Web Development

Lecture 2 - Intro to PHP

Last Time

Looked at :-

- Web Architecture
- HTTP request-response
- Web Servers - Apache HTTPD
- Intro to PHP
- XAMPP

Todays Lecture

- Review of last week
- Look at XAMPP config files and how Apache is configured
- PHP
  - Variables
  - Data structures
  - Loops
  - Forms
- HTTP GET/POST Methods

XAMPP - Configuration

Let’s take a look at a few of the XAMPP components and their settings

Bring up the XAMPP console

Bring up a command tools

In command tool navigate to c:\xampp\ i.e. >cd c:\xampp

Issue dir command to list all folders and files in this directory i.e. >dir

We’re going to look at the php, htdocs and config directories (folders)

PHP folder

Navigate to php folder i.e. >cd php

Issues dir command >dir

In this directory we can see the php interpreter php.exe

This is effectively what Apache uses to parse or execute php files

**Demo executing php file from the command line

Create simple php file in the www directory (folder)

Web Server Alternative

If you are having issues with installing and running Apache HTTP in XAMPP there is an alternative

A web server comes bundled with php installation

To run a web server from the php folder

- Bring up a command tool
- Navigate to c:/xampp/php and issue the command
- c:/xampp/php>php -s localhost:8080
- Store php files in the **www** folder
- To request files in browser type http://localhost:8080/www/filename.php

Apache httpd - config file

We'll take a look at some of the configuration settings in the httpd.config file

This file can be accessed via the XAMPP console or using you text editor

Using a text editor find the file c:\xampp\apache\conf\httpd.conf

Lets look at some of the settings

Line 223 **ServerName localhost:80**

Http address

Port setting: change here if want to start Apache on different port

Apache httpd - config file

Line 58: **Listen 80** - which port Apache is listening for Requests

Line 247: **DocumentRoot "C:/xampp/htdocs"**

Under the **htdocs** folder we place our folders and files we want Apache web server to serve up to us on request

e.g.

http:\\localhost\lecture2\form.php <=> c:\xampp\htdocs\lecture2\form.php

** must use left hand side in order for Apache web server to execute php code

PHP

PHP is an interpretive language

PHP is typeless

What is an example of a strongly typed language ?

Variables

A variable is a placeholder for data. You can assign a value to it, and from then on, any time PHP encounters your variable, it will use the value instead.

In PHP, variable names are case-sensitive, so $UserName is a different variable from $username.

PHP - Variables

The **type** a variable assumes is determined on assignment

```
$str = “this is a string”;
```

The variable $str is now of type **String**

To determine type of a variable use

```
gettype($var_name);
```

Experiment in repl.it

PHP supports the following data types:

- String.
- Integer.
- Float (floating point numbers - also called double)
- Boolean.
- Array.
- Object.
- NULL.

PHP Variables - Testing type

- is_array() - Finds whether a variable is an array
- is_bool() - Finds out whether a variable is a boolean
- is_callable() - Verify that the contents of a variable can be called as a function
- is_float() - Finds whether the type of a variable is float
- is_int() - Find whether the type of a variable is integer
- is_null() - Finds whether a variable is NULL
- is_numeric() - Finds whether a variable is a number or a numeric string
- is_object() - Finds whether a variable is an object
- is_resource() - Finds whether a variable is a resource
- is_scalar() - Finds whether a variable is a scalar
- is_string() - Find whether the type of a variable is string
- function_exists() - Return TRUE if the given function has been defined

e.g.

```
$firstname = ‘Sean’;
echo “\n”.is_string($firstname); // returns true;
echo “\n”.is_int($firstname); // returns false;
```

PHP Variables - undefined

If a variable has not been defined or has not been assigned a value we can use

The ***isset($var_name)*** function

```
if (isset($vardef))
    echo "\nvariable defined";
 else
    echo "\nvariable not set or undefined";
```

PHP Strings

We can create strings using single or double quotes

$str = ‘This is a string’;

$str = “And so is this”;

Note: Variables begin with **$**
end each statement with semi-colon **;**

To include special characters in a string e.g. “ use a backslash

```
$str = “This is an example \”of a quote\” in a string”;
echo $str;
```

Try it in repl.it

PHP Strings

Adding two strings together (concatenation)

```
$firstName = ‘Sean’;

$lastName = ‘Citizen’;

echo $firstName . “ “ . $lastName;
```

Dot . notation

Add Blank space

Try it in repl.it !

