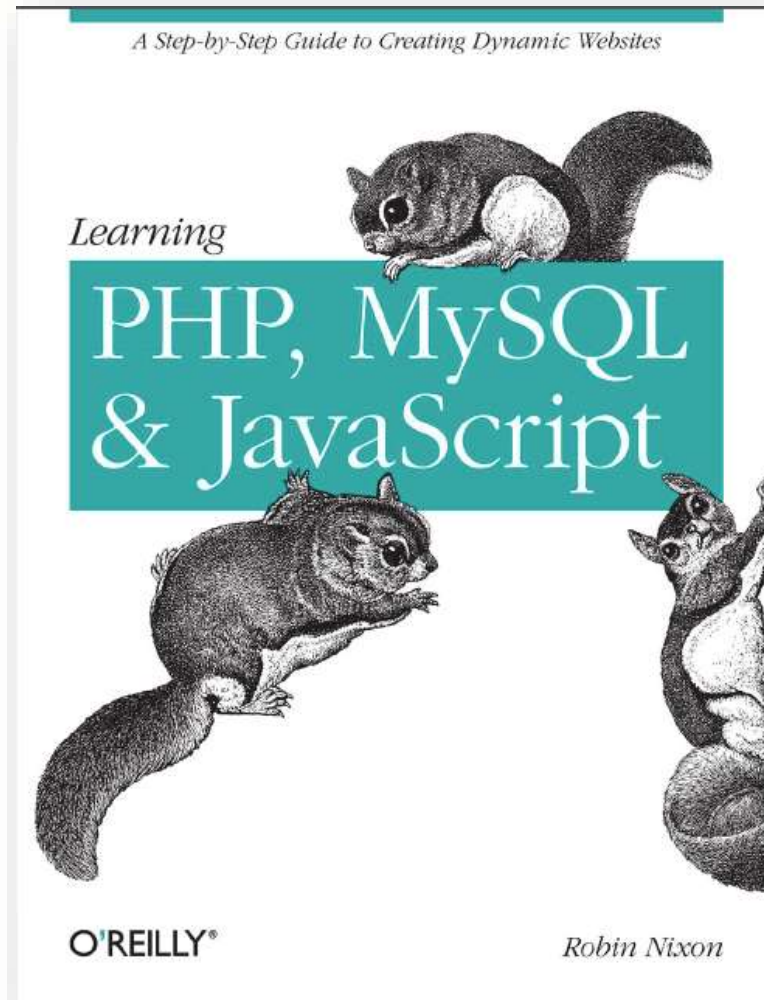


CENG420

Chapter 5 PHP



CHAPTER 3 & 4 IN



WHAT IS IT?

- **Server side scripting**
 - Server sends file to a scripting engine before sending it to the user
 - Very powerful language that can take care of all your operations in a website:
 - variables
 - Functions
 - Database entry and modification
 - form handling
 - maintaining sessions and cookies
 - etc
 - Similar to the C language syntax and to perl language (Variables start with \$ sign)



HOW TO USE IT?

PHP can be embedded in your page along with html and javascript.

NOTE

whenever your page contains any PHP, no matter how small the code is, the whole page is considered a php type and should be saved with a .php extension.

