file\_exists()” is the PHP function that returns true if the file exists and false if it does not exist.
* “$file\_name” is the path and name of the file to be checked

The code below uses file\_exists function to determine if the file my\_settings.txt exists.

<?php

if (file\_exists('my\_settings.txt'))

{

echo 'file found!';

}

else

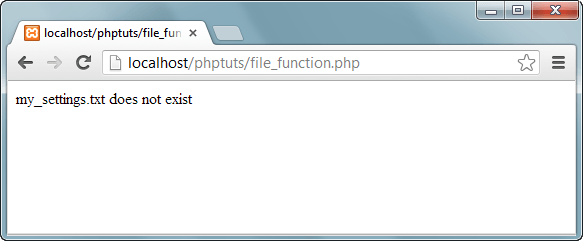
{

echo 'my\_settings.txt does not exist';

}

?>

  Save the above code in a file named file\_function.php Assuming you saved the file in phptuts folder in htdocs, open the URL **http://localhost/phptuts/file\_function.php** in your browser You will get the following results.

[](https://www.guru99.com/images/2013/04/file_exists.png)

**PHP Fopen Function**

The fopen function is used to open files. It has the following syntax

<?php

fopen($file\_name,$mode,$use\_include\_path,$context);

?>

  HERE,

* “fopen” is the PHP open file function
* “$file\_name” is the name of the file to be opened
* “$mode” is the mode in which the file should be opened, the table below shows the modes

|  |  |
| --- | --- |
| **Mode** | **Description** |
| r | * Read file from beginning. * Returns false if the file doesn’t exist. * Read only |
| r+ | * Read file from beginning * Returns false if the file doesn’t exist. * Read and write |
| w | * Write to file at beginning * truncate file to zero length * If the file doesn’t exist attempt to create it. * Write only |
| w+ | * Write to file at beginning, truncate file to zero length * If the file doesn’t exist attempt to create it. * Read and Write |
| a | * Append to file at end * If the file doesn’t exist attempt to create it. * Write only |
| a+ | * Php append to file at end * If the file doesn’t exist attempt to create it * Read and write |

* “$use\_include\_path” is optional, default is false, if set to true, the function searches in the include path too.
* “$context” is optional, can be used to specify the context support.

**PHP Fwrite Function**

The fwrite function is used to write files.

It has the following syntax

<?php

fwrite($handle, $string, $length);

?>

  HERE,

* “fwrite” is the PHP function for writing to files
* “$handle” is the file pointer resource
* “$string” is the data to be written in the file.
* “$length” is optional, can be used to specify the maximum file length.

**PHP Fclose Function**

Is is used to close a file in php which is already open

It has the following syntax.

<?php

fclose($handle);

?>

  HERE,

* “fclose” is the PHP function for closing an open file
* “$handle” is the file pointer resource.

Let’s now look at an example that creates my\_settings.txt.

We will use the following functions.

* Fopen
* Fwrite
* fclose

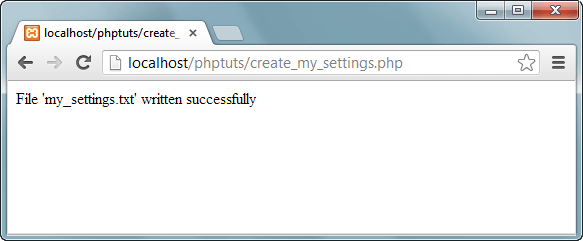
The code below “create\_my\_settings\_file.php” implements the above example.

|  |  |
| --- | --- |
| Open a file | <?php  $fh = fopen("my\_settings.txt", 'w')  or  die("Failed to create file"); ?> |
| Closing a file | <?php  fclose($fh);  ?> |
| Create File | <?php  $fh = fopen("my\_settings.txt", 'w') or die("Failed to create file");  $text = <<<\_END  localhost;root;pwd1234;my\_database  \_END;  fwrite($fh, $text) or die("Could not write to file");  fclose($fh);  echo "File 'my\_settings.txt' written successfully"; ?> |

**Testing the code**

Open the URL **http://localhost/phptuts/create\_my\_settings.php** in your browser.

You will get the following page

. [](https://www.guru99.com/images/2013/04/write_file.png)

Note: if your disk is full or you do not have permission to write files, you will get an error message.

Switch back to the URL **http://localhost/phptuts/file\_function.php** .

What results do you get?

**PHP Fgets Function**

The fgets function is used to read php files line by line. It has the following basic syntax. fgets($handle); HERE,

* “$fgets” is the PHP function for reading file lines
* “$handle” is the file pointer resource.

Let’s now look at an example that reads my\_settings.txt file using the fopen and fgets functions.

The code below read\_my\_settings.php implements the above example.

<?php

$fh = fopen("my\_settings.txt", 'r') or die("File does not exist or you lack permission to open it");

$line = fgets($fh);

echo $line; fclose($fh);

?>

  HERE,

* “fopen” function returns the pointer to the file specified in the file path
* “die()” function is called if an error occurs. It displays a message and exists execution of the script

**PHP Copy Function**

The PHP copy function is used to copy files. It has the following basic syntax. copy($file,$copied\_file); HERE,

* “$file” specifies the file path and name of the file to be copied.
* “copied\_file” specified the path and name of the copied file

The code below illustrates the implementation

<?php

copy('my\_settings.txt', 'my\_settings\_backup.txt') or die("Could not copy file");

echo "File successfully copied to 'my\_settings\_backup.txt'";

?>

**Deleting a file**

The unlink function is used to delete the file. The code below illustrates the implementation.

<?php

if (!unlink('my\_settings\_backup.txt'))

{

echo "Could not delete file";

}

else

{

echo "File 'my\_settings\_backup.txt' successfully deleted";

}

?>

**PHP File\_get\_contents Function**

The file\_get\_contents function is used to read the entire file contents.

The code below illustrates the implementation.

The difference between file\_get\_contents and fgets is that file\_get\_contents returns the file data as a string while fgets reads the file line by line.

<?php

echo "<pre>"; // Enables display of line feeds

echo file\_get\_contents("my\_settings.txt");

echo "</pre>"; // Terminates pre tag

?>

**Summary**

* A file is a resource for storing data
* PHP has a rich collection of built in functions that simplify working with files.
* Common file functions include fopen, fclose, file\_get\_contents
* The table below shows a summary of the functions covered

|  |  |
| --- | --- |
| **Function** | **Description** |
| File\_exists | Used to determine if a file exists or not |
| fopen | Used to open a file. Returns a pointer to the opened file |
| fwrite | Used to write to files |
| fclose | Used to open closed files |
| fgets | Used to read a file line by line |
| copy | Used to copy an existing file |
| unlink | Used to delete an existing file |
| file\_get\_contents | Used to return the contents of a file as a string |

**PHP Try Catch Example: Exception & Error Handling Tutorial**

**What is an Exception?**

An error is an unexpected program result that cannot be handled by the program itself.

Errors are resolved by fixing the program. An example of an error would be an infinite loop that never stops executing.

An exception is unexpected program result that can be handled by the program itself.

Examples of exception include trying to open a file that does not exist.

This exception can be handled by either creating the file or presenting the user with an option of searching for the file.

**In this tutorial, you will learn-**

* [Why handle exception?](https://www.guru99.com/error-handling-and-exceptions.html#1)
* [PHP Error handling](https://www.guru99.com/error-handling-and-exceptions.html#2)
* [Error handling examples](https://www.guru99.com/error-handling-and-exceptions.html#3)
* [Difference between Errors and Exception](https://www.guru99.com/error-handling-and-exceptions.html#4)
* [Multiple Exceptions](https://www.guru99.com/error-handling-and-exceptions.html#5)
* [Testing the code](https://www.guru99.com/error-handling-and-exceptions.html#6)

**Why handle exception?**

* Avoid unexpected results on our pages which can be very annoying or irritating to our end users
* Improve the security of our applications by not exposing information which malicious users may use to attack our applications
* Php Exceptions are used to change the normal flow of a program if any predictable error occurs.

**PHP Error handling**

When an error occurs, depending on your configuration settings, PHP displays the error message in the web browser with information relating to the error that occurred.

  PHP offers a number of ways to handle errors.

We are going to look at three (3) commonly used methods;

1. **Die statements**– the die function combines the echo and exit function in one. It is very useful when we want to output a message and stop the script execution when an error occurs.

1. **Custom error handlers** – these are user defined functions that are called whenever an error occurs.

1. **PHP error reporting** – the error message depending on your PHP error reporting settings. This method is very useful in development environment when you have no idea what caused the error. The information displayed can help you debug your application.

**Error handling examples**

Let’s now look at some simple examples with error handling routines.

Let’s suppose that we have developed an application that uses text files to store data. We might want to check for the file’s existence before we attempt to read data from it.

**The code below implements the above example.**

<?php

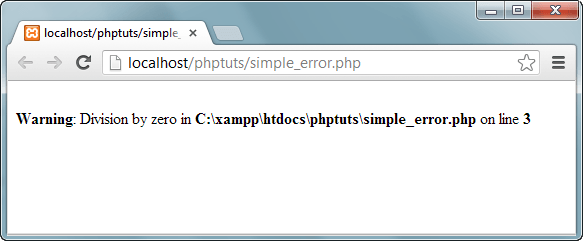
$denominator = 0;

echo 2 / $denominator;

?>

Assuming you saved the file simple\_error.php in phptuts folder, open the URL **http://localhost/phptuts/simple\_error.php**

You will get the following results

[](https://www.guru99.com/images/2013/04/error.png)

As you can see from the above results, it makes our application look unprofessional and can be annoying to the user.

We will modify the above code and write an error handler for the application

<?php

$denominator = 0;

if ($denominator != 0) {

echo 2 / $denominator;

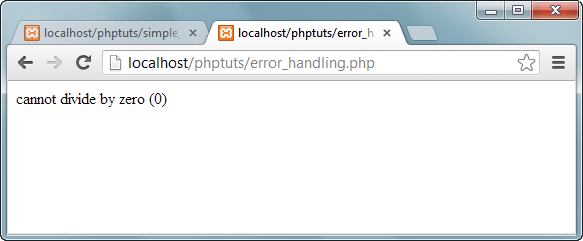
} else {

echo "cannot divide by zero (0)";

}

?>

Assuming you saved the above code as error\_handling.php, open the URL **http://localhost/phptuts/error\_handling.php**

[](https://www.guru99.com/images/2013/04/division_by_zero_error.png)

*Note: it’s a good security practice to display a message as the one shown above instead of showing the message like “File not found”.*

Let’s look at another example that uses a custom error handler.

The custom error handler will be set as the default PHP error handling function and will basically display an error number and message.

The code below illustrates the implementation of the above example

<?php

function my\_error\_handler($error\_no, $error\_msg)

{

echo "Opps, something went wrong:";

echo "Error number: [$error\_no]";

echo "Error Description: [$error\_msg]";

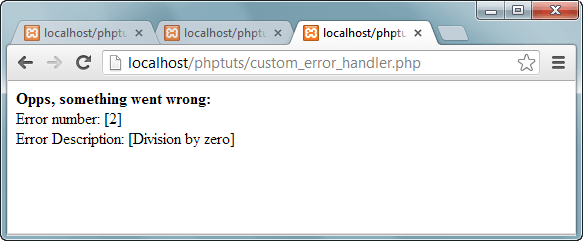
}

set\_error\_handler("my\_error\_handler");

echo (5 / 0);

?>

  Open the URL **http://localhost/phptuts/custom\_error\_handler.php** you will get the following results

. [](https://www.guru99.com/images/2013/04/custom_error_handler.png)

As you can see from the above example, custom error handlers are powerful in the sense that

* They allow us to customize the error messages.
* The custom error handler can also include error logging in a file/database, emailing the developer etc.

Let’s now look at the third type of error handling. We will be using the PHP built in function error\_reporting function. It has the following basic syntax

<?php

error\_reporting($reporting\_level);

?>

  HERE,

* “error\_reporting” is the PHP error reporting function
* “$reporting\_level" is optional, can be used to set the reporting level. If no reporting level has been specified, PHP will use the default error reporting level as specified in the php.ini file.

| **Reporting Level** | **Description** | **Example** |
| --- | --- | --- |
| E\_WARNING | Displays warning messages only. Does not halt the execution of the script | error\_reporting(E\_WARNING); |
| E\_NOTICE | Displays notices that can occur during normal execution of a program or could be an error. | error\_reporting(E\_ NOTICE); |
| E\_USER\_ERROR | Displays user generated errors i.e. custom error handler | error\_reporting(E\_ USER\_ERROR); |
| E\_USER\_WARNING | Displays user generated warning messages | error\_reporting(E\_USER\_WARNING); |
| E\_USER\_NOTICE | Displays user generated notices | error\_reporting(E\_USER\_NOTICE); |
| E\_RECOVERABLE\_ERROR | Displays error that are not fatal and can be handled using custom error handlers | error\_reporting(E\_RECOVERABLE\_ERROR); |
| E\_ALL | Displays all errors and warnings | error\_reporting(E\_ ALL); |

**Difference between Errors and Exception**

* Exceptions are thrown and intended to be caught while errors are generally irrecoverable.
* Exceptions are handled in an object oriented way.

