Web Development

Lecture 4

CA Schedule

Moodle Quiz will take place on the 14th Nov. Will cover weeks 1 - 7. 20% overall mark

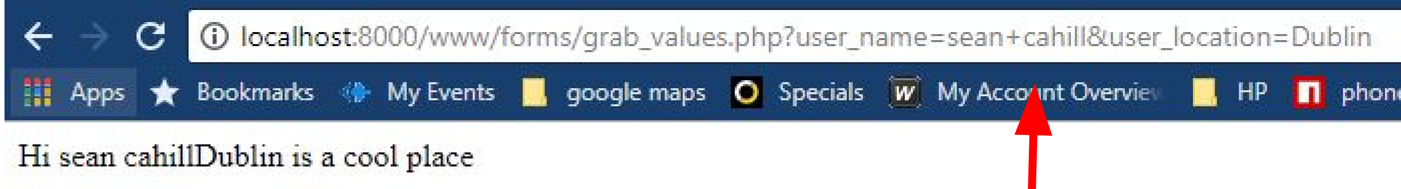
Project details at end of class next week. To be delivered Dec. 15th. 30% overall mark. Covering all topics week 1 - 12

Written exam in Jan 2018. Covers all lectures. 50% overall marks

This Session

- HTTP GET and POST
- The $_SERVER Global Variable
- Sessions
- Move to Single Page

HTTP Get


```
localhost:8000/www/forms/grab_values.php?user_name=sean+cahill&user_location=Dublin
```

Hi sean cahillDublin is a cool place

Problem with HTML GET method is that the values sent to the server are visible over the browser address bar, therefore sending sensitive data such as passwords, secret keys using GET method is not advisable.

Amount of data can be sent using this method is limited to the length of characters allowed in URL by different browsers. Here are few things to note while using GET method :

- GET parameters remain in browser history because they are part of the URL
- GET requests can be bookmarked.
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

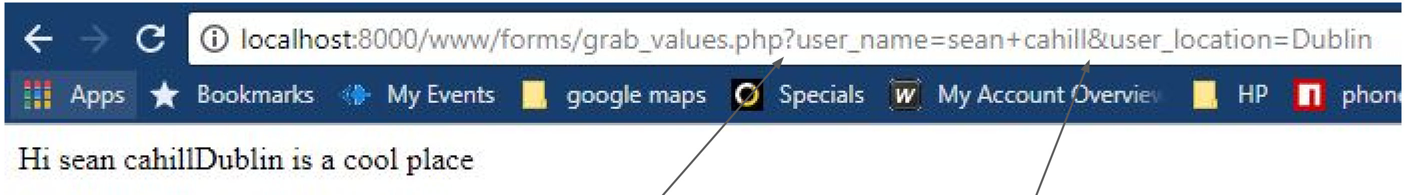
So the preferred way of sending data to server has been the POST method.

HTTP Get

Good technique and standards requirements say that you never use **GET** for an operation that has "side effects," or that actually *does* something.

For example, we’re just looking at data, so no side effects affect the operation. But, ultimately, we're going to add the data to the database, which is, by definition, a side effect.

URLs With Parameters



```
localhost:8000/www/forms/grab_values.php?user_name=sean+cahill&user_location=Dublin
```

Hi sean cahillDublin is a cool place

Parameters can be passed with a URL request and then accessed in to calling program or service. Parameters are key/value pairs.

