Chapter - II

2.1. File Handling and Directories

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.

```
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.

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

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 a 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
        ?>
```

PHP Read File - fgets()

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);
        ?>
```

PHP Read File - fgetc()

Syntax

```
string fgetc ( resource $handle )
```

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

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 ] )
```

```
<?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";
?>
```

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

```
<?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";
?>
```

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
$status=unlink('data.txt');
if($status){
echo "File deleted successfully";
}else{
echo "Sorry!";
}
?>
```

2.2. Creating a Dynamic HTML Form with PHP

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";
?>
```

Form Processing

- For example, if you want to get the details of visitors to your website, and send them good thoughts, you can collect the user information by means of form processing. Then, the information can be validated either at the client-side or on the server-side. The final result is sent to the client through the respective web browser. To create a HTML form, form tag should be used.

Attributes of Form Tag:

Attribute	Description
name or id	It specifies the name of the form and is used to identify individual forms.
action	It specifies the location to which the form data has to be sent when the form is submitted.

method	It specifies the HTTP method that is to be used when the form is submitted. The possible values are get and post. If get method is used, the form data are visible to the users in the url. Default HTTP method is get.
encType	It specifies the encryption type for the form data when the form is submitted.
novalidate	It implies the server not to verify the form data when the form is submitted.

Controls used in forms: Form processing contains a set of controls through which the client and server can communicate and share information. The controls used in forms are:

- Textbox: Textbox allows the user to provide single-line input, which can be used for getting values such as names, search menu and etc.
- Textarea: Textarea allows the user to provide multi-line input, which can be used for getting values such as an address, message etc.
- DropDown: Dropdown or combobox allows the user to provide select a value from a list of values.
- Radio Buttons: Radio buttons allow the user to select only one option from the given set of options.
- CheckBox: Checkbox allows the user to select multiple options from the set of given options.
- Buttons: Buttons are the clickable controls that can be used to submit the form.

Form Validation:

Form validation is done to ensure that the user has provided the relevant information. Basic validation can be done using HTML elements. For example, in the above script, the email address text box is having a type value as "email", which prevents the user from entering the incorrect value for an email. Every form field in the above script is followed by a required attribute, which will intimate the user not to leave any field empty before submitting the form. PHP methods and arrays used in form processing are:

- isset(): This function is used to determine whether the variable or a form control is having a value or not.
- $_GET[]: It is used the retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the url as the parameter.
- $_POST[]: It is used the retrieve the information from the form control through the HTTP POST method. IT takes name attribute of corresponding form control as the parameter.
- $_REQUEST[]: It is used to retrieve an information while using a database.

```
<?php
if (isset($_POST['submit']))
{
        if ((!isset($_POST['firstname'])) || (!isset($_POST['lastname'])) ||
                (!isset($_POST['address'])) || (!isset($_POST['emailaddress'])) ||
                (!isset($_POST['password'])) || (!isset($_POST['gender'])))
        {
                $error = "*" . "Please fill all the required fields";
        }
        else
        {
                $firstname = $_POST['firstname'];
                $lastname = $_POST['lastname'];
                $address = $_POST['address'];
                $emailaddress = $_POST['emailaddress'];
                $password = $_POST['password'];
                $gender = $_POST['gender'];
        }
}
?>
<html>

<head>
        <title>Simple Form Processing</title>
</head>
```

```
<body>
        <h1>Form Processing using PHP</h1>
        <fieldset>
                <form id="form1" method="post" action="form.php">
                        <?php
                                if (isset($_POST['submit']))
                                {
                                        if (isset($error))
                                        {
                                                echo "<p style='color:red;'>"
                                                . $error . "</p>";
                                        }
                                }
                                ?>

                                FirstName:
                                <input type="text" name="firstname"/>
                                <span style="color:red;">*</span>
                                <br>
                                <br>
                                Last Name:
                                <input type="text" name="lastname"/>
                                <span style="color:red;">*</span>
                                <br>
                                <br>
                                Address:
                                <input type="text" name="address"/>
                                <span style="color:red;">*</span>
                                <br>
                                <br>
                                Email:
                                <input type="email" name="emailaddress"/>
                                <span style="color:red;">*</span>
                                <br>
                                <br>
                                Password:
                                <input type="password" name="password"/>
                                <span style="color:red;">*</span>
```

```
                                        <br>
                                        <br>
                                        Gender:
                                        <input type="radio"
                                                  value="Male"
                                                  name="gender"> Male
                                        <input type="radio"
                                                  value="Female"
                                                  name="gender">Female
                                        <br>
                                        <br>
                                        <input type="submit" value="Submit" name="submit"
/>
                    </form>
          </fieldset>
          <?php
          if(isset($_POST['submit']))
          {
                    if(!isset($error))
                    {
                                        echo"<h1>INPUT RECEIVED</h1><br>";
                                        echo "<table border='1'>";
                                        echo "<thead>";
                                        echo "<th>Parameter</th>";
                                        echo "<th>Value</th>";
                                        echo "</thead>";
                                        echo "<tr>";
                                        echo "<td>First Name</td>";
                                        echo "<td>".$firstname."</td>";
                                        echo "</tr>";
                                        echo "<tr>";
                                        echo "<td>Last Name</td>";
                                        echo "<td>".$lastname."</td>";
                                        echo "</tr>";
                                        echo "<tr>";
                                        echo "<td>Address</td>";
                                        echo "<td>".$address."</td>";
                                        echo "</tr>";
                                        echo "<tr>";
                                        echo "<td>Email Address</td>";
                                        echo "<td>" .$emailaddress."</td>";
                                        echo "</tr>";
                                        echo "<tr>";
                                        echo "<td>Password</td>";
                                        echo "<td>".$password."</td>";
                                        echo "</tr>";
                                        echo "<tr>";
                                        echo "<td>Gender</td>";
                                        echo "<td>".$gender."</td>";
                                        echo "</tr>";
                                        echo "</table>";
                    }
          }
          ?>
</body>

</html>
```