This means when an exception is thrown; an exception object is created that contains the exception details.

The table below shows the exception object methods

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| getMessage() | Displays the exception’s message | <?php  echo $e->getMessage();  ?> |
| getCode() | Displays the numeric code that represents the exception | <?php  echo $e->getCode();  ?> |
| getFile() | Displays the file name and path where the exception occurred | <?php  echo $e->getFile();  ?> |
| getLine() | Displays the line number where the exception occurred | <?php  echo $e->getLine();  ?> |
| getTrace() | Displays an array of the backtrace before the exception | <?php  print\_r( $e->getTrace());  ?> |
| getPrevious() | Displays the previous exception before the current one | <?php  echo $e->getPrevious();  ?> |
| getTraceAsString() | Displays the backtrace of the exception as a string instead of an array | <?php  echo $e->getTraceAsString();  ?> |
| \_\_toString() | Displays the entire exception as a string | <?php  echo $e->\_\_toString();  ?> |

  Below is the basic syntax for throwing an exception.

<?php

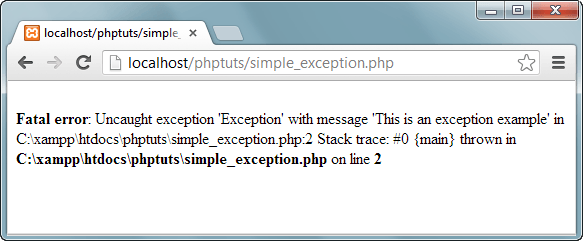
throw new Exception("This is an exception example");

?>

HERE,

* “throw” is the keyword used to throw the exception
* “new Exception(…)” creates an exception object and passes “This is an exception example “ string as the message parameter.

The above code outputs the following message.

[](https://www.guru99.com/images/2013/04/uncaught_exception.png)

We are now going to look at an example that implements the throw and catch exceptions.

We will modify the above example and include the try, throw and catch.

It has the following basic syntax.

<?php

try {

//code goes here that could potentially throw an exception

}

catch (Exception $e) {

//exception handling code goes here

}

?>

  HERE,

* “try{…}” is the block of code to be executed that could potentially raise an exception
* “catch(Exception $e){…}” is the block of code that catches the thrown exception and assigns the exception object to the variable $e.

The code below shows the basic exception example with the try, throw and catch exception implemented.

The program deliberately throws an exception which it then catches.

<?php

try {

$var\_msg = "This is an exception example";

throw new Exception($var\_msg);

}

catch (Exception $e) {

echo "Message: " . $e->getMessage();

echo "";

echo "getCode(): " . $e->getCode();

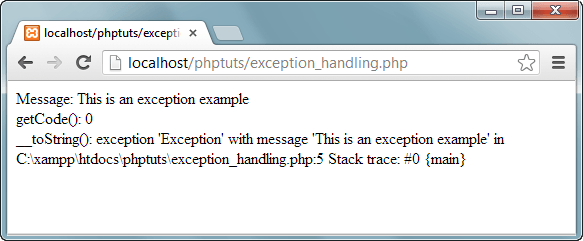
echo "";

echo "\_\_toString(): " . $e->\_\_toString();

}

?>

  Open the URL **http://localhost/phptuts/exception\_handling.php** You will get the following results.

[](https://www.guru99.com/images/2013/04/exception_info.png)

It’s also possible to create multiple exceptions for one php try statement depending on the type of exception thrown.

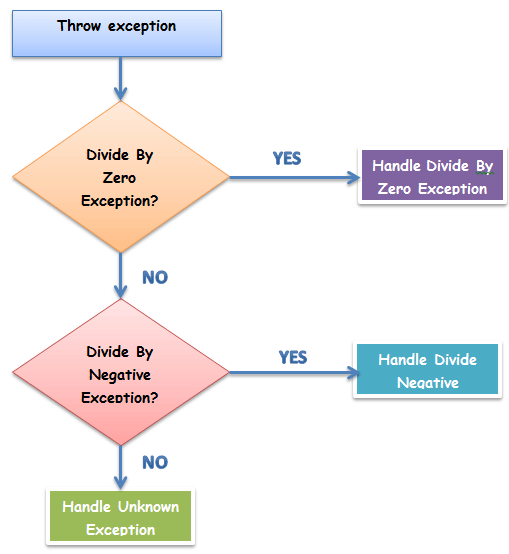
See the article on MySQL, PHP data access… for implementation examples of multiple exceptions

**Multiple Exceptions**

Multiple exception use multiple try catch blocks to handle the thrown exceptions. Multiple exceptions are useful when;

* You want to display a customized message depending on the exception thrown
* You want to perform a unique operation depending on the exception thrown

The flowchart below illustrates the how multiple exceptions work

[](https://www.guru99.com/images/2013/04/exception_flowchart.png)

Let’s look at an example that uses multiple exceptions.

We will modify the code that divides a number by the passed in denominator.

We expect two types of exceptions to occur;

* Division by zero
* Division by a negative number

For simplicity’s sake, we will only display the exception type in our catch blocks.

The PHP built in Exception class is used to throw exceptions.

We will create two classes that extend the exception class and use them to throw exceptions.

The code below shows the implementation.

<?php

class DivideByZeroException extends Exception {};

class DivideByNegativeException extends Exception {};

function process($denominator)

{

try

{

if ($denominator == 0)

{

throw new DivideByZeroException();

}

else if ($denominator < 0)

{

throw new DivideByNegativeException();

}

else

{

echo 25 / $denominator;

}

}

catch (DivideByZeroException $ex)

{

echo "DIVIDE BY ZERO EXCEPTION!";

}

catch (DivideByNegativeException $ex)

{

echo "DIVIDE BY NEGATIVE NUMBER EXCEPTION!";

}

catch (Exception $x)

{

echo "UNKNOWN EXCEPTION!";

}

}

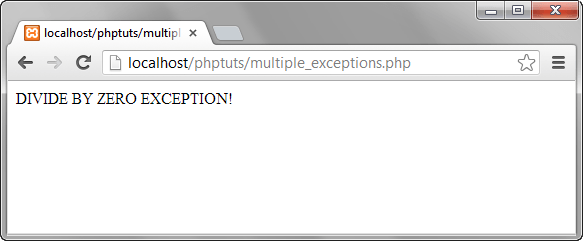
process(0);

?>

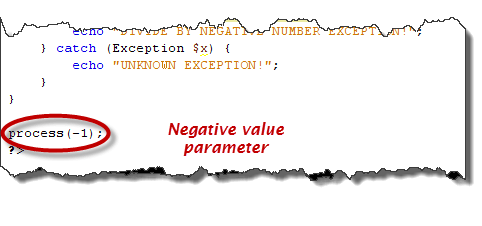
**Testing the code**

We will assume you saved multiple\_exceptions.php in phptuts folder.

Browse to the URL **http://localhost/phptuts/multiple\_exceptions.php**

[](https://www.guru99.com/images/2013/04/division_by_zero_exception.png)

Switch back to the PHP file and pass -1 as the parameter as shown in the following diagram.

[](https://www.guru99.com/images/2013/04/negative_value_exception.png)

Browse to the URL **http://localhost/phptuts/multiple\_exceptions.php**.

 What results do you get? Pass 3 as the parameter.

What results do you get?

**Summary**

* Errors are unexpected results produced by PHP code
* Error handling improves the application performance
* PHP has built in functions that can be used to customize the way PHP reports errors
* Exceptions are like errors, but they can be caught using the catch block when thrown.
* Displaying error messages that show error information is considered a bad security practice.

**PHP Regular Expression/Regex: preg\_match() | preg\_replace()**

**What is Regular expression in PHP?**

**PHP Regular Expression** also known as regex are powerful pattern matching algorithm that can be performed in a single expression. Regular expressions use arithmetic operators such as (+,-,^) to create complex expressions. They can help you accomplish tasks such as validating email addresses, IP address etc.

**Why use regular expressions**

* PHP Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time.
* When validating user input such as email address, domain names, telephone numbers, IP addresses,
* Highlighting keywords in search results
* When creating a custom HTML template. Regex in PHP can be used to identify the template tags and replace them with actual data.

In this PHP Regex tutorial, you will learn:

* [Built-in Regular expression Functions in PHP](https://www.guru99.com/php-regular-expressions.html#1)
* [PHP Preg\_match()](https://www.guru99.com/php-regular-expressions.html#2)
* [PHP Preg\_split()](https://www.guru99.com/php-regular-expressions.html#3)
* [PHP Preg\_replace()](https://www.guru99.com/php-regular-expressions.html#4)
* [Regular Expression Metacharacters](https://www.guru99.com/php-regular-expressions.html#5)
* [Explaining the pattern](https://www.guru99.com/php-regular-expressions.html#6)

**Built-in Regular expression Functions in PHP**

PHP has built in functions that allow us to work with regular functions which we will learn in this PHP Regular Expressions tutorial. Let’s look at the commonly used regular expression functions in PHP.

* preg\_match() in PHP – this function is used to perform pattern matching in PHP on a string. It returns true if a match is found and false if a match is not found.
* preg\_split() in PHP – this function is used to perform a pattern match on a string and then split the results into a numeric array
* preg\_replace() in PHP – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as PHP preg\_match(), PHP preg\_split() or PHP preg\_replace().

<?php

function\_name('/pattern/',subject);

?>

HERE,

* "function\_name(...)" is either PHP preg\_match(), PHP preg\_split() or PHP preg\_replace().
* "/.../" The forward slashes denote the beginning and end of our PHP regex tester function
* "'/pattern/'" is the pattern that we need to matched
* "subject" is the text string to be matched against

Let’s now look at practical examples that implement the above PHP regex functions.

**PHP Preg\_match()**

The first example uses the preg\_match() in PHP function to perform a simple pattern match for the word guru in a given URL.

The code below shows the implementation for preg\_match() tester function for the above example.

<?php

$my\_url = "www.guru99.com";

if (preg\_match("/guru/", $my\_url))

{

echo "the url $my\_url contains guru";

}

else

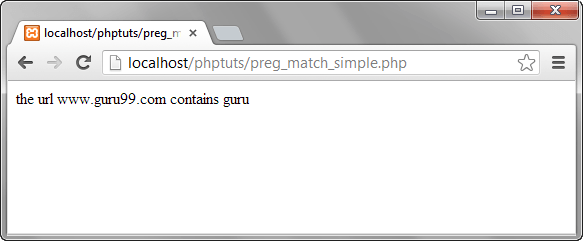
{

echo "the url $my\_url does not contain guru";

}

?>

  Browse to the URL **http://localhost/phptuts/preg\_match\_simple.php**

[](https://www.guru99.com/images/2013/04/preg_match.png)

Let’s examine the part of the code responsible for our output "*preg\_match('/guru/', $my\_url)"*   HERE,

* "preg\_match(...)" is the PHP regex function
* "'/guru/'" is the regular expression pattern to be matched
* "$my\_url" is the variable containing the text to be matched against.

The diagram below summarizes the above points

**PHP Preg\_split()**

Let’s now look at another example that uses the preg\_split() in PHP function.

We will take a string phrase and explode it into an array; the pattern to be matched is a single space.

The text string to be used in this example is "I Love Regular Expressions".

The code below illustrates the implementation of the above example.

<?php

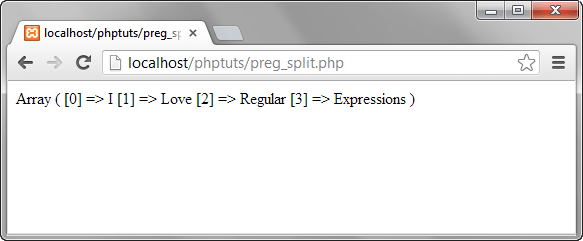
$my\_text="I Love Regular Expressions";

$my\_array = preg\_split("/ /", $my\_text);

print\_r($my\_array );

?>

  Browse to the URL **http://localhost/phptuts/preg\_split.php**

[](https://www.guru99.com/images/2013/04/preg_split.png)

**PHP Preg\_replace()**

Let’s now look at the preg\_replace() in PHP function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word guru in a string.

It replaces the word guru with the word guru surrounded by css code that highlights the background colour.