Parameter list begins with **?** and are separated by **&**

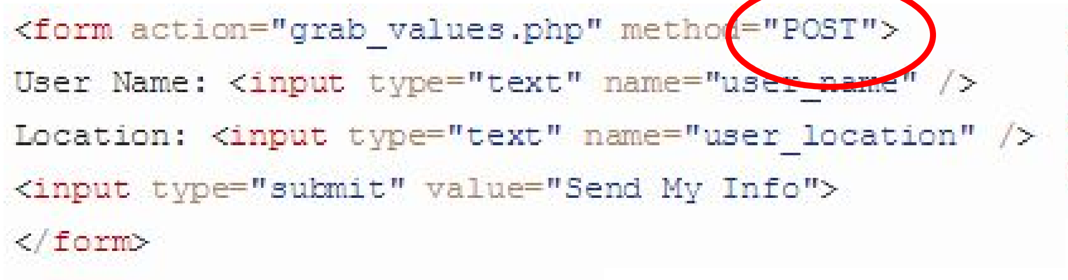
user_name=sean+cahill

key/value

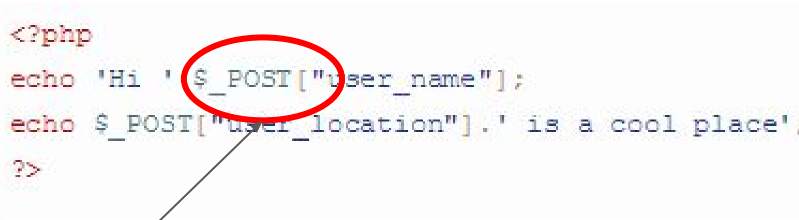
HTML POST Method

A large sized post request can be sent using POST method. POST method works similar way as GET, but it is safer than GET because the values are not visible on the address bar and will not be stored in browser history. HTTP **POST** requests supply additional data from the client (browser) to the server in the message body.

To use POST just change element method to POST as shown below :



```
<form action="grab_values.php" method="POST">
User Name: <input type="text" name="user_name" />
Location: <input type="text" name="user_location" />
<input type="submit" value="Send My Info">
</form>
```



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

And to collect the input field values, we will just use PHP ***$_POST*** Predefined Global Variable:

HTTP Post

Benefits of using POST method.

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length
- POST variable are not included in url address

$_Server Variable

When we issue a http request we can use the PHP Global variable

$_SERVER to access information about the http request

Bring up your files from last week or download Form Example files on moodle

To your file “secondpage.php” Add the line

```
print_r($_SERVER);
```

Have a look of the output from the print statement when you submit the form . One of the elements of this associative array is the **‘REQUEST_METHOD’** element, which tells us if the request is a POST or GET request

HTTP Get

Why use GET request ?

In your **form.php** file add the line, outside the form tag but inside the body tags

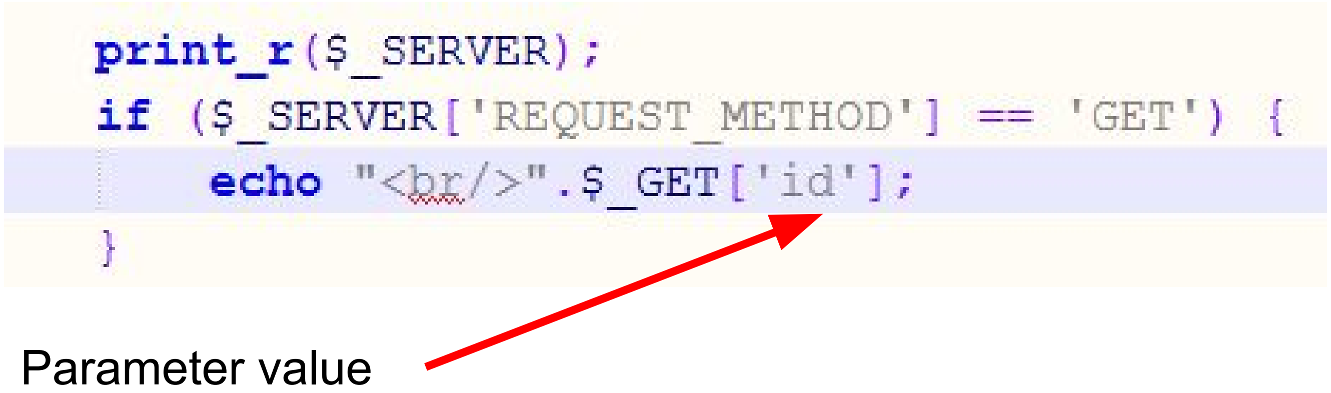
```
<a href=”secondpage.php?id=25”>Get Request</a>
```

By default this is a GET request, here we are passing a parameter to our next page

$_SERVER Variable

Let's use the 'REQUEST_METHOD' element to determine if the request was a POST or GET. We can examine also if there was a parameter passed

At the top of the "secondpage.php" file use an **if** statement to print out the parameter value if a get request was made



```
print_r($_SERVER);
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    echo "<br/>".$_GET['id'];
}
```

