K.T.S.P.Madal’s

Hutatma Rajguru Mahavidyalaya, Rajgurunagar

Tal-Khed, Dist.-Pune 410505.

TY.BSc (Computer Science)

Semester-VI

Subject- Web Technologies II

According to new CBCS syllabus w.e.f.2019-2020

Prof.S.V.Patole

Department of Computer Science

Hutatma Rajguru Mahavidyalaya,

Rajgurunagar.

Chapter 1. Introduction to web Techniques

1. Variables

In PHP, a variable is declared using a $ sign followed by the variable name. Here, some important points to know about variables:

As PHP is a loosely typed language, so we do not need to declare the data types of the variables.It automatically analyzes the values and makes conversions to its correct datatype. After declaring a variable, it can be reused throughout the code. Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

$variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

2. Server Information

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

```
<!DOCTYPE html>

<html>

<body>


<?php

echo $_SERVER['PHP_SELF'];

echo "<br>";
echo $_SERVER['SERVER_NAME'];

echo "<br>";

echo $_SERVER['HTTP_HOST'];

echo "<br>";

echo $_SERVER['HTTP_REFERER'];

echo "<br>";

echo $_SERVER['HTTP_USER_AGENT'];

echo "<br>";

echo $_SERVER['SCRIPT_NAME'];

?>


</body>

</html>
```

Output:

```
/demo/demo_global_server.php

35.194.26.41

35.194.26.41

https://tryphp.w3schools.com/showphp.php?filename=demo_global_server

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/122.0.0.0 Safari/537.36

/demo/demo_global_server.php
```

4. Processing Forms

The example below displays a simple HTML form with two input fields and a submit button:

```
<!DOCTYPE HTML>

<html>

<body>


<form action="welcome.php" method="post">
```

```
Name: <input type="text" name="name"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>


</body>

</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method. To display the submitted data you could simply echo all the variables.

The "**welcome.php**" looks like this:

```
<html>

<body>


Welcome <?php echo $_POST["name"]; ?><br>

Your email address is: <?php echo $_POST["email"]; ?>


</body>

</html>
```

4.1 When to use GET?

Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases. GET may be used for sending non-sensitive data.

4.2 When to use POST?

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.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

4.2 Automatic Quoting of Parameters

By default, PHP ships with the magic_quotes_gpc option enabled in the php.ini configuration file. This option instructs PHP to automatically call addslashes() on all cookie data, GET parameters, and POST parameters. The purpose of this automatic quoting is to escape special characters (such as single quotes) to prevent SQL injection attacks when interacting with databases. However, it’s essential to note that this feature is considered outdated and deprecated. Modern PHP applications should not rely on magic_quotes_gpc. Developers should handle parameter quoting and escaping explicitly using prepared statements or parameterized queries when working with databases.

4.3 self processing pages

HTML forms are used to send the user information to the server and returns the result back to the browser. For example, if you want to get the details of visitors to your website, and send them good thoughts, you can collect the user information by means of form processing. Then, the information can be validated either at the client-side or on the server-side. The final result is sent to the client through the respective web browser. To create a HTML form, **form** tag should be used.

```
<!DOCTYPE html>
<html>

<head>
    <title>Simple Form Processing</title>
</head>

<body>
    <form id="form1" method="post">
        FirstName:
        <input type="text" name="firstname" required/>
        <br>
        <br>
        LastName
        <input type="text" name="lastname" required/>
        <br>
        <br>
        Address
        <input type="text" name="address" required/>
        <br>
        <br>
        Email Address:
        <input type="email" name="emailaddress" required/>
        <br>
        <br>
        Password:
        <input type="password" name="password" required/>
        <br>
        <br>
        <input type="submit" value="Submit”/>
    </form>
</body>
</html>
```

4.4 Sticky Forms

A technique in which the result of a query is accompanied by a search forms whose values are those of the previous queries

Example.

If we give any "search string" to a search engine, the top of the result page contains another search box, which already contains the same "search string". To refine our search further we can add extra keywords to the "search string".

```
<html>

<head>

<title>Temperature conversion program</title>

</head>

<body>

<?php

Sfahr-5 GET['Fahrenheit'];
```

```
<form action="<?php echo $_SERVER['PHP SELF']?>" method="GET"> Fahrenheit
temperature: <input type="text" name "Fahrenheit" value"<?php echo $fahr?>"/>

<br/>

<input type="submit" name="Conversion"/><br/>

</form>

<?php

?>

if(lis_null(Sfahr))

Scelsius (Sfahr-32)*5/9, printf("%.2fF is %.2f0", Sfahr, Scelsius);

</body>

</html>
```

5. File Upload

PHP allows you to upload single and multiple files through few lines of code only.PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

$_FILES['filename']['name']

returns file name.

$_FILES['filename']['type']

returns MIME type of the file.

$_FILES['filename']['size']

returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

returns error code associated with this file.

6. Form Validation

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

- **Empty String**

```
if (emptyempty ($_POST["name"])) {
   $errMsg = "Error! You didn't enter the Name.";
           echo $errMsg;
} else {
   $name = $_POST["name"];
}
```

- **Validate String**

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
   $ErrMsg = "Only alphabets and whitespace are allowed.";
           echo $ErrMsg;
} else {
   echo $name;
}
```

- **Validate Number**

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
   $ErrMsg = "Only numeric value is allowed.";
   echo $ErrMsg;
} else {
   echo $mobileno;  }
```

7. Setting Response Header

The HTTP response that a server sends back to a client contains headers that identify the type of content in the body of the response, the server that sent the response, how many bytes are in the body, when the response was sent, etc. PHP and Apache normally take care of the headers for you, identifying the document as HTML, calculating the length of the HTML page, and so on. Most web applications never need to set headers themselves. However, if you want to send back something that's not HTML, set the expiration time for a page, redirect the client's browser, or generate a specific HTTP error, you'll need to use the header( ) function.

8. PHP Error Handling

Error handling is a method of capturing an error thrown by a program and applying appropriate action to resolve that error. PHP displays an error message with the file name and line number by default.

There are two popular ways to handle errors in PHP:

1. Applying dai () method
2. Custom errors handler and error triggers

Applying die()

```
<?php

/* Check if the file exists */

if(!file_exists("example.csv")) {

  die("error: The required file is missing.");

}

/* File opening in read mode */

$fp = fopen("example.csv", "r");

?>
```

Custom errors handler

In PHP, we can define our custom error handling function, which is straightforward to create. This function requires a minimum of two parameters and supports a maximum of five parameters

```
<?php

/* User-defined error handling function */

function customErrorHandlar($errorNo, $errorMessage, $errorFile, $errorLine) {

  echo "Error Message: [$errorNo] $errorMessage<br>";

  echo "Error on line $errorLine in $errorFile";

}


/* Set Error Handler */

set_error_handler("customErrorHandlar");


echo $hello;

?>
```