Lesson 5

# PHP, Forms and Databases

PHP only works on a live server. This is where software such as MAMP will come in handy as you it will allow us to test PHP on a local server or you can use FTP (Cyber duck or another program) to publish to a remote server.

### Sending an email PHP

|  |  |
| --- | --- |
| We’ll write a PHP mail script that will process form data.  This simple script is responsible for checking that the form data is valid and then sends the email.  Open mail.php and add the following:  <?php  // Get the form fields and remove whitespace.  $name = strip\_tags(trim($\_POST["name"]));  $name = str\_replace(array("\r","\n"),array(" "," "),$name);  $email = filter\_var(trim($\_POST["email"]), FILTER\_SANITIZE\_EMAIL);  $message = trim($\_POST["message"]);  $response = "";  $recipient = "[info@brightdaystudio.co.uk](mailto:info@brightdaystudio.co.uk)";  $subject = "New contact from $name";  $content = "Name: $name \n Email: $email \n\n Message: \n$message";  $headers = "From: $name <$email>" . "\r\n" . "Reply-to: $name <$email>";  $headers .= "MIME-Version: 1.0\r\n";  $headers .= "Content-type: text/html\r\n";  // Check the data.  if ( empty($name) OR empty($message) OR !filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {  $response = "Oops! There was a problem with your submission. Please try again.";  Exit;  }  //send mail  if (mail($recipient, $subject, $content, $headers)) {  $response = "Thank You $name! Your message has been sent.";  } else {  $response = "Oops! Something went wrong and we couldn't send your message.";  }  ?>  Finally into the HTML section of our php page we add this:  <h2><?php echo $response ?></h2> | |
| We extract the form data into three variables $name, $email, and $message. We use the PHP trim method to remove any unnecessary whitespace. We then create variables for the email recipient, subject, email content, and email headers.  Check that no fields are blank; if one or more is blank return an error message.  Then send the mail.  Note: Setting the email headers is optional. It’s important to note that manipulating the headers can cause the email to be marked as spam by some email clients.  If the mail is successful return a success message, if not return an error message. | |
| Publish your pages using FTP (cyberduck) to a server.  Test your form on a live server! | |

### 

### MySQL Databases

|  |  |
| --- | --- |
| Open data.php  First we set up php variables for important data we need to connect to the database, and the data we collect from the form. Enter this:  <?php  // setup our variables  $servername = "160.153.133.168";  $username = "editor";  $password = "RQgH8xZnF2dvewf";  $dbname = "testDB";  $table = "messages";  // Get the form fields and remove whitespace.  $name = strip\_tags(trim($\_POST["name"]));  $name = str\_replace(array("\r","\n"),array(" "," "),$name);  $email = filter\_var(trim($\_POST["email"]), FILTER\_SANITIZE\_EMAIL);  $message = trim($\_POST["message"]);  // Create connection  $conn = new mysqli($servername, $username, $password, $dbname);  // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  $sql = "INSERT INTO $table (name, email, message) VALUES ('$name', '$email', '$message')";  if ($conn->query($sql) === TRUE) {  $response = "New record created successfully";  } else {  $response = "Error: " . $sql . "<br>" . $conn->error;  }  $conn->close();  ?>  Finally into the HTML section of our php page we add this:  <h2><?php echo $response ?></h2> | |
| Here we respond if the connection was successful, or return an error if it fails, then close the connection. | |
| Open the page on the server.  Test your form, check phpmyadmin that the data has been passed into the database.  That’s it! You’ve added a database entry using php! | |