PHP Strings

Number of characters in a string

$str = “Sean Citizen”;

*echo **strlen**($str); // 12 n.b. includes blank character*

Trim characters from front and end of string

$str = “ abc defg “;

echo trim($str,” “); // remove blank char from string “abc defg”

Other string functions see http://php.net/manual/en/ref.strings.php

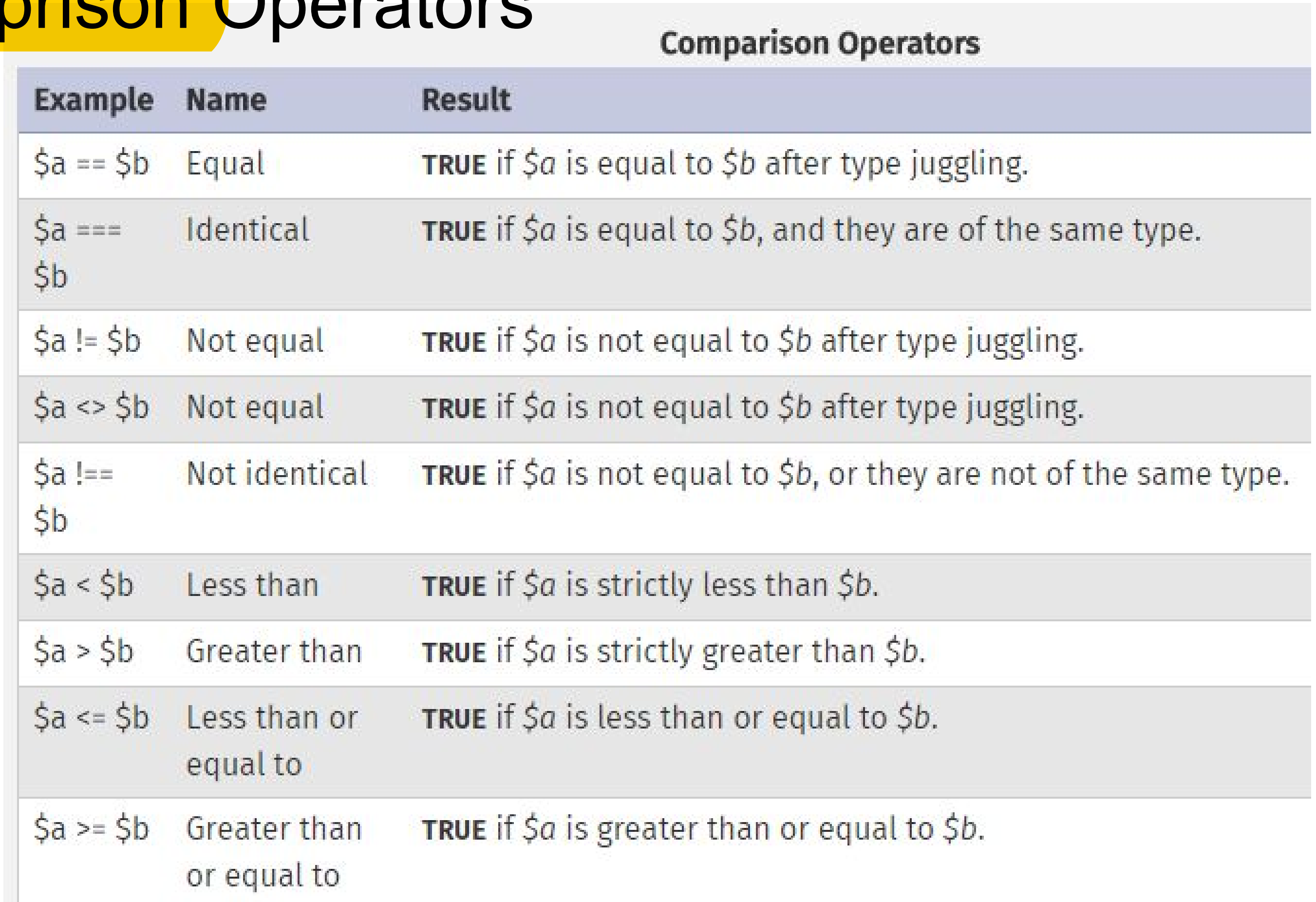
PHP - Conditional Statement

if - else conditional

```
If ( condition is true) {
        statement1;
        statement2; }
else {
        statement1;
        statement2; }
```