HTTP GET	HTTP POST
In GET method we can not send large amount of data rather limited data of some number of characters is sent because the request parameter is appended into the URL.	In POST method large amount of data can be sent because the request parameter is appended into the body.
GET request is comparatively better than Post so it is used more than the Post request.	POST request is comparatively less better than Get method, so it is used less than the Get request.
GET requests are only used to request data (not modify)	POST requests can be used to create and modify data.

GET request is comparatively less secure because the data is exposed in the URL bar.	POST request is comparatively more secure because the data is not exposed in the URL bar.
Request made through GET method are stored in Browser history.	Request made through POST method is not stored in Browser history.
GET method request can be saved as bookmark in browser.	POST method request can not be saved as bookmark in browser.
Request made through GET method are stored in cache memory of Browser.	Request made through POST method are not stored in cache memory of Browser.
Data passed through GET method can be easily stolen by attackers as the data is visible to everyone.GET requests should never be used when dealing with sensitive data	Data passed through POST method can not be easily stolen by attackers as the URL Data is not displayed in the URL
In GET method only ASCII characters are allowed.	In POST method all types of data is allowed.

2.3. Database Connectivity with MySQL,Performing basic database operations (CRUD)

PHP MySQL Connect

Since PHP 5.5, mysql_connect() extension is deprecated. Now it is recommended to use one of the 2 alternatives.

- mysqli_connect()
- PDO::__construct()

PHP mysqli_connect()

PHP mysqli_connect() function is used to connect with MySQL database. It returns resource if connection is established or null.

Syntax

    resource mysqli_connect (server, username, password)

PHP mysqli_close()

PHP mysqli_close() function is used to disconnect with MySQL database. It returns true if connection is closed or false.

Syntax

    bool mysqli_close(resource $resource_link)

PHP MySQL Connect Example

```
<?php
$host = 'localhost:3306';  //write your port number
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

PHP MySQLi Create Database Example - mysqli_query()

```
<?php
$host = 'localhost:3306'; //Write your port number
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
```

```
{
  die('Could not connect: ' . mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'CREATE Database mydb';
if(mysqli_query( $conn,$sql)){
  echo "Database mydb created successfully.";
}else{
echo "Sorry, database creation failed ".mysqli_error($conn);
}
mysqli_close($conn);
?>
```

PHP MySQL Create Table

```
<?php
$host = 'localhost:3306';  //Write your port number
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = "create table emp5(id INT AUTO_INCREMENT,name
VARCHAR(20) NOT NULL,
emp_salary INT NOT NULL,primary key (id))";
if(mysqli_query($conn, $sql)){
 echo "Table emp5 created successfully";
}else{
echo "Could not create table: ". mysqli_error($conn);
}

mysqli_close($conn);
```

```
?>
```

PHP MySQLi Insert Record Example

```
<?php
$host = 'localhost:3306';  //Write your port number

$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
if(mysqli_query($conn, $sql)){
 echo "Record inserted successfully";
}else{
echo "Could not insert record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQLi Update Record Example

```
<?php
$host = 'localhost:3306';  //Write your port number

$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
```

```
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$id=2;
$name="Rahul";
$salary=80000;
$sql = "update emp4 set name=\"$name\", salary=$salary where id=$id";
if(mysqli_query($conn, $sql)){
 echo "Record updated successfully";
}else{
echo "Could not update record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQLi Delete Record Example

```
?php
$host = 'localhost:3306';  //Write your port number

$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$id=2;
$sql = "delete from emp4 where id=$id";
```

```
if(mysqli_query($conn, $sql)){
 echo "Record deleted successfully";
}else{
echo "Could not deleted record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQL Select Query

There are two other MySQLi functions used in select query.

- mysqli_num_rows(mysqli_result $result): returns number of rows.
- mysqli_fetch_assoc(mysqli_result $result): returns row as an associative array. Each key of the array represents the column name of the table. It return NULL if there are no more rows.

