PHP Self Processing Forms

For this lab, you will combine the skills you have learned in the previous two labs, to create a self-processing form in PHP.

1. Create a file named **lab06.php**. You will be able to reuse most of your code from the previous labs, but make sure to test your code frequently. Copying and pasting code is a very common way to introduce bugs into your scripts.
2. Here is the first block of PHP code that will be in lab06.php. This is very similar to the code block you created for Lab 5. Compare the code blocks and make necessary changes, taking note of the differences listed in the next step.

```php
<?php
// define variables and set to empty values
$nameErr = $emailErr = $websiteErr = $commentErr = "";
$name = $email = $website = $comment = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = $_POST["name"];
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = $_POST["email"];
    }

    if (empty($_POST["website"])) {
        $websiteErr = "Website is required";
    } else {
        $website = $_POST["website"];
    }

    if (empty($_POST["comment"])) {
        $commentErr = "Comment is required";
    } else {
        $comment = $_POST["comment"];
    }
}
?>
```

3. The changes you will need to make to the code block from your previous lab include:
    a. The variable declarations at the top. These variables will store the submitted values from the $_POST array or populate error message values for missing required values.
    b. The selection statements, if statement blocks, have been changed to no longer echo the values or error messages, but rather store them in the new variables.
    c. The else if statement that displayed an error messaged when the HTTP request was set to GET has been removed. When your lab06.php file is first requested, your browser will use GET by default; this is no longer an error state.
    d. Make sure the <php … ?> block is closed at the bottom. You will put your static HTML form right after this block.
4. You will reuse the form code from Lab 5. Copy the contents of **lab05.html** and paste it directly below your PHP code block in **lab06.php.** You will need to change HTML title to Lab 6.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lab 5</title>
</head>
<body>
<form action="lab05.php" method="post">
    Name: <input type="text" name="name"><br>
    E-mail: <input type="text" name="email"><br>
    Website: <input type="text" name="website"><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea><br>
    <input type="submit">
</form>
</body>
</html>
```

5. In your form element, change the **action** attribute to match the code below. The PHP snippet embedded in the action attribute quotations will return the name of the currently running script, which is also the script you want to process the form. With this code, a form will designate the current script as the action. You can hard code the file name in the action as well, “lab06.php”, but the PHP code below will protect you from mistakenly changing the name of the script and forgetting to change the corresponding action attribute.

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

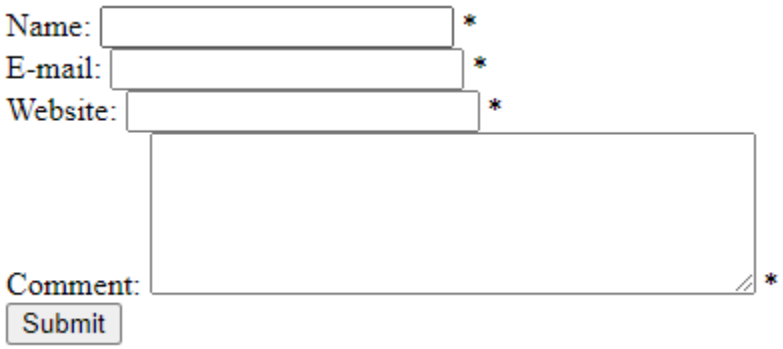
6. Next, add **** tags to your form. If the form is submitted without the required values filled in, the PHP processing block at the top of your code will populate the error variables with either an empty string or an actual error message. The error variables are echoed every time the form processes. If the required values are in $_POST, the error string will be an empty string, and thus not seen. Otherwise, if the required values are not in $_POST, the error string will contain a message to show next to the input field in the form.

```
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
    Name: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr;?></span><br>
    E-mail: <input type="text" name="email">
    <span class="error">* <?php echo $emailErr;?></span><br>
    Website: <input type="text" name="website">
    <span class="error">* <?php echo $websiteErr;?></span><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea>
    <span class="error">* <?php echo $commentErr;?></span><br>

    <input type="submit">
</form>
```

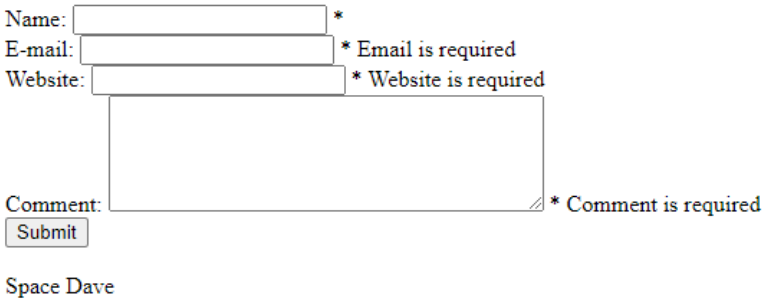
7. Lastly, add a block of PHP code below your form to display the values stored from $_POST. This is for testing. In later labs you will collect the values submitted via the form and save them to a file or database.

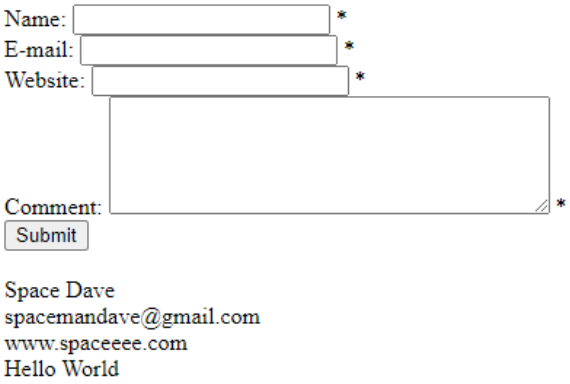
```
</form>

<?php
echo "<br>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
?>

</body>
```

8. As previously stated, the first time you request lab06.php from your browser, it will use GET by default. Your form should look like this. The PHP code block at the top of your script will do nothing because it only processes form values or error messages when executed as a POST request.



9. However, when the submit button is pressed, the HTTP request will be set to POST via your form's method attribute. The form sends a POST request for the current file, via the action attribute we set to PHP_SELF. The form reloads and the code block at the top will then process the POST input accordingly.





10. Notice the form does not retain values that have been entered by the user. In the event a user has missed a required value, they will have to fill out the entire form again. You can fix this by using PHP to echo the values that your top block is collecting from POST.

```
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
    Name: <input type="text" name="name" value="<?php echo $name;?>">
    <span class="error">* <?php echo $nameErr;?></span><br>
    E-mail: <input type="text" name="email" value="<?php echo $email;?>">
    <span class="error">* <?php echo $emailErr;?></span><br>
    Website: <input type="text" name="website" value="<?php echo $website;?>">
    <span class="error">* <?php echo $websiteErr;?></span><br>
    Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>
    <span class="error">* <?php echo $commentErr;?></span><br>

    <input type="submit">
</form>
```

11. After adding the echo statement above, your form will retain the values submitted when processed, allowing the user to correct any form errors.

Name: Space Dave *
E-mail: spacemandave@gmail.com *
Website: * Website is required
Comment: * Comment is required
Submit

Space Dave
spacemandave@gmail.com

12. In the next lab, you will work to improve your self-processing “sticky” form by making it more secure. You will also learn how to direct the user to a new page, if all the required form values have been submitted. Test this script extensively before moving on.

Deliverables

1. Demonstrate the code and execution to the instructor during this lab, during office hours, or during the next lab period.
2. Upload the instructor approved .php file (source code only) to the Lab 6 D2L drop box.