Using Autoglobals

PHP includes various predefined global arrays, called autoglobals or superglobals, which contain client, server, and environment information that you can use in your scripts.

|  |  |
| --- | --- |
| Array | Description |
| $\_COOKIE | An array of values passed to the current script as HTTP cookies |
| $\_ENV | An array of environment information |
| $\_FILES | An array of information about uploaded files |
| $\_GET | An array of values from a form submitted with the “get” method |
| $\_POST | An array of values from a form submitted with the “post” method |
| $\_REQUEST | An array of all the elements in the $\_COOKIE, $\_GET, and $\_POST arrays |
| $\_SERVER | An array of information about the Web server that served the current script |
| $\_SESSION | An array of session variables that are available to the current script |
| $GLOBALS | An array of reference to all variables that are defined with global scope |

NOTE: Autoglobals are associative arrays, which are arrays whose elements are referred to with an alphanumeric key instead of an index number.

PHP Forms and User Input

The PHP $\_GET and $\_POST variables are used to retrieve information from forms, like user input.

The $\_GET variable

The $\_GET variable is used to collect values from a form with method = “get”. Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser’s address bar) and it has limits on the amount of information to send (max. 100 characters).

The $\_GET variable is an array of variable names and values sent by the HTTP GET method.

|  |
| --- |
| <form action="week05\_get.php" method="get">Customer Name:<input type="text" name="cust\_name"><br>Customer Number:<input type="text" name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form> |

When the user clicks the “Send Form” button, the URL could look like this:

|  |
| --- |
| http://localhost/mysql\_php/week05/week05\_get.php?cust\_name=abc&cust\_num=123&Submit=Send+Form |

The “week05\_get.php” file can now use the $\_GET variable to catch the form data.

|  |
| --- |
| <?php$cust\_name = $\_GET['cust\_name'];$cust\_num = $\_GET['cust\_num'];echo "GET Method: Cust\_name: $cust\_name. Cust\_num: $cust\_num.";?> |

Why use $\_GET?

NOTE: when using the $\_GET variable all variable names and values are displayed in the URL. So this method should not be used when sending passwords or other sensitive information. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

The $\_POST Variable

The $\_POST variable is used to collect values from a form with method=”post”. Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

The $\_POST variable is an array of variable names and values sent by the HTTP POST method.

|  |
| --- |
| <form action="week05\_post.php" method="post">Customer Name:<input type="text" name="cust\_name"><br>Customer Number:<input type="text" name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form> |

When the user clicks the “Send Form” button, the URL will not contain any form data, and will look like this:

|  |
| --- |
| <http://localhost/mysql_php/week05/week05_post.php> |

The “week05\_post.php” file can now use the $\_POST variable to catch the form data.

|  |
| --- |
| <?php$cust\_name = $\_POST['cust\_name'];$cust\_num = $\_POST['cust\_num'];echo "Post method: Cust\_name: $cust\_name. Cust\_num: $cust\_num.";?> |

The $\_REQUEST Variable

The PHP $\_REQUEST variable contains the contents of both $\_GET, $\_POST, and $\_COOKIE.

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

|  |
| --- |
| <form action="week05.php" method="get">Customer Name:<input type="text" name="cust\_name"><br>Customer Number:<input type="text" name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form> |

|  |
| --- |
| <?php$cust\_name = $\_REQUEST['cust\_name'];$cust\_num = $\_REQUEST['cust\_num'];echo "Request Variable: Cust\_name: $cust\_name. Cust\_num: $cust\_num.";?> |

Form Validation

User input should be validated on the browser whenever possible (by client script – JavaScript). Browser validation is faster and you reduce the server load.

You should consider using server validation if the user input will be inserted into a database. PHP built in functions for validation: is\_double(), is\_float(), is\_int(), is\_integer(), is\_long(), is\_null(), is\_numeric(), is\_object(), is\_real(), and is\_string().

NOTE: empty($variable) = true (if "", 0, "0", NULL, FALSE, empty array, variable was declared but no data assign to it).

A good way to validate a form on the server is to post the form to itself (all-in-one-form), instead of jumping to a different page. The user will then get the error message on the same page as the form. This makes it easier to discover the error.

