Lecture 3 - Tutorial

We are going to create an application that posts some form data to a separate php file on submission

Step 1.

Create a new file in Notepad++ (or whatever is your favourite editor)

Enter the following to create a simple html page

```
<!DOCTYPE>
<html>
        <head>
        </head>
        <body>
        </body>
</html
```

Save as **form.php** in a new folder ***"lecture3/tutorial***" under the **htdocs** folder

Step 2.

Add the form to your web page, between the <body> and </body> tags enter

```
<h2>Form Enter</h2>
<form action="grab_values.php" method="GET">
User Name: <input type="text" name="user_name" />
Location: <input type="text" name="user_location" />
Password: <input type="password" name="password" />
<input type="submit" value="submit">
</form>
```

Save your file and test in a browser by typing the address

localhost/lecture3/tutorial/form.php

Step 3.

Create a new php program in the same folder as above called **grab_values.php**

```
<?php
echo 'Hi '.$_GET["user_name"];
echo $_GET["user_location"].' is a cool place';
?>
```

Save and run by typing into browser address i.e.(make sure apache running)

localhost/lecture3/tutorial/form.php

Step 4.

Enter data to form and press submit. Observer the address bar, what do you notice ?

Step 5.

Lets add another input field, this time for entering a password. To **form.php** add

```
Password: <input type=”password”   name=”password” />
```

Just before the submit input field

Step 6.

Refresh your browser to bring up an new form i.e.

> *localhost/lecture3/tutorial/form.php*

Enter form data and press submit. What do you see now ? Why is this bad ?

Step 7

Let’s change our request method to POST instead of GET. Change **form.php** to use “POST” as the method in the form tag.

Change **grab_values.php** and replace all occurrences of $_GET with $_POST. Save and refresh form .

Enter data to form and submit.

What is the difference in the address bar of the browser ? Can you see the password field ?

Step 8.

Finally we will put some checking into our ***grab_values.php*** program to ensure it cannot be run outside of a post form action .

To help us with this we will use a PHP global variable $_SERVER, this is an associative array which contains a bunch of information about the http request that has been made. We are in particular interested in the value of the element ‘REQUEST_METHOD’ which tells us if the form action was a GET or POST (or something else). In our case we want the value to be “POST”.

To see all elements of the $_SERVER variable we can use the ***print_r()*** function which allows us to view the contents of an array.

If the value of $_SERVER[‘REQUEST_METHOD’] != “POST” then we want to flag an error and not continue executing .

So our ***grab_values.php*** program will now look like

```
<?php
 print_r($_SERVER);
 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        echo 'Hi '.$_POST["user_name"];
        echo $_POST["user_location"].' is a cool place';
        }
 else {
                echo "<br/>"."Bad Request , must come via POST";
        }
?>
```

To test this try running ***grab_values.php*** outside of the post method i.e. by entering, in the browser address bar

localhost/lecture3/tutorial/grab_values.php

Run the form thru the browser by going back to

localhost/lecture3/tutorial/form.php

Enter some form data, press submit and observe the values of $_SERVER variable.

Note: -

We have a potential problem. We are able to refresh the ***grab_values.php*** file once submission has happen. This potentially means we could resubmit the form multiple times. In the next lecture we will see potential ways to resolve this using sessions and redirects.