PHP PROGRAMMING

UNIT-3

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.

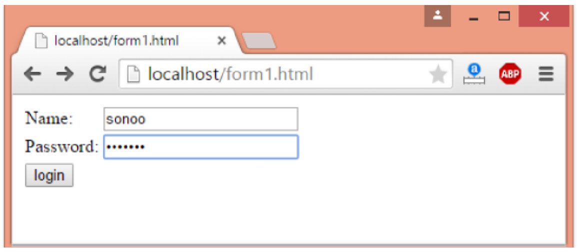
Let's see a simple example to receive data from post request in PHP.

File: form1.html

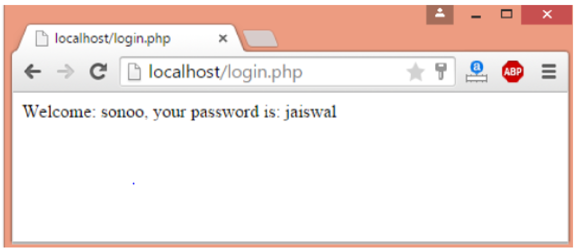
```
<form action="login.php" method="post">

<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";

?>
```



Name: sonoo
Password: •••••••
login



Welcome: sonoo, your password is: jaiswal

The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE. We will discuss $_COOKIE variable when we will explain about cookies.

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

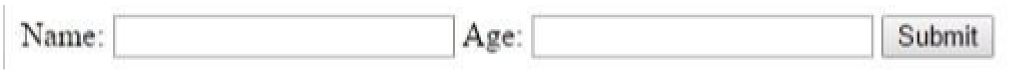
Try out following example by putting the source code in test.php script.

```
<?php
```

```
 if( $_REQUEST["name"] || $_REQUEST["age"] ) {
 echo "Welcome ". $_REQUEST['name']. "<br />";
 echo "You are ". $_REQUEST['age']. " years old.";
 exit();
 }
?>
<html>
 <body>
 <form action = "<?php $_PHP_SELF ?>" method = "POST">
 Name: <input type = "text" name = "name" />
 Age: <input type = "text" name = "age" />
 <input type = "submit" />
 </form>
</body>
</html>
```

Here $_PHP_SELF variable contains the name of self script in which it is being called.

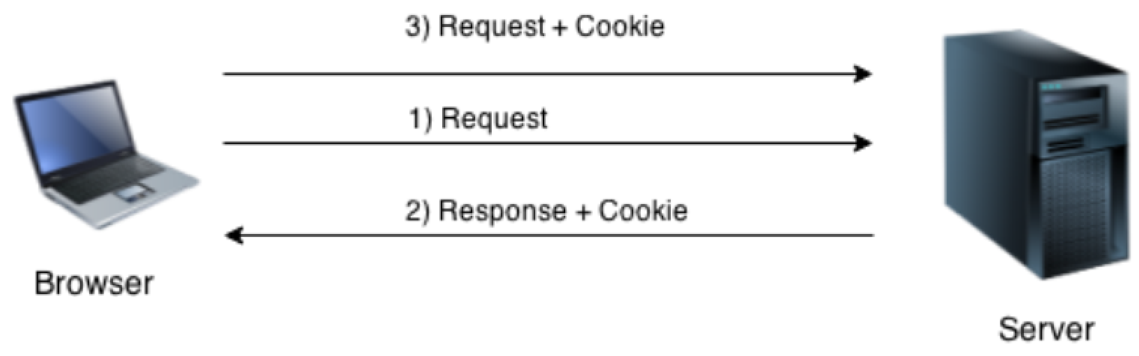
It will produce the following result



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.

There are three steps involved in identifying returning users –

• Server script sends a set of cookies to the browser. For example name, age,

or identification number etc.

• Browser stores this information on local machine for future use.

