Lab 04: Server-side Variables

In this lab, you will experiment with server-side variables. These variables are available to programmers to capture and use the information around the application execution environment. It is especially useful when your application needs to capture information not normally available through user input.

Note that server-side variables are server and version dependent.

Several predefined variables in PHP are superglobals, which means that they are always accessible, regardless of scope. You can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

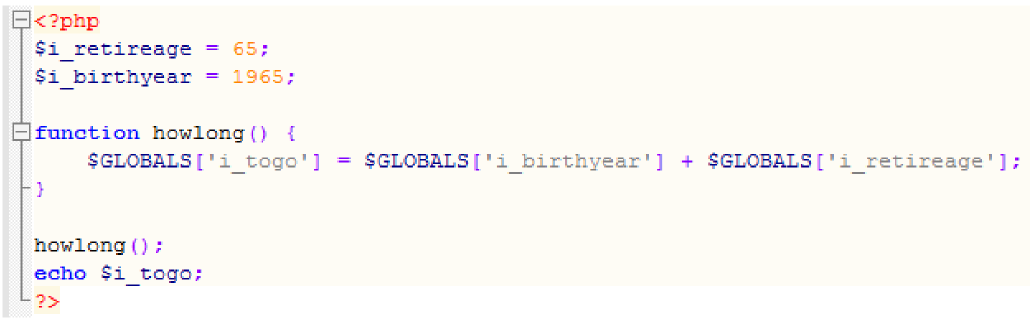
1.1 $GLOBALS

$GLOBALS is used to access global variables from any part of your PHP script, including from within a function or method.

Format for this keyword is $GLOBALS [<variable name>]

<variable name> is the name of the variable you want to access.

Figure 1 below shows an example of how this may be used.

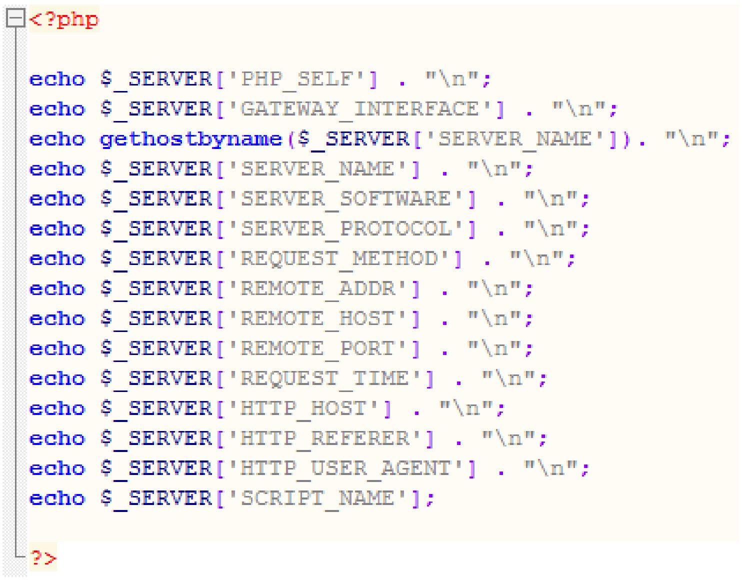


```php
<?php
$i_retireage = 65;
$i_birthyear = 1965;

function howlong() {
    $GLOBALS['i_togo'] = $GLOBALS['i_birthyear'] + $GLOBALS['i_retireage'];
}

howlong();
echo $i_togo;
?>
```

Figure 1: PHP code using $GLOBALS keyword to manage variable computation.

1.2 $_SERVER

$_SERVER holds information about headers, paths, and script locations. Figure 2 below shows an example of how to use some of the elements in $_SERVER.



```php
<?php

echo $_SERVER['PHP_SELF'] . "\n";
echo $_SERVER['GATEWAY_INTERFACE'] . "\n";
echo gethostbyname($_SERVER['SERVER_NAME']). "\n";
echo $_SERVER['SERVER_NAME'] . "\n";
echo $_SERVER['SERVER_SOFTWARE'] . "\n";
echo $_SERVER['SERVER_PROTOCOL'] . "\n";
echo $_SERVER['REQUEST_METHOD'] . "\n";
echo $_SERVER['REMOTE_ADDR'] . "\n";
echo $_SERVER['REMOTE_HOST'] . "\n";
echo $_SERVER['REMOTE_PORT'] . "\n";
echo $_SERVER['REQUEST_TIME'] . "\n";
echo $_SERVER['HTTP_HOST'] . "\n";
echo $_SERVER['HTTP_REFERER'] . "\n";
echo $_SERVER['HTTP_USER_AGENT'] . "\n";
echo $_SERVER['SCRIPT_NAME'];

?>
```

Figure 2: Sample codes of how to use some of $_SERVER elements.

1.3 $_REQUEST, $_POST, $_COOKIE and $_GET

When a user submits data through an HTML form, there are several ways to retrieve the submitted data.

$_REQUEST is an associative array that by default contains the contents of $_GET, $_POST and $_COOKIE.

$_POST is widely used to collect form data with POST method. 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. POST method also supports advanced functionality such as multi-part binary input while uploading files to server.

$_GET is used to collect form data with GET method. $_GET can also collect data sent in the URL. Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET method has a limitation of about 2000 characters. Note that GET method should not be used for sending passwords or other sensitive information.

$_COOKIE is used to collect cookies data transmitted via HTTP cookies. More about this will be discussed in other lab exercise.

The format to use the above keywords is: <keyword> [<name>]

Example:	$_REQUEST [“first_name”];
	$_POST [“first_name”];
	$_GET [“first_name”];
	$_COOKIE [“first_name”];

1.4 $_FILES

When a user uploads a file to the server, $_FILES keyword is used to retrieve and handle the uploaded file. File is generally uploaded using POST method. More about this in the next lab exercise.

You tasks:

1. Create a page containing a list of server variable options for the user to select.
2. Display on web browser the outputs of selected option.

-- END --