The Different Ways of Input Method, Syntax and Techniques in PHP Programming

Zalwyn Base

BSIT-1C

Homework no. 1

GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? Character.

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

```
<?php
  if( $_GET["name"] || $_GET["age"] ) {
    echo "Welcome ". $_GET['name']. "<br />";
```

```
      echo "You are ". $_GET['age']. " years old.";

      exit();

   }

?>

<html>

   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "GET">

         Name: <input type = "text" name = "name" />

         Age: <input type = "text" name = "age" />

         <input type = "submit" />

      </form>


   </body>

</html>
```

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.

- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

```
<?php

 if( $_POST["name"] || $_POST["age"] ) {
     if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
        die ("invalid name and name should be alpha");
     }
     echo "Welcome ". $_POST['name']. "<br />";
     echo "You are ". $_POST['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>
```

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.

```
<?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>
```

CLASS
Basic class definitions begin with the keyword class, followed by a class name, followed by a pair of curly braces which enclose the definitions of the properties and methods belonging to the class.
The class name can be any valid label, provided it is not a PHP reserved word. A valid class name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: ^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$. A class may contain its own constants, variables (called "properties"), and functions (called "methods").

The pseudo-variable $this is available when a method is called from within an object context. $this is a reference to the calling object (usually the object to which the method belongs, but possibly another object, if the method is called statically from the context of a secondary object). As of PHP 7.0.0 calling a non-static method statically from an incompatible context results in $this being undefined inside the method. Calling a non-static method statically from an incompatible context has been deprecated as of PHP 5.6.0. As of PHP 7.0.0 calling a non-static method statically has been generally deprecated (even if called from a compatible context). Before PHP 5.6.0 such calls already triggered a strict notice.

NEW

Base, Zalwyn
BSIT-1C
Homework 1

To create an instance of a class, the new keyword must be used. An object will always be created unless the object has a constructor defined that throws an exception on error. Classes should be defined before instantiation (and in some cases this is a requirement).
If a string containing the name of a class is used with new, a new instance of that class will be created. If the class is in a namespace, its fully qualified name must be used when doing this.
Example #3 Creating an instance

```
<?php $instance = new SimpleClass();
 // This can also be done with a variable: $className = 'SimpleClass';
 $instance = new $className();
// new SimpleClass() ?>
```

In the class context, it is possible to create a new object by new self and new parent.

REFERENCES

https://www.tutorialspoint.com/php/php_get_post.htm

http://php.net/manual/en/language.oop5.basic.php#language.oop5.basic.new

Base, Zalwyn
BSIT-1C
Homework 1