3.1 Working with Number

One thing to notice about PHP is that it provides automatic data type conversion.
So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
This automatic conversion can sometimes break your code.

1 PHP Integers

An integer is a number without any decimal part.
2, 256, -256, 10358, -179567 are all integers. While 7.56, 10.0, 150.67 are floats.
So, an integer data type is a non-decimal number between -2147483648 and 2147483647. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).
Here are some rules for integers:

- An integer must have at least one digit
- 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)

PHP has the following functions to check if the type of a variable is integer:

- is_int()
- is_integer() - alias of is_int()
- is_long() - alias of is_int()

Example
Check if the type of a variable is integer:

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

$x = 59.85;
var_dump(is_int($x));
?>
```

2 PHP Floats

A float is a number with a decimal point or a number in exponential form.
2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.
The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following functions to check if the type of a variable is float:

- is_float()
- is_double() - alias of is_float()

Example

Check if the type of a variable is float:

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

3 PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

- is_nan()

However, the PHP var_dump() function returns the data type and value:

Example

Invalid calculation will return a NaN value:

```
<?php
$x = acos(8);
 var_dump($x);
?>
```

4 PHP Numerical Strings

The PHP is_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

Example

Check if the variable is numeric:

```
<?php
$x = 5985;
var_dump(is_numeric($x));
$x = "5985";
var_dump(is_numeric($x));
$x = "59.85" + 100;
var_dump(is_numeric($x));
$x = "Hello";
var_dump(is_numeric($x));
?>
```

5 PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.
The (int), (integer), or intval() function are often used to convert a value to an integer.
Example
Cast float and string to integer:

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;

echo "<br>";

// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

6 Number Function

Don't think that PHP's power is limited to strings only: the language has over 50 built-in functions for working with numbers, ranging from simple formatting functions to functions for arithmetic, logarithmic, and trigonometric manipulations. Table given below lists some of these 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

Function	Description	Example	Output
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

Ceil	Round off a number with decimal points to the nearest large number.	<?php
		echo ceil(3.4);
		echo ceil(3.9);
		?>
			4
			4

Floor	Round off a number with decimal points to the nearest small number.	<?php
		echo floor(3.4);
		echo floor(3.9);
		?>
			3
			3

round	Round off a number with decimal points to the nearest whole number.	<?php
		echo round(3.4);
		echo round(3.8);
		?>
			3
			4

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	<?php	3.1415926535898

Function	Description	Example	Output
	returns the value of PI	echo pi(); ?>	

3.2 Working with String

Basic PHP String Functions

In this section, we will discuss a few basic string functions which are most commonly used in PHP scripts.

1. Getting length of a String

PHP has a predefined function to get the length of a string. Strlen() displays the length of any string. It is more commonly used in validating input fields where the user is limited to enter a fixed length of characters.

Syntax

Strlen(string);

Example

```
<?php
echo strlen("Welcome to Cloudways");//will return the length of given string
?>
```

Output

```
20
```

2. Counting of the number of words in a String

Another function which enables display of the number of words in any specific string is str_word_count(). This function is also useful in validation of input fields.

Syntax

Str_word_count(string)

Example

```
<?php
echo str_word_count("Welcome to Cloudways");//will return the number of words in a string
?>
```

Output

```
3
```

3. Reversing a String

Strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

Syntax

Strev(string)

Example

```
<?php
echo strrev(“Welcome to Cloudways”);// will return the string starting from the end
?>
```

Output

```
syawduolC ot emocleW
```

4. Finding Text Within a String

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return “False”. Strops() is most commonly used in validating input fields like email.

Syntax

Strpos(string,text);

Example

```
<?php
echo strpos(“Welcome to Cloudways”,”Cloudways”);
?>
```

Output

```
11
```

5. Replacing text within a string

Str_replace() is a built-in function, basically used for replacing specific text within a string.

Syntax

Str_replace(string to be replaced,text,string)

Example

```
<?php
echo str_replace(“cloudways”, “the programming world”, “Welcome to cloudways”);
?>
```

Output

```
Welcome to the programming world
```

6. Converting lowercase into Title Case

Ucwords() is used to convert first alphabet of every word into uppercase.

Syntax

Ucwords(string)

Example

