Tutorial 09 – PHP Form Handling

Table of Contents

Learning Outcomes

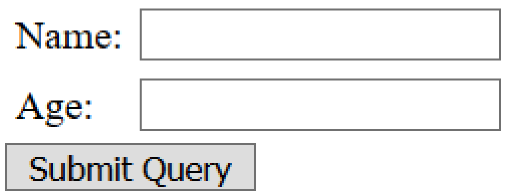
1. Form Handling
2. Reserved variable: $_GET
3. Reserved variable: $_POST
4. Reserved variable: $_REQUEST

PHP Form Handling

- Any form element in an HTML page will automatically be available to your PHP scripts.
- $_GET and $_POST variables are used to retrieve information from forms, like user input.

Example

The example below contains an HTML form with two input fields and a submit button:

php03_01_form.html	Sample Output
<html><body> <form action="php03_01_form_response.php" method="post"> Name: <input type="text" name="fname"/> Age: <input type="text" name="age"/> <input type="submit" /> </form> </body></html>	

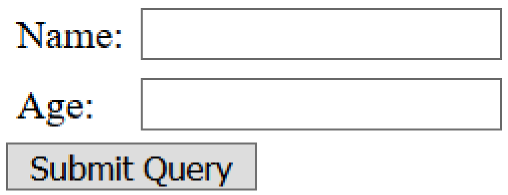
- When a user fills out the form above and click on the submit button, the form data is sent to a PHP file, called "php03_01_form_response.php":

php03_01_form_response.php	Sample Output
<html> <body> Welcome <?php echo $_POST["fname"]; ?>!<br /> You are <?php echo $_POST["age"]; ?> years old. </body> </html>	Welcome John! You are 28 years old.

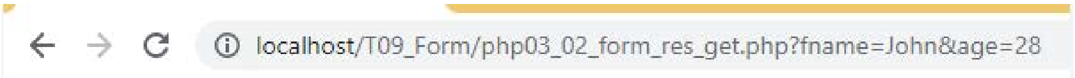
The $_GET Function

- The built-in $_GET function is used to collect values from a form sent 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
  - has limits on the amount of information to send (max. 100 characters).

php03_02_form_get.html	Sample Output

`<html>`
`<body>`
`<form`
`action="php03_02_form_res_get.php"`
`method="get">`
`Name: <input type="text"`
`name="fname" />`
`Age: <input type="text" name="age"`
`/>`
`<input type="submit" />`
`</form>`
`</body>`
`</html>`
	

- When the user clicks the "Submit" button, the URL sent to the server could look something like this:



localhost/T09_Form/php03_02_form_res_get.php?fname=John&age=28

- The "php03_02_form_res_get.php" file can now use the $_GET function to collect form data (the names of the form fields will automatically be the keys in the $_GET array):

php03_02_form_res_get.php	Sample Output

`<html>`
`<body>`
`Welcome`
`<?php echo $_GET["fname"]; ?>`
`.<br />`
`You are`
`<?php echo $_GET["age"]; ?>`
`years old!`
`</body>`
`</html>`
	Welcome John!
	You are 28 years old.

When to use method="get"?

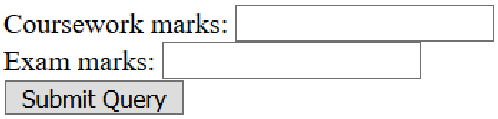
- When using method="get" in HTML forms, all variable names and values are displayed in the URL.
- This method should **NOT** be used when sending passwords or other **sensitive** information!
- However, because the variables are displayed in the URL, it is **possible to bookmark** the page. This can be useful in some cases.
- The get method is **not suitable for large variable** values; the value cannot exceed 100 characters.

The $_POST Function

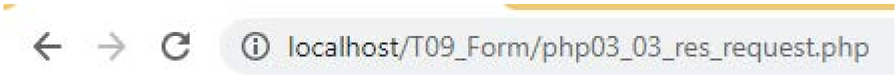
- The built-in $_POST function is used to collect values from a form sent with method="post".
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- Note: However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).

Example

php03_03_form_request.html	Sample Output

`<html>`
`<body>`
`<form action="php03_03_res_request.php" method="post">`
`Coursework marks: <input type="text" name="coursework_marks" /><br>`<br>`Exam marks: <input type="text" name="exam_marks" /><br>`<br>`<input type="submit" />`
`</form>`
`</body>`
`</html>`
	
	Coursework marks: <br>Exam marks:
	Submit Query

When the user clicks the "Submit" button, the URL will look like this:



localhost/T09_Form/php03_03_res_request.php

When to use 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.
- However, because the variables are not displayed in the URL, it is **not possible to bookmark** the page.

The PHP $_REQUEST Function

- The PHP built-in $_REQUEST function contains the contents of both $_GET and $_POST.
- The $_REQUEST function can be used to collect form data sent with both the GET and POST methods.

