**PHP – Sessions**

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored.

This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the php.ini file called session save path . Bore using any session variable make sure you have setup this path.

When a session is started following things happen −

PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as

3c7foj34c3jj973hjkop2fc937e3443.

A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.

A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sass\_ i.e.

sess\_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

A PHP session is easily started by making a call to the session start () function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to Session start () at the beginning of the page.

Session variables are stored in associative array called $\_SESSION []. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.

Make use of asset () function to check if session variable is already set or not.

Put this code in a test.php file and load thisfile many times to see the result −

<?php

session\_start();

if( isset( $\_SESSION['counter'] ) )

{

$\_SESSION['counter'] += 1;

}

else

{

$\_SESSION['counter'] = 1;

}

$msg = "You have visited this page ".

$msg .= "in this session.";

?>

<html>

<head>

<title>Setting up a PHP session</t

</head>

<body>

<?php echo ( $msg ); ?>

</body>

</html>

It will produce the following result −

You have visited this page 1in this session.

A PHP session can be destroyed by

session\_destroy() function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.

Here is the example to unset a single variable

−

<?php

unset($\_SESSION['counter']);

?>

Here is the call which will destroy all the session variables −

<?php

session\_destroy();

?>

You don't need to call start session () function to start a session when a user visits your site if you can set session. Auto starts variable to 1 in php.ini file.

There may be a case when a user does not allow storing cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session name=session\_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

<?php

session\_start();

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

$\_SESSION['counter'] = 1;

}

else {

$\_SESSION['counter']++;

}

$msg = "You have visited this page ".

$msg .= "in this session.";

echo ( $msg );

?>

<p>

To continue click following link <br

<a href="nextpage.php?<?php echo htm

</p>

It will produce the following result −

You have visited this page 1in this session.

To continue click following link

When you have an import script that takes long to execute, the browser seem to lock up and you cannot access the website anymore. This is because a request is reading and locking the session file to prevent corruption.

You can either

- use a different session handler with

Session set save handler ()

- use session write close () in the import script as soon you don't need session anymore (best moment is just before the long during part takes place), you can session start whenever you want and as many times you like if your import script requires session variables changed.

Example

<?php

session\_start (); //

initiate / open session

$\_SESSION [ 'count' ] = 0 ; //

store something in the

session

session\_write\_close (); //now

close it,

# from here every other

script can be run (and makes

it seem like multitasking)

for( $i = 0; $i <=100; $i ++){

//do 100 cycles

session\_start (); //open

the session again for editing

a variable

$\_SESSION [ 'count' ] +=

1 ; //change variable

session\_write\_close ();

//now close the session

again!

sleep ( 2 ); //every cycle

sleep two seconds, or do a

heavy task

}

?>

13

4 years ago

dave1010 at gmail dot com

PHP locks the session file

until it is closed. If you

have 2 scripts using the

same session (i.e. from the

same user) then the 2nd

script will not finish its

call to session\_start()

until the first script

finishes execution.

If you have scripts that run

for more than a second and

users may be making more

than 1 request at a time

then it is worth calling

session\_write\_close() as

soon as you've finished

writing session data.

<?php

// a lock is places on the

session, so other scripts

will have to wait

session\_start ();

// do all your writing to

$\_SESSION

$\_SESSION [ 'a' ] = 1 ;

// $\_SESSION can still be

read, but writing will not

update the session.

// the lock is removed and

other scripts can now read

the session

session\_write\_close ();

do\_something\_slow ();

?>