PHP – Superglobals

Objectives

1. Check PHP settings with `phpinfo()`
2. Understand how data is transferred from forms and then retrieved with `$_POST`
3. Understand how data is transferred from forms and then retrieved with `$_GET`
4. Review for data is transferred with query strings using `$_GET`
5. Understand how `$_POST` and `$_GET` are associate arrays and can be looped
6. Use conditional logic to check for specific values

Introduction

In this lab we'll check that PHP is up and running on your web space, experiment with PHP variables and get to grips with Superglobals.

Is it working?

To check if PHP is running we'll create a very simple PHP page.

In your *public_html* folder create a subfolder called *superglobals* and create a file called *info.php*.

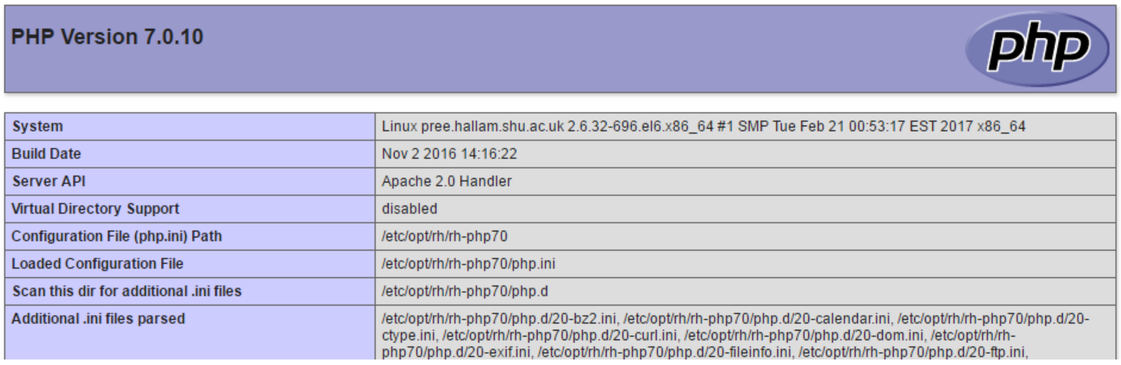
Add the following code to your file.

```
<?php
phpinfo();
?>
```

To view your file you will need to navigate to it through http by using an address such as:

```
http://homepages.shu.ac.uk/~<yourid>/WAD/superglobals/info.p
hp
```

You should see a page such as:



PHP Version 7.0.10

System	Linux pree.hallam.shu.ac.uk 2.6.32-696.el6.x86_64 #1 SMP Tue Feb 21 00:53:17 EST 2017 x86_64
Build Date	Nov 2 2016 14:16:22
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/opt/rh/rh-php70
Loaded Configuration File	/etc/opt/rh/rh-php70/php.ini
Scan this dir for additional .ini files	/etc/opt/rh/rh-php70/php.d
Additional .ini files parsed	/etc/opt/rh/rh-php70/php.d/20-bz2.ini, /etc/opt/rh/rh-php70/php.d/20-calendar.ini, /etc/opt/rh/rh-php70/php.d/20-ctype.ini, /etc/opt/rh/rh-php70/php.d/20-curl.ini, /etc/opt/rh/rh-php70/php.d/20-dom.ini, /etc/opt/rh/rh-php70/php.d/20-exif.ini, /etc/opt/rh/rh-php70/php.d/20-fileinfo.ini, /etc/opt/rh/rh-php70/php.d/20-ftp.ini,

Working with Variables

Create a second PHP page. This time use a boiler plate HTML document as follows:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>PHP Variables</title>
</head>
<body>
</body>
</html>
```

Then add the following code in the `<body>` of the document.