```
<?php
echo ucwords(“welcome to the php world”);
?>
```

Output

```
Welcome To The Php World
```

7. Converting a whole string into UPPERCASE

Strtoupper() is used to convert a whole string into uppercase.

Syntax

Strtoupper(string);

Example

```
<?php
echo strtoupper(“welcome to cloudways”);// It will convert all letters of string into uppercase
?>
```

Output

```
WELCOME TO CLOUDWAYS
```

8. Converting whole String to lowercase

Strtolower() is used to convert a string into lowercase.

Syntax

Strtolower(string)

Example

```
<?php
echo strtolower(“WELCOME TO CLOUDWAYS”);
?>
```

Output : welcome to cloudways

9. Repeating a String

PHP provides a built-in function for repeating a string a specific number of times.

Syntax

Str_repeat(string,repeat)

Example

```
<?php
echo str_repeat(“=”,13);
?>
```

Output

```
=============
```

10. Comparing Strings

You can compare two strings by using strcmp(). It returns output either greater than zero, less than zero or equal to zero. If string 1 is greater than string 2 then it returns greater than zero. If string 1 is less than string 2 then it returns less than zero. It returns zero, if the strings are equal.

Syntax

Strcmp(string1,string2)

Example

```
<?php
echo strcmp(“Cloudways”,”CLOUDWAYS”);
echo “<br>”;
echo strcmp(“cloudways”,”cloudways”);//Both the strings are equal
echo “<br>”;
echo strcmp(“Cloudways”,”Hosting”);
echo “<br>”;
echo strcmp(“a”,”b”);//compares alphabetically
echo “<br>”;
echo strcmp(“abb baa”,”abb baa caa”);//compares both strings and returns the result in terms of number of characters.
?>
```

Output

```
1
0
-1
-1
-4
```

11. Displaying part of String

Through substr() function you can display or extract a string from a particular position.

Syntax

substr(*string,start,length*)

Example

```
<?php
echo substr(“Welcome to Cloudways”,6).”<br>”;
echo substr(“Welcome to Cloudways”,0,10).”<br>”;
?>
```

Output

```
e to Cloudways
Welcome to
```

12. Removing white spaces from a String

Trim() is dedicated to remove white spaces and predefined characters from a both the sides of a string.

Syntax

trim(*string,charlist*)

Example

```
<?php
$str = “Wordpess Hosting”;
echo $str . “<br>”;
echo trim(“$str”,”Wording”);
?>
```

Output

```
Wordpess Hosting
pess Host
```

3.3 Working with Dates and Time

1 Date/Time

The PHP date() function is used to format a date and/or a time.

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

date(*format,timestamp*)

Parameter	**Description**
Format	Required. Specifies the format of the timestamp
Timestamp	Optional. Specifies a timestamp. Default is the current date and time

2 Get a Date

The required *format* parameter of the date() function specifies how to format the date

(or time).
Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.
The example below formats today's date in three different ways:
Example

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

Use the date() function to automatically update the copyright year on your website:
Example

```
&copy; 2010-<?php echo date("Y");?>
```

Get a Time
Here are some characters that are commonly used for times:

- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:
Example

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

3 Get Your Time Zone

If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.
So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.
The example below sets the timezone to "America/New_York", then outputs the current

time in the specified format:
Example

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

4 Create a Date With mktime()

The optional *timestamp* parameter in the date() function specifies a timestamp. If omitted, the current date and time will be used (as in the examples above).
The PHP mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.
Syntax
mktime(*hour, minute, second, month, day, year*)
The example below creates a date and time with the date() function from a number of parameters in the mktime() function:
Example

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

5 Create a Date From a String With strtotime()

The PHP strtotime() function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).
Syntax
strtotime(*time, now*)
The example below creates a date and time from the strtotime() function:
Example

```
<?php
$d=strtotime("10:30pm April 15 2014");
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

PHP is quite clever about converting a string to a date, so you can put in various values:
Example