Note: drop {} if only one statement

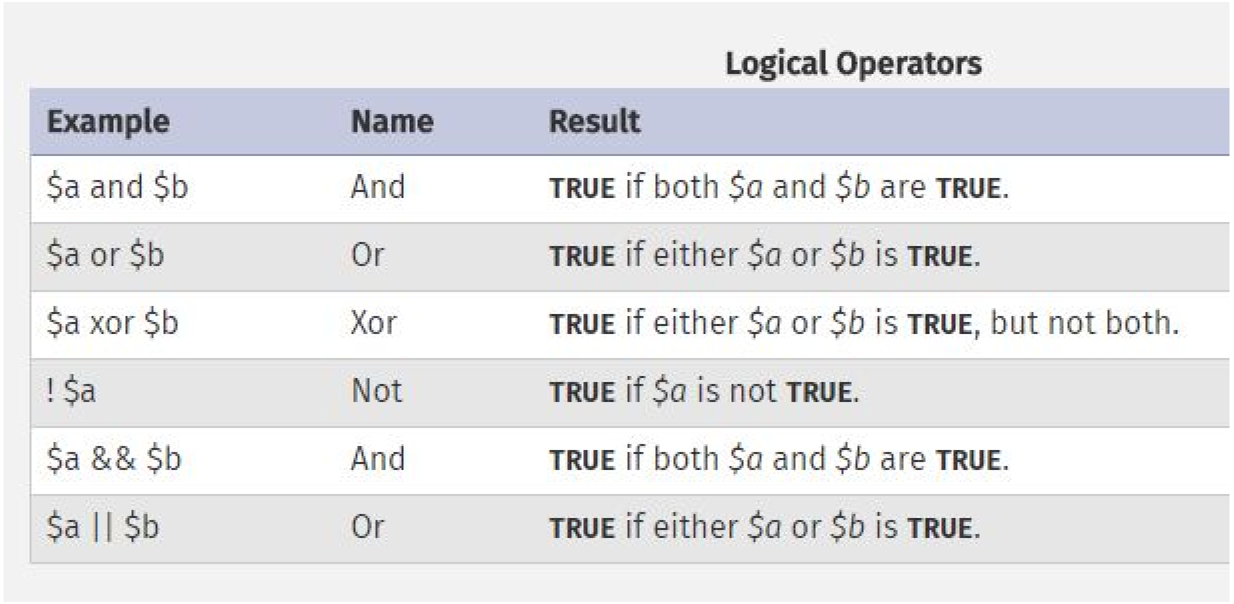
PHP - Comaprison Operators



Comparison Operators

Example	Name	Result
$a == $b	Equal	**TRUE** if *$a* is equal to *$b* after type juggling.
$a === $b	Identical	**TRUE** if *$a* is equal to *$b*, and they are of the same type.
$a != $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a <> $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a !== $b	Not identical	**TRUE** if *$a* is not equal to *$b*, or they are not of the same type.
$a < $b	Less than	**TRUE** if *$a* is strictly less than *$b*.
$a > $b	Greater than	**TRUE** if *$a* is strictly greater than *$b*.
$a <= $b	Less than or equal to	**TRUE** if *$a* is less than or equal to *$b*.
$a >= $b	Greater than or equal to	**TRUE** if *$a* is greater than or equal to *$b*.

PHP - Logical Operators



Logical Operators

Example	Name	Result
$a and $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a or $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.
$a xor $b	Xor	**TRUE** if either *$a* or *$b* is **TRUE**, but not both.
! $a	Not	**TRUE** if *$a* is not **TRUE**.
$a && $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a || $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.

PHP - for loop

The syntax of a *for* loop is:

```
for (expr1; expr2; expr3)
    statement
```

Place statement in { } if more than one statement

expr1 - is evaluated (executed) once unconditionally at the beginning of the loop.e.g $i=0;

At the beginning of each iteration, **expr2** is evaluated. If it evaluates to TRUE, the loop continues and the nested statement(s) are executed. If it evaluates to FALSE, the execution of the loop ends.