• When next time browser sends any request to web server then it sends those

cookies information to the server and server uses that information to identify

the user

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 value
```

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

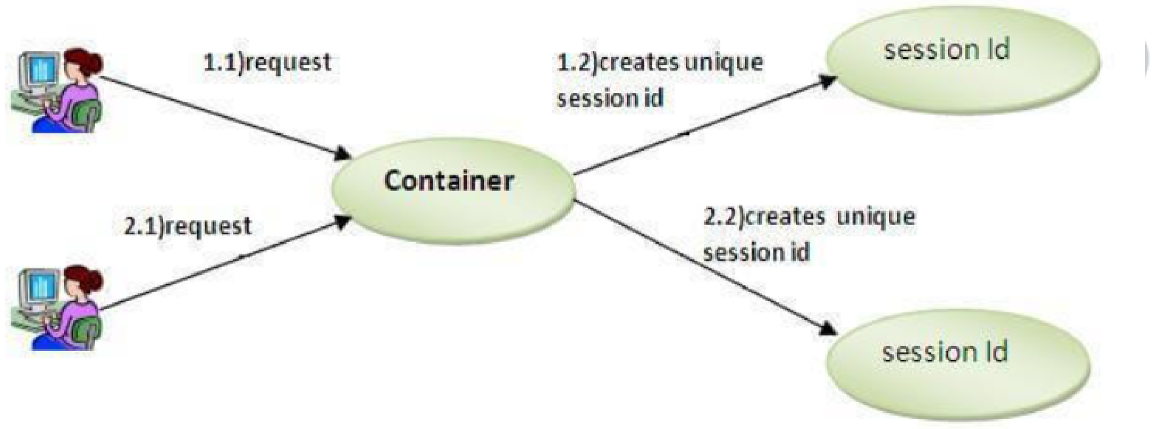
```
<?php

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.



Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new

PHP session and set some session variables:

Example

```
<?php

// Start the session
```

```
session_start();

?>

<!DOCTYPE html>

<html>

<body>

<?php

// Set session variables

$_SESSION["favcolor"] = "green";

$_SESSION["favanimal"] = "cat";

echo "Session variables are set.";

?>

</body>

</html>
```

Note: The session_start() function must be the very first thing in your document.

Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page

(session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php

session_start();

?>

<!DOCTYPE html>
```

```
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSION);
?>
</body>
</html>
```

<u>How does it work? How does it know it's me?</u>

Most sessions set a user-key on the user's computer that looks something like this:

765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
```

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.

or

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP readfile() Function

The readfile() function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup Language

PHP = PHP Hypertext Preprocessor

SQL = Structured Query Language

SVG = Scalable Vector Graphics

XML = EXtensible Markup Language

The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

Example

```php
<?php
echo readfile("webdictionary.txt");
?>
```

The readfile() function is useful if all you want to do is open up a file and read its contents.

PHP Open File - fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

We will use the text file, "webdictionary.txt", during the lessons:

AJAX = Asynchronous JavaScript and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup Language

PHP = PHP Hypertext Preprocessor

SQL = Structured Query Language

SVG = Scalable Vector Graphics

XML = EXtensible Markup Language

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified

file:

Example

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Tip: The fread() and the fclose() functions will be explained below.

The file may be opened in one of the following modes:

Modes Description

r:

Open a file for read only. File pointer starts at the beginning of the file

w:

Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a:

Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x:

Creates a new file for write only. Returns FALSE and an error if file already exists

r+:

Open a file for read/write. File pointer starts at the beginning of the file

w+

Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a+

Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x+

Creates a new file for read/write. Returns FALSE and an error if file already exists

PHP Read File - fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose()

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

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen("testfile.txt", "w")
```

<u>PHP File Permissions</u>

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite()

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php

$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");

$txt = "John Doe\n";

fwrite($myfile, $txt);

$txt = "Jane Doe\n";

fwrite($myfile, $txt);

fclose($myfile);

?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe".

After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:

```
John Doe

Jane Doe
```

PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
```

```
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");

$txt = "Mickey Mouse\n";

fwrite($myfile, $txt);

$txt = "Minnie Mouse\n";

fwrite($myfile, $txt);

fclose($myfile);

?>
```

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

```
Mickey Mouse

Minnie Mouse
```

Exception Handling in PHP

Exception handling is a powerful mechanism of PHP, which is used to handle runtime errors (runtime errors are called exceptions). So that the normal flow of the application can be maintained.

The main purpose of using exception handling is to maintain the normal execution of the application.

Note: Exception handling is required when an exception interrupts the normal execution of the program or application.

Basic Use of Exceptions

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception"message.

"Exception handling is almost similar in all programming languages. It changes the normal flow of the program when a specified error condition occurs, and this condition is known as exception. PHP offers the following keywords for this purpose:"

<u>try:</u>The try block contains the code that may have an exception or where an exception can arise. When an exception occurs inside the try block during runtime of code, it is caught and resolved in catch block. The try block must be followed by catch or finally block. A try block can be followed by minimum one and maximum any number of catch blocks.

<u>catch:</u>The catch block contains the code that executes when a specified exception is thrown. It is always used with a try block, not alone. When an exception occurs, PHP finds the matching catch block.

<u>throw -</u>It is a keyword used to throw an exception. It also helps to list all the exceptions that a function throws but does not handle itself.

<u>Remember that each throw must have at least one "catch</u>".