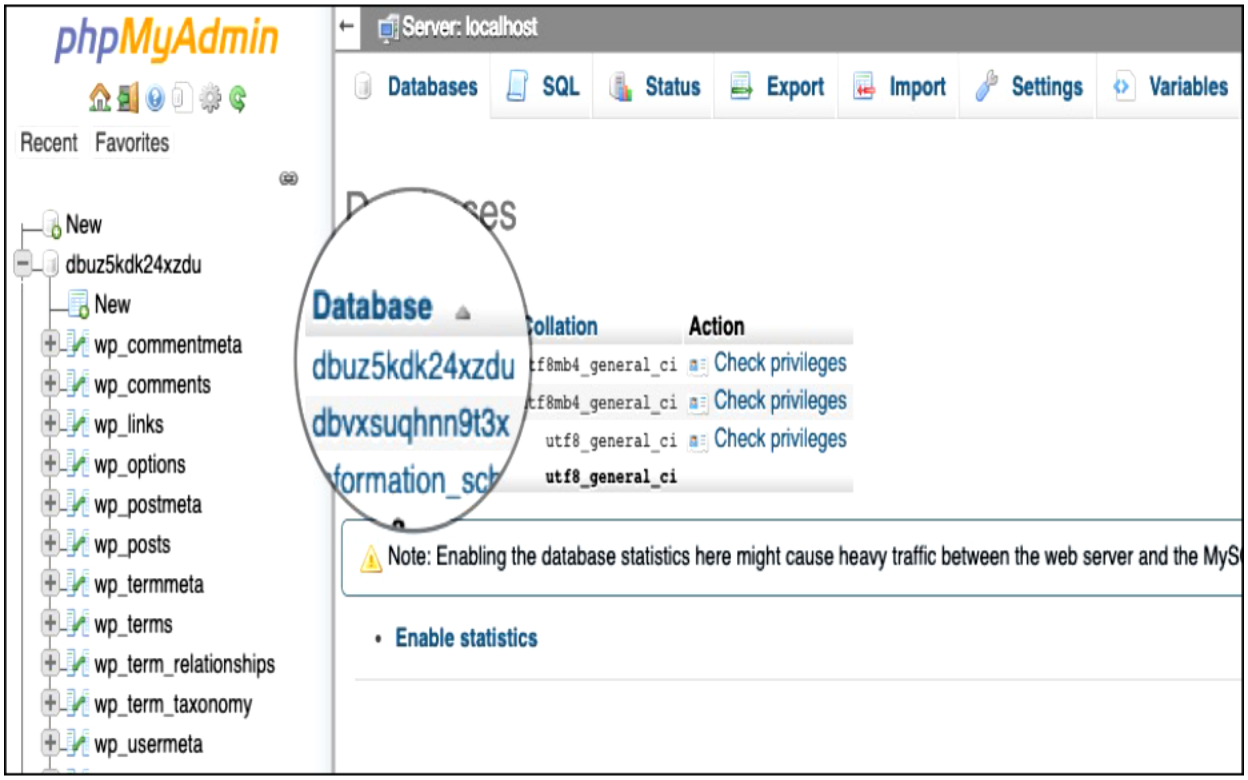
```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";
```

```
$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";
$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

3.4 Creating tables using PhpMyAdmin

1 Creating tables using PhpMyAdmin

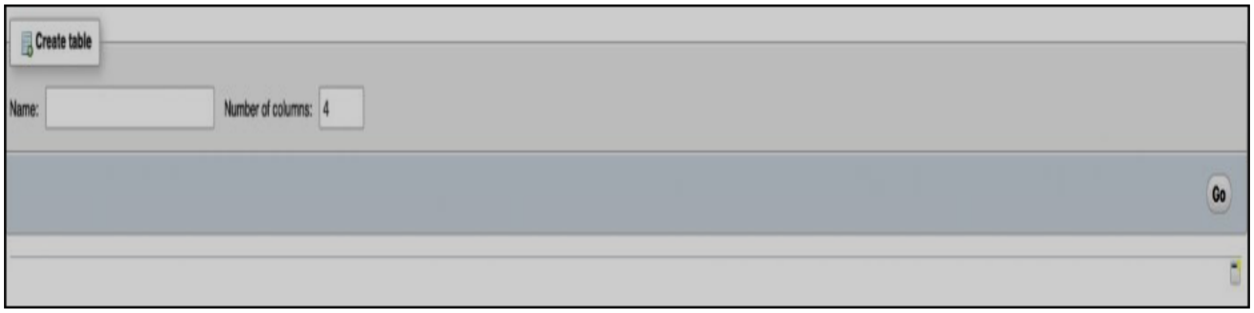
To create new tables inside a database, open the phpMyAdmin tool, click on the **Databases** tab and click on the name of the desired database.



Database	Collation	Action
dbuz5kdk24xzdu	utf8mb4_general_ci	Check privileges
dbvxsuqhnn9t3x	utf8mb4_general_ci	Check privileges
formation_sch	utf8_general_ci	Check privileges
	utf8_general_ci	

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MyS

On the new page that opens you will see a list of all the current tables inside the database and a section named **Create table**. In that section, in the **Name** field, input the desired new name of the table and then select the number of columns that the table should have via the **Number of columns** drop-down. When ready, click on **Go** to create the table.

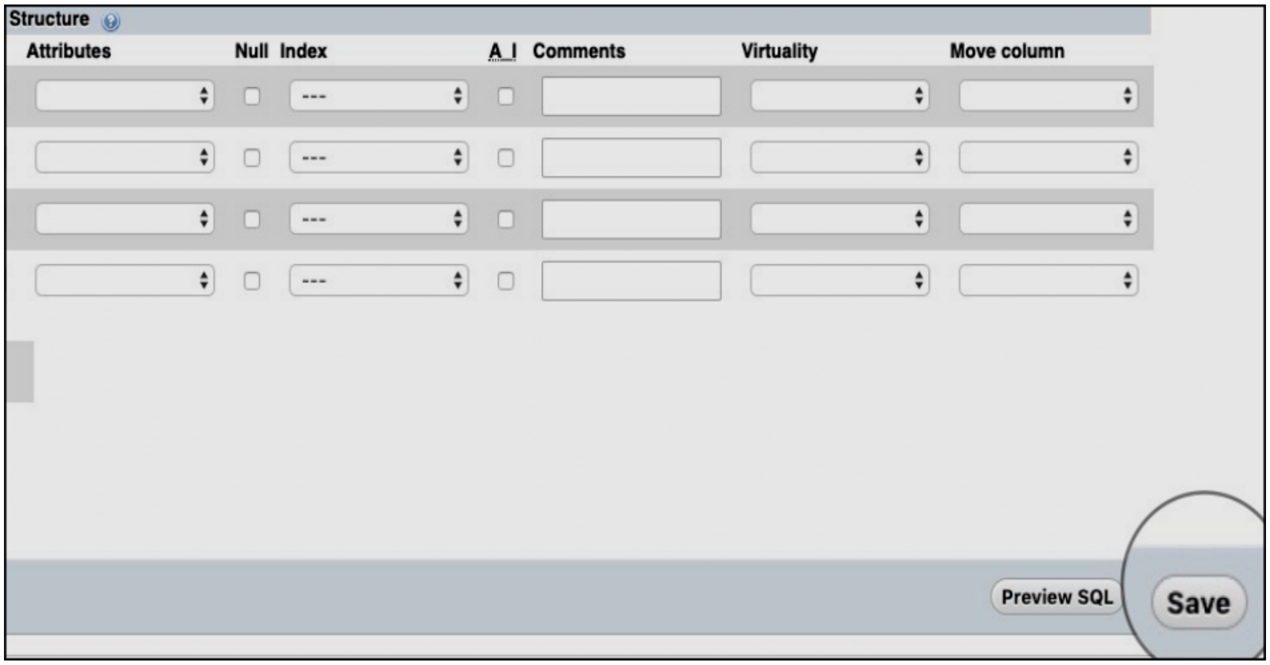


On the next page, you can configure the structure of the columns in the new table. The different fields there are:

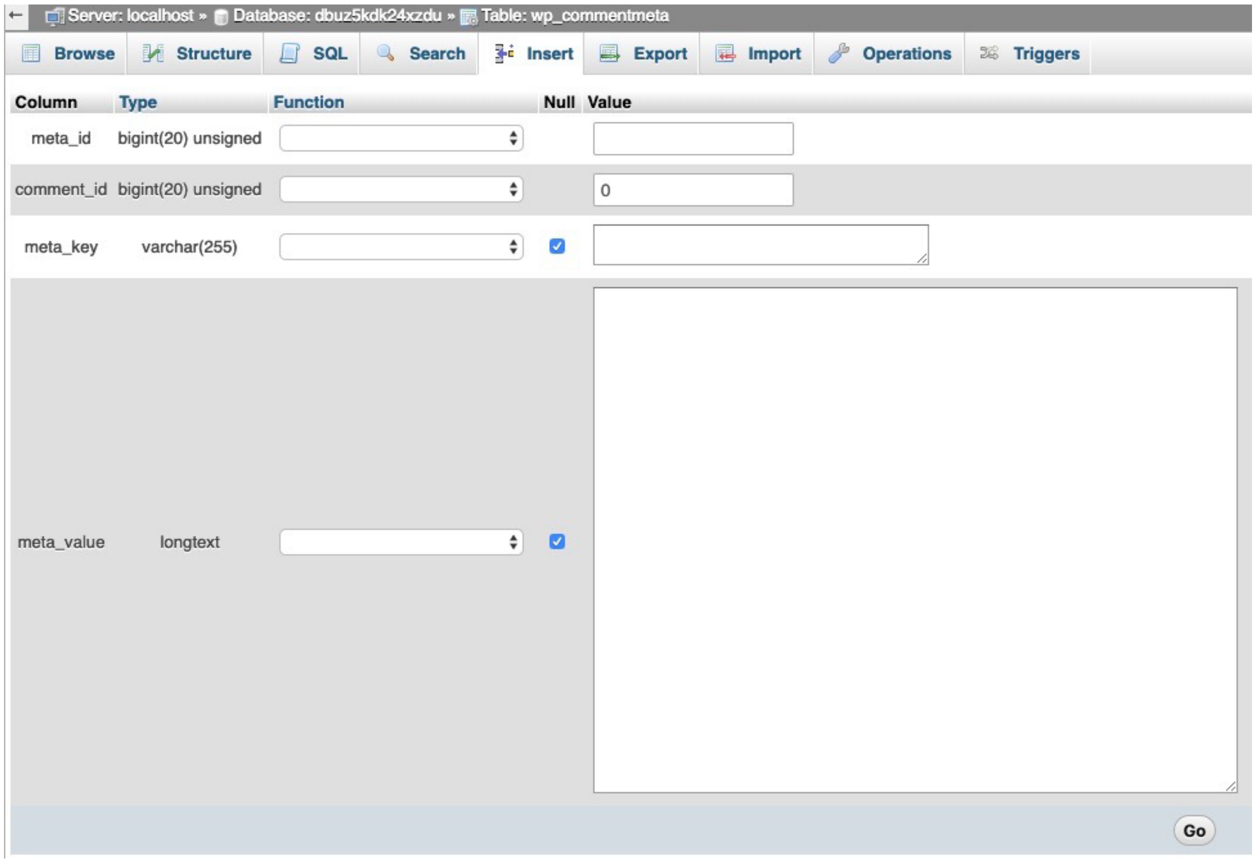
- **Name** – The name of the column;