At the end of each iteration, **expr3** is evaluated (executed). e.g. i++;

PHP - Other loops

while loop

```
    while (condition true)
        Statement
```

do - while loop

```
    do {
        statements
    } while (condition true)
```

Question: What's the difference between two

PHP Data Structures - Numeric Arrays

A Numeric Array element consists of an association between an **index** and a **value**, where the **index** is a number and the **value** can be any object e.g. string, number etc.

```
$car[1] = “ford”;
```

PHP Data Structures - Numeric Arrays

Define an array using build **in array()** function

```
$fruit = array(“orange”,”apple”,”plum”,”pear”);
```

Access array use square bracket notation with index

```
echo $fruit[0] ;//    “orange”
echo $fruit[1] ;//    “apple”
echo $fruit[2] ;//    “plum”
echo $fruit[3] ;//    “pear”
```

Note: Index starts from 0

New Element

```
$fruit[4] = “grape”;
```

PHP Data Structures - Associative Array

Numeric Array *$employee[0] = “sean”*

Associative Arrays *$employee[“firstName”] = “sean”*

Associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as a string so that you can establish a more meaningful relationship between the index and the value.

In Associative Arrays we sometimes replace the word **index** for **key** and refer to the association between keys and values as **key/value pairs**

PHP Data Structures - Associative Array

Dictionary

What is the ‘Key’

What is the ‘Value’

Key - the word we’re searching for

Value - the definition of that word

PHP Data Structures - Associative Arrays

Two Ways to create:

```
/* First method to create associate array. */
 $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);
```

Note:
the arrow (**=>**) notation for creating the association

```
/* Second method to create array. */
 $salaries["mohammad"] = "high";
 $salaries["qadir"] = "medium";
 $salaries["zara"] = "low";
```

PHP Data Structures - Array Functions

Count function: elements in array

```
$fruit = array("apple","orange","pear","plum");
echo "\n".count($fruit); // 4

$groceries = array("fruit" => array("apple","orange","pear","plum"),
                   "veggies" => array("beans","carrots","cabbage","potato"));
echo "\n".count($groceries);      // 2
```

PHP Data Structures - Array Functions

Print out value of array using print_r function

```
$fruit = array("apple","orange","pear","plum");
print_r($fruit);
//
        Array
        (
            [0] => apple
            [1] => orange
            [2] => pear
            [3] => plum )
```

PHP Data Structures - Array Functions

Remove Element from end of an array, ***array_pop(**$arrayname**)***

```
$a=array("red","green","blue");
array_pop($a);
print_r($a);
// Array(“red”,”green”);
```

Add one or more elements to end of array, *array_push*(*$arrayname,$element1,$element2,.....)*

```
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);    // Array(“red”,”green”,”blue”,”yellow”)
```

PHP Data Structures - Array Functions

Associative Array

Return array of all key values ***array_keys($arrayname)***

```
$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
print_r(array_keys($a));        //Array (  [0] => “Volvo”,
                                           [1] => “BMW”,
                                           [2] => “Toyota”)
```

More Array functions
https://www.w3schools.com/php/php_ref_array.asp

PHP Data Structures - Array Iteration

The ***foreach*** construct provides an easy way to iterate over arrays. *foreach* works only on arrays and objects There are two syntaxes:

```
foreach (array_expression as $value)
    Statement
```

The first form loops over the array given by *array_expression*. On each iteration, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element).

```
foreach (array_expression as $key => $value)
    Statement
```

The second form will additionally assign the current element's key to the *$key* variable on each iteration.

PHP Data Structures - Array Iteration

```
$fruit = array("apple","orange","pear","plum");
print_r($fruit);

$cars=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
print_r(array_keys($cars));

foreach ($fruit as $value)
  echo "\n".$value;

foreach ($cars as $key => $value)
  echo "\n".$key." ".$value;
```

Alternative

```
for ($i=0;$i<count($fruit);$i++  )
    echo "\n".$fruit[$i];
```

PHP Strings cont...

explode(“delimiter”,string.);/ delimiter blank space / : etc.*

```
<?php
// Example 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2
?>
```

Forms

Forms allow the user of a website to enter in some information to your website.