```
<?php
$firstname = "Bob";
$surname = "Smith";
echo $firstname;
echo $surname;
?>
```

Experiment with the PHP to output the variables with HTML formatting for example:

```
<?php
$firstname = "Bob"; $surname = "Smith";
echo "<h1>" . $firstname. " ". $surname . "</h1>";
?>
```

Superglobals

One of the key features of web applications is the ability to pass data from one page in your web application to another.

One method to do this is HTML forms. The data from HTML form is sent for processing using either the ‘post’ or ‘get’ methods. Where the data is sent and whether it is sent with post or get is controlled by the attributes of the form element ie:

```
<form action="process.php" method="post">
```

Tip: If no `method` *value is set the default is for a form to send values using get. If no* `action` *is set then when the form is submitted it reloads the page - sending the data with it.*

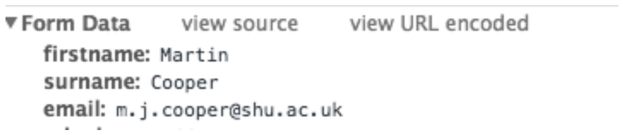
The values that the user entered in the form are then available to the targeted action page for processing. In a HTML form, each form element should have a unique name/id. This is the fieldname of the value to be submitted.

```
<input name="email" type="text" id="email" />
```

That is the name of the name/value pairs that will get submitted when the data is sent. So if you have a form field named ‘email’ and the users enters the value ‘bob@shu.ac.uk’ the name/value pair submitted is equal to email=m.j.cooper@shu.ac.uk.

HTTP Protocol

The way data is sent using 'get' and 'post' forms part of the HTTP protocol and therefore is not unique to PHP. You can use modern browser developer tools to see the data been sent in the HTTP requests.



firstname: Martin
surname: Cooper
email: m.j.cooper@shu.ac.uk

$_POST and $_GET

In PHP the values are made available to the receiving page. How they are retrieved depends on which method was used to send the data in the form ie post or get.

- `$_POST['fieldname']` - retrieves values from a HTML form that has used the method ‘’post’.
- `$_GET['fieldname']` - retrieves values from a HTML form that has used the method ‘get’.

(`$_POST` and `$_GET` are known as superglobal variables.)

The HTML form method ‘post’ sends the name/values pairs to the next page in the header of the http page request. As such the values are hidden from the user.

The ‘get’ methods places the name/values pairs in URL of the page request.

```
ie:
http://www.mysite.com/process.php?email=m.j.cooper@shu.ac.uk
```

Multiply values will be concatenated in the URL with ampersands as follows:

```
http://www.mysite.com/process.php?email=m.j.cooper@shu.ac.uk
&name=Martin
```

This is what is known as a 'query string'.

A 'query string' can be created by appending a HTML link with a ‘?’ and then the name/value pair.

As indicated multiply values can be sent by using the ‘&’ concatenation character.

Looping the Super Global Associate Array

Both $_POST and $_GET are associate arrays. You can loop through the value and keys using foreach() ie:

```
foreach ($_POST as $key=> $value) {
  echo $key;
  echo $value;
}
```

Checking for values with isset(), Ternary and the Coalescing Operator

The PHP method `isset()` is used to check is a value is present. It will return a Boolean value ie

```
if(isset($_POST['firstname']){
…
}
```

An alternative to an if/else statement is the Ternary operator. This has the syntax

```
Question ? value if true : value if false
```

For example the following uses `isset()` to see if there is a posted value for firstname and assigns it to the `$firstname` variable if found, else a value of 'No Name' is assigned

```
$firstname = isset($_POST['firstname']) ?
$_POST['firstname'] : 'No Name';
```

(the above should all be on one line)

In PHP the above construct can be refactored with Coalascing Operator (two question marks) as follows:

```
$firstname = $_POST['firstname'] ?? 'No Name';
```

Task 1 - POST from a Form

1. Open the file *formpost.php*. Notice that the file contains a HTML form with various form elements. Also note that the form has a method of `post`. Add an `action` pointing the form to *formpostprocess.php*.

2. Open the file *formpostprocess.php*.

3. Use the `$_POST` superglobal variable to retrieve and display the values from the form using echo.

4. The superglobal `$_POST` represents an associate array of values sent in the http headers. You can prove this by outputting the array with `print_r()`.

```
print_r($_POST);
```

5. You can also loop the array using the a `foreach()`.