$_SERVER Variable

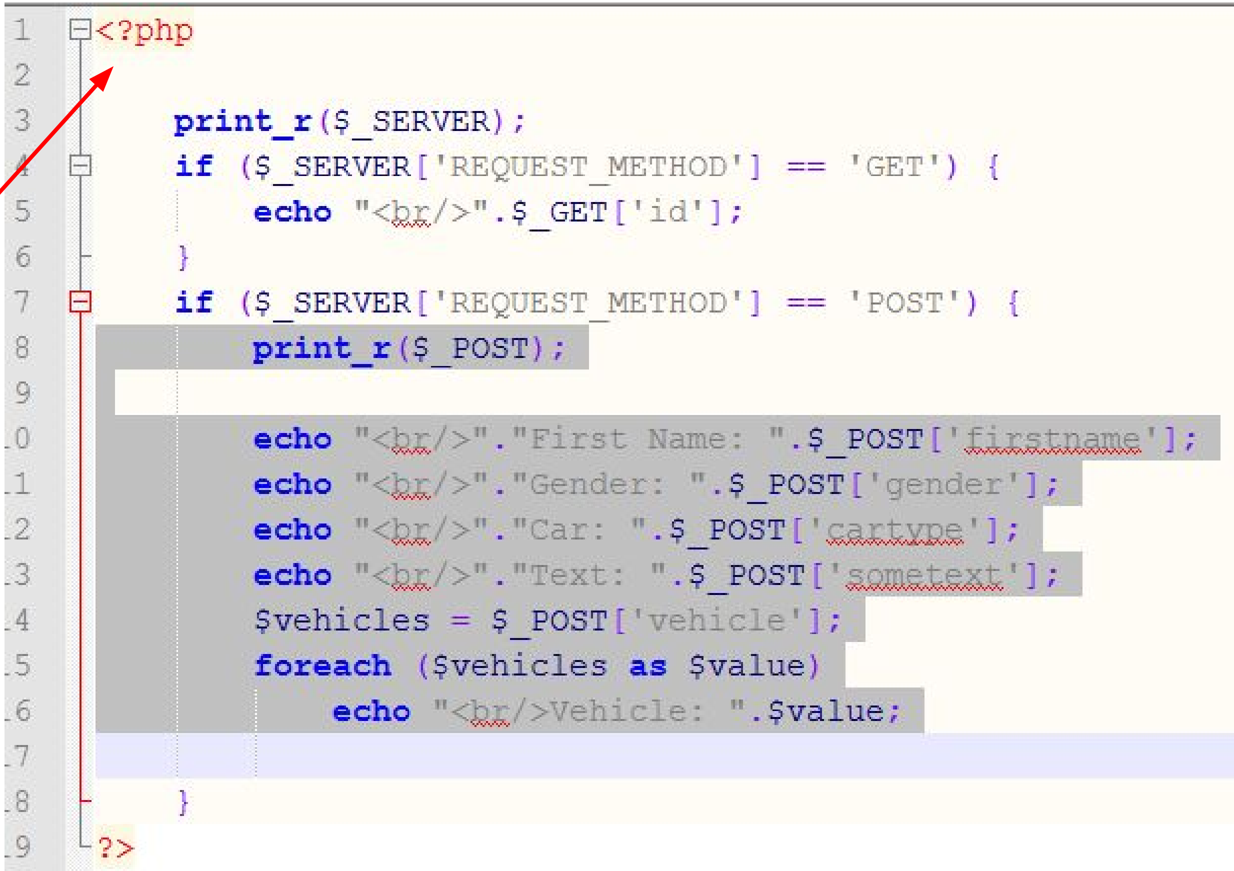
You will notice that when we click on the href link we get errors because the POST values from our form are empty.

We can add a check in “secondpage.php” to ensure we only process the form values if a post request has been issued.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        echo "<br/>"."First Name: ".$_POST['firstname'];
        echo "<br/>"."Gender: ".$_POST['gender'];
        echo "<br/>"."Car: ".$_POST['cartype'];
        echo "<br/>"."Text: ".$_POST['sometext'];
        $vehicles = $_POST['vehicle'];
        foreach ($vehicles as $value)
                echo "<br/>Vehicle: ".$value;

        }
```

$_SERVER Variable

So our “secondpage.php” file should now look like, with the GET and POST checks in place.