All websites use forms in some way, or use elements from forms for different purposes e.g. Registration form, airline booking etc.

Forms

When we want to make a form in HTML, we use the **form** html tag!

> *<form>*
>
> *</form>*

Each form has a number of different forms elements that can be added. We will take a look at time of the HTML form elements that we can use.

Create a new folder under the htdocs folder,say “**lecture2**”. Open a new file in this folder and work along with me by adding to this file. We’ll start by adding some basic html and save as, say, **form.php**

Forms -Input field

```
<input type='text' name='firstname'></input>
```

The **type** is set as text Try change this to password and see what happens! (it will hide the characters!)

This is the **name** we want to give that field (we will use the name later)

Forms - Radio Button

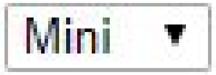
```
<input type='radio' name='gender' value='Male'></input>
<input type='radio' name='gender' value='Female'></input>
```

the type is a set to **radio .** We set the **value** that we give if the option is picked. You will notice that both of the names are set to the same **name**, this is because we want to treat both of these options as part of the same group, so we give them the same name.

Select Menu

The name we are giving this select menu

```
<select name='cartype'>
  <option value='Mini'>Mini</option>
  <option value='Volvo'>Volvo</option>
  <option value='BMW'>BMW</option>
  <option value='Saab'>Saab</option>
</select>
```



If they pick Saab, we will save audi as the value

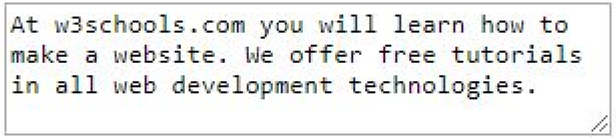
This is what it looks like. If you click the little arrow, the menu drops down.

Text Area

<textarea rows=’3’ cols=’40’ name=’sometext’>

At w3schools.com you will learn how to make a website. We offer free tutorials in all web development technologies.

</textarea>



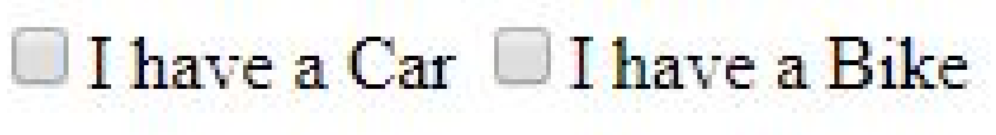
```
At w3schools.com you will learn how to
make a website. We offer free tutorials
in all web development technologies.
```

rows = number of rows
cols = number of columns

Checkbox

Note: Indicates an array of values

<input type=’checkbox’ name=’vehicle[]’ value=’Car’>I have a Car</input>
<input type=’checkbox’ name=’vehicle[]’ value=’Bike’>I have a Bike</input>

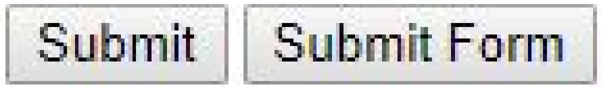


For a complete list of input types see:-

https://www.w3schools.com/tags/att_input_type.asp

Submit Button

<input type='submit' ></input'>



If you want to change the text to something else, you can add value attribute:

<input type='submit' value='Submit Form'></input'>

Form Submission

What happens when we hit submit?

At the moment nothing will really happen at all when the submit button has been pressed.

We need to tell the page what to do when the button is clicked!

Http Request Methods

We mentioned in previous lecture that the method of request is included in the header information sent with the request. In our example we saw we were using a GET method.

Http has a number of methods for sending a http request. We will look specifically a the GET and POST methods.

Setting Action and Method

The **action** is the page we want to send the values to

```
<form action=”secondpage.php” method=”post”>

    Username: <input type=’text’ name=”username”></input>
    <input type=”Submit”></input>

</form>
```

The method we use is called a **post**

Form Submission - Action

When we submit a form we populate PHP global variables with data relating to the values we have entered on our form

PHP **$_GET** & **$_POST** are part of PHP global variables, and can be called anywhere in PHP scripts, both $_GET & $_POST captures data sent with HTML post and get methods.

Form Submission

On the second page..

Assume the values have been sent to the second page.

Now we want to gather them and print them out.