- **Type** – The type of data, which will be stored in the corresponding column. More details about the possible choices can be found in the official MySQL Data Types documentation;
- **Length/Values** – The length of the field;
- **Default** – With this option, you can specify if the fields in the column would have a default value. This is useful when you want to have timestamps for the entries in each row;
- **Collation** – The data collation for each of the fields;
- **Attributes** – assign any special attributes to the fields;
- **Null** – Define whether the field value can be *NULL*. More about the *NULL* value can be found in the MySQL documentation;
- **Index** – Set the Index of the row. More information about the MySQL column indexes can be found in the MySQL documentation;
- **A_I** – Short for Auto Increment. If this option is enabled then the values in the fields of the column will be auto incremented;
- **Comments** – Here add comments, which will be included in the database SQL code.

After you configured the different columns, specify the **Collation** and **Engine** of the new table via their respective drop-downs.

When ready, click on **Save** to create the new table.



How to Add Content in a Database Table
To add records inside a database table, open the table with phpMyAdmin and click on the **Insert** tab.


Server: localhost » Database: dbuz5kdk24xzdu » Table: wp_commentmeta
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers
Go

Column	Type	Function	Null	Value
meta_id	bigint(20) unsigned			
comment_id	bigint(20) unsigned			0
meta_key	varchar(255)		✓	
meta_value	longtext		✓	

Enter the desired data in the corresponding fields and click on **Go** to store it. You can see the newly inserted record by clicking on the **Browse** tab.

3.5 Interaction with HTML form

1 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.
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.

2 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

3 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.

4 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.
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

5 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.

5.1 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.

5.2 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

6. 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'])): ?> //executed when form is submit
ted
      <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.

3.6 Validating HTML Form

Form Validation in PHP

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers
4. Validate Email
5. Input length
6. Button click

1. Empty String

The code below checks that the field is not empty. If the user leaves the required field empty, it will show an error message. Put these lines of code to validate the required field.