<?php

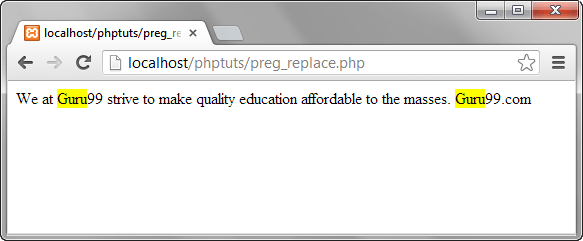
$text = "We at Guru99 strive to make quality education affordable to the masses. Guru99.com";

$text = preg\_replace("/Guru/", '<span style="background:yellow">Guru</span>', $text);

echo $text;

?>

  Assuming you have saved the file preg\_replace.php, browser to the URL **http://localhost/phptuts/preg\_replace.php**

[](https://www.guru99.com/images/2013/04/preg_replace.png)

**Regular Expression Metacharacters**

The above examples used very basic patterns; metacharacters simply allow us to perform more complex pattern matches such as test the validity of an email address. Let’s now look at the commonly used metacharacters.

| **Metacharacter** | **Description** | **Example** |
| --- | --- | --- |
| . | Matches any single character except a new line | /./ matches anything that has a single character |
| ^ | Matches the beginning of or string / excludes characters | /^PH/ matches any string that starts with PH |
| $ | Matches pattern at the end of the string | /com$/ matches guru99.com,yahoo.com Etc. |
| \* | Matches any zero (0) or more characters | /com\*/ matches computer, communication etc. |
| + | Requires preceding character(s) appear at least once | /yah+oo/ matches yahoo |
| \ | Used to escape meta characters | /yahoo+\.com/ treats the dot as a literal value |
| [...] | Character class | /[abc]/ matches abc |
| a-z | Matches lower case letters | /a-z/ matches cool, happy etc. |
| A-Z | Matches upper case letters | /A-Z/ matches WHAT, HOW, WHY etc. |
| 0-9 | Matches any number between 0 and 9 | /0-4/ matches 0,1,2,3,4 |

  The above list only gives the most commonly used metacharacters in regular expressions.

Let’s now look at a fairly complex example that checks the validity of an email address.

<?php

$my\_email = "[name@company.com](mailto:name@company.com)";

if (preg\_match("/^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/", $my\_email)) {

echo "$my\_email is a valid email address";

}

else

{

echo "$my\_email is NOT a valid email address";

}

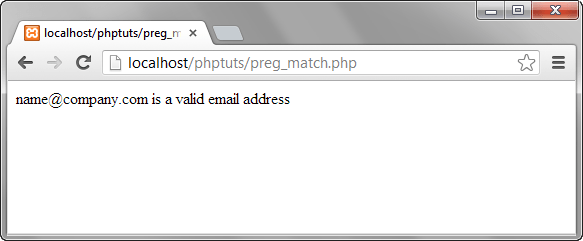
?>

**Explaining the pattern "[/^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/]"**

HERE,

* "'/.../'" starts and ends the regular expression
* "^[a-zA-Z0-9.\_-]" matches any lower or upper case letters, numbers between 0 and 9 and dots, underscores or dashes.
* "+@[a-zA-Z0-9-]" matches the @ symbol followed by lower or upper case letters, numbers between 0 and 9 or dashes.
* "+\.[a-zA-Z.]{2,5}$/" escapes the dot using the backslash then matches any lower or upper case letters with a character length between 2 and 5 at the end of the string.

Browse to the URL **http://localhost/phptuts/preg\_match.php**

[](https://www.guru99.com/images/2013/04/validate_email.png)

As you can see from the above example breakdown, metacharacters are very powerful when it comes to matching patterns.

**Summary**

* A Regular Expression or Regex in PHP is a pattern match algorithm
* Regular expressions are very useful when performing validation checks, creating HTML template systems that recognize tags etc.
* [PHP](https://www.guru99.com/php-tutorials.html) has built in functions namely PHP preg\_match(), PHP preg\_split() and PHP preg\_replace() that support regular expressions.
* Metacharacters allow us to create complex patterns

# How to Send Email using PHP mail() Function

## What is PHP mail?

PHP mail is the built in PHP function that is used to send emails from PHP scripts.

The mail function accepts the following parameters;

* Email address
* Subject
* Message
* CC or BC email addresses
  + It’s a cost effective way of notifying users on important events.
  + Let users contact you via email by providing a contact us form on the website that emails the provided content.
  + Developers can use it to receive system errors by email
  + You can use it to email your newsletter subscribers.
  + You can use it to send password reset links to users who forget their passwords
  + You can use it to email activation/confirmation links. This is useful when registering users and verifying their email addresses

In this tutorial, you will learn-

* [Why/When to use the PHP mail](https://www.guru99.com/php-mail.html#2)
* [Simple Mail Transmission Protocol](https://www.guru99.com/php-mail.html#3)
* [Sanitizing email user inputs](https://www.guru99.com/php-mail.html#4)
* [Secure Mail](https://www.guru99.com/php-mail.html#5)

## Why/When to use the mail PHP

### Sending mail using PHP

The PHP mail function has the following basic syntax

<?php

mail($to\_email\_address,$subject,$message,[$headers],[$parameters]);

?>

  HERE,

* “$to\_email\_address” is the email address of the mail recipient
* “$subject” is the email subject
* “$message” is the message to be sent.
* “[$headers]” is optional, it can be used to include information such as CC, BCC
  + CC is the acronym for carbon copy. It’s used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.
  + BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

## Simple Mail Transmission Protocol (SMTP)

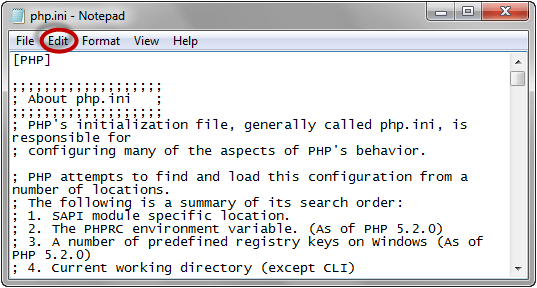
PHP mailer uses Simple Mail Transmission Protocol (SMTP) to send mail.

On a hosted server, the SMTP settings would have already been set.

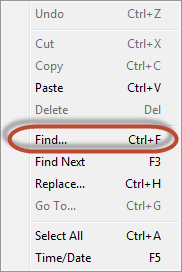
The SMTP mail settings can be configured from “php.ini” file in the PHP installation folder.

Configuring SMTP settings on your localhost Assuming you are using xampp on windows, locate the “php.ini” in the directory “C:\xampp\php”.

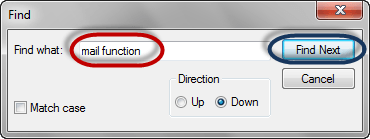
* Open it using notepad or any text editor. We will use notepad in this example. Click on the edit menu

[](https://www.guru99.com/images/2013/04/php_ini.png)

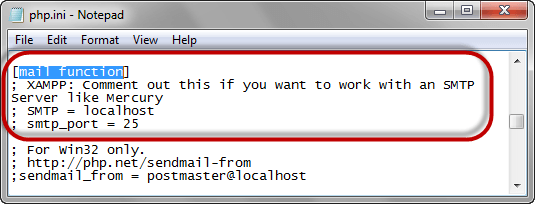
* Click on Find… menu

[](https://www.guru99.com/images/2013/04/find.png)

* The find dialog menu will appear

[](https://www.guru99.com/images/2013/04/find2.png)

* Click on Find Next button

[](https://www.guru99.com/images/2013/04/php_ini2.png)

* Locate the entries
  + *[mail function]*
  + *; XAMPP:* Don’t remove the semi column if you want to work with an SMTP Server like Mercury
  + ; SMTP = localhost
  + ; smtp\_port = 25
  + Remove the semi colons before SMTP and smtp\_port and set the SMTP to your smtp server and the port to your smtp port. Your settings should look as follows
    - SMTP = smtp.example.com
    - smtp\_port = 25
    - *Note the SMTP settings can be gotten from your web hosting providers.*
    - If the server requires authentication, then add the following lines.
      * auth\_username = [example\_username@example.com](mailto:example_username@example.com)
      * auth\_password = example\_password
      * Save the new changes.
      * Restart[Apache](https://www.guru99.com/apache.html)server.

**Php Mail Example**

Let’s now look at an example that sends a simple mail.

<?php

$to\_email = 'name @ company . com';

$subject = 'Testing PHP Mail';

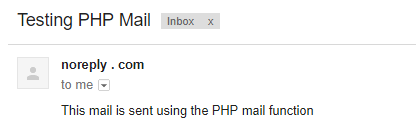
$message = 'This mail is sent using the PHP mail function';

$headers = 'From: noreply @ company . com';

mail($to\_email,$subject,$message,$headers);

?>

**Output:**

[](https://www.guru99.com/images/1/phpemail_1.png)

*Note: the above example only takes the 4 mandatory parameters.*

*You should replace the above fictitious email address with a real email address.*

## Sanitizing email user inputs

The above example uses hard coded values in the source code for the email address and other details for simplicity.

Let’s assume you have to create a contact us form for users fill in the details and then submit.

* Users can accidently or intentional inject code in the headers which can result in sending spam mail
* To protect your system from such attacks, you can create a custom function that sanitizes and validates the values before the mail is sent.

Let’s create a custom function that validates and sanitizes the email address using the filter\_var built in function.

Filter\_var function The filter\_var function is used to sanitize and validate the user input data.

It has the following basic syntax.

<?php

filter\_var($field, SANITIZATION TYPE);

?>

  HERE,

* “filter\_var(…)” is the validation and sanitization function
* “$field” is the value of the field to be filtered.
* “SANITIZATION TYPE” is the type of sanitization to be performed on the field such as;
  + **FILTER\_VALIDATE\_EMAIL** – it returns true for valid email addresses and false for invalid email addresses.
  + **FILTER\_SANITIZE\_EMAIL**– it removes illegal characters from email addresses. info\@domain.(com) returns [info@domain.com](mailto:info@domain.com).
  + **FILTER\_SANITIZE\_URL** – it removes illegal characters from URLs. http://www.example@.comé returns >http://www.example@.com
  + **FILTER\_SANITIZE\_STRING**- it removes tags from string values. <b>am bold</b> becomes am bold.

The code below implements uses a custom function to send secure mail.

<?php

function sanitize\_my\_email($field) {

$field = filter\_var($field, FILTER\_SANITIZE\_EMAIL);

if (filter\_var($field, FILTER\_VALIDATE\_EMAIL)) {

return true;

} else {

return false;

}

}

$to\_email = 'name @ company . com';

$subject = 'Testing PHP Mail';

$message = 'This mail is sent using the PHP mail ';

$headers = 'From: noreply @ company. com';

//check if the email address is invalid $secure\_check

$secure\_check = sanitize\_my\_email($to\_email);

if ($secure\_check == false) {

echo "Invalid input";

} else { //send email

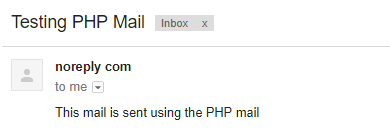
mail($to\_email, $subject, $message, $headers);

echo "This email is sent using PHP Mail";

}

?>

**Output:**

[](https://www.guru99.com/images/1/phpemail_2.png)

## Secure Mail

Emails can be intercepted during transmission by unintended recipients.

This can exposure the contents of the email to unintended recipients.

Secure mail solves this problem by transmitting emails via Hypertext Transfer Protocol Secure (HTTPS).

HTTPS encrypts messages before sending them.

## Summary

* The PHP built in function mail() is used to send mail from PHP scripts
* Validation and sanitization checks on the data are essential to sending secure mail
* The PHP built in function filter\_var() provides an easy to use and efficient way of performing data sanitization and validation

**PHP MySQLi Functions: mysqli\_query, mysqli\_connect, mysqli\_fetch\_array**

PHP has a rich collection of built in functions for manipulating MySQL databases.

**In this tutorial, you will learn-**

* [mysqli\_connect function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#1)
* [mysqli\_select\_db function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#2)
* [mysqli\_query function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#3)
* [mysqli\_num\_rows function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#4)
* [mysqli\_fetch\_array function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#5)
* [mysqli\_close function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#6)
* [PHP Data Access Object PDO](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#7)

**PHP mysqli\_connect function**

The PHP mysql connect function is used to connect to a MySQL database server.

It has the following syntax.

<?php;

$db\_handle = mysqli\_connect($db\_server\_name, $db\_user\_name, $db\_password);

?>

HERE,

* “$db\_handle” is the database connection resource variable.
* “mysqli\_connect(…)” is the function for php database connection
* “$server\_name” is the name or IP address of the server hosting MySQL server.
* “$user\_name” is a valid user name in MySQL server.
* “$password” is a valid password associated with a user name in MySQL server.

**PHP mysqli\_select\_db function**

The mysqli\_select\_db function is used to select a database.

It has the following syntax.