All we need to do is access the **post** object in PHP and pass the name of the field we want

```
<?php
    $name = $_POST['username'];
    echo $name;
?>
```

Breakdown of the Code

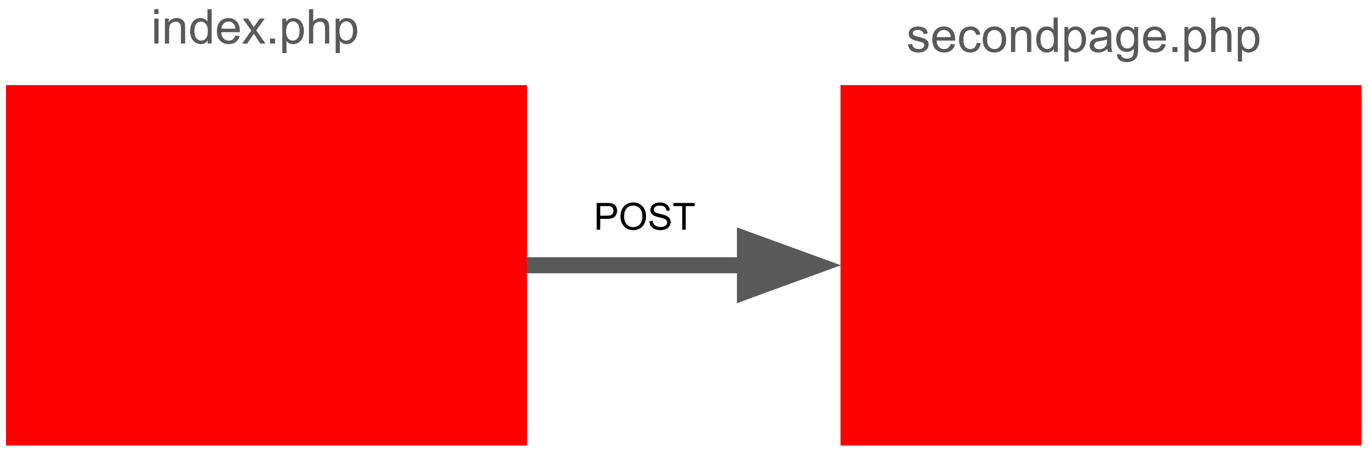
```
$name = $_POST[‘username’]; // Access the post object
```

We make a variable called $name

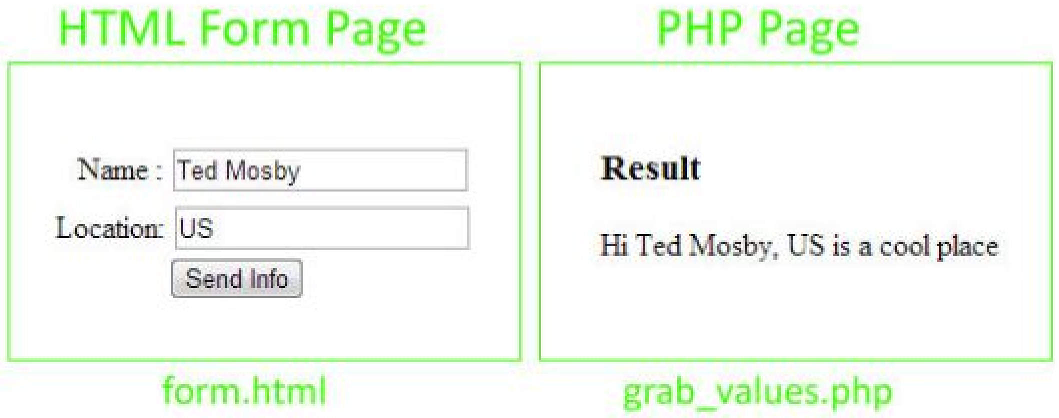
We are telling PHP to give us the field called **username** from the POST object. You will notice that we had a field called username in the previous piece of HTML.

Form Submission

The values from the first page will be gathered and sent to the second page.