```
if (empty ($_POST["name"])) {
   $errMsg = "Error! You didn't enter the Name.";
          echo $errMsg;
} else {
   $name = $_POST["name"];
}
```

2. Validate String

The code below checks that the field will contain only alphabets and whitespace, for example - name. If the name field does not receive valid input from the user, then it will show an error message:

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
   $ErrMsg = "Only alphabets and whitespace are allowed.";
          echo $ErrMsg;
} else {
   echo $name;
}
```

3. Validate Number

The below code validates that the field will only contain a numeric value. **For example -** Mobile no. If the *Mobile no* field does not receive numeric data from the user, the code will display an error message:

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
   $ErrMsg = "Only numeric value is allowed.";
   echo $ErrMsg;
} else {
   echo $mobileno;
}
```

4. Validate Email

A valid email must contain @ and . symbols. PHP provides various methods to validate the email address. Here, we will use regular expressions to validate the email address.

The below code validates the email address provided by the user through HTML form. If the field does not contain a valid email address, then the code will display an error message:

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
   $ErrMsg = "Email is not valid.";
         echo $ErrMsg;
} else {
   echo "Your valid email address is: " .$email;
}
```

5. Input Length Validation

The input length validation restricts the user to provide the value between the specified

range, for Example - Mobile Number. A valid mobile number must have 10 digits.
The given code will help you to apply the length validation on user input:

```
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
   $ErrMsg = "Mobile must have 10 digits.";
         echo $ErrMsg;
} else {
   echo "Your Mobile number is: " .$mobileno;
}
```

6. Button Click Validate

The below code validates that the user click on submit button and send the form data to the server one of the following method - get or post.

```
if (isset ($_POST['submit']) {
   echo "Submit button is clicked.";
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      echo "Data is sent using POST method ";
   }
} else {
   echo "Data is not submitted";
}
```

3.7 Error checking or Exiting

PHP Error Handling

Error handling in PHP is simple. An error message with filename, line number and a message describing the error is sent to the browser.

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

Different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:
Example

```
<?php
$file=fopen("mytestfile.txt","r");
?>
```

If the file does not exist you might get an error like this:

```
Warning: fopen(mytestfile.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:
Example

```
<?php
if(file_exists("mytestfile.txt"))
{
        $file = fopen("mytestfile.txt", "r");
} else {
        die("Error: The file does not exist.");
}
?>
```

Or you can also write like this

```
<?php
        $file = fopen("mytestfile.txt", "r") or die("Error: The file does not exist.");
?>
```

Now if the file does not exist you get an error like this:

```
Error: The file does not exist.
```

3.8 Introduction to Regular Expression

What is Regular Expression

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

```
$exp = "/SPU/i";
```

In the example above, / is the delimiter, SPU is the pattern that is being searched for, and i is a modifier that makes the search case-insensitive.

The delimiter can be any character that is not a letter, number, backslash or space. The most common delimiter is the forward slash (/), but when your pattern contains forward slashes it is convenient to choose other delimiters such as # or ~.

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions. The preg_match(), preg_match_all() and preg_replace() functions are some of the most commonly used ones:

Function	**Description**
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

Using preg_match()

The preg_match() function will tell you whether a string contains matches of a pattern.

Example:

Use a regular expression to do a case-insensitive search for "SPU" in a string:

```
<?php
        $str = "Visit SPU";
        $pattern = "/SPU/i";
        echo preg_match($pattern, $str);
        // Outputs 1
?>
```

Using preg_match_all()

The preg_match_all() function will tell you how many matches were found for a pattern in a string.

Example:

Use a regular expression to do a case-insensitive count of the number of occurrences of "ain" in a string:

```
<?php
        $str = "The rain in SPAIN falls mainly on the plains.";
        $pattern = "/ain/i";
        echo preg_match_all($pattern, $str);
        // Outputs 4
?>
```

Using preg_replace()

The preg_replace() function will replace all of the matches of the pattern in a string with another string.

Example:

Use a case-insensitive regular expression to replace Microsoft with SPU in a string:

```
        <?php
        $str = "Visit Microsoft!";
        $pattern = "/microsoft/i";
        echo preg_replace($pattern, "SPU", $str);
        // Outputs "Visit SPU!"
        ?>
```

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	Description
I	Performs a case-insensitive search
M	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
U	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description

[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	**Description**
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	**Description**
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X n*'s

n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

Note: If your expression needs to search for one of the special characters you can use a backslash (\ ) to escape them. For example, to search for one or more question marks you can use the following expression: $pattern = '/\?+/';

Grouping

You can use parentheses ( ) to apply quantifiers to entire patterns. They also can be used to select parts of the pattern to be used as a match.

Example

Use grouping to search for the word "banana" by looking for ba followed by two instances of na:

```
<?php
$str = "Apples and bananas.";
$pattern = "/ba(na){2}/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

3.8 File Handling

PHP File Handling

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

PHP fopen() function is used to open file or URL and returns resource. The fopen() function accepts two arguments: $filename and $mode. The $filename represents the file to be opended and $mode represents the file mode for example read-only, read-write, write-only etc.

Syntax: resource fopen (string $filename, string $mode [,bool $use_include_path = false])

Example

```
<?php
        $handle = fopen("c:\\folder\\file.txt", "r");
?>
```

PHP Open File Mode:

Mode	Description

R	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
W	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
A	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
X	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in **read-write** mode.
C	Opens file in **write-only** mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in **read-write** mode.

PHP Close File - fclose()

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

Syntax: fclose (resource $handle)

Example

```
<?php
        fclose($handle);
?>
```

<u>Reading file in php</u>

PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character. The available PHP file read functions are given below.

- fread()
- fgets()
- fgetc()

<u>PHP Read File - fread()</u>

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax: string fread (resource $handle , int $length)

Example

```
<?php
        $filename = "c:\\myfile.txt";
        $handle = fopen($filename, "r");//open file in read mode
        $contents = fread($handle, filesize($filename));//read file
        echo $contents;//printing data of file
        fclose($handle);//close file
?>
```

Output:

```
hello php file
```

<u>PHP Read File - fgets()</u>

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

Syntax: string gets (resource $handle , int $length)

Example

```
<?php
        $fp = fopen("c:\\file1.txt", "r");//open file in read mode
        echo fgets($fp);
        fclose($fp);
?>
```

<u>PHP Read File - fgetc()</u>

The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax

string fgetc (resource $handle)

Example

```
<?php
        $fp = fopen("c:\\file1.txt", "r");//open file in read mode
        while(!feof($fp)) {
          echo fgetc($fp);
        }
        fclose($fp);
?>
```

PHP Write File - fwrite()

PHP fwrite() and fputs() functions are used to write data into file. To write data into file, you need to use w, r+, w+, x, x+, c or c+ mode.

The PHP fwrite() function is used to write content of the string into file.

Syntax: int fwrite (resource $handle , string $string [, int $length])

Example

```
<?php
        $fp = fopen('data.txt', 'w');//open file in write mode
        fwrite($fp, 'hello ');
        fwrite($fp, 'php file');
        fclose($fp);
        echo "File written successfully";
?>
```

Output : File written successfully

Note for overwriting the file:

The above example will overwrite the content of the file. If it requires to append the content of the file just replace the “a” with “w” in file open function.

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax : bool unlink (string $filename [, resource $context])

Example

```
<?php
        unlink('data.txt');
        echo "File deleted successfully";
?>
```