Example

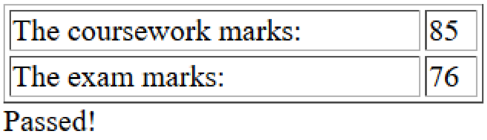
php03_03_form_request.html

```
<html><body>
<?php
   $cmarks =
$_REQUEST["coursework_marks"];
   $emarks =
$_REQUEST["exam_marks"];
?>

<table border = "1" width =
"250px">
<tr>
<td>The coursework marks:</td>
<td><?php echo $cmarks ?></td>
</tr>
<tr>
<td>The exam marks:</td>
<td><?php echo $emarks ?></td>
</tr>
</table>

<?php
if ($cmarks >= 50 && $emarks >= 40)
   echo "Passed!";
else
   echo "Failed!";
?>
</body></html>
```

Sample Output



The coursework marks:	85
The exam marks:	76

Passed!

- Note: if-else will be covered in next Tutorial, you may refer to the following tutorial for exercise purpose: https://www.w3schools.com/php7/php7_if_else.asp

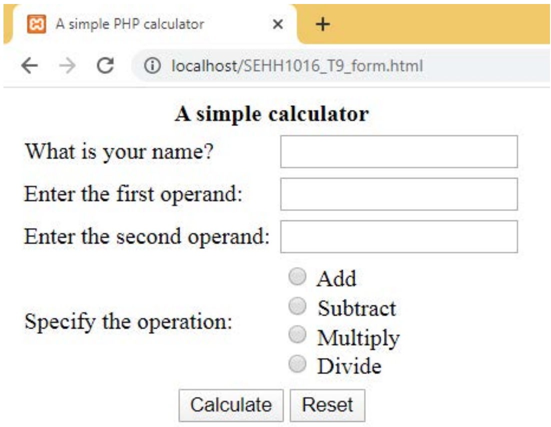
Further Reading

1. https://www.php.net/manual/en/reserved.variables.get.php
2. https://www.php.net/manual/en/reserved.variables.post.php
3. https://www.php.net/manual/en/reserved.variables.request.php

PHP Form handling Exercise

Create a PHP calculator and test it with a Web server

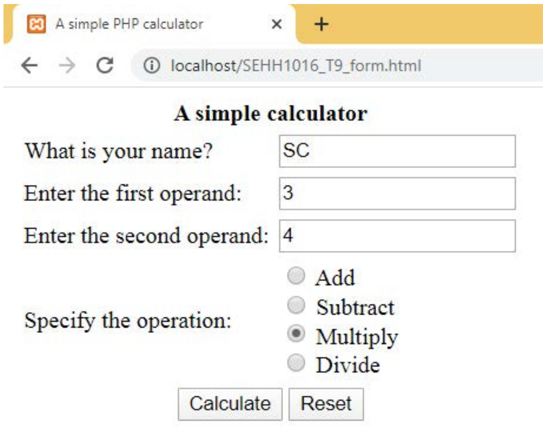
1. Create an HTML form similar to the one below. Save the file as “SEHH1016_T9_form.html” **in the root directory** of the Web server. You can use the template provided to start your work.



A simple calculator

What is your name?	
Enter the first operand:	
Enter the second operand:	
Specify the operation:	○ Add ○ Subtract ○ Multiply ○ Divide

Calculate Reset



A simple calculator

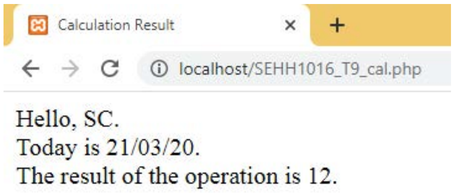
What is your name?	SC
Enter the first operand:	3
Enter the second operand:	4
Specify the operation:	○ Add ○ Subtract ● Multiply ○ Divide

Calculate Reset

Note the following about the form:

- Use the POST method in the form
- When the user clicks the “Calculate” button, the form is sent to a dynamic HTML page called “SEHH1016_T9_cal.php”. This PHP file will be put in the root directory of your Web server. You will create this file in step 3 below.
- Use suitable “name” for the input elements. These names will be referenced again in the PHP file in step 3.

2. Save the HTML form. Use a browser to view the result.
3. Create a PHP file called “SEHH1016_T9_cal.php”. The page should look similar to the one below (after the user inputs in the HTML form). It displays the following information:
   - A greeting message (in the first line)
   - The date of today (in the second line)
   - The result of the calculation (in the third line)



```
Hello, SC.
Today is 21/03/20.
The result of the operation is 12.
```

4. Note the following about the PHP file.
   - By using the `$_POST` variable in the PHP file, get the values input by the user.
     Hint: To get the input from <u>one particular input element</u> from the HTML form, use <u>“name” of that input element as the index</u> to the `$_POST` variable)
   - To obtain a <u>user’s radio button selection</u>, the **“value”** of the <u>selected</u> radio button will become the stored value in the `$_POST` variable of the set of radio buttons.

- Determine which arithmetic operation (+, -, *, /) should be carried out based on the operation selected by the user. Use conditional statements (if…elseif…else) to control the execution

5. Save the PHP file. Make sure the PHP file is in the root directory of the web server. Test the PHP calculator: (1) enter values in the HTML form, (2) click the calculate button, and (3) view the result displayed in the PHP page.