<?php

mysqli\_select\_db($db\_handle,$database\_name);

?>

  HERE,

* “mysqli\_select\_db(…)” is the database selection function that returns either true or false
* “$database\_name” is the name of the database
* “$link\_identifier” is optional, it is used to pass in the server connection link

**PHP mysqli\_query function**

The mysqli\_query function is used to execute[SQL](https://www.guru99.com/sql.html)queries.

The function can be used to execute the following query types;

* Insert
* Select
* Update
* delete

It has the following syntax.

<?php

mysqli\_query($db\_handle,$query) ;

?>

  HERE,

* “mysqli\_query(…)” is the function that executes the SQL queries.
* “$query” is the SQL query to be executed
* “$link\_identifier” is optional, it can be used to pass in the server connection link

**PHP mysqli\_num\_rows function**

The mysqli\_num\_rows function is used to get the number of rows returned from a select query.

It has the following syntax.

<?php

mysqli\_num\_rows($result);

?>

  HERE,

* “mysqli\_num\_rows(…)” is the row count function
* “$result” is the mysqli\_query result set

**PHP mysqli\_fetch\_array function**

The mysqli\_fetch\_array function is used fetch row arrays from a query result set.

It has the following syntax.

<?php

mysqli\_fetch\_array($result);

?>

  HERE,

* “mysqli\_fetch\_array(…)” is the function for fetching row arrays
* “$result” is the result returned by the mysqli\_query function.

**PHP mysqli\_close function**

The mysqli\_close function is used to close an open database connection.

It has the following syntax.

<?php

mysqli\_close($db\_handle);

?>

  HERE,

* “mysqli\_close(…)” is the PHP function
* “$link\_identifier” is optional, it is used to pass in the server connection resource

Let’s look at practical examples that take advantage of these functions.

Creating the MySQL database This tutorial assumes knowledge of MySQL and SQL, if these terms are unfamiliar to you, refer to our MySQL and SQL tutorials.

We will create a simple database called my\_personal\_contacts with one table only.

Below are the steps to create the database and table.

* Connect to MySQL using your favorite access tool such as MySQL workbench, phpMyAdmin etc.
* Create a database named my\_person\_contacts
* Execute the script shown below to create the table and insert some dummy data

<?php

CREATE TABLE IF NOT EXISTS `my\_contacts` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `full\_names` varchar(255) NOT NULL,

  `gender` varchar(6) NOT NULL,

  `contact\_no` varchar(75) NOT NULL,

  `email` varchar(255) NOT NULL,

  `city` varchar(255) NOT NULL,

  `country` varchar(255) NOT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=5 ;

INSERT INTO `my\_contacts` (`id`, `full\_names`, `gender`, `contact\_no`, `email`, `city`, `country`) VALUES

(1, 'Zeus', 'Male', '111', 'zeus @ olympus . mt . co', 'Agos', 'Greece'),

(2, 'Anthena', 'Female', '123', 'anthena @ olympus . mt . co', 'Athens', 'Greece'),

(3, 'Jupiter', 'Male', '783', 'jupiter @ planet . pt . co', 'Rome', 'Italy'),

(4, 'Venus', 'Female', '987', 'venus @ planet . pt . co', 'Mars', 'Italy');

?>

  We now have a database set up that we will manipulate from PHP.

Reading records from the database We will now create a program that prints the records from the database.

<?php

$dbh = mysqli\_connect('localhost', 'root', 'melody');

//connect to MySQL server if (!$dbh)

die("Unable to connect to MySQL: " . mysqli\_error());

//if connection failed output error message

if (!mysqli\_select\_db($dbh,'my\_personal\_contacts'))

die("Unable to select database: " . mysqli\_error());

//if selection fails output error message

$sql\_stmt = "SELECT \* FROM my\_contacts";

//SQL select query

$result = mysqli\_query($dbh,$sql\_stmt);

//execute SQL statement

if (!$result)

die("Database access failed: " . mysqli\_error());

//output error message if query execution failed

$rows = mysqli\_num\_rows($result);

// get number of rows returned

if ($rows) {

while ($row = mysqli\_fetch\_array($result)) {

echo 'ID: ' . $row['id'] . '<br>';

echo 'Full Names: ' . $row['full\_names'] . '<br>';

echo 'Gender: ' . $row['gender'] . '<br>';

echo 'Contact No: ' . $row['contact\_no'] . '<br>';

echo 'Email: ' . $row['email'] . '<br>';

echo 'City: ' . $row['city'] . '<br>';

echo 'Country: ' . $row['country'] . '<br><br>';

}

}

mysqli\_close($dbh); //close the database connection

?>

  Executing the above code returns the results shown in the diagram shown below

[](https://www.guru99.com/images/2013/04/greek_gods.png)

Inserting new records

Let’s now look at an example that adds a new record into our table. the code below shows the implementation.

<?php

$dbh = mysqli\_connect('localhost', 'root', 'melody');

//connect to MySQL server if (!$dbh)

die("Unable to connect to MySQL: " . mysqli\_error());

//if connection failed output error message

if (!mysqli\_select\_db($dbh,'my\_personal\_contacts'))

die("Unable to select database: " . mysql\_error());

//if selection fails output error message

$sql\_stmt = "INSERT INTO `my\_contacts` (`full\_names`,`gender`,`contact\_no`,`email`,`city`,`country`)";

$sql\_stmt .= " VALUES('Poseidon','Mail','541',' poseidon @ sea . oc ','Troy','Ithaca')";

$result = mysqli\_query($dbh,$sql\_stmt); //execute SQL statement

if (!$result)

die("Adding record failed: " . mysqli\_error());

//output error message if query execution failed echo "Poseidon has been successfully added to your contacts list";

mysqli\_close($dbh); //close the database connection

?>

  Executing the above code outputs “Poseidon has been successfully added to your contacts list” go back to the select query example and retrieval your contacts again.

See if Poseidon has been added to your list.

Updating records Let’s now look at an example that updates a record in the database.

Let’s suppose that Poseidon has changed his contact number and email address.

<?php

$dbh = mysqli\_connect('localhost', 'root', 'melody');

//connect to MySQL server

if (!$dbh)

die("Unable to connect to MySQL: " . mysqli\_error());

//if connection failed output error message

if (!mysqli\_select\_db($dbh,'my\_personal\_contacts'))

die("Unable to select database: " . mysql\_error());

//if selection fails output error message

$sql\_stmt = "UPDATE `my\_contacts` SET `contact\_no` = '785',`email` = ' poseidon @ ocean . oc ';

//SQL select query $sql\_stmt .= " WHERE `id` = 5";

$result = mysqli\_query($dbh,$sql\_stmt);

//execute SQL statement if (!$result)

die("Deleting record failed: " . mysqli\_error());

//output error message if query execution failed

echo "ID number 5 has been successfully updated";

mysqli\_close($dbh); //close the database connection

?>

**Deleting records**

Let’s now look at an example that removes records from the database. Let’s suppose that Venus has a restraining order against us, and we must remove her contacts info from our database.

<?php

$dbh = mysqli\_connect('localhost', 'root', 'melody');

//connect to MySQL server

if (!$dbh)

die("Unable to connect to MySQL: " . mysqli\_error());

//if connection failed output error message

if (!mysqli\_select\_db($dbh,'my\_personal\_contacts'))

die("Unable to select database: " . mysqli\_error());

//if selection failes output error message $id = 4;

//Venus's ID in the database

$sql\_stmt = "DELETE FROM `my\_contacts` WHERE `id` = $id";

//SQL Delete query

$result = mysqli\_query($dbh,$sql\_stmt);

//execute SQL statement

if (!$result)

die("Deleting record failed: " . mysqli\_error());

//output error message if query execution failed

echo "ID number $id has been successfully deleted";

mysqli\_close($dbh); //close the database connection

?>

**PHP Data Access Object PDO**

The PDO is a class that allows us to manipulate different database engines such as MySQL, PostGres, MS SQL Server etc.

The code below shows the database access method using the PDO object.

*Note: the code below assumes knowledge of SQL language, arrays, exception handling and foreach loop.*

<?php

try {

$pdo = new PDO("mysql:host=localhost;dbname=my\_personal\_contacts", 'root', 'melody');

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo->exec('SET NAMES "utf8"');

$sql\_stmt = "SELECT \* FROM `my\_contacts`";

$result = $pdo->query($sql\_stmt);

$result->setFetchMode(PDO::FETCH\_ASSOC);

$data = array();

foreach ($result as $row) {

$data[] = $row;

}

print\_r($data);

}

catch (PDOException $e) {

echo $e->getMessage();

}

?>

  HERE,

* “try{…catch…}” is the exception handling block
* “$pdo = new PDO("mysql…” creates an instance of the PDO object and passes the database drivers, server and database names, user id and password.
* “$pdo->setAtt…” sets the PDO error mode and exception mode attributes
* “$pdo->exec('SET NA…” sets the encoding format

ODBC ODBC is the acronym for Open Database Connectivity. It has the following basic syntax.

<?php $conn = odbc\_connect($dsn, $user\_name, $password); ?>

  HERE,

* “odbc\_connect” is the PHP built in function
* “$dsn” is the ODBC data source name.
* “$user\_name” is optional, it is used for the ODBC user name
* “$password” is optional, it is used for the ODBC password

The example used assumes you;

* Are Using Windows OS
* You have created an ODBC link to the northwind Microsoft Access database named northwind

Below is the implementation code for ODBC data access

<?php

$dbh = odbc\_connect('northwind', '', '');

if (!$dbh) {

exit("Connection Failed: " . $dbh);

}

$sql\_stmt = "SELECT \* FROM customers";

$result = odbc\_exec($dbh, $sql\_stmt);

if (!$result) {

exit("Error access records");

}

while (odbc\_fetch\_row($result)) {

$company\_name = odbc\_result($result, "CompanyName");

$contact\_name = odbc\_result($result, "ContactName");

echo "<b>Company Name (Contact Person):</b> $company\_name ($contact\_name) <br>";

}

odbc\_close($dbh);

?>

  Summary

* MySQL is an open source relational database management available on most web hosting servers
* PHP has a rich collection of built in functions that simplify working with MySQL
* PDO is the acronym for PHP Data Object; it is used to connect to different database engines using the same object
* PHP uses the odbc\_connect function to manipulate databases via ODBC

**PHP Object Oriented Programming (OOPs) concept Tutorial with Example**

**What is OOPs?**

Object Oriented is an approach to software development that models application around real world objects such as employees, cars, bank accounts, etc. A class defines the properties and methods of a real world object. An object is an occurrence of a class.

The three basic components of object orientation are;

* Object oriented analysis – functionality of the system
* Object oriented designing – architecture of the system
* Object oriented programming – implementation of the application

**Object Oriented Programming Principles**

The three major principles of OOP are;

* **Encapsulation** – this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to;
  + Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.
  + Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.
  + The internal implementation of the class can be changed without worrying about breaking the code that uses the class.
* **Inheritance**– this is concerned with the relationship between classes. The relationship takes the form of a parent and child. The child uses the methods defined in the parent class. The main purpose of inheritance is;
  + Re-usability– a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database.
* **Polymorphism** – this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is;
  + Simplify maintaining applications and making them more extendable.

**OOPs Concepts in PHP**

PHP is an object oriented scripting language; it supports all of the above principles. The above principles are achieved via;

* **Encapsulation** - via the use of “get” and “set” methods etc.
* **Inheritance** - via the use of extends keyword
* **Polymorphism** - via the use of implements keyword

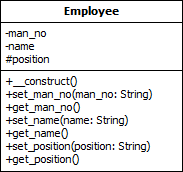
Now that we have the basic knowledge of OOP and how it is supported in PHP, let us look at examples that implement the above principles

**What is UML?**

Unified Modeling Language UML is a technique used to design and document object oriented systems.

UML produces a number of documents, but we will look at the class diagram which is very important to object oriented php programming.

**Class Diagram Example**

[](https://www.guru99.com/images/2013/04/employee_class.png)

**Class Diagram Key**

* The **Upper box** contains the class name
* The **middle box** contains the class variables
* The **lower box** contains the class methods
* The **minus (-)** sign meansprivate scope
* The **plus (+)** sign means public scope
* The **hash (#)** sign means protected scope

**How to Create a class in PHP**

The class keyword is used to define a class in PHP. Below are the rules for creating a class in PHP.

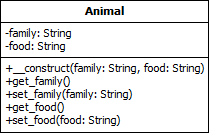
* The class name should start with a letter
* The class name cannot be a PHP reserved word
* The class name cannot contain spaces

Let’s say we want to create a class for representing animals.

We will start with identifying the features that are common to all animals.

