# PHP

## EXERCISE 4: RECEIVING VALUES FROM FORMS

We have already seen how PHP scripts can receive values from the user in the URL, but a more flexible way is to receive the contents of an HTML form.

Here is a minimal HTML form to send the same values to the **time1.php** page as we previously sent in the URL. Save this is a file called **form1.html** and upload it to the server alongside **time1.php**.

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<title>Form 1</title>**  **</head>**  **<body>**  **<form action='time1.php' method='get'>**  **First name <input type='text' name='firstname'>**  **Last name <input type='text' name='lastname'>**  **<input type='submit'>**  **</form>**  **</body>**  **</html>** |

Note that this form sends its values in the URL (you can see them there) just the same way as we did by hand before. Look at the highlighted parts of the form tag in the code above. The ***action*** attribute names the PHP script the form should be sent to, in this case **time1.php**. The ***method*** attribute controls how the data is sent. Here it is set to ‘get’, which means the values are sent in the URL. More commonly, the ‘post’ method is used for forms. Note the input fields are given the names we then need to refer to in the PHP script to access the values.

If a form has its method set to ‘post’ the values are available in PHP in exactly the same way as with ‘get’ except a different array must be used. Instead of $\_GET it is $\_POST.

Save the following in a file called **post.php**.

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<title>Post Form Receiver</title>**  **</head>**  **<body>**  **<?php**  **if ( isset( $\_POST['firstname']) && isset( $\_POST['lastname'])) {**  **print "<p>Hello ".$\_POST['firstname']." ".$\_POST['lastname']."!</p>\n";**  **} else {**  **print "<p>You didn't tell me your first and last names!</p>\n";**  **}**  **?>**  **</body>**  **</html>** |

This is the same code as used in **time1.php** but converted to receive post value instead of get. Change the form in **form1.html** to point at this script and use the **post** method.

Now try this script. Save it with the name **autoform.php** and view it in the web browser.

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<title>Post Form Receiver</title>**  **</head>**  **<body>**  **<?php**  **if ( isset( $\_POST['firstname']) && isset( $\_POST['lastname'])) {**  **print "<p>Hello ".$\_POST['firstname']." ".$\_POST['lastname']."!</p>\n";**  **print "<p><a href=''>Go again!</a></p>\n";**  **} else {**  **?>**  **<form action='' method='post'>**  **First name <input type='text' name='firstname'>**  **Last name <input type='text' name='lastname'>**  **<input type='submit'>**  **</form>**  **<?php**  **}**  **?>**  **</body>**  **</html>** |

Once you have experimented with this script, carefully examine the code and write an explanation of how it works in your report. This technique, where a PHP script both displays the form AND receives the input from it is very commonly used when building web applications.

Here’s a similar form, but tagged up with w3css classes to make it look nice. It also includes a **textarea** message field. All three input fields have the **required** attribute. Find out what this does and mention it in your report.

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<title>Contact Form</title>**  **<meta name='viewport' content='width=device-width, initial-scale=1'>**  **<link rel='stylesheet' href='http://www.w3schools.com/lib/w3.css'>**  **</head>**  **<body class='w3-padding-large'>**  **<h1>Contact Form</h1>**  **<form action='sendemail.php' method='post' class='w3-padding-large w3-card-16'>**  **<div class='w3-group'>**  **<input type='text' name='firstname' class='w3-input' required>**  **<label class='w3-label'>First name</label>**  **</div>**  **<div class='w3-group'>**  **<input type='text' name='lastname' class='w3-input' required>**  **<label class='w3-label'>Last name</label>**  **</div>**  **<div class='w3-group'>**  **<textarea name='message' class='w3-input' required></textarea>**  **<label class='w3-label'>Message</label>**  **</div>**  **<input type='submit' value='Send to website owner' class='w3-btn'>**  **</form>**  **</body>**  **</html>** |

Save this code in a file called **contact-form.html**. Note that this form is pointing at a script called **sendemail.php**.

The code for sendemail.php is below, but it needs fixing in various ways. Save it in a file called sendemail.php, make the required changes (see instructions after the code) and upload it together with contact-form.html to the web server.

|  |
| --- |
| **<!DOCTYPE html>**  **<html>**  **<head>**  **<title>Email Confirmation</title>**  **<meta name='viewport' content='width=device-width, initial-scale=1'>**  **<link rel='stylesheet' href='http://www.w3schools.com/lib/w3.css'>**  **</head>**  **<body class='w3-padding-large'>**  **<h1>Contact Confirmation Screen</h1>**  **<?php**  **// FIX THIS**  **$message = "This needs to be changed to be whatever the user enters.";**  **// CHANGE THE LINE BELOW TO AN EMAIL ADDRESS YOU RECEIVE MESSAGES AT.**  **$toaddress = "noOne@NoWhere"; // PLEASE don't leave noOne@NoWhere in this - it won't work...**  **// create a subject line for the email - THIS SHOULD GIVE THE NAME ENTERED INTO THE FORM**  **$subjectline = "Message from John Smith";**  **// set the from address for the email; this is required**  **$headers = "from:bus.support@lsbu.ac.uk\n"; // Daydream will send an email only if this header is present**  **$success = false;**  **// Send the email. The mail command is commented out. Remove the // to activate it.**  **// It is commented out because if the server does not support sending mails it might crash the script.**  **//$success = mail( $toaddress, $subjectline, $message, $headers);**  **// THIS TEXT SHOULD ALSO GIVE THE NAME ENTERED BY THE USER, NOT John Smith.**  **print "<p>Thank you John Smith, the following message has been sent to the website owner:</p>";**  **print "<div class='w3-card-16 w3-padding-large'>\n";**  **print "<p><b>Subject:</b> ".$subjectline."</p>\n";**  **print "<p>".$message."</p>\n";**  **print "<d/div>\n";**  **if ( !$success) {**  **print "<h2>Error: Message not sent!</h2>\n";**  **}**  **?>**  **</body>**  **</html>** |

You will need to fix the following (all highlighted in red in the code above);

1. The text of the message should come from the form input.
2. The value for the variable $toAddress should be a real email address you own. This is where the email will get sent.
3. The subject line should not say John Smith, but give the names the user entered on the form.
4. The thank you message should also not say John Smith, but the name entered on the form.
5. The mail command itself has been disabled by preceding it with the comment symbol //. Once you have fixed all the other problems listed above and tested that it looks right, remove the // before the mail command and hopefully you will actually receive the email at the address you gave for $toAddress.

|  |
| --- |
| **EXERCISE 4: RECEIVING VALUES FROM FORMS** |
| ***Required in your report (basic exercise).*** |
| * Clickable links to your Daydream account for **form1.html**, **autoform.php** and **contact-form.html**. * An explanation of how the **autoform.php** script works. * An explanation of what the **required** attribute does in HTML form fields. * Reflection on the tasks and all activities involved. |
| ***Extended Tasks*** |
| * Add other HTML form elements to the contact form and include their values in the email. Add select, radio-buttons and check boxes. * Follow the tutorial at <http://www.w3schools.com/php/php_form_validation.asp> and demonstrate that neither of the forms developed above are secure from users entering special characters. * Improve the forms in the light of this by use of **htmlspecialcharacters()** function. |
| *Original additional work:*  *Investigate and experiment with any related subject matter that interests you.* |