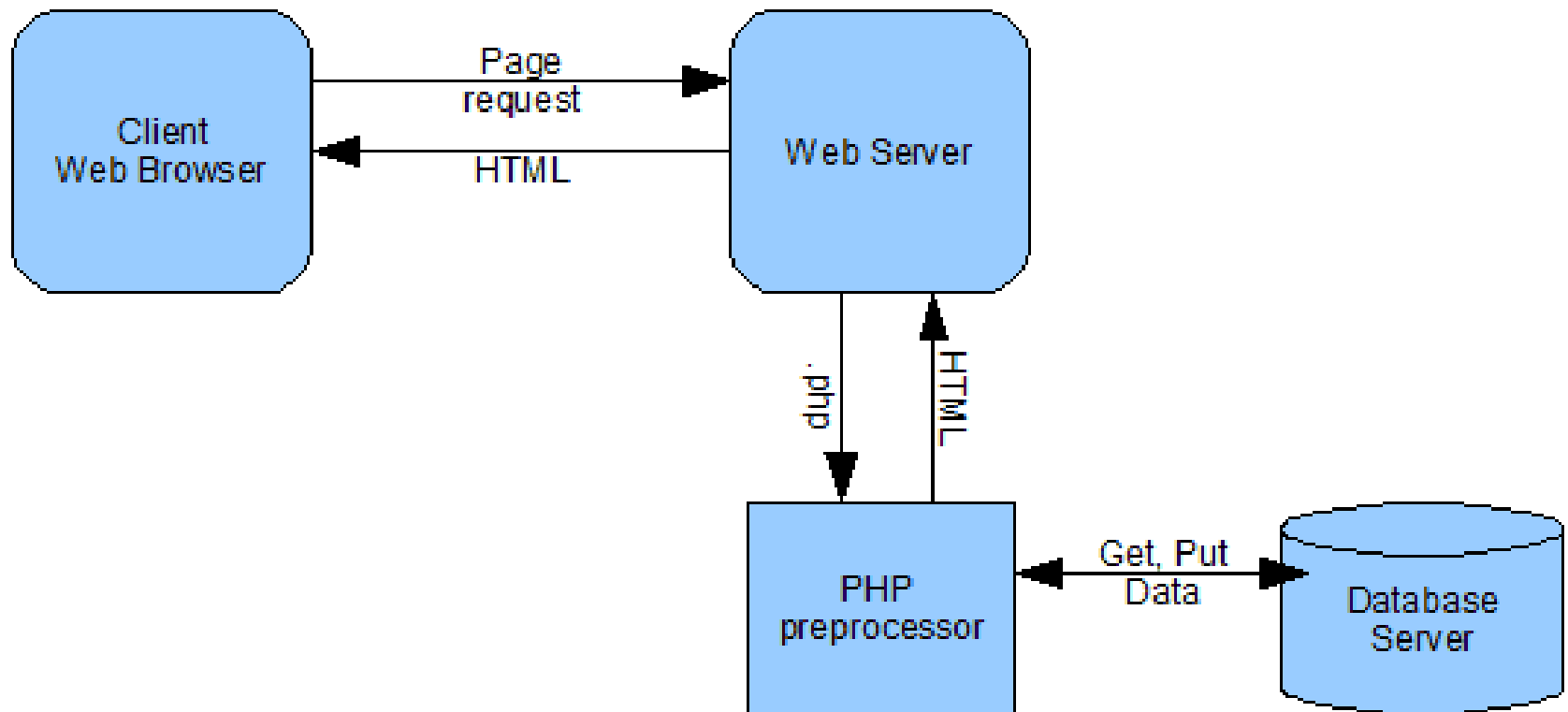


WHERE ARE THEY STORED?

- PHP scripts are stored at the server.
- Whenever the user requests a page, the server executes the script, converts it into html (and JS/CSS if any) and then sends to the browser.
- The browser doesn't receive any single php code nor it does know how to execute it.
- In normal cases, we place all our website (a collection of html, css, js, and php files) into a folder (and subfolders) on the server.
- In this class, we will place them all in a local server. Mainly we will use XAMPP (or MAMP for mac users)