* All animals belong to a family such as a herbivore, carnival, etc.
* All animals eat food

The diagram below shows the diagram for the animal

[](https://www.guru99.com/images/2013/04/animal_class.png)

Let’s now code our animal class

<?php

class Animal

{

private $family;

private $food;

public function \_\_construct($family, $food)

{

$this->family = $family;

$this->food = $food;

}

public function get\_family()

{

return $this->family;

}

public function set\_family($family)

{

$this->family = $family;

}

public function get\_food()

{

return $this->food;

}

public function set\_food($food)

{

$this->food = $food;

}

}

?>

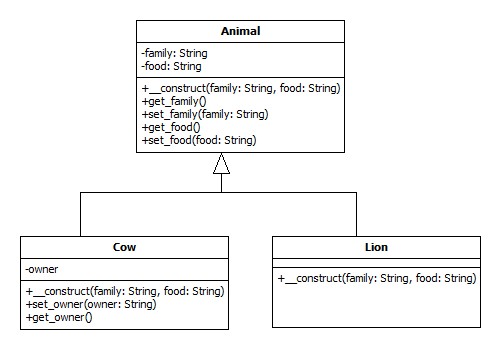
  HERE,

* “private $family, $food” means the variables cannot be accessed directly outside the class (Encapsulation).
* “public function \_\_construct($family…)” is the php constructor method. This function is called whenever an instance of the class has been created. In this case, we are setting the family and food.
* “public function get…()” is the method used to access the family or food value (Encapsulation)
* “public function set…()” is the method used to set the family or food value (Encapsulation)

**How implement Inheritance in PHP**

We will work with a cow and a lion. Both the cow and lion inherit from the Animal class.

The class diagram below shows the relationships.

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

Note the cow inherits from the animal class and defines its own variable and methods too.

Let’s now code the Cow class

<?php

class Cow extends Animal

{

private $owner;

public function \_\_construct($family, $food)

{

parent::\_\_construct($family, $food);

}

public function set\_owner($owner)

{

$this->owner = $owner;

}

public function get\_owner()

{

return $this->owner;

}

}

?>

  Let’s now code the Lion class

<?php

class Lion extends Animal

{

public function \_\_construct($family, $food)

{

parent::\_\_construct($family, $food);

}

}

?>

  HERE,

* “class … extends Animal” makes the cow and lion use methods from the Animal class (Inheritance).

**How to Create object of the class**

The Animal, Cow, and Lion classes should all be in the same directory for simplicity’s sake.

Let’s now create the application that uses our classes.

PHP Class Example

<?php

require 'Animal.php';

require 'Cow.php';

require 'Lion.php';

$cow = new Cow('Herbivore', 'Grass');

$lion = new Lion('Canirval', 'Meat');

echo '<b>Cow Object</b> <br>';

echo 'The Cow belongs to the ' . $cow->get\_family() . ' family and eats ' . $cow->get\_food() . '<br><br>';

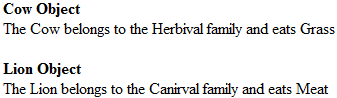
echo '<b>Lion Object</b> <br>';

echo 'The Lion belongs to the ' . $lion->get\_family() . ' family and eats ' . $lion->get\_food();

?>

**Testing our application**

Let’s now view our application in a web browser

. [](https://www.guru99.com/images/2013/04/animal_object.png)

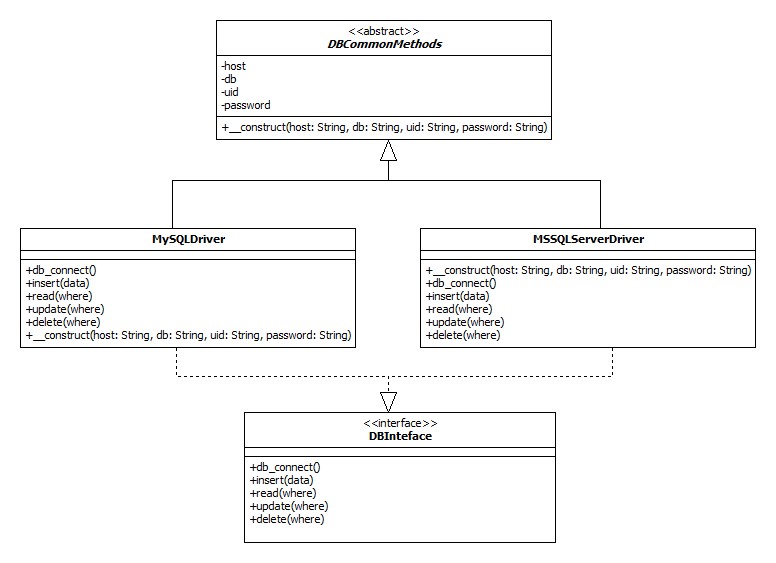
Fantastic right! Let’s now look at the third principle of OOP, polymorphism.

Let’s say we want to develop an application that connects to different database engines such as MySQL and[SQL](https://www.guru99.com/sql.html)Server but use the same uniform interface.

We can create an interface that defines the standard methods and an abstract class that implements the common methods.

* **Interface** – it is similar to a class. It only defines the methods and parameters.
* **Abstract class**– it is a class that cannot be used to create an object directly. Its purpose is to provide partial or whole implementations of common methods.

The class diagram below illustrates the relationship among our abstract class, interface, and implementation classes.

[](https://www.guru99.com/images/2013/04/inheritance2.jpg)

Let’s now create our abstract class

<?php

abstract class DBCommonMethods

{

private $host;

private $db;

private $uid;

private $password;

public function \_\_construct($host, $db, $uid, $password)

{

$this->host = $host;

$this->db = $db;

$this->uid = $uid;

$this->password = $password;

}

}

?>

  HERE,

* “abstract class” means the class cannot be used directly to php create object
* “$host,$db…” are class variables common to all implementations
* “function \_\_construct(…)” is the php class constructor method that sets the common variables values at initialization

Let’s now create the interface that contains the standard methods which will be implemented differently depending on the database engine.

<?php

interface DBInterface

{

public function db\_connect();

public function insert($data);

public function read($where);

public function update($where);

public function delete($where);

}

?>

  HERE,

* “interface” is the keyword for creating interfaces
* “public function…(…)” are the standard methods that should be implemented

Let’s now create the concrete classes that will extend the DBCommonMethods class and extend the DBInterface interface. MySQLDriver.php

<?php class MySQLDriver extends

DBCommonMethods implements DBInterface { public function \_\_construct($host, $db, $uid, $password)

{

parent::\_\_construct($host, $db, $uid, $password); }

public function db\_connect() { //connect code goes here }

public function delete($where) { //delete code goes here }

public function insert($data) { //insert code goes here }

public function read($where) { //read code goes here }

public function update($where) { //update code goes here }

} ?>

  MSSQLServerDriver.php

<?php

class MSSQLServerDriver extends

DBCommonMethods implements DBInterface { public function \_\_construct($host, $db, $uid, $password)

{

parent::\_\_construct($host, $db, $uid, $password); }

public function db\_connect() { //connect code goes here }

public function delete($where) { //delete code goes here }

public function insert($data) { //insert code goes here }

public function read($where) { //read code goes here }

public function update($where) { //update code goes here }

} ?>

  HERE,

* “class … extends DBCommonMethods” use the methods in the DBCommonMethods
* “… implements DBInterface” ensures that the class provides standard methods regardless of the database driver used.

Usage of the above code The code using the above class would look like this

<?php $db = new MySQLDriver($host,$db,$uid,$password); ?>

Or

<?php $db = new MSSQLServerDriver ($host,$db,$uid,$password); ?>

The rest of the code would be the same for both drivers such as;

<?php

$db->db\_connect();

$db->insert($data);

?>

**Summary**

* Object Oriented Programming OOP is a powerful technical that models applications after real world objects
* A class is a representation of real world objects with properties and methods
* The three basic principles of OOP are;
  + Encapsulation
  + Inheritance
  + Polymorphism

**PHP Security Function: strip\_tags, filter\_var, Md5 and sha1**

**Potential security threats**

They are basically two groups of people that can attack your system

* Hackers – with the intent to gain access to unauthorized data or disrupt the application
* Users – they may innocently enter wrong parameters in forms which can have negative effects on a website or web application.

The following are the kinds of attacks that we need to look out for.

**SQL Injection** – This type of attack appends harmful code to[SQL](https://www.guru99.com/sql.html)statements.

This is done using either user input forms or URLs that use variables.

The appended code comments the condition in the WHERE clause of an SQL statement. The appended code can also;

* insert a condition that will always be true
* delete data from a table
* update data in a table
* This type of attack is usually used to gain unauthorized access to an application.

**Cross-site scripting –** this type of attack inserts harmful code usually JavaScript. This is done using user input forms such as contact us and comments forms. This is done to;

* Retrieve sensitive information such as cookies data
* Redirect the user to a different URL.
* Other threats can include – PHP code injection, Shell Injection, Email Injection, Script Source Code Disclosure etc.

**PHP Application Security Best Practices**

Let’s now look at some of the PHP Security best practices that we must consider when developing our applications.

**PHP strip\_tags**

The strip\_tags functions removes HTML,[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)or PHP tags from a string.

This function is useful when we have to protect our application against attacks such as cross site scripting.

Let’s consider an application that accepts comments from users.

<?php

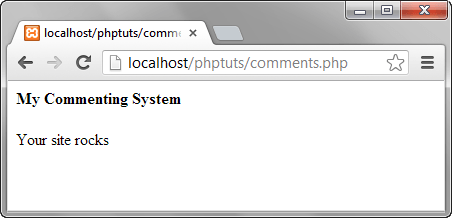
$user\_input = "Your site rocks";

echo "<h4>My Commenting System</h4>";

echo $user\_input;

?>

Assuming you have saved comments.php in the phptuts folder, browse to the URL**http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/normal_comment.png)

Let’s assume you receive the following as the user input   <script>alert('Your site sucks!');</script>

<?php

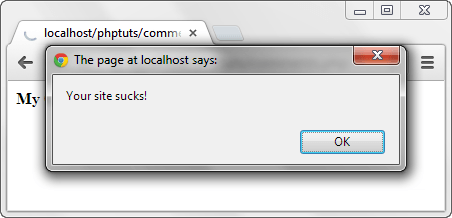
$user\_input = "<script>alert('Your site sucks!');</script>";

echo "<h4>My Commenting System</h4>";

echo $user\_input;

?>

  Browse to the URL **http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/attacked_comment.png)

Let’s now secure our application from such attacks using strip\_tags function.

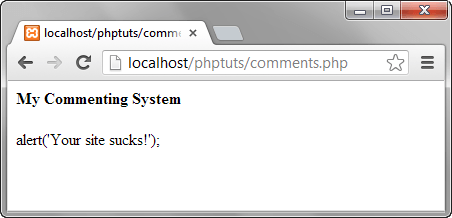
<?php

$user\_input = "<script>alert('Your site sucks!');</script>";

echo strip\_tags($user\_input);

?>

Browse to the URL**http://localhost/phptuts/comments.php**

[](https://www.guru99.com/images/2013/04/sanitized_comment.png)

**PHP filter\_var function**

The filter\_var function is used to validate and sanitize data.

Validation checks if the data is of the right type. A numeric validation check on a string returns a false result.

Sanitization is removing illegal characters from a string.

Check this link for the complete reference [filter\_var](http://www.php.net/manual/en/function.filter-var.php)

The code is for the commenting system.

It uses the filter\_var function and FILTER\_SANITIZE\_STRIPPED constant to strip tags.

<?php

$user\_input = "<script>alert('Your site sucks!');</script>";

echo filter\_var($user\_input, FILTER\_SANITIZE\_STRIPPED);

?>

**Output:**

alert('Your site sucks!');

  mysqli\_real\_escape\_string function This function is used to protect an application against SQL injection.

Let’s suppose that we have the following SQL statement for validating the user id and password.

<?php

SELECT uid,pwd,role FROM users WHERE uid = 'admin' AND password = 'pass';

?>

  A malicious user can enter the following code in the user id text box. ' OR 1 = 1 -- And 1234 in the password text box Let’s code the authentication module

<?php

$uid = "' OR 1 = 1 -- ";

$pwd = "1234";

$sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd';";

echo $sql;

?>

  The end result will be

SELECT uid,pwd,role FROM users WHERE uid = '' OR 1 = 1 -- ' AND password = '1234';

HERE,

* “SELECT \* FROM users WHERE user\_id = ''” tests for an empty user id
* “' OR 1 = 1 “ is a condition that will always be true
* “--" comments that part that tests for the password.

The above query will return all the users Let’s now use mysqli\_real\_escape\_string function to secure our login module.

<?php

$uid = mysqli\_real\_escape\_string("' OR 1 = 1 -- ");

$pwd = mysqli\_real\_escape\_string("1234");

$sql = "SELECT uid,pwd,role FROM users WHERE uid = '$uid' AND password = '$pwd';";

echo $sql;

?>

  The above code will output

SELECT uid,pwd,role FROM users WHERE uid = '\' OR 1 = 1 -- ' AND password = '1234';

*Note the second single quote has been escaped for us, it will be treated as part of the user id and the password won’t be commented.*

**PHP Md5 and PHP sha1**

Md5 is the acronym for Message Digest 5 and sha1 is the acronym for Secure Hash Algorithm 1.

