**Project 02\_04\_01**

In this Project we will build a series of PHP applications to test our knowledge of the unit. Be sure to document all of your code to indicate your understanding.

 

**Part 01**

In this part of the Project, you will create a Web form that will help in creating Jumble puzzles. It will have four input fields for words, plus a Reset and submit button. You will create a form processing script that will verify that all four words have been entered, that each is between 4 and 7 letters. After verification, they will be converted to uppercase and jumbled.

1. Create a new file called ***JumbleMaker.html***. Scaffold a basic HTML code layout into it. Set the <title> to ***Jumble Maker***. Make sure to have the ***modernizr*** <script> linked in.
2. Inside the ***<body>*** element, place the following code:  
   <body>:  
    ***<h1>Jumble Maker</h1>  
    <form action="processJumbleMaker.php" method="post">  
    <p>  
    word 1: <input type="text" name="word1"><br>  
    word 2: <input type="text" name="word2"><br>  
    word 3: <input type="text" name="word3"><br>  
    word 4: <input type="text" name="word4"><br>  
    </p>  
    <input type="reset" value="Clear Form">&nbsp;&nbsp;  
    <input type="submit" name="Submit" value="Send Form">  
    </form>***</body>  
   Perform a server/browser test.
3. Create a new file called ***processJumbleMaker.php***. Scaffold a basic HTML code layout into it. Set the <title> to ***Process Jumble Maker***. Make sure to have the ***modernizr*** <script> linked in. Perform a server/browser test.
4. Add an ***<h2>*** element to the body with the content set to ***Process Jumble Maker***. Place a set of PHP standard ***script*** ***delimiters*** in the <body> of the document:  
   <body>  
    ***<h2>Process Jumble Maker</h2>  
    <?php  
      
    ?>***</body>  
   Perform a server/browser test to make sure it processes.
5. At the top of the script, declare and initialize some ***global*** variables, one to hold a count of the errors, and an array to hold the words:  
    <?php  
    ***$errorCount = 0;  
    $words = array();*** ?>
6. Below the variables, add a ***displayError()*** function to the script. The function should handle all error display by taking defining parameters, the name of the field and an error message:  
    ***function displayError($fieldName, $errorMsg) {  
    global $errorCount;  
    echo "Error for \"$fieldName\": $errorMsg<br>\n";  
    ++$errorCount;  
    }***Do a server/browser test for syntax.
7. Below the first function scaffold a second function named ***validateWord()***. This one will take two parameters, the string to be validated, and the name of the form field that the data is coming from. After we build it out, it will eventually clean up and return the data, as well as make use of the global ***$errorCount***:  
    ***function validateWord($data, $fieldName) {  
    global $errorCount;  
    $retval = "";  
      
    return $data;  
    }***Do a server/browser test for syntax.
8. At the bottom of the script, let’s build the code that will retrieve the form data from the ***$\_POST*** superglobal array and call the ***validateWord()*** function. Add a debug foreach loop to test:  
    ***$words[] = validateWord($\_POST['word1'], "Word 1");  
    $words[] = validateWord($\_POST['word2'], "Word 2");  
    $words[] = validateWord($\_POST['word3'], "Word 3");  
    $words[] = validateWord($\_POST['word4'], "Word 4");  
    foreach ($words as $word) {  
    echo "$word<br>";  
    }***Do a server/browser test.
9. Before we finish building out our ***validateWord()*** function, let’s build the closing part of the script that will produce the results. Replace the debug ***foreach*** loop with the following ***if…else*** clause:  
    ***if ($errorCount > 0) {  
    echo "Please use the \"Back\" button to re-enter any   
    missing data.<br>\n";  
    }  
    else {  
    $wordNum = 0;  
    foreach ($words as $word) {  
    echo "Word " . ++$wordNum . ": $word<br>\n";  
    }  
    }***Do a server/browser test.
10. Now let’s go back and construct some more of our ***validateWord()*** function. Let’s start off by validating for missing data. Under the variable declarations, enter the following code, be sure to fix the value returned by the ***return*** statement:  
     ***if (empty($data)) {  
     displayError($fieldName, "This field is required");  
     ++$errorCount;  
     $retval = "";  
     }  
     return $retval;***Do a server/browser test.
11. Now, let’s have ***validateWord()*** clean up any ***whitespace*** before or after the data and strip out any extra ***escape*** characters. We will also check it for acceptable ***length***, and make sure it uses only letters by employing ***Regular*** ***Expressions***:  
     ***else {  
     $retval = trim($data);  
     $retval = stripslashes($retval);  
     if (strlen($retval) < 4 || strlen($retval) > 7) {  
     displayError($fieldName, "Words must be between 4   
     and 7 characters in length");  
     }  
     if (preg\_match("/^[A-Za-z]+$/i", $retval) == 0) {  
     displayError($fieldName, "Words must consist only of   
     letters");  
     }  
     }***Do a server/browser test.
12. Finally, let’s have ***validateWord()*** convert the data to ***uppercase***, then ***shuffle*** the characters to give us our Jumble. Enter the calls at the bottom of the function, just before the return:  
     ***$retval = strtoupper($retval);  
     $retval = str\_shuffle($retval);  
     return $retval;***Do a server/browser test.

**Part 02**

In this Part of the Project, you will create an all-in-one form that is a working Contact Me page. The page will have inputs for the subject, the user’s name, e-mail address, and the message. The form will set up the necessary steps to send an e-mail to the user. It will probably not execute the actual send, because of limitations of our server.