TO VISUALIZE THIS

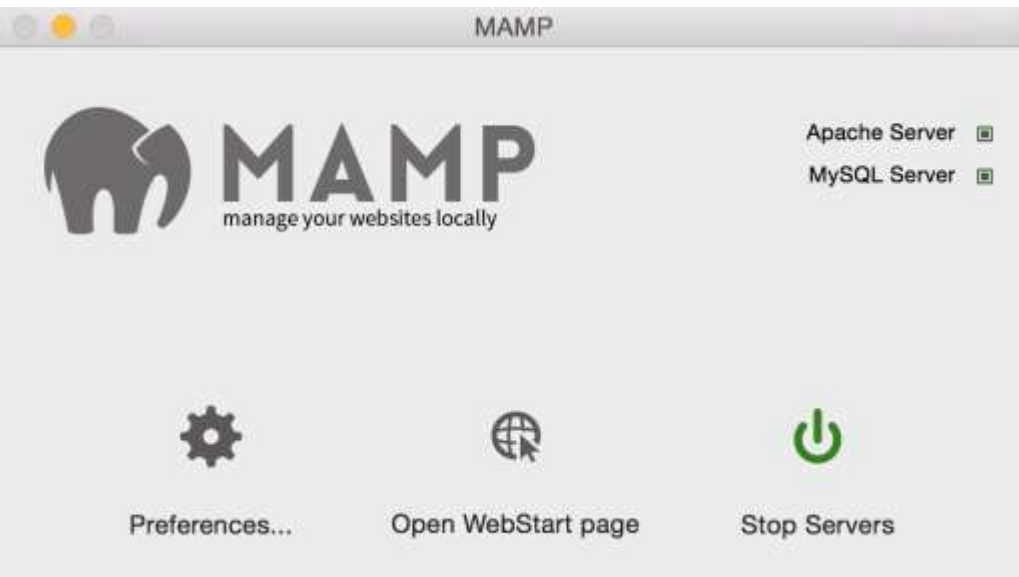


XAMPP

- **XAMPP** is a free and open source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.
- It will be also used to create our databases later and connect to them via PHP.
- Once you install xampp, open it and launch the apache and mysql servers.
- If they fail to launch, it might be a port problem. Apache and vmware virtual machine use the same port. You might need to override and change one port.



XAMPP/MAMP RUNNING STATE



PHP
[phpinfo](#) shows the current configuration of PHP.

MySQL
MySQL can be administered with [phpMyAdmin](#).
To connect to the MySQL server from your own scripts use the following connection

MAMP Version
3.4 → Your version is up-to-date.

News
[MAMP PRO for Windows now on sale](#)

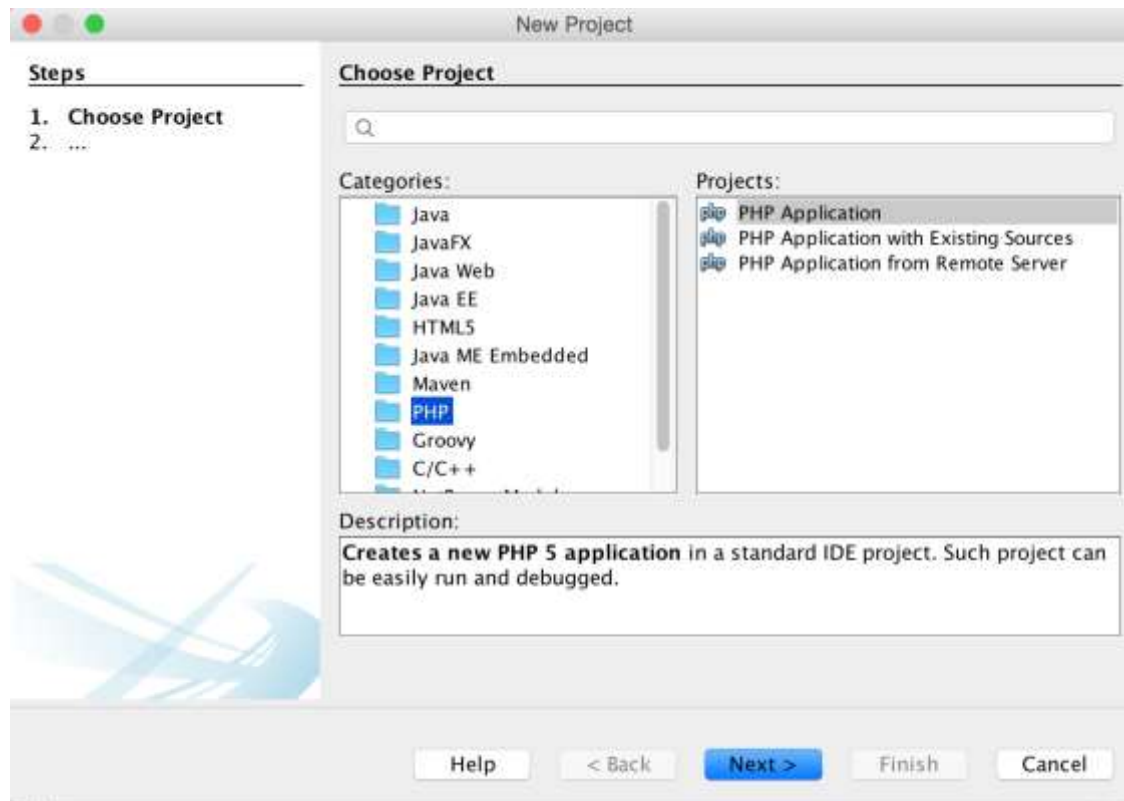


Creating the project in NetBeans



IN NETBEANS

- Create a new project of type “PHP”.
- Normally, if you have xampp installed, the default folder for saving the files will be *xampp/htdocs/projectname*



THIS IS A MAC VERSION. ON WINDOWS, WILL BE
XAMPP/HTDOCS...