HTTP Get Method



Name : Ted Mosby

Location: US

Result

Hi Ted Mosby, US is a cool place

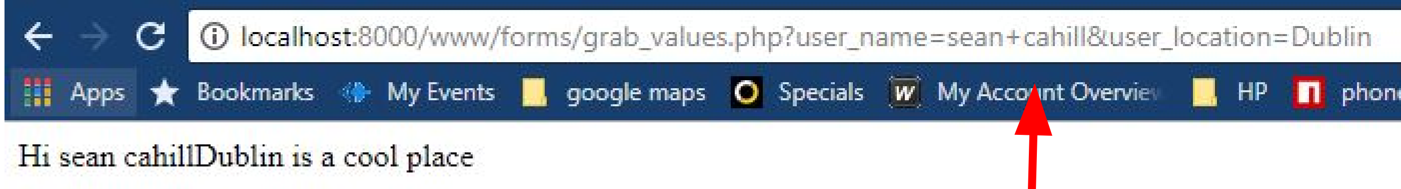
```html
<form action="grab_values.php" method="GET">
User Name: <input type="text" name="user_name" />
Location: <input type="text" name="user_location" />
<input type="submit" value="Submit">
</form>
```

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

When user fills form elements and clicks submit button, the data is sent to “grab_values.php”, which is specified in *action* form element.

Picture above illustrates a typical HTML form procedure with GET method.

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

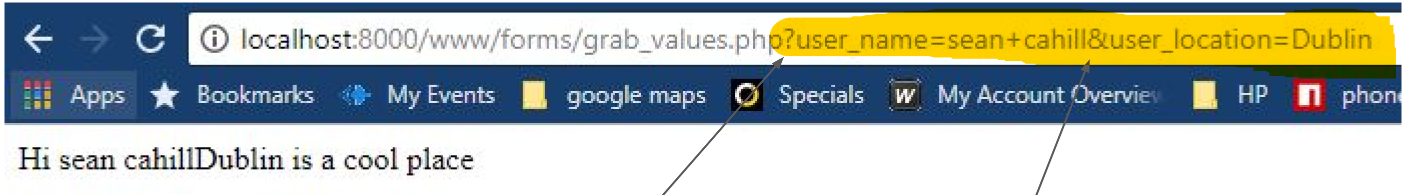
So the preferred way of sending data to server has been the POST method.

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 **&**

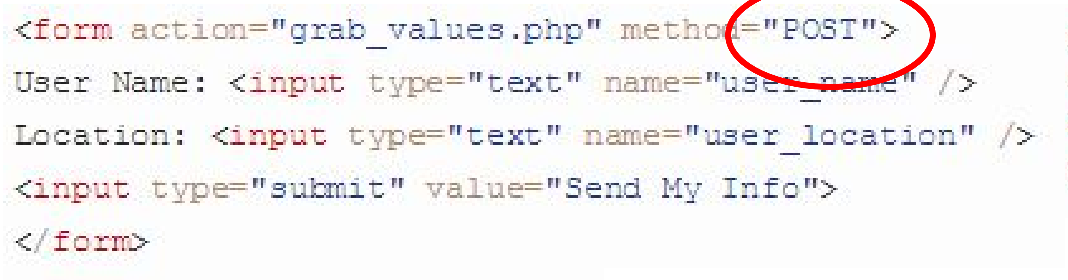
user_name=sean+cahill

key/value

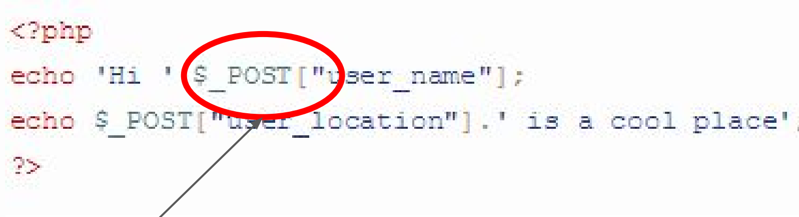
HTML POST Method

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

In Class Exercise

Let’s go back to the form we have been building

Create a new PHP page e.g. ‘*form_posted.php*’ in same folder to display the values entered via the Form. How do we display the checkbox values ?

So action=”*form_posted.php*” and method = ‘POST’

Next Time….

PHP cont…

- Sessions
- Cookies
- PHP Functions
- PHP Objects

Server Side Validation

PHP - Include files

Finally

A complete form web page with some additional css up on moodle

More css styling for forms
https://www.sanwebe.com/2014/08/css-html-forms-designs