```
<?php

    print_r($_SERVER);
    if ($_SERVER['REQUEST_METHOD'] == 'GET') {
        echo "<br/>".$_GET['id'];
    }
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        print_r($_POST);

        echo "<br/>"."First Name: ".$_POST['firstname'];
        echo "<br/>"."Gender: ".$_POST['gender'];
        echo "<br/>"."Car: ".$_POST['cartype'];
        echo "<br/>"."Text: ".$_POST['sometext'];
        $vehicles = $_POST['vehicle'];
        foreach ($vehicles as $value)
            echo "<br/>Vehicle: ".$value;

    }
?>
```

Single Page

Up till now we have been processing our form in a separate php file i.e “secondpage.php”

But before doing any further processing e.g. storing to a database we may need to return to the form if the user has made an error(s).

It makes sense to have the validation of our form on the same page as the form itself to avoid the complication of returning to the form from another file.

We will see more on validation in a later lecture.

So we will combine our two files into one file

Single Page

Take the contents of “secondpage.php” and paste the entire file to the top of the **form.php** file, including the <?php and ?> tags. But without the GET request code. So at the top of form.php we should have

```
<?php
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        print_r($_POST);

        echo "<br/>"."First Name: ".$_POST['firstname'];
        echo "<br/>"."Gender: ".$_POST['gender'];
        echo "<br/>"."Car: ".$_POST['cartype'];
        echo "<br/>"."Text: ".$_POST['sometext'];
        $vehicles = $_POST['vehicle'];
        foreach ($vehicles as $value)
            echo "<br/>Vehicle: ".$value;

    }
?>
```

Single Page

But we now need to make a change so that we don’t move to a new php file once we have submitted our form, but remain on the same file and highlight any errors to the user.

To do that we need to make a change to the “action” of the form i.e. instead of the action moving to “secondpage.php” it now just executes itself i.e. **form.php**

```
<form action="form.php" method="POST">
```

This is another reason to have the **if condition** checking if there was a POST request. As the first time that the file is executed the value of $_SERVER[‘REQUEST_METHOD’] will not be set so our form processing code will not be executed

Sessions

Why use sessions? A session is a chunk of data maintained at the server that maintains state between HTTP requests. HTTP is fundamentally a stateless protocol; sessions are used to give it statefulness.

But can’t we pass values as parameters from one file to another ?

Yes but we’ve seen that it’s not secure, especially when using a GET request

Sessions are useful, particularly when, for example a user has logged into an application. We can keep a record of their login details from file to file using a session

Sessions

Are Sessions the same as Cookies?

We’re going to see more of cookies later in the lectures but they are not quite the same

A cookie is a bit of data stored by the browser and sent to the server with every request.

A session is a collection of data stored on the server and associated with a given user

The main **difference between** a **session** and a **cookie** is that **session** data is stored on the server, whereas **cookies** store data in the visitor's browser. **Sessions** are more secure than **cookies** as it is stored in server. **Cookie** can be turn off from browser and/or deleted by the client.

Sessions

How do we use sessions?

PHP global variable $_SESSION is basically a variable that stores temporary information about current user, which can be retrieved later from any page within the site, as long as the session is active.

PHP $_SESSION creates a unique id for each user. PHP session can exactly identify each user and serve them stored data, and it is considered safe too, because session information are not accessible from the client browser, and there are no easy way to hijack a session from remote computer.

Sessions

Starting PHP Session

To start a PHP session, we use session_start() function. It must be on the top of HTML or any text. Something like this :

```
<?php
session_start();  // starts PHP session

//other PHP codes
?>
<HTML>
</HTML>
```

When session_start() is called, it either starts a new session or resumes the existing session.

Sessions

When the session is active, we can store values in PHP *$_SESSION*, later which can be accessed from any PHP script.

```
<?php
// starts PHP session
session_start();
//store a session variable
$_SESSION["name"] = "Sean";
//retrieve and output stored session variable
echo $_SESSION["name"];
?>
```

Once we have stored the variable, we can retrieve it anywhere within the same website, all we have to do is resume existing session we’ve created in above example and output the value.

```
<?php
// starts PHP session
session_start();
//retrieve and output stored session variable
echo $_SESSION["name"];
?>
```

Confirmation Page

Once we have validated our form it makes sense to move to another php file to confirm that we have a valid form

We can use a session to retain information from a previous page

At the very top of **form.php** add the code

```
<?php
// starts PHP session
session_start();
?>
```

Confirmation page

Inside our form.php file we will add some code to save a session variable and pass it on to a new file, **confirm.php**