```
foreach ($_POST as $key=> $value) {
  echo $key;
  echo $value;
}
```

Task 2 - GET from a Form

1. Open the file *formget.php*. Notice that the file contains a HTML form with various form elements. Also note that the form has a method of `get`. Add an `action` pointing the form to *formgetprocess.php*.

2. Open the file *formgetprocess.php*.

3. Use the `$_GET` superglobal variable to retrieve and display the values from the form.

4. The superglobal `$_GET` represents an associate array of values sent in the query string. You can prove this by outputting the array with `print_r()`.

5. Use a foreach loop to output the name / value pairs.

Task 3 - GET from a Query String

1. Open the file *querystring.php*.

2. This file contains a link to the page *querystringprocess.php*. Appended the link with a "?" to add a name/value pair for the fieldname "email".

3. Extend the file to include two more variables.

4. Open the file *querystringprocess.php*. Use the appropriate superglobal variable to retrieve and display the values from the hyperlink.

Task 4 - isset(), Ternary and the Coalescing Operator

Extend the querystring example to set a default value is no value is included in the URL.

1. Use `isset()` in combination with the ternary operator to see if a specific value was sent ie:

```
$email = isset($_GET['email']) ? $_GET['email'] : 'No Email
Received;
echo $email;
```

2. With PHP 7 we can use the coalescing operator to refactor this comment construct as:

```
$email = $_GET['email'] ?? 'No Name Received ';
```

Form Values and isset()

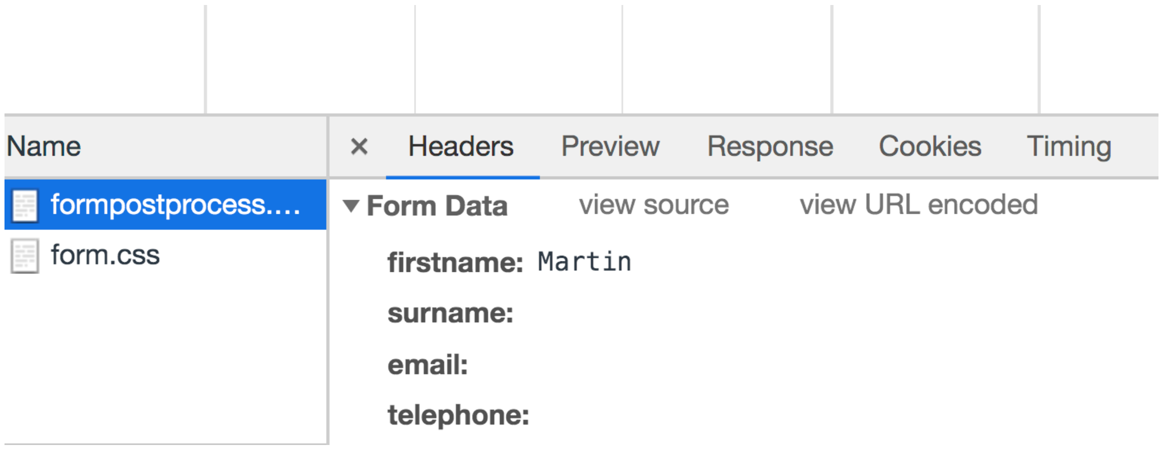
If you try either of these examples with the either the `$_POST` or the `$_GET` FORM, then they won't work as expected. If a text form field is left blank in a submitted form it will return an empty string. The value therefore exists but is empty.

However, if a checkbox is left unchecked then no value, empty or otherwise is returned.

In the `$_POST` form example you could use the following to detect if there is a value for `firstname` and whether the checkbox for `mailingList` was checked:

```
// firstname is a text field so check if empty
$firstname = ($_POST['firstname'] !== "" ) ?
$_POST['firstname'] : 'No Name';
echo $firstname;
// mailingList is a checkbox so check is exists
$mailingList = (isset($_POST['mailingList'])) ?
$_POST['mailingList'] : 0;
echo $mailingList;
```

Use the Chrome Inspector to view the data submitted ie:



firstname: Martin
surname:
email:
telephone:

The above shows that a value for `firstname` was submitted but also that an empty string for `surname` was submitted.