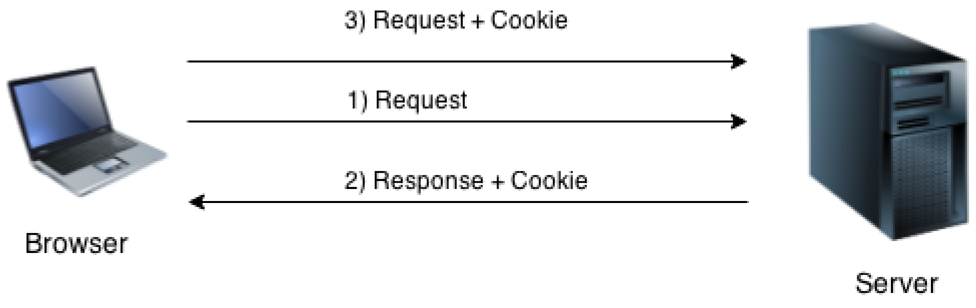
```
<?php
$host = 'localhost:3306';  //Write your port number

$user = '';
$pass = '';
$dbname = 'test';
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'SELECT * FROM emp4';
$retval=mysqli_query($conn, $sql);

if(mysqli_num_rows($retval) > 0){
 while($row = mysqli_fetch_assoc($retval)){
    echo "EMP ID :{$row['id']}  <br> ".
         "EMP NAME : {$row['name']} <br> ".
         "EMP SALARY : {$row['salary']} <br> ".
```

```
        "--------------------------------<br>";
    } //end of while
    }else{
    echo "0 results";
    }
    mysqli_close($conn);
    ?>
```

2.4 Working with GET, POST, REQUEST, SESSION, and COOKIE Variables

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.

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 ]]]]]] )
```

Example

```
setcookie("CookieName", "CookieValue");/* defining name and value only*/
setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)
setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);
```

PHP $_COOKIE

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

```
$value=$_COOKIE["CookieName"];//returns cookie valu
```

PHP Cookie Example

File: cookie1.php

```
<?php
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
   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: Sonoo

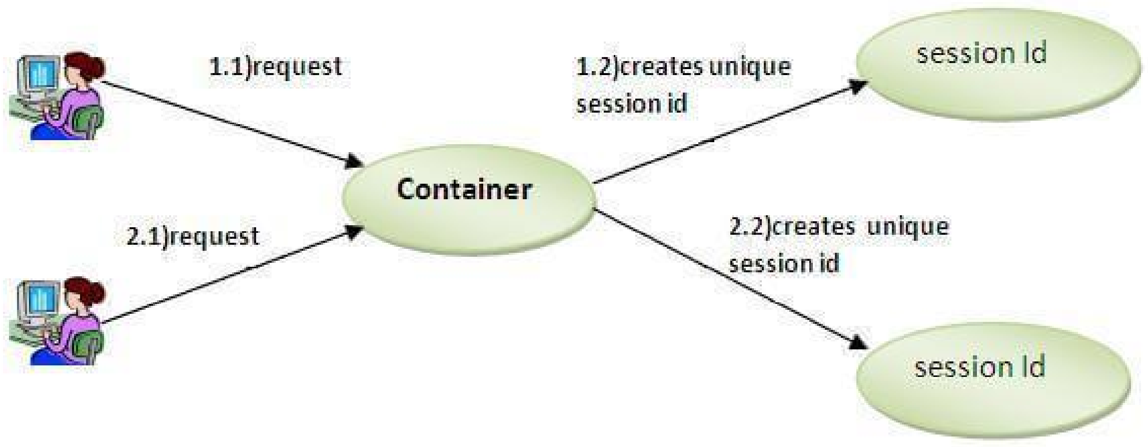
PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.
File: cookie1.php

```
<?php
setcookie ("CookieName", "", time() - 3600);// set the expiration date to one
hour ago
?>
```

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

```
echo $_SESSION["user"];
```

PHP Session Example

File: session1.php

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
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();
        ?>
```

$_SERVER

- $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

```
<!DOCTYPE html>
<html>
<body>

<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>

</body>
</html>
```

$_REQUEST

- $_REQUEST is a PHP super global variable which contains submitted form data, and all cookie data.
- In other words, $_REQUEST is an array containing data from $_GET, $_POST, and $_COOKIE.

- You can access this data with the $_REQUEST keyword followed by the name of the form field, or cookie, like this:

  $_REQUEST['firstname']

Using $_REQUEST on $_POST Requests

POST request are usually data submitted from an HTML form.

```
HTML form
<html>
<body>

<form method="post" action="demo_request.php">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

</body>
</html>
```

When a user clicks the submit button, the form data is sent to a PHP file specified in the action attribute of the <form> tag.
In the action file we can use the $_REQUEST variable to collect the value of the input field.

```
PHP file
$name = $_REQUEST['fname'];
echo $name;
```

In the example below we have put the HTML form and PHP code in the same PHP file.

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

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = htmlspecialchars($_REQUEST['fname']);
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```