Different ways of coding PHP

(A). Two –part form

HTML --- calls --- PHP

|  |
| --- |
| <html><body><h1>Two Part form</h1><form action="week05\_two\_part\_form.php" method="get">Customer Name:<input type="text" name="cust\_name"><br>Customer Number:<input type="text" name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form></body></html> |

|  |
| --- |
| <?php$cust\_name = $\_REQUEST['cust\_name'];$cust\_num = $\_REQUEST['cust\_num'];$errCount = 0;/\* Check for string \*/if (empty($cust\_name)) { echo "Customer name cannot be empty!<br>"; $errCount++;}/\* check for number \*/if (empty($cust\_num)) { echo "Customer number cannot be empty!<br>"; $errCount++;}if (!(is\_numeric($cust\_num))) { echo "Customer number is not a number!<br>"; $errCount++;}if ($errCount > 0) { echo "Error - Validation Error - Error!<br>"; echo "Click back button to re-enter data!<br>";} else { echo "Good Data: Cust\_name: $cust\_name. Cust\_num: $cust\_num.";}?> |

(B) Use of Advanced Escaping HTML

HTML --- calls --- PHP (calls PHP repeatedly)

|  |
| --- |
| <html><body><h1>Advanced Escaping HTML</h1><form action="week05\_advanced\_escaping.php" method="get">Customer Name:<input type="text" name="cust\_name"><br>Customer Number:<input type="text" name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form></body></html> |

|  |
| --- |
| <?php$cust\_name = $\_REQUEST['cust\_name'];$cust\_num = $\_REQUEST['cust\_num'];$errCount = 0;/\* Check for string \*/if (empty($cust\_name)) { echo "Customer name cannot be empty!<br>"; $errCount++;}/\* check for number \*/if (empty($cust\_num)) { echo "Customer number cannot be empty!<br>"; $errCount++;}if (!(is\_numeric($cust\_num))) { echo "Customer number is not a number!<br>"; $errCount++;}if ($errCount > 0) { echo "Error - Validation Error - Error!<br>"; /\* Advanced Escaping HTML \*/?><form action="week05\_advanced\_escaping.php" method="get">Customer Name:<input type="text" value="<?php echo $cust\_name; ?>"name="cust\_name"><br>Customer Number:<input type="text" value="<?php echo $cust\_num; ?>"name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form><?php/\* End Advanced Escaping HTML \*/} else { echo "Good Data: Cust\_name: $cust\_name. Cust\_num: $cust\_num.";}?> |

(C) All-in-one Form

PHP --- calls (self PHP repeatedly)

|  |
| --- |
| <html><body><h1>All-In-One Form</h1><?php$cust\_name = "";$cust\_num = "";$errCount = 0;$displayForm = TRUE;if (isset($\_REQUEST['Submit'])) { $cust\_name = $\_REQUEST['cust\_name']; $cust\_num = $\_REQUEST['cust\_num']; /\* Check for string \*/ if (empty($cust\_name)) { echo "Customer name cannot be empty!<br>"; $errCount++; } /\* check for number \*/ if (empty($cust\_num)) { echo "Customer number cannot be empty!<br>"; $errCount++; } if (!(is\_numeric($cust\_num))) { echo "Customer number is not a number!<br>"; $errCount++; } if ($errCount > 0) { echo "Error - Validation Error - Error!<br>"; $displayForm = TRUE; } else { $displayForm = FALSE; }}if ($displayForm) { /\* Advanced Escaping HTML \*/?><form action="week05\_all\_in\_one.php" method="get">Customer Name:<input type="text" value="<?php echo $cust\_name; ?>"name="cust\_name"><br>Customer Number:<input type="text" value="<?php echo $cust\_num; ?>"name="cust\_num"><input type="submit" name="Submit" value="Send Form"></form><?php/\* End Advanced Escaping HTML \*/} else { echo "Good Data: Cust\_name: $cust\_name. Cust\_num: $cust\_num."; echo "<h1>Display content of \$\_REQUEST. </h1>"; echo "<table border='1'>"; foreach ($\_REQUEST as $k => $v) { echo "<tr><td>" . $k . "</td><td>" . $v . "</td></tr>"; } echo "</table>";}?></body></html> |

References:

a). Copied from <http://www.w3schools.com/php/default.asp>