<!DOCTYPE html> **PHP Form Validation**

<html>

<head>

<style>

.error { color: #FF0000;}

</style>

</head>

<body>

<?php

//define variables and set to empty

$name\_err = $email\_err = $gender\_err = $website\_err = "";

$name = $email = $gender = $comment = $website = "";

//On a form with the method POST, check if the submit button has been clicked.

if($\_SERVER["REQUEST\_METHOD"] == "POST") {

//Check if the name input box is empty.

//If yes, assign an error message to the string variable '$name\_err'.

//Otherwise, call the function 'test\_input' with the 'name' value as its parameter, and assign it to the string variable '$name'.

//Use preg\_match and regex to check the syntax of the name and display an error message if necessary.

if(empty($\_POST["name"])) {

$name\_err = "Please enter a valid name";

}

else {

$name = test\_input($\_POST["name"]);

if(!preg\_match("/^[a-zA-Z-']\*$/",$name)) {

$name\_err = "Only letters and white spaces allowed";

}

}

}

//Repeat the process for all input boxes

if(empty($\_POST["email"])) {

$email\_err = "Please enter a valid email address"

}

else {

$email = test\_input($\_POST["email"]);

if(!filter\_var($email;, FILTER\_VALIDATE\_EMAIL)) {

$email\_err = "The email address is incorrect";

}

}

//The website field is not required so the variable '$website' is assigned the value "" instead of displaying an error.

if(empty($\_POST["website"])) {

$website = "";

}

else {

$website = test\_input($\_POST["website"]);

if(!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website)) {

$website\_err = "Enter a valid website URL";

}

}

if(empty($\_POST["comment"])) {

$comment = "";

}

else {

$comment = test\_input($\_POST["comment"]);

}

if(empty($\_POST["gender])) {

$gender\_err = "Please select a gender";

}

else {

$gender = test\_input($\_POST["gender"]);

}

}

//function test\_data() calls three other functions:

//1. The trim() function strips unnecessary characters (extra space, tab, newline) from the user input data.

//2. The stripslashes() function removes backslashes from the user input data (it literally strips the slashes)

//3. The htmlspecialchars() function converts special characters to HTML entities.

//This means that it will replace HTML characters like < and > with &lt; and &gt;.

//This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

//Without htmlspecialchars(), the $\_SERVER["PHP\_SELF"] variable is not secure - IT MUST BE INCLUDED

function test\_input ($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>PHP Form Validation Example</h2>

<p><span class="error">\* required field</span></p>

<form method="POST" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

FullName: <input type="text" name="name">

<span class="error">\* <?php echo $name\_err;?></span>

<br> <br>

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

<span class="error">\* <?php echo $email\_err;?></span>

<br> <br>

Website: <input type="text" name="website">

<span class="error">\* <?php echo $website\_err;?></span>

<br> <br>

Comment: <textarea name="comment" rows="2" cols="10"></textarea>

<br> <br>

Gender:

<input type="radio" name="gender" value="female">Female

<input type="radio" name="gender" value="male">Male

<span class="error">\* <?php echo $gender\_err;?></span>

<br> <br>

<input type="submit" name="submit" value="Submit">

</form>

<?php>

echo "<h2>Your Input:</h2>;

echo $name;

echo "<br>";

echo $email;

echo "<br>";

echo $website;

echo "<br>";

echo $comment;

echo "<br>";

echo $gender;>

?>

</body>

</html>