```
$_SESSION["name"] = $_POST['firstname'];
```

Add this code in the **form.php** file inside the if statement

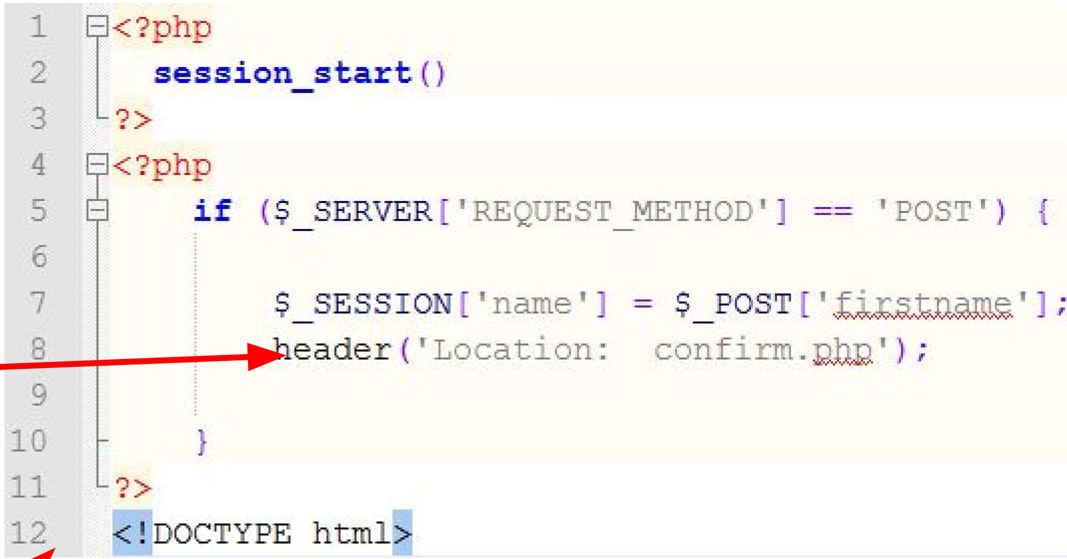
Confirmation page

How do we pass control to a new php file ?

We can use the php **header** function

header('Location: confirm.php');

We need to remove any print or echo statements, so that our php code at the top of **form.php** becomes

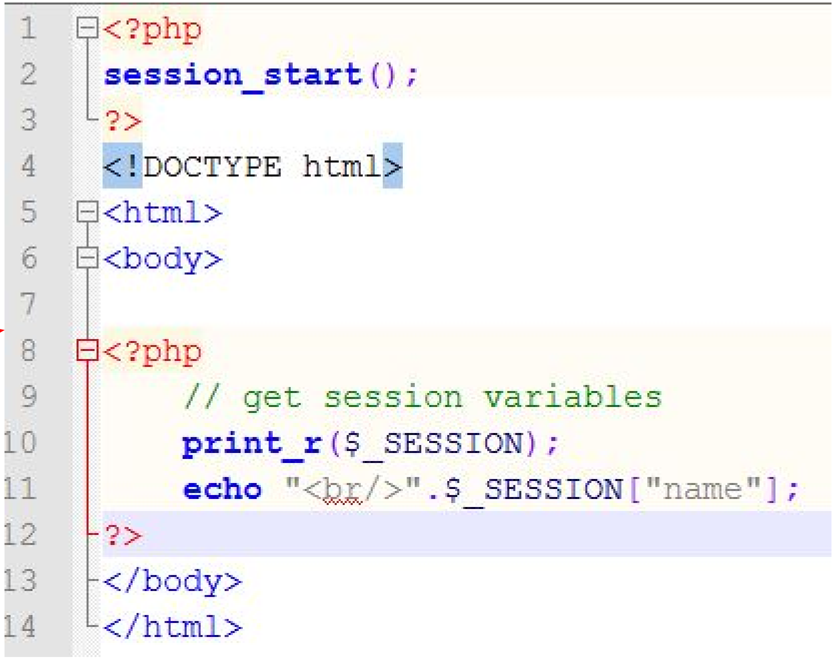


```
<?php
  session_start()
?>
<?php
    if ($_SERVER['REQUEST_METHOD'] == 'POST') {

        $_SESSION['name'] = $_POST['firstname'];
        header('Location:  confirm.php');

    }
?>
<!DOCTYPE html>
```

Confirmation Page

We now need to create our confirmation page **confirm.php** and we can output the value of our session variable.

Create the following file in the same folder as **form.php**



```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
	// get session variables
	print_r($_SESSION);
	echo "<br/>".$_SESSION["name"];
?>
</body>
</html>
```

Next Time...

We’ll move on to MySql and how to

Create

Read

Update

Delete

CRUD

Data in a MySql database using PHP code