1. Create a new file called ***ContactForm.php***. Scaffold a basic HTML code layout into it. Set the <title> to ***Contact Form***. Make sure to have the ***modernizr*** <script> linked in. Place a set of PHP standard ***script*** ***delimiters*** in the <body> of the document:  
   <body>  
    ***<?php  
      
    ?>***</body>  
   Perform a server/browser test.
2. At the top of the script, declare and initialize some ***global*** variables:  
    <?php  
    ***$showForm = true;  
    $errorCount = 0;  
    $sender = "";  
    $email = "";  
    $subject = "";  
    $message = "";*** ?>  
   Perform a server/browser test.
3. Next we can build a function named ***displayForm()*** which will take care of displaying the form, both initially, or if there are errors in the submitted data. It will be a ***Sticky*** ***Form***, remembering any values the user has already entered, so it will need them as parameters. It will use ***multiple*** PHP scripts, so be careful with the script delimiters. Place a ***debug*** line below the function definition to test it:  
    ***function displayForm($sender, $email, $subject, $message) {  
    ?>  
    <h2 style="text-align: center">Contact Me</h2>  
    <form name="contact" action="ContactForm.php"   
    method="post">  
    <p>Your name:<br> <input type="text"   
    name="Sender" value="<?php echo $sender;   
    ?>"></p>  
    <p>Your E-mail:<br> <input type="text"   
    name="Email" value="<?php echo $email;   
    ?>"></p>  
    <p>Subject:<br> <input type="text" name="Subject"   
    value="<?php echo $subject; ?>"></p>  
    <p>Message:<br>  
    <textarea name="Message"><?php echo   
    $message?></textarea>  
    </p>  
    <p>  
    <input type="reset" value="Clear   
    Form">&nbsp;&nbsp;  
    <input type="submit" name="Submit" value="Send   
    Form">  
    </p>  
    </form>  
    <?php  
    }  
    displayForm($sender, $email, $subject, $message); // debug  
    ?>***Perform a server/browser test.
4. Before we build out the validation routines, let’s build some of the end of the script. Replace the debug displayForm call with the following code:  
    <?php  
    }  
      
    ***if ($showForm) {  
    if ($errorCount > 0) {  
    echo "<p>Please re-enter the form information   
    below.</p>\n";  
    }  
    displayForm($sender, $email, $subject, $message);  
    }  
    else {  
    $result = true; // debug  
    if ($result) {  
    echo "<p>Your message has been sent. Thank you, " .  
    $sender . ".</p>\n";  
    }  
    else {  
    echo "<p>There was an error sending your message, " .   
    $sender . ".</p>\n";  
    }  
    }*** ?>  
   Perform a server/browser test, first setting ***$errorCount*** to 1. Then give it a test with ***$errorCount*** set to 0.
5. Now lets build a function ***validateInput()*** that will validate our required input fields. Enter this code above the ***displayForm()*** function. It will take in the ***data*** and the ***field*** ***name*** as parameters:  
    ***function validateInput($data, $fieldName) {  
    global $errorCount;  
    if (empty($data)) {  
    echo "\"$fieldName\" is a required field.<br>\n";  
    ++$errorCount;  
    $retval = "";  
    }  
    else {  
    $retval = trim($data);  
    $retval = stripslashes($retval);  
    }  
    return $retval;  
    }***Perform a server/browser test.
6. Now we can use the ***$\_POST*** superglobal array to test for a ***submit*** and validate some of our data. Enter the following code just above the ***if (showform)*** statement. We will validate all of the required fields:  
    ***if (isset($\_POST['Submit'])) {  
    $sender = validateInput($\_POST['Sender'], "Your   
    Name");  
    $email = validateInput($\_POST['Email'], "Your E-mail");  
    $subject = validateInput($\_POST['Subject'], "Subject");  
    $message = validateInput($\_POST['Message'],   
    "Message");  
    if ($errorCount == 0) {  
    $showForm = false;  
    }  
    else {  
    $showForm = true;  
    }  
    }***Perform a server/browser test.
7. Lastly, let’s validate the e-mail field for a proper e-mail using ***Regular*** ***Expressions***. Below the ***validateInput()*** function, we can build a ***validateEmail()*** function:   
    ***function validateEmail($data, $fieldName) {  
    global $errorCount;  
    if (empty($data)) {  
    echo "\"$fieldName\" is a required field.<br>\n";  
    ++$errorCount;  
    $retval = "";  
    }  
    else {  
    $retval = trim($data);  
    $retval = stripslashes($retval);  
    $pattern = "/^[\w-]+(\.[\w-]+)\*@" .   
    "[\w-]+(\.[\w-]+)\*" .   
    "(\.[a-z]{2,})$/i";  
    if (preg\_match($pattern, $retval) == 0) {  
    echo "\"$fieldName\" is not a valid e-mail   
    address.<br>\n";  
    ++$errorCount;  
    }  
    }  
    return $retval;  
    }***Perform a server/browser test.
8. In the ***if (isset…*** clause, change the ***$email*** assignment to call ***validateEmail()*** instead of ***validateInput()***:  
    $sender = validateInput($\_POST['Sender'], "Your Name");  
    ***$email = validateEmail($\_POST['Email'], "Your E-mail");*** $subject = validateInput($\_POST['Subject'], "Subject");  
   Perform a good bank of browser/server tests.
9. Now let’s set up a PHP e-mail send. We will do this at the top of the last ***else*** clause in the script. Replace the ***debug $result = true;*** statement with the following code. Please change my ***email*** to your own:  
    else {  
    ***$senderAddress = "$sender <$email>";  
    $headers = "From: $senderAddress\nCC:$senderAddress";  
    $result = mail("mark.buckler@west-mec.org", $subject,   
    $message, $headers);***Give this a browser/server test.