


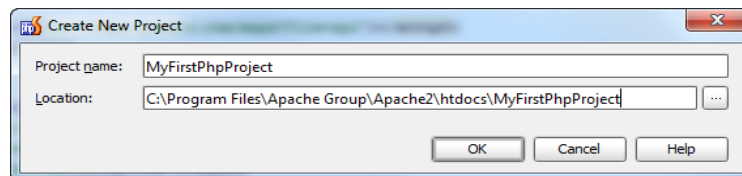
## PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

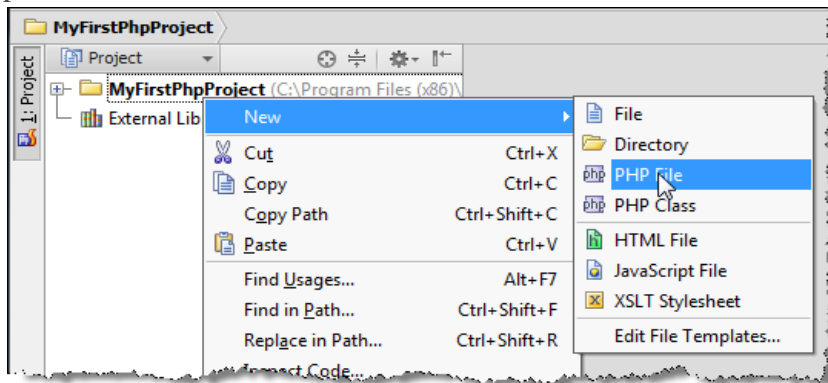
- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

To create and run your first php project

1. Create a project. For that, choose File | Create New Project on the main menu.
2. In the Create New Project dialog box that opens specify the name of the project in theName text box, for example, typeMyFirstPhpProject.
3. Click the Browse button  next to theLocation text box.
4. In the dialog that opens select the path to the.\htdocs folder and click OK. PhpStorm composes the path to the project folder as follows:



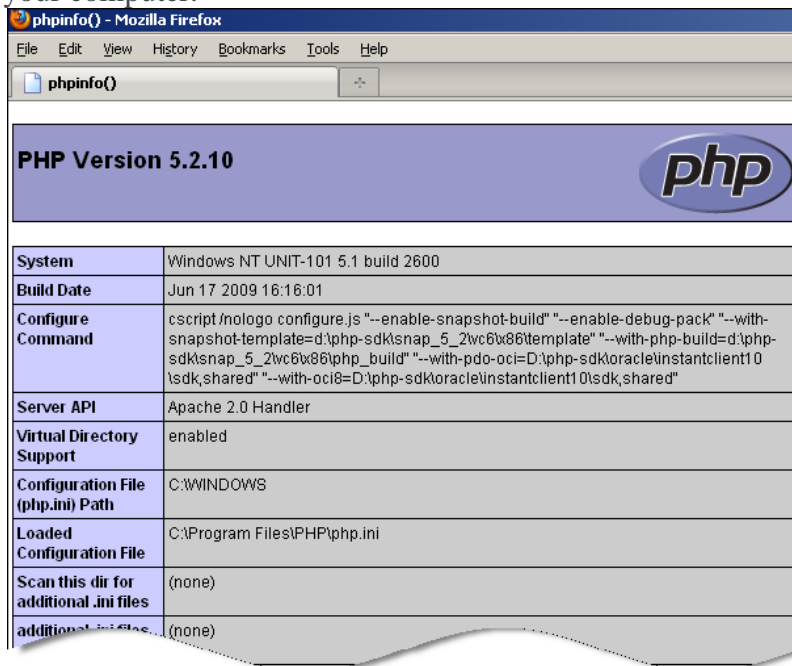
5. Create a PHP file. To do that, right-click the project directory in the Project tool window, point to New on the context menu, and choose PHP File.



6. In the New PHP File dialog box that opens type MyFile and click OK. PhpStorm will create the stub file for you and open it in the dedicated editor tab.
7. Type this sample code:

```
echo phpinfo();
```

8. To save the file, choose File | Save All or press Ctrl+S
9. Open a Web browser and type the following URL address:
10. <http://localhost:<port>/MyFirstPhpProject/MyFile.php>
11. The PHP Information page opens displaying the PHP engine configuration settings on your computer.



A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**:

```
<?php
// PHP code goes here
?>
```

## PHP Variables

Variables are "containers" for storing information.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

## PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no \$ sign before the constant name).

**Note:** Unlike variables, constants are automatically global across the entire script.

To create a constant, use the define() function.