They are both used to encrypt strings.

Once a string has been encrypted, it is tedious to decrypt it.

Md5 and sha1 are very useful when storing passwords in the database.

The code below shows the implementation of md5 and sha1

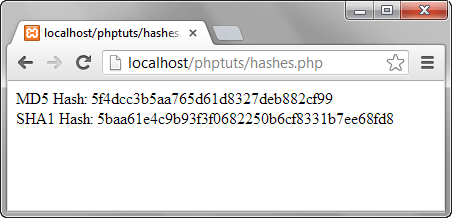
<?php

echo "MD5 Hash: " . md5("password");

echo "SHA1 Hash: " . sha1("password");

?>

  Assuming you have saved the file hashes.php in phptuts folder, browse to the URL

[](https://www.guru99.com/images/2013/04/hashes.png)

As you can see from the above hashes, if an attacker gained access to your database, they still wouldn’t know the passwords for them to login.

**Summary**

* Security refers to measures put in place to protect an application from accidental and malicious attacks.
* strip\_tags function is used to remove tags such as <script></script> from input data
* filter\_var function validates and php sanitize input data
* mysqli\_real\_escape\_string is used to sanitize SQL statement. It removes malicious characters from the statements
* both MD5 and SHA1 are used to encrypt password.

**PHP Date() & Time Function: How to Get Current Timestamp?**

**PHP date() Function**

PHP date function is an in-built function that simplify working with date data types. The PHP date function is used to format a date or time into a human readable format. It can be used to display the date of article was published. record the last updated a data in a database.

**In this tutorial, you will learn date and time function in PHP-**

* [PHP Date Syntax & Example](https://www.guru99.com/php-date-functions.html#3)
* [What is a TimeStamp?](https://www.guru99.com/php-date-functions.html#4)
* [Getting a list of available time zone identifiers](https://www.guru99.com/php-date-functions.html#5)
* [PHP set Timezone Programmatically](https://www.guru99.com/php-date-functions.html#6)
* [PHP Mktime Function](https://www.guru99.com/php-date-functions.html#7)
* [PHP Date function](https://www.guru99.com/php-date-functions.html#8)
* [Time parameters](https://www.guru99.com/php-date-functions.html#9)
* [Day parameters](https://www.guru99.com/php-date-functions.html#10)
* [Month Parameters](https://www.guru99.com/php-date-functions.html#11)
* [Year Parameters](https://www.guru99.com/php-date-functions.html#12)

**PHP Date Syntax & Example**

PHP Date the following basic syntax

<?php

date(format,[timestamp]);

?>

HERE,

* “date(…)” is the function that returns the current timestamp in PHP on the server.
* “format” is the general format which we want our output to be i.e.;
  + “Y-m-d” for PHP date format YYYY-MM-DD
  + “Y” to display the current year
  + “[timestamp]” is optional. If no timestamp has been provided, PHP will get the current PHP date time on the server.

Let’s look at a basic example that displays the current year.

<?php

echo date("Y");

?>

**Output:**

2018

**What is a TimeStamp?**

 A timestamp in PHP is a numeric value in seconds between the current time and value as at 1st January, 1970 00:00:00 Greenwich Mean Time (GMT).

 The value returned by the time function depends on the default time zone.

The default time zone is set in the php.ini file.

It can also be set programmatically using date\_default\_timezone\_set function.

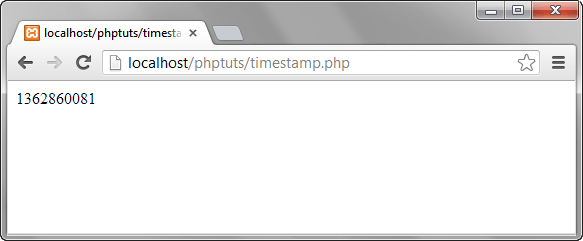
The code below displays the current time stamp

<?php

echo time();

?>

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

[](https://www.guru99.com/images/2013/04/timestamp.png)

Note: the value of the timestamp PHP is not a constant. It changes every second.

**Getting a list of available time zone identifiers**

Before we look at how to set the default time zone programmatically, let’s look at how to get a list of supported time zones.

<?php

$timezone\_identifiers = DateTimeZone::listIdentifiers();

foreach($timezone\_identifiers as $key => $list){

echo $list . "<br/>";

}

?>

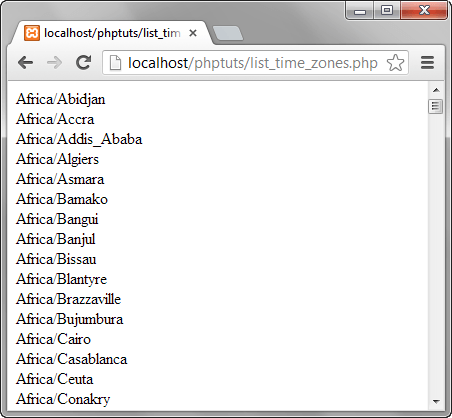
  HERE,

* “$timezone\_identifiers = DateTimeZone::listIdentifiers();” calls the listIdentifiers static method of the DateandTime Zone built in class.

The listIdentifiers method returns a list of constants that are assigned to the variable $timezone\_identifiers.

* “foreach{…}” iterates through the numeric array and prints the values.

Assuming you saved the file list\_time\_zones.php in phptuts folder, browse to the URL **http://localhost/phptuts/list\_time\_zones.php**

[](https://www.guru99.com/images/2013/04/list_time_zones.png)

**PHP set Timezone Programmatically**

The date\_default\_timezone\_set function allows you to set the default time zone from a PHP script.

The set time zone will then be used by all date in PHP function scripts. It has the following syntax.

<?php

date\_default\_timezone\_set ( string $timezone\_identifier );

?>

HERE,

* “date\_default\_timezone\_set()” is the function that sets the default time zone
* “string $timezone\_identifier” is the time zone identifier

The script below displays the time according to the default time zone set in php.ini.

It then changes the default time zone to Asia/Calcutta and displays the time again.

<?php

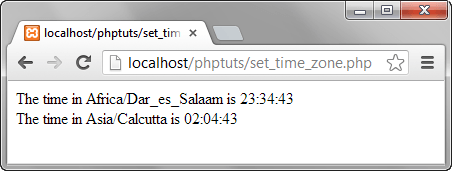
echo "The time in " . date\_default\_timezone\_get() . " is " . date("H:i:s");

date\_default\_timezone\_set("Asia/Calcutta");

echo "The time in " . date\_default\_timezone\_get() . " is " . date("H:i:s");

?>

  Assuming you have saved the file set\_time\_zone.php in the phptuts folder, browse to the URL**http://localhost/phptuts/set\_time\_zone.php**

[](https://www.guru99.com/images/2013/04/timezone.png)

**PHP Mktime Function**

The mktime function returns the timestamp in a[Unix](https://www.guru99.com/unix-linux-tutorial.html)format.

It has the following syntax.

<?php

mktime(hour, minute, second, month, day, year, is\_dst);

?>

HERE,

* “mktime(…)” is the make PHP timestamp function
* “hour” is optional, it is the number of hour
* “minute” is optional, it is the number of minutes
* “second” is optional, it is the number of seconds
* “month” is optional, it is the number of the month
* “day” is optional, it is the number of the day
* “year” is optional, it is the number of the year
* “is\_dst” is optional, it is used to determine the day saving time (DST). 1 is for DST, 0 if it is not and -1 if it is unknown.

Let’s now look at an example that creates a timestamp for the date 13/10/2025 using the mktime function.

<?php

echo mktime(0,0,0,10,13,2025);

?>

  HERE,

* “0,0,0” is the hour, minute and seconds respectively.
* “13” is the day of the month
* “10” is the month of the year
* “2025” is the year

**Output:**

1760328000

**PHP Date function reference**

The table below shows the common parameters used when working with the PHP date functions.

**PHP Time parameters**

| **Parameter** | **Description** | **Example** |
| --- | --- | --- |
| “r” | Returns the full date and time | <?php  echo date("r");  ?> |
| “a”,”A” | Returns whether the current time is am or pm, AM or PM respectively | <?php  echo date("a");  echo date("A");  ?> |
| “g”,”G” | Returns the hour without leading zeroes [1 to 12], [0 to 23] respectively | <?php  echo date("g");  echo date("G");  ?> |
| “h”,”H” | Returns the hour with leading zeros [01 to 12],[00 to 23] respectively | <?php  echo date("h");  echo date("H");  ?> |
| “i”,”s” | Returns the minutes/seconds with leading zeroes [00 to 59] | <?php  echo date("i");  echo date("s");  ?> |

**Day parameters**

| **Parameter** | **Description** | **Example** |
| --- | --- | --- |
| “d” | Returns the day of the month with leading zeroes [01 to 31] | <?php  echo date("d");  ?> |
| “j” | Returns the day of the month without leading zeroes [1 to 31] | <?php  echo date("j");  ?> |
| “D” | Returns the first 3 letters of the day name [Sub to Sat] | <?php  echo date("D");  ?> |
| “l” | Returns day name of the week [Sunday to Saturday] | <?php  echo date("l");  ?> |
| “w” | Returns day of the week without leading zeroes [0 to 6] Sunday is represent by zero (0) through to Saturday represented by six (6) | <?php  echo date("w");  ?> |
| “z” | Returns the day of the year without leading spaces [0 through to 365] | <?php  echo date("z");  ?> |

**Month Parameters**

| **Parameter** | **Description** | **Example** |
| --- | --- | --- |
| “m” | Returns the month number with leading zeroes [01 to 12] | <?php  echo date("m");  ?> |
| “n” | Returns the month number without leading zeroes [01 to 12] | <?php  echo date("n");  ?> |
| “M” | Returns the first 3 letters of the month name [Jan to Dec] | <?php  echo date("M");  ?> |
| “F” | Returns the month name [January to December] | <?php  echo date("F");  ?> |
| “t” | Returns the number of days in a month [28 to 31] | <?php  echo date("t");  ?> |

**Year Parameters**

| **Parameter** | **Description** | **Example** |
| --- | --- | --- |
| “L” | Returns 1 if it’s a leap year and 0 if it is not a leap year | <?php  echo date("L");  ?> |
| “Y” | Returns four digit year format | <?php  echo date("Y");  ?> |
| “y” | Returns two (2) digits year format (00 to 99) | <?php  echo date("y");  ?> |

**Summary**

* The date function in PHP is used to format the timestamp into a human desired format.
* The timestamp is the number of seconds between the current time and 1st January, 1970 00:00:00 GMT. It is also known as the UNIX timestamp.
* All PHP date() functions use the default time zone set in the php.ini file
* The default time zone can also be set programmatically using PHP scripts.

**PHP XML Tutorial: Create, Parse, Read with Example**

**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>';

}

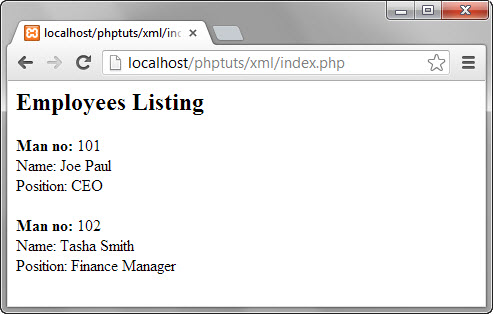
?>

  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";

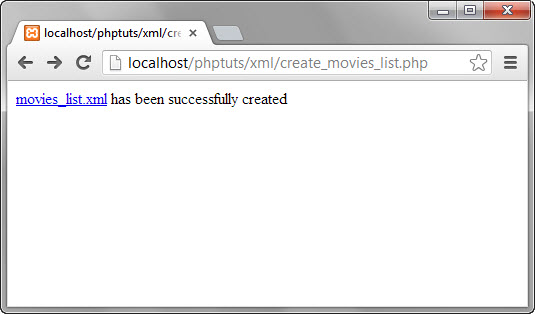
?>

  HERE,

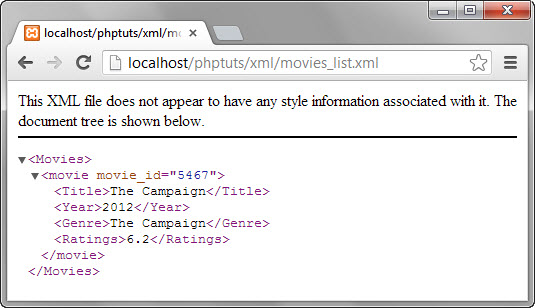
* “$dom = new DOMDocument();” creates an instance of DOMDocument class.
* “$dom->encoding = 'utf-8';” sets the document encoding to utf-8
* “$dom->xmlVersion = '1.0';” specifies the version number 1.0
* “$dom->formatOutput = true;” ensures that the output is well formatted
* “$root = $dom->createElement('Movies');” creates the root node named Movies
* “$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.

