# PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $\_GET and $\_POST.

The form request may be get or post. To retrieve data from get request, we need to use $\_GET, for post request $\_POST.

## PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

*File: form1.html*

<form action="welcome.php" method="get">

Name: <input type="text" name="name"/>

<input type="submit" value="visit"/>

</form>

*File: welcome.php*

<?php

$name=$\_GET["name"];//receiving name field value in $name variable

echo "Welcome, $name";

?>

## PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

*File: form1.html*

<form action="login.php" method="post">

<table>

<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>

<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>

<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>

</table>

</form>

*File: login.php*

<?php

$name=$\_POST["name"];//receiving name field value in $name variable

$password=$\_POST["password"];//receiving password field value in $password variable

echo "Welcome: $name, your password is: $password";

?>

PHP - A Simple HTML Form

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

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

User Name:   
E-mail:   


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["username"]; ?><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:

### Example

<html>  
<body>  
  
<form action="welcome\_get.php" method="get">  
Name: <input type="text" name="username"><br>  
E-mail: <input type="text" name="email"><br>  
<input type="submit">  
</form>  
  
</body>  
</html>

Name:   
E-mail:   


and "welcome\_get.php" looks like this:

<html>  
<body>  
  
Welcome <?php echo $\_GET["username"]; ?><br>  
Your email address is: <?php echo $\_GET["email"]; ?>  
  
</body>  
</html>

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET vs. POST

Both GET and POST create an array (e.g. array( key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $\_GET and $\_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$\_GET is an array of variables passed to the current script via the URL parameters.

$\_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

**Note:** GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

**Developers prefer POST for sending form data.**

## Example 1: GET METHOD

Welcome

<?php echo $\_GET["username"]; ?>

<br>

Your email address is:

<?php echo $\_GET["email"]; ?>

<html>

<body>

<form action="<?php $\_PHP\_SELF ?>" method="get">

Name: <input type="text" name="username"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>

</body>

</html>

## Example 2: POST METHOD

Welcome

<?php echo $\_POST["username"]; ?>

<br>

Your email address is:

<?php echo $\_POST["email"]; ?>

<html>

<body>

<form action="<?php $\_PHP\_SELF ?>" method="post">

Name: <input type="text" name="username"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>

</body>

</html>

Welcome hhh  
Your email address is: dfdfs

Top of Form

Name:   
E-mail:   


Bottom of Form

# PHP Form Validation

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

**PHP Form Validation Example**

\* required field

Top of Form

Name:  \*  
  
E-mail:  \*  
  
Website:   
  
Comment:   
  
Gender: Female Male Other \*  
  


Bottom of Form

**Your Input:**

The validation rules for the form above are as follows:

he validation rules for the form above are as follows:

|  |  |
| --- | --- |
| **Field** | **Validation Rules** |
| Name | Required. + Must only contain letters and whitespace |
| E-mail | Required. + Must contain a valid email address (with @ and .) |
| Website | Optional. If present, it must contain a valid URL |
| Comment | Optional. Multi-line input field (textarea) |
| Gender | Required. Must select one |

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

Name: <input type="text" name="name">  
E-mail: <input type="text" name="email">  
Website: <input type="text" name="website">  
Comment: <textarea name="comment" rows="5" cols="40"></textarea>

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:  
<input type="radio" name="gender" value="female">Female  
<input type="radio" name="gender" value="male">Male  
<input type="radio" name="gender" value="other">Other

The Form Element

The HTML code of the form looks like this:

<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

When the form is submitted, the form data is sent with method="post".

**What is the $\_SERVER["PHP\_SELF"] variable?**  
  
The $\_SERVER["PHP\_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $\_SERVER["PHP\_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

**What is the htmlspecialchars() function?**  
  
The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

## Validate Form Data With PHP

## What is Validation ?

Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows −

* **Client-Side Validation** − Validation is performed on the client machine web browsers.
* **Server Side Validation** − After submitted by data, The data has sent to a server and perform validation checks in server machine.

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test\_input().

Now, we can check each $\_POST variable with the test\_input() function, and the script looks like this:

## Example 3: FORM VALIDATION

*<form.php>*

<html>

<head>

<title>PHP Form Validation</title>

</head>

<body>

<?php

// define variables and set to empty values

$name = $email = $gender = $comment = $website = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$name = test\_input($\_POST["name"]);

$email = test\_input($\_POST["email"]);

$website = test\_input($\_POST["website"]);

$comment = test\_input($\_POST["comment"]);

$gender = test\_input($\_POST["gender"]);