New PHP Project

Steps

1. Choose Project
2. **Name and Location**
3. Run Configuration
4. PHP Frameworks
5. Composer

Name and Location

Project Name:

Sources Folder:

Document root for MAMP

PHP Version:

PHP version is used only for hints

Default Encoding:

N.B.

If you can't find php type for new projects, you might need to install the plugin from netbeans.

Go to tools-> plugins->available plugins and download the php.



PART 1

Creating PHP code blocks



1 STANDARD DELIMITERS

`<?php statements; ?>`

- This could be embedded anywhere in your page.

Example:

`<?php echo "Explore Africa"; ?>`

- **echo** is similar to print.



2 SHORT DELIMITERS

<? *statements*; ?>

- Not always supported
- To be safe, use the standard



EXAMPLE 1

`<body>`

`<h1> Multiple Script Sections </h1>`

`<h2> First Script Section </h2>`

 `<?php echo "<p>Output from the first script section.
</p>"; ?>`

`<h2> Second Script Section </h2>`

 `<?php echo "<p>Output from the second script section.
</p>"; ?>`

`</body>`



EXAMPLE 2

```
<h1>First Phplesson</h1>
```

```
<p>
```

```
<?php
```

```
    echo "Hi there.";
```

```
    $answer = 6 * 7;
```

```
    echo "The answer is $answer, what ";
```

```
    echo "was the question again?";
```

```
?>
```

```
</p>
```

```
<p> Yes another paragraph.</p>
```



COMMENTS

```
// this is a comment
```

```
# this is a comment
```

```
/* this is
```

```
    All a
```

```
    Comment
```

```
*/
```



PART 2

Variables, Arrays, & functions



VARIABLES

- Case sensitive
- Start with a dollar (\$) sign

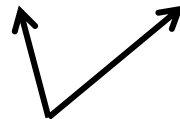
```
$variable_name = variable_value
```

```
$myname = "john";
```

```
$x=$y;
```

```
echo $x;
```

```
echo "<p>the age is", $x, ".</p>";
```



EXAMPLE 2

```
<?php  
$x = "15" + 27;  
echo($x);  
echo("\n");  
?>
```

42



ARRAYS

```
$team = array('abc','def','ghi');
```

```
echo $team[2] //will give 'ghi'
```

```
$team[] = "xyz"; //this will add to the first available  
index.
```



VARIABLES (CONT.)

- ✓ Variable names are case sensitive
- ✓ Operators are the same: +, -, ++, --, +=,
- ✓ .= (for strings)
- ✓ Comparison operators are the same as well: ==, !=, ...
- ✓ Logical operators: &&, ||, !, and, or, xor
- ✓ The and, or and xor have low precedence but used for databases (later)



STRINGS

The “.” operator concatenates strings

```
echo “hello ”.”I am here”  
//produces “hello I am here”
```

```
echo “I have “.$x.“apples”  
//produces I have 5 apples
```

`$b .= $c` will concatenate c to b.



LOOK AT THIS

`$info='I have $x apples' → I have $x apples`

`$info="I have $x apples" → I have 5 apples`

The double quotes behaves like "." operators.

`$info = 'My sister\'s car is a \'Mercedes\''`

`$info = 'Date \t Name \t Payment'`

`\t` is tab

`\n` is newline

`\r` is carriage return



VARIABLE TYPING AND IMPLICIT CASTING

PHP is similar to JavaScript in terms of implicit casting

```
<? $number=12345*76890;  // = 949207050  
echo substr($number,3,1);  
// will return 2, $number becomes string.  
?>
```

- Substr will be covered later.

```
$pi = "3.14";
```

```
$radius=5;
```

```
echo $pi*($radius*$radius) // turns string into number.
```



CONSTANTS

```
define("Root_location","/usr/local");  
$dir = Root_location;
```



FUNCTIONS

```
function longdate($timestamp)
{ return date("l F j S y", $timestamp);}
Echo longdate( time());
// time(), date() are php functions.(will be covered later)
```

GLOBAL VARIABLES

```
global $is_logged_in;
```



EXPLICIT CASTING

```
$speed="55mph";  
$speed=(int)$speed*2;
```

```
echo gettype($x);  
echo is_numeric($x);  
echo is_string($x);  
echo is_int($x);
```



LOOPS

Same as all others:

- `if() {} else {}`
- `do {} while {}`
- `while() {}`
- `switch() { case: ...}`
- `for() {}`



LOOPS (PART 2)

Additional loops:

foreach: used with arrays

Two syntaxes available:

```
foreach($arrayname as $variablename){}
```

```
foreach($Aname as $indexname => $vname){}
```



foreach SYNTAX 1

```
$DaysOfWeek =  
    array("Monday", "Tuesday",  
        "Wednesday", "Thursday", "Friday",  
        "Saturday", "Sunday");  
foreach ($DaysOfWeek as $Day)  
{  
    echo "<p>$Day</p>";  
}
```

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

Sunday

foreach SYNTAX 2

```
foreach ($DaysOfWeek as $DayNumber => $Day)
{
    echo "<p>Day $DayNumber is $Day</p>";
}
```

```
Day 0 is Monday
Day 1 is Tuesday
Day 2 is Wednesday
Day 3 is Thursday
Day 4 is Friday
Day 5 is Saturday
Day 6 is Sunday
```



PART 3

Strings & String functions



STRINGS

```
$quote="<p>hello</p>"
```

```
echo $quote → hello
```

```
$quote='<p>'hello'</p>'
```

```
echo $quote → "hello"
```

Vice versa:

```
$quote="<p>'hello'</p>"
```

```
echo $quote → 'hello'
```



STRINGS

```
$vegetable =“tomato”;
```

```
Echo “<p>I have a $vegetable? </p>”
```

→ I have a tomato?

```
Echo “<p>I have 5 $vegetables </p>”
```

→ I have 5 (empty because ***\$vegetables*** is considered 1 word and the variable is undefined)



SOLUTION TO THE PREVIOUS PROBLEM

“<p> I have 5 {\$vegetable}s </p>”

Or

<p>I have 5 \${vegetable}s </p>

Or

echo "<p> I have 5 ",\$vegetable,"s </p>";



CAUTION

I have 5 { \$vegetable}s
Returns
I have 5 { tomato}s



STRING FUNCTIONS

You can check <http://www.php.net/manual/en/ref.strings.php> for complete API.

[strlen — Get string length](#)

[str_word_count — Return information about words used in a string](#)

[ucfirst — Make a string's first character uppercase](#)

[lcfirst — Make a string's first character lowercase](#)

[strtok — Tokenize string](#)

[strtolower — Make a string lowercase](#)

[strtoupper — Make a string uppercase](#)



STRING FUNCTIONS

htmlspecialchars_decode —
Convert special HTML entities
back to characters

```
<?php
$str = "<p>this -&gt; &quot;</p>\n";

echo htmlspecialchars_decode($str);
```

```
<p>this -> "</p>
```

htmlspecialchars — Convert
special characters to HTML
entities: **&** becomes **&**, **<**
becomes **<**

md5 — Calculate the md5 hash of
a string

- used to store passwords in
databases for security
reasons

trim — Strip whitespace (or
other characters) from the
beginning and end of a string



STRING FUNCTIONS

substr_compare — Binary safe comparison of two strings from an offset, up to length characters

substr_count — Count the number of substring occurrences

substr_replace — Replace text within a portion of a string

substr — Return part of a string



SUBSTR

```
string substr ( string $string , int $start [,  
int $length ] )
```

Start:

- If **start** is non-negative, the returned string will start at the **start**'th position in **string**, counting from zero. For instance, in the string *'abcdef'*, the character at position 0 is 'a', the character at position 2 is 'c', and so forth.
- If **start** is negative, the returned string will start at the **start**'th character from the end of **string**.
- If **string** is less than or equal to **start** characters long, FALSE will be returned.



SUBSTR

```
<?php
```

```
$rest = substr("abcdef", -1);    // returns "f"
```

```
$rest = substr("abcdef", -2);    // returns "ef"
```

```
$rest = substr("abcdef", -3, 1); // returns "d"
```

```
?>
```

- **Length:**
- If **length** is given and is positive, the string returned will contain at most **length** characters beginning from **start** (depending on the length of **string**).
- If **length** is given and is negative, then that many characters will be omitted from the end of **string** (after the start position has been calculated when a **start** is negative). If **start** denotes the position of this truncation or beyond, false will be returned.
- If **length** is given and is 0, FALSE or NULL an empty string will be returned.
- If **length** is omitted, the substring starting from **start** until the end of the string will be returned.



SUBSTR EXAMPLES