# XML Tutorial for Beginners

## What is XML?

**XML** stands for e**X**tensible **M**arkup **L**anguage. It is a language (not> a programming language) that uses the markup and can extend. It is derived from **S**tandard **G**eneralized **M**arkup **L**anguage(SGML). XML also uses DTDs (**D**ocument **T**ype **D**efinitions) to define the XML document structure.

XML is not for handling computational operations and algorithms. Thus, XML is not a programming language. The main goal is to transport data not to display information. XML bridges the gap between human readability and machine readability. Unlike HTML tags, XML tags are self-descriptive.

XML is an open format. The filename extension of XML is .xml

In this XML tutorial, you will learn:

* [What is XML?](https://www.guru99.com/xml-tutorials.html#1)
* [History of XML](https://www.guru99.com/xml-tutorials.html#2)
* [XML Features](https://www.guru99.com/xml-tutorials.html#3)
* [XML Encoding](https://www.guru99.com/xml-tutorials.html#4)
* [XML Syntax](https://www.guru99.com/xml-tutorials.html#5)
* [XML Declaration](https://www.guru99.com/xml-tutorials.html#6)
* [XML Comments](https://www.guru99.com/xml-tutorials.html#7)
* [XML Tags and Elements](https://www.guru99.com/xml-tutorials.html#8)
* [XML Attributes](https://www.guru99.com/xml-tutorials.html#9)
* [Attribute versus Element](https://www.guru99.com/xml-tutorials.html#10)
* [XML Entities](https://www.guru99.com/xml-tutorials.html#11)
* [HTML versus XML](https://www.guru99.com/xml-tutorials.html#12)
* [JSON versus XML](https://www.guru99.com/xml-tutorials.html#13)
* [XML DTD](https://www.guru99.com/xml-tutorials.html#14)
* [XML DOM](https://www.guru99.com/xml-tutorials.html#15)
* [XML Validation](https://www.guru99.com/xml-tutorials.html#16)
* [XML Namespaces](https://www.guru99.com/xml-tutorials.html#17)
* [XML Editors](https://www.guru99.com/xml-tutorials.html#18)
* [XML Parsers](https://www.guru99.com/xml-tutorials.html#19)
* [SAX](https://www.guru99.com/xml-tutorials.html#20)
* [XML Data Binding](https://www.guru99.com/xml-tutorials.html#21)
* [XML Schemas](https://www.guru99.com/xml-tutorials.html#22)
* [Advantages of XML](https://www.guru99.com/xml-tutorials.html#23)
* [Disadvantages of XML](https://www.guru99.com/xml-tutorials.html#24)

## History of XML

XML started way back in 1996 and was first published in 1998. **W**orld **W**ide **W**eb **C**onsortium (W3C) is the developer of **XML**, and it became a **W3C recommendation** in 1998.

There are two versions of XML.

1. XML 1.0
2. XML 1.1

**XML 1.1** is the latest version. Yet, **XML 1.0**is the most used version.

Editors of XML are:

* Tim Bray,
* Jean Paoli,
* C. M. Sperberg,
* Eve Maler,
* François Yergeau.

## XML Features

Here are some important features of XML:

* It is extensible and human-readable.
* It is platform and language independent.
* It preserves white space.
* Overall simplicity.
* Self-descriptive nature.
* It separates data from HTML.
* XML tags are not predefined. You need to define your customized tags.
* XML was designed to carry data, not to display that data.
* Mark-up code of XML is easy to understand for a human.
* Well-structured format is easy to read and write from programs.
* XML is an extensible markup language like HTML.

## XML Encoding

Encoding is the conversion of Unicode characters to their binary representation.UTF is use for XML encoding. **UTF** stands for **U**CS (**UCS** stands for **U**niversal **C**haracter **S**et) **T**ransformation **F**ormat.

Mainly, there are two types of UTF encoding.

1. UTF-8 : UTF-8 uses 8-bits to represent the characters.

Example:

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

1. UTF-16

It uses 16-bits to represent the characters.

Example:

<?xml version="1.0" encoding="UTF-16"?>

You can use encoding inside the XML declaration. UTF-8 is the default encoding in XML.

## XML Syntax

The below code segment shows the basic XML syntax.

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

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

## XML Declaration

XML declaration consists of the XML version, character encoding or/and standalone status. The declaration is optional.

### Syntax for XML Declaration

The below code segment shows the syntax for XML declaration.

<?xml version="version\_number," encoding="character\_encoding" standalone="yes\_or\_no" ?>

### XML Declaration Rules

Following are XML declaration rules.

* If the XML declaration is present, it must be the first thing that appears.
* The XML declaration is case sensitive, and it must start with the lowercased **<?xml**.
* It has no closing tag.

### Example of XML Declaration

Following code segment shows an example of an XML declaration.

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

## XML Comments

Comments are optional. Adding comments help to understand the document content.

### Syntax for XML Comments

A comment begins with **<!--** and ends with **-->**.

Following code segment shows the syntax for XML comments.

<!-- Add your comment here -->

## XML Tags and Elements

Tags work as pairs except for declarations. Every tag pair consists of an **opening tag** (also known as the **start tag**) and a **closing tag** (also known as the **end tag**).

Tag names are enclosed in **<>**. For a particular tag pair, the start and end tags must be identical except the end tag has **/** after the **<**.

<name>...</name>

Anything between the opening and closing tags is referred to as **content**.

Opening tag, content, and closing tag, altogether, is referred to as an **element**.

Opening tag + content + closing tag = an element

**Note:** Elements may also contain attributes. You will learn the attributes very soon.

Let us consider the below element.

<age>20</age>

In the above element,

* **age** is the **name of the element.**

**Note: Tag name** also referred to as an **element** or **element name**.

* **<age>** - opening tag
* **25** - content
* **</age>** - closing tag.

If there is no content between the tags, as shown below, it referred to as **empty tags**.

<result></result>

### XML Tag and Element Rules

Following list shows XML tag and element rules.

* Tags are case sensitive.

Example:

Correct:

<age>20</age>

Wrong:

<age>20</Age>

**Note: AGE**, **Age,** and **age** are three different names in XML.

* All XML documents must contain a single root element.
* All elements must have a closing tag (except for declarations).
* A tag name must begin with a letter or an underscore, and it cannot start with the **XML.**
* A tag name can contain letters, digits, hyphens, underscores, and periods. Hyphens underscore, and periods are the only punctuation marks allowed.
* A tag name cannot contain spaces.
* All elements must be nested properly.

**Example:**

Correct:

<b><u>This text is bold and italic</u></b>

Wrong:

<b><u>This text is bold and italic.</b></u>

## XML Attributes

Attribute for an element is placed after the tag name in the start tag. You can add more than one attribute for a single element with different attribute names.

Let's consider the below XML document.

<company name="ABC Holdings" location="London">

<chairman>Mr. John</chairman>

<gm>Mr. Wood</gm>

</company>

There are two attributes in the **company** element, i.e. **name** and **location**.

Let's study the **name** attribute,

* **name="ABC Holdings"** - an attribute
* **name** - attribute name
* **ABC Holdings** – attribute value

**Note**: An **attribute name** is also known as an **attribute**.

Also, note that in the above example, the **company** is the **root** element.

### XML Attribute Rules

The below list shows XML attribute rules.

* Attribute values must be within quotes.
* An element cannot contain several attributes with the same name.

## Attribute versus Element

Are you still confused about the difference between an attribute and an element? Here is another example.

Let's consider documents A and B given below.

Document A:

<teacher subject="English">

<name>Mr. John</name>.

<qualification>Graduate</qualification>

</teacher>

Document B:

<teacher>

<subject>English</subject>

<name>Mr. John</name>

<qualification>Graduate</qualification>

</teacher>

In document A, the **subject** is an **attribute.**

In document B, the **subject** is an **element**.

## XML Entities

### What are XML Entities?

In simple terms, entities are a way of representing special characters. Entities are also known as **entity** **references**.

### Why You Need XML Entities?

Some characters (such as **"**, **&** **<**, and so on) are reserved in XML. They are referred to as **special characters** and cannot be directly used for other purposes.

For example, the **<**and**>** symbols a used for **tags**. You cannot directly type from the keyboard for **less than** and **greater than** signs. Instead, you need to use entities.

Following table shows some of the popular XML entities.

| **Character** | **Description** | **Entity Name** | **Usage** |
| --- | --- | --- | --- |
| " | Quotation mark (double quote) | **quot** | &quot; |
| & | Ampersand | **amp** | &amp; |
| ' | Apostrophe (single quote) | **apos** | &apos; |
| < | Less than sign | **lt** | &lt; |
| > | Greater than sign | **gt** | &gt; |

Example:

<friend>

<name>My friends are Alice **&amp;** Jane.</name>

</friend>

## HTML versus XML

### Similarities between HTML and XML

Following list shows the similarities between HTML and XML.

* Both are open formats.
* Both are markup languages.
* Both use tags and attributes to describe the content.

### Differences between HTML and XML

Even though XML is like HTML, XML is not a replacement for HTML. There are some significant differences between HTML and XML as well.

Following list table show a comparison between HTML and XML.

|  | **HTML** | **XML** |
| --- | --- | --- |
| **Stands for** | Hypertext Markup Language | Extensible Markup Language |
| **Type of language** | A predefined markup language. | A framework for specifying markup languages. |
| **Structural details** | Not provided. | Provided. |
| **Purpose** | Used to display data. | Used to transport data |
| **Driven by** | Format driven. | Content-driven. |
| **Nature** | Has a static nature. | Has a dynamic nature. |
| **Tag type** | Predefined tags. | User-defined tags. |
| **Tag limit** | A limited number of tags are available. | Tags are extensible. |
| **Closing tags** | It is not necessary to use closing tags (but recommended to use closing tags). | Closing tags are mandatory. |
| **Namespace support** | Not supported. | Supported. |
| **Case sensitivity** | Tags are not case-sensitive. | Tags are case-sensitive. |
| **White space** | White space cannot preserve (can ignore white space). | White space preserved (cannot ignore white space). |
| **Parsing in JavaScript** | Not needed any extra application. | Need DOM implementation. |
| **Code nesting** | Not necessarily needed. | Needed. |
| **Errors** | Can ignore small errors. | Errors are not allowed. |
| **Filename Extension** | **.html** or **.htm** | **.xml** |
| **Size** | Comparatively large. | Comparatively small. |
| **Quotes** | Quotes are not required for attribute values. | Required for XML attribute values. |
| **Object support** | Offers native object support. | Objects have to be expressed by conventions. |
| **Null support** | Natively recognizes the null value. | Need to use xsi:nil on elements. |
| **Formatting decisions** | Provides direct mapping for application data. | Require more significant effort. |
| **Learning curve** | Less steep learning curve compared to XML. | Steep learning curve. |
| **Website** | <https://html.spec.whatwg.org/> | <https://www.w3.org/TR/xml11/> |

### Basic HTML Syntax

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

</body>

</html>

### Basic XML Syntax

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

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

### Same example with HTML and XML

With HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<p>Book</p>

<p>Name: Anna Karenina</p>

<p>Author: Leo Tolstoy</p>

<p>Publisher: The Russian Messenger</p>

</body>

</html>

With XML

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

<book>

<name>Anna Karenina</name>

<author>Leo Tolstoy</author>

<publisher>The Russian Messenger</publisher>

</book>

## JSON versus XML

### Similarities between JSON and XML

The below list shows the similarities between JSON and XML.

* Both are open formats.
* Both are self-describing.
* Both have a hierarchical structure.
* Both can parse and use by several programming languages.

### Differences between JSON and XML

There are several differences between XML and JSON as well.

The below tables show a comparison between JSON and XML.

|  | **JSON** | **XML** |
| --- | --- | --- |
| **Stands for** | JavaScript Object Notation | Extensible Markup Language |
| **Extended from** | JavaScript | SGML |
| **Data storage** | Data stored as key-value pairs. | Data stored as a tree structure. |
| **Namespaces** | No support for namespaces. | Supports namespaces. |
| **Comments** | Adding comments is not supported. | Can add comments. |
| **Data accessibility** | Readily accessible as JSON objects. | Data need to be parsed. |
| **Metadata** | Adding metadata is not supported. | Can write metadata. |
| **Types** | JSON types: string, number, array, Boolean. | All XML data should be strings. |
| **Data types of support** | Supports **text** and **number** data types only. | Support many data types (text, numbers, images, so on) |
| **Array's support** | More support for arrays compared to XML. | No or less support for arrays. |
| **Object's support** | Native support for object. | The object has to be express by conventions. |
| **AJAX toolkit support** | Supported. | Not fully supported. |
| **Retrieving values** | Easy. | Difficult. |
| **Deserializing/serializing** | Fully automated. | Developers have to write JavaScript code. |
| **Browser support** | Supported by most browsers. | Cross-browser XML parsing can be tricky. |
| **Encoding** | Only supports UTF-8 encoding. | It supports various encoding. |
| **Display capabilities** | No display capabilities. | Offer display capabilities. |
| **Document size** | Smaller than XML. | Large than JSON. |
| **Filename Extension** | **.json** | **.xml** |
| **Security** | Less secured. | More secure than JSON. |
| **Easy to read** | Relatively easy. | Relatively difficult. |
| **Learning curve** | Easy to learn. | Steep learning curve. |
| **Website** | <https://www.json.org/json-en.html> | <https://www.w3.org/TR/xml11/> |

### Basic JSON Syntax

{string:value, .......}

### Same example with JSON and XML

With JSON

{"books,":[

{"name":"Anna Karenina", "author":"Leo Tolstoy"},

{"name":"One Hundred Years of Solitude", "author":"Gabriel Garcia Marquez"},

{"name":"The Great Gatsby", "author":"Scott Fitzgerald"},

{"name":"Invisible Man", "author":"Ralph Ellison"}

]}

With XML

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

<books>

<book>

<name>Anna Karenina</name>

<author>Leo Tolstoy</author>

</book>

<book>

<name>One Hundred Years of Solitude</name>

<author>Gabriel Garcia Marquez</author>

</book>

<book>

<name>The Great Gatsby</name>

<author>Scott Fitzgerald</author>

</book>

<book>

<name>Invisible Man</name>

<author>Ralph Ellison</author>

</book>

</books>

## XML DTD

### What is DTD?

DTD stands for **D**ocument **T**ype **D**efinition. It defines the structure of an XML document using some legal elements. XML DTD is optional.

### DTD Rules

Following list shows DTD rules.

* If DTD is present, it must appear at the start of the document (only the XML declaration can appear above the DTD).
* The element declaration must start with an **!** mark.
* The DTD name and element type of the root element must be the same.

### Examples of DTD

**Example of an internal DTD:**

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

<!DOCTYPE student [

<!ELEMENT student (firstname,lastname,school)>

<!ELEMENT firstname (#PCDATA)>

<!ELEMENT lastname (#PCDATA)>

<!ELEMENT school (#PCDATA)>

]>

<student>

<firstname>Mark</firstname>

<lastname>Wood</lastname>

<school>Hills College</school>

</student>

In the above example,

* **!DOCTYPE student**indicates the beginning of the DTD declaration. And the **student** is the root element of the XML document.
* **!ELEMENT student** indicates the **student** element must contain **firstname, lastname**and**school**elements.
* **!ELEMENT firstname** indicates the **firstname** element is of type **#PCDATA** (**P**arsed **C**haracter **Data**).
* **!ELEMENT lastname** indicates the **lastname** element is of type **#PCDATA**.
* **!ELEMENT school** indicates the **school** element is of type **#PCDATA**.

**Example of an external DTD:**

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

<!DOCTYPE student SYSTEM "student.dtd">

<student>

<firstname>Mark</firstname>

<lastname>Wood</lastname>

<school>Hills College</school>

</student>

The DTD file content (student.dtd) as follows.

<!ELEMENT student (firstname,lastname,school)>

<!ELEMENT firstname (#PCDATA)>

<!ELEMENT lastname (#PCDATA)>

<!ELEMENT school (#PCDATA)>

## XML DOM

### What is DOM?

DOM stands for **D**ocument **O**bject **M**odel. It defines a standard manner of accessing and manipulating XML documents. DOM has a (hierarchical) tree structure.

### Example of DOM

Let's consider the below XML document.

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

<school>

<student>

<name>

<first\_name>Alex</first\_name>

<last\_name>Clarke</last\_name>

</name>

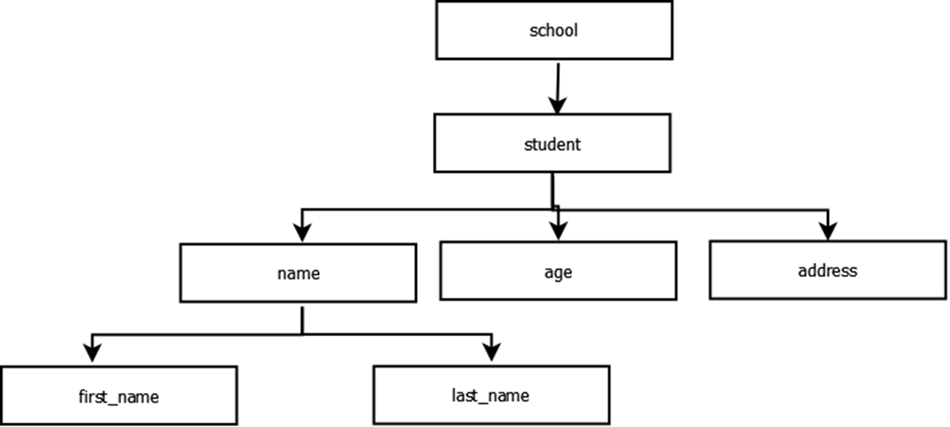
<age>14</age>

<address>No. 35, Flower Road, Leeds</address>

</student>

</school>

The tree structure of the above XML file would look like the following image.

[](https://www.guru99.com/images/2/041321_0541_XMLTutorial1.png)XML DOM Example

## XML Validation

### What are Well-formed XML Documents?

**Well-formed** XML documents are XML documents with correct syntax.

### What are Valid XML Documents?

**Valid**XML documents are well-formed and also conform to the DTD rules.

## XML Namespaces

### Why Namespaces?

Namespaces help to avoid element name conflicts.

### Namespace Declaration

Following shows the syntax for the namespace declaration.

<element xmlns:name="URL">

In the above declaration,

* The **xmlns**keyword indicates the beginning of the namespace.
* The **name** is the prefix of the namespace.
* The **URL** is the namespace identifier.

### Examples of Namespaces

Following code segment shows an example of namespaces.

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

<abt:about xmlns:abt="https://www.guru99.com/about-us.html">

<abt:founder>Krishna</abt:founder>

<abt:vision>Fun and Free Education for ALL</abt:vision>

</abt:about>

## XML Editors

There are several XML editors available. Any text editor (such as notepad and so on) can use as an XML editor.

The following list shows some of the popular XML editors in 2021.

### 1) XML Notepad:

XML Notepad is an open-source editor for XML . It has a tree view and XSL Output on the left pane and node text on the right. It has an error-debugging window at the bottom.

#### Key Statistics:

* Type - XML editor
* Developer - Microsoft
* Supported operating system - Microsoft Windows.
* Price - Free

**Link:** <http://microsoft.github.io/XmlNotepad/>

### 2) Stylus Studio:

Stylus Studio is an IDE written in C++ for Extensible Markup Language ( XML ). It allows a user to edit and transform XML documents, data such as electronic data interchange(EDI), CSV, and relational data.

#### Key Statistics:

* Type - Integrated development environment (IDE) for XML
* Developer - Progress Software Corporation
* Supported operating system - Microsoft Windows.
* Price - Paid (Please refer to the website given below for the latest price), Free trial available.

**Link:** <http://www.stylusstudio.com/>

### 3) Altova XMLSpy:

XMLSpy is primarily marketed as a JSON and XML Editor. It has a built-in schema designer and editor. It includes Visual Studio And Eclipse integration.

#### Key Statistics:

* Type - XML Editor
* Developer - Altova
* Supported operating system - Microsoft Windows.
* Price - Paid (Please refer to the website given below for the latest price), Free trial available.

**Link:** <https://www.altova.com/xmlspy-xml-editor>

### 4) Oxygen XML Editor:

Oxygen XML is a cross-platform editor developed in Java. It helps to validate schemas like DTD, W3C XML Schema, RELAX NG, Schematron, NRL, and NVDL schemas.

#### Key Statistics:

* Type - XML editor
* Developer - SyncRO Soft Ltd
* Supported operating system - Windows, Linux, and Mac OS X
* Price - Paid (Please refer to the website given below for the latest price

**Link:** <https://www.oxygenxml.com/>

### 5) Xmplify:

Xmplify XML Editor provides a fully XML-aware editing environment with DTD and XML Schema-based auto, automatic document validation, etc.

#### Key Statistics:

* Type - XML Editor
* Developer - MOSO Corporation
* Supported operating system - Mac OS.
* Price - Paid (Please refer to the website given below for the latest price

**Link:** <http://xmplifyapp.com/>

## XML Parsers

An XML parser is a software library that provides an interface to work with XML documents. It checks whether the format of the XML document is correct. Some parsers can also validate the XML documents. Modern-day browsers come with XML parsers.

## SAX

**SAX** stands for **S**imple **A**PI for **X**ML. It is an **a**pplication **p**rogram **i**nterface (**API**) for parsing XML documents. They behave similarly to the event handlers in Java.

Unlike DOM, SAX is an example of an event-based XML parser.

Here are some important differences between the SAX and DOM.

|  | **SAX** | **DOM** |
| --- | --- | --- |
| **Stands for** | Simple API for XML | Document Object Model |
| **Type of parser** | Event-based | Object-based |
| **Read and write XML** | Read-only | Both read and write |
| **Insert/update/delete nodes** | Cannot insert/update/delete nodes | Can insert/update/delete nodes |
| **Memory efficiency** | Good memory efficiency | Varies |
| **Speed** | Slower than DOM Parser | Faster than SAX Parser |
| **Suitable for** | Small-sized files | Large-sized files |

## XML Data Binding

XML data binding is the representation of data in an XML document as a business object in the memory of a computer.

There are three approaches for XML data binding.

* **XML schema-based data binding:** Corresponding XML classes are created based on the schema.
* **Class-based data binding:** A corresponding XML schema is created based on classes.
* **Mapping-based data binding:**It describes how an existing XML schema maps to a set of classes (and vice-versa).

There are XML data binding frameworks also.

Examples:

* [gSOAP](https://www.genivia.com/dev.html)
* [xStream](http://x-stream.github.io/)

XML data binding is easy with frameworks. The data binding framework generates a large amount of code for you. You need to feed in a DTD or XML schema.

## XML Schemas

XML **schema**(also known as **XML schema definition** or **XSD**) use to describe the XML document structure. It is an alternative to DTD.

### Why Schema is Important?

DTD is not powerful as schema as it is not extensible and flexible enough. So, it may not be suitable for some situations. In such a situation schema is important. The main purpose of using XML schema is to define the elements and attributes of an XML document.

### How XML Schema is Different from DTD?

The following comparison shows how XSD (XML Schema) is different from DTD.

|  | **DTD** | **XSD** |
| --- | --- | --- |
| **Stands for** | Document Type Definition | XML Schema Definition |
| **Extensibility** | Not extensible | Extensible |
| **Control on XML structure** | Less control | More control |
| **Data types of support** | Not supported | Supported |
| **Namespace Support** | Not supported | Supported |

Following code segment shows an example of XML schema.

xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

<xs:element name = "employee">

<xs:complexType>

<xs:sequence>

<xs:element name = "firstname" type = "xs:string" />

<xs:element name = "lastname" type = "xs:string" />

<xs:element name = "phone" type = "xs:int" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

## Advantages of XML

Here, pros/benefits of XML:

* It made it easy to transport and share data.
* XML improves the exchange of data between various platforms.
* It is a markup language, which is a set of characters or/and symbols placed in a text document.
* XML indicates how the XML document should look after it is displayed.
* It simplifies the platform change process.
* It enhances data availability.
* It supports multilingual documents and Unicode.
* Provide relatively easy to learn and code.
* It is a markup language, which is a set of characters or/and symbols placed in a text document.
* It performs validation using DTD and Schema.
* Makes documents transportable across systems and applications. With the help of XML, you can exchange data quickly between different platforms.
* XML separates the data from HTML.

## Disadvantages of XML

Here are the cons/drawback of using XML:

* XML requires a processing application.
* The XML syntax is similar to another alternative 'text-based' data transmission formats, which is sometimes confusing.
* No intrinsic data type support
* The XML syntax is redundant.
* Does not allow the user to create his tags.

### Summary

* XML stands for eXtensible Markup Language. XML is a language (not a programming language) that uses the markup and can extend.
* The main goal is to transport data, not to display data.
* XML 1.1 is the latest version. Yet, XML 1.0 is the most used version.
* Tags work as pairs except for declarations.
* Opening tag + content + closing tag = an element
* Entities are a way of representing special characters.
* DTD stands for Document Type Definition. It defines the structure of an XML document using some legal elements. XML DTD is optional.
* DOM stands for Document Object Model. It defines a standard manner of accessing and manipulating XML documents.
* Well-formed XML documents are XML documents with correct syntax.
* Valid XML documents are well-formed and also conform to the DTD rules.
* Namespaces help to avoid element name conflicts.