Syntax

```
define(name, value, case-insensitive)
```

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

The example below creates a constant with a **case-sensitive** name:

## Example

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

## PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

## PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

## Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

## PHP Integer

An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

Rules for integers:

- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example \$x is an integer. The PHP var\_dump() function returns the data type and value:

#### Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

#### PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example \$x is a float. The PHP var\_dump() function returns the data type and value:

#### Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

#### PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

#### PHP Array

An array stores multiple values in one single variable.

In the following example \$cars is an array. The PHP var\_dump() function returns the data type and value:

#### Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

#### PHP Object

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

#### Example

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

#### PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

**Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

#### Example

```
<?php
$x = "Hello world!";
$x = null;
```

```
var_dump($x);  
?>
```

## PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

## PHP Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes some code if a condition is true and another code if the condition is false
- **if...elseif...else statement** - specifies a new condition to test, if the first condition is false
- **switch statement** - selects one of many blocks of code to be executed

## PHP - The if Statement

The if statement is used to execute some code **only if a specified condition is true**.

### Syntax

```
if (condition) {  
    code to be executed if condition is true;  
}
```

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

### Example

```
<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
}
```

```
}  
?>
```

## PHP - The if...else Statement

Use the if...else statement to execute some code **if a condition is true and another code if the condition is false.**

### Syntax

```
if (condition) {  
    code to be executed if condition is true;  
} else {  
    code to be executed if condition is false;  
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

### Example

```
<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

## PHP - The if...elseif....else Statement

Use the if...elseif...else statement to **specify a new condition to test, if the first condition is false.**

### Syntax

```
if (condition) {  
    code to be executed if condition is true;  
} elseif (condition) {  
    code to be executed if condition is true;  
} else {  
    code to be executed if condition is false;  
}
```



The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

#### Example

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

#### The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

#### Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

This is how it works: First we have a single expression  $n$  (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.

Use **break** to prevent the code from running into the next case automatically.

The **default** statement is used if no match is found.

## Example

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, or green!";
}
?>
```

## PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

### The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

### Syntax

```
while (condition is true) {
    code to be executed;
}
```

The example below first sets a variable \$x to 1 (\$x = 1). Then, the while loop will continue to run as long as \$x is less than, or equal to 5 (\$x <= 5). \$x will increase by 1 each time the loop runs (\$x++):

#### Example

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

#### The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

#### Syntax

```
do {
    code to be executed;
} while (condition is true);
```

The example below first sets a variable \$x to 1 (\$x = 1). Then, the do while loop will write some output, and then increment the variable \$x with 1. Then the condition is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5:

#### Example

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

#### The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

## Syntax

```
for (init counter; test counter; increment counter) {  
    code to be executed;  
}
```

### Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

The example below displays the numbers from 0 to 10:

### Example

```
<?php  
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}  
?>
```

## The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

## Syntax

```
foreach ($array as $value) {  
    code to be executed;  
}
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array (\$colors):

### Example

```
<?php  
$colors = array("red", "green", "blue", "yellow");  
  
foreach ($colors as $value) {  
    echo "$value <br>";  
}
```

```
}  
?>
```

## PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function.

### Create a User Defined Function in PHP

A user defined function declaration starts with the word "function":

#### Syntax

```
function functionName() {  
    code to be executed;  
}
```

**Note:** A function name can start with a letter or underscore (not a number).

**Tip:** Give the function a name that reflects what the function does!

Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ( { ) indicates the beginning of the function code and the closing curly brace ( } ) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

#### Example

```
<?php  
function writeMsg() {  
    echo "Hello world!";  
}  
  
writeMsg(); // call the function  
?>
```

## PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (\$fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

#### Example

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

## PHP Arrays

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

## Create an Array in PHP

In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

## PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";
```

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:

### Example

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";  
?>
```

## PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";
```

The named keys can then be used in a script:

### Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

## PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

## PHP Form Handling

The PHP superglobals `$_GET` and `$_POST` are used to collect form-data.

### PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

#### Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
```



```
</body>  
</html>
```

The output could be something like this:

```
Welcome John  
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:

## PHP MySQL Database

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP.

What is MySQL?

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

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

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

- Employees
- Products
- Customers
- Orders

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

## Database Queries

A query is a question or a request.

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

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

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

A database consists of one or more tables.

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

### Create a MySQL Database Using MySQLi and PDO

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

The following examples create a database named "myDB":

#### Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```