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>Student Registration</h2>

<form method = "post" action = "<?php $\_PHP\_SELF ?>">

<table>

<tr>

<td>Name:</td>

<td><input type = "text" name = "name"></td>

</tr>

<tr>

<td>E-mail:</td>

<td><input type = "text" name = "email"></td>

</tr>

<tr>

<td>Specific Time:</td>

<td><input type = "text" name = "website"></td>

</tr>

<tr>

<td>Class details:</td>

<td><textarea name = "comment" rows = "5" cols = "40"></textarea></td>

</tr>

<tr>

<td>Gender:</td>

<td>

<input type = "radio" name = "gender" value = "female">Female

<input type = "radio" name = "gender" value = "male">Male

</td>

</tr>

<tr>

<td>

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

</td>

</tr>

</table>

</form>

<?php

echo "<h2>Your Given details are as :</h2>";

echo $name;

echo "<br>";

echo $email;

echo "<br>";

echo $website;

echo "<br>";

echo $comment;

echo "<br>";

echo $gender;

?>

</body>

</html>

# PHP - COMPLETE FORM

This page explains about time real-time form with actions. Below example will take input fields as text, radio button, drop down menu, and checked box.

## Example 4: FORM VALIDATION WITH ACTIONS

<html>

<head>

<style>

.error {color: #FF0000;}

</style>

</head>

<body>

<?php

// define variables and set to empty values

$nameErr = $emailErr = $genderErr = $websiteErr = "";

$name = $email = $gender = $comment = $website = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (empty($\_POST["name"])) {

$nameErr = "Name is required";

}else {

$name = test\_input($\_POST["name"]);

}

if (empty($\_POST["email"])) {

$emailErr = "Email is required";

}else {

$email = test\_input($\_POST["email"]);

// check if e-mail address is well-formed

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

$emailErr = "Invalid email format";

}

}

if (empty($\_POST["website"])) {

$website = "";

}else {

$website = test\_input($\_POST["website"]);

}

if (empty($\_POST["comment"])) {

$comment = "";

}else {

$comment = test\_input($\_POST["comment"]);

}

if (empty($\_POST["gender"])) {

$genderErr = "Gender is required";

}else {

$gender = test\_input($\_POST["gender"]);

}

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>Student registration</h2>

<p><span class = "error">\* required field.</span></p>

<form method = "post" action = "<?php

echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

<table>

<tr>

<td>Name:</td>

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

<span class = "error">\* <?php echo $nameErr;?></span>

</td>

</tr>

<tr>

<td>E-mail: </td>

<td><input type = "text" name = "email">

<span class = "error">\* <?php echo $emailErr;?></span>

</td>

</tr>

<tr>

<td>Time:</td>

<td> <input type = "text" name = "website">

<span class = "error"><?php echo $websiteErr;?></span>

</td>

</tr>

<tr>

<td>Classes:</td>

<td> <textarea name = "comment" rows = "5" cols = "40"></textarea></td>

</tr>

<tr>

<td>Gender:</td>

<td>

<input type = "radio" name = "gender" value = "female">Female

<input type = "radio" name = "gender" value = "male">Male

<span class = "error">\* <?php echo $genderErr;?></span>

</td>

</tr>

<td>

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

</td>

</table>

</form>

<?php

echo "<h2>Your given values are as:</h2>";

echo $name;

echo "<br>";

echo $email;

echo "<br>";

echo $website;

echo "<br>";

echo $comment;

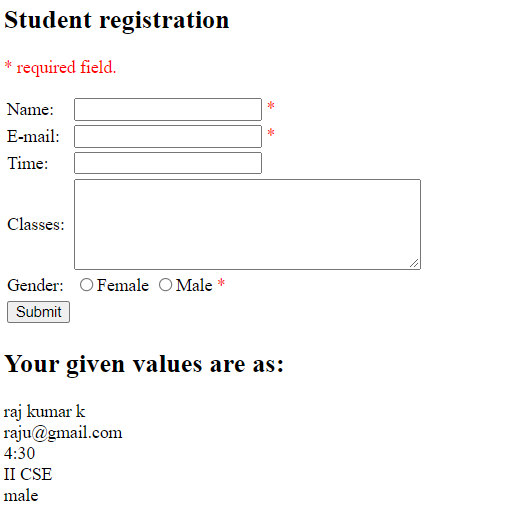
echo "<br>";

echo $gender;

?>

</body>

</html>



PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

***Note: PHP Cookie must be used before <html> tag***

## PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $\_COOKIE superglobal variable.

**Syntax :** bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path  [, string $domain  [,  bool $secure = false  [, bool $httponly = false ]]]]]] )

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

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

**Example:**

setcookie("CookieName", "CookieValue");/\* defining name and value only\*/

setcookie("CookieName", "CookieValue", 84600); // one day i.e. 84600 seconds

//using expiry in 1 hour(1\*60\*60 seconds or 3600 seconds)  24\*60\*60

setcookie("CookieName", "CookieValue", time()+24\*60\*60, "/mypath/", "mydomain.com", 1);

## PHP $\_COOKIE

PHP $\_COOKIE superglobal variable is used to get cookie.

## PHP Cookie Example

*File: cookie1.php*

<?php

setcookie("user", "Raju");  //name is user and value is raju

?>

<html>

<body>

<?php

**if**(!isset($\_COOKIE["user"])) {   //consider value is ramu

    echo "Sorry, cookie is not found!";

} **else** {

    echo "<br/>Cookie Value: " . $\_COOKIE["user"];

}

?>

</body>

</html>

Output:

Sorry, cookie is not found!

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

Cookie Value: Raju

## Example 2:

*File: cookie2.php*

## <?php $cookie\_name = "user"; $cookie\_value = "Raju"; setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/"); // 86400 = 1 day now total expire time is 30 days ?>

## <html> <body> <?php if(!isset($\_COOKIE[$cookie\_name])) {   echo "Cookie named '" . $cookie\_name . "' is not set!"; } else {   echo "Cookie '" . $cookie\_name . "' is set!<br>";   echo "Value is: " . $\_COOKIE[$cookie\_value]; } ?> </body> </html>

## PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

*File: cookie3.php*

<?php

setcookie ("CookieName", "", *time() - 3600*);

// set the expiration date to one hour ago

?>

<html>

<head>

<title>Deleting Cookies with PHP</title>

</head>

<body>

<?php echo "Deleted Cookies" ?>

</body>

</html>

# PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



## PHP session\_start() function

PHP session\_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

**Syntax :** bool session\_start ( void )

**Example:** session\_start();

## PHP $\_SESSION

PHP $\_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

**Example: Store information :** $\_SESSION["user"] = "Sachin";

**Example: Get information : e**cho $\_SESSION["user"];

## PHP Session Example

*File: session1.php*

<?php

session\_start();

?>

<html>

<body>

<?php

$\_SESSION["user"] = "Ramu";  //only one session

// $\_SESSION["user2"] = "Raju"; 2nd session

echo "Session information are set successfully.<br/>";

?>

<a href="session2.php">Visit next page</a>

</body>

</html>

*File: session2.php*

<?php

session\_start();

?>

<html>

<body>

<?php

echo "User is: ".$\_SESSION["user"];

?>

</body>

</html>

## PHP Session Counter Example

*File: sessioncounter.php*

<?php

   session\_start();

**if** (!isset($\_SESSION['counter'])) {

      $\_SESSION['counter'] = 1;

   } **else** {

      $\_SESSION['counter']++;

   }

   echo ("Page Views: ".$\_SESSION['counter']);

?>

## PHP Destroying Session

PHP session\_destroy() function is used to destroy all session variables completely.

*File: session3.php*

<?php

session\_start();

session\_destroy();

?>

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()

The PHP fopen() function is used to open a file.

**Syntax:** fopen ( string $filename , string $mode [, bool $use\_include\_path = false [, resource $context ]] )

**Example:**

<?php

$handle = fopen("c:\\folder\\file.txt", "r");

?>

## PHP Close File - fclose()

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

**Syntax: b**ool fclose ( resource $handle )

**Example**

<?php

fclose($handle);

?>

## PHP Read File - fread()

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

**Syntax: s**tring fread ( resource $handle , int $length )

**Example**

<?php

$filename = "readfile.txt";

$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//open and read file

echo $contents;//printing data of file

fclose($handle);//close file

?>

Output

hello php file

## PHP Write File - fwrite()

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('writefile.txt', 'w');//open file in write mode

fwrite($fp, 'hello ');

fwrite($fp, 'php file');

fwrite($fp, 'welcome');

fclose($fp);

echo "File written successfully";

?>

Output

File written successfully

## 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";

?>

PHP Open File

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 opened 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 [, resource $context ]] )

## 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 Open File Example

<?php

$handle = fopen("c:\\folder\\file.txt", "r");

?>

PHP Read File

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()

## PHP Read File - fread()

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

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

**$handle** represents file pointer that is created by fopen() function.

**$length** represents length of byte to be read.

## Example

<?php

$filename = "c:\\file1.txt";

$fp = fopen($filename, "r");//open file in read mode

$contents = fread($fp, filesize($filename));//read file

echo "<pre>$contents</pre>";//printing data of file

fclose($fp);//close file

?>

Output:

this is first line

this is another line

this is third line

## PHP Read File - fgets()

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

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

## Example

<?php

$fp = fopen("c:\\file1.txt", "r");//open file in read mode

echo fgets($fp);

fclose($fp);

?>

Output

this is first line

## PHP Read File - fgetc()

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);

?>

Output

this is first line this is another line this is third line

PHP Write File

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.

## PHP Write File - fwrite()

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');//opens file in write-only mode

fwrite($fp, 'welcome ');

fwrite($fp, 'to php file write');

fclose($fp);

echo "File written successfully";

?>

Output: data.txt

welcome to php file write

## PHP Overwriting File

If you run the above code again, it will erase the previous data of the file and writes the new data. Let's see the code that writes only new data into data.txt file.

<?php

$fp = fopen('data.txt', 'w');//opens file in write-only mode

fwrite($fp, 'hello');

fclose($fp);

echo "File written successfully";

?>

Output: data.txt

hello

## PHP Append to File

If you use **a** mode, it will not erase the data of the file. It will write the data at the end of the file. Visit the next page to see the example of appending data into file

# PHP Append to File

You can append data into file by using a or a+ mode in fopen() function. Let's see a simple example that appends data into data.txt file.

Let's see the data of file first.

data.txt

welcome to php file write

## PHP Append to File - fwrite()

The PHP fwrite() function is used to write and append data into file.

**Example**

<?php

$fp = fopen('data.txt', 'a');//opens file in append mode

fwrite($fp, ' this is additional text ');

fwrite($fp, 'appending data');

fclose($fp);

echo "File appended successfully";

?>

Output: data.txt

welcome to php file write this is additional text appending data

PHP Delete File

In PHP, we can delete any file using unlink() function. The unlink() function accepts one argument only: file name. It is similar to UNIX C unlink() function.

PHP unlink() generates E\_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE.

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

**$filename** represents the name of the file to be deleted.

## PHP Delete File Example

<?php

$status=unlink('data.txt');    //true

**if**($status) //if the status is true then file is deleted

{

echo "File deleted successfully";

}**else**{

echo "Sorry!";    // if you want to delete the file that file is not exist in the current directory

}

?>

Output

File deleted successfully

# PHP: LIST ALL FILES IN A DIRECTORY

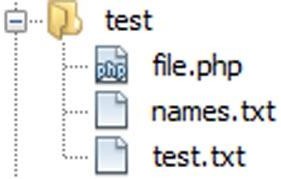
It deals with the process of how to list all files in a directory using PHP.

We will do this using PHP’s glob function, which allows us to retrieve a list of file pathnames that match a certain pattern.

For this example, I have created a folder called “test”. Inside the folder, I have created three files:

* + test.txt
  + names.txt
  + file.php

Here is a screenshot of the directory:



<?php

//Get a list of file paths using the glob function.

$fileList = glob('test/\*'); // test is the folder or directory … wrt this directory how many files are available

//Loop through the array that glob returned. Here \* indicates all the types of files

foreach($fileList as $filename)

{

//Simply print them out onto the screen.

echo $filename, '<br>';

}

The result will look something like this:

1. test/file.php
2. test/names.txt
3. test/test.txt

However, what if we wanted to list all files with a particular file extension? i.e. What if we only want to list the .txt files and not the **.php** file that is currently present?

// $fileList = glob('test/\*');

Well, the solution is pretty simple:

//Get a list of all files ending in .txt 2 $fileList = glob('test/\*.txt');

$fileList = glob('test/\*.php'); //display only php files

In the code snippet above, we told the glob function to return a list of file pathnames that ended .txt

Warning: In some cases, the folder may have subdirectories. In cases where you are listing everything that is inside a specified folder, these subdirectories will be returned by the glob function. To avoid printing out or interacting with subdirectories, you can simply use the is\_file function to confirm that the file pathname in question leads to an actual file:

<?php

$fileList = glob('test/\*');

foreach($fileList as $filename){

//Use the is\_file function to make sure that it is not a directory.

if(is\_file($filename))

{

echo $filename, '<br>';

}

}

# PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

## PHP $\_FILES

The PHP global $\_FILES contains all the information of file. By the help of $\_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is filename.

### $\_FILES['filename']['name'] returns file name.

### $\_FILES['filename']['type'] returns MIME type of the file.

### $\_FILES['filename']['size'] returns size of the file (in bytes).

### $\_FILES['filename']['tmp\_name'] returns temporary file name of the file which was stored on the server.

### $\_FILES['filename']['error'] returns error code associated with this file.

## move\_uploaded\_file() function

The move\_uploaded\_file() function moves the uploaded file to a new location. The move\_uploaded\_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

**Syntax:** bool move\_uploaded\_file ( string $filename , string $destination )

## PHP File Upload Example

*File: uploadform.html*

<form action="uploader.php" method="post">

    Select File:

    <input type="file" name="fileToUpload"/>

    <input type="submit" value="Upload Image" name="submit"/>

</form>

*File: uploader.php*

<?php

$target\_path = "e:/";

$target\_path = $target\_path.basename( $\_FILES['fileToUpload']['name']);

**if**(move\_uploaded\_file($\_FILES['fileToUpload']['tmp\_name'], $target\_path)) {

    echo "File uploaded successfully!";

} **else**{

    echo "Sorry, file not uploaded, please try again!";

}

?>