SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19SB602 FULL STACK DEVELOPMENT FOR NEXT GENERATION IOT

III YEAR / VI SEMESTER

Unit III- **CORE PHP WITH MODEL–VIEW–CONTROLLER**

Topic : Arrays, Form Validation

PHP Arrays

An array stores multiple values in one single variable

An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

Example

```
$cars = array("Volvo", "BMW", "Toyota");
```

PHP Array Types

In PHP, there are three types of arrays:

Indexed arrays - Arrays with a numeric index
Associative arrays - Arrays with named keys
Multidimensional arrays - Arrays containing one or more arrays

Working With Arrays

In this tutorial you will learn how to work with arrays, including:

Create Arrays
Access Arrays
Update Arrays
Add Array Items
Remove Array Items
Sort Arrays

Array Items

Array items can be of any data type.

The most common are strings and numbers (int, float), but array items can also be objects, functions or even arrays.

You can have different data types in the same array.

Example

Array items of four different data types:

```
$myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);
```

Array Functions

The real strength of PHP arrays are the built-in array functions, like the count() function for counting array items:

Example

How many items are in the $cars array:

```
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
```

PHP Indexed Arrays

In indexed arrays each item has an index number.

By default, the first item has index 0, the second item has item 1, etc.

Example

Create and display an indexed array:

```
$cars = array("Volvo", "BMW", "Toyota");
var_dump($cars);
```

PHP Sorting Arrays

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

The following PHP array sort functions:

sort() - sort arrays in ascending order
rsort() - sort arrays in descending order
asort() - sort associative arrays in ascending order, according to the value
ksort() - sort associative arrays in ascending order, according to the key
arsort() - sort associative arrays in descending order, according to the value
krsort() - sort associative arrays in descending order, according to the key

PHP Form Handling

The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form

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

Example

```
<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>
```

Result Size: 668 x 492

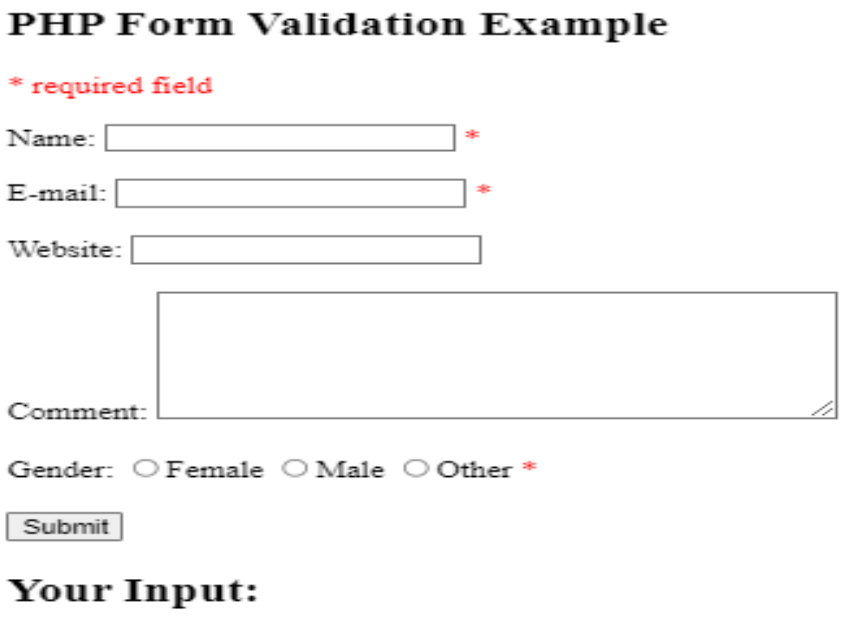
Name:

E-mail:

Submit

PHP Form Validation

The HTML form we will be working on browser, contains various input fields: required and optional text fields, radio buttons, and a submit button:



The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea.

The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5"
cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:

```
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>
- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
Remove backslashes \ from the user input data (with the PHP stripslashes() function)
The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

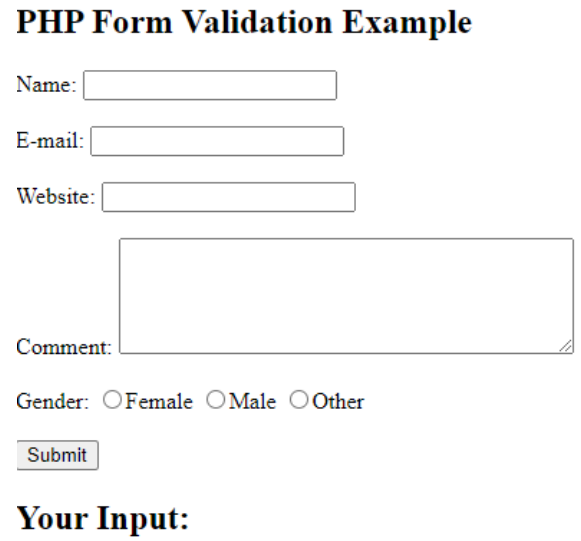
Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```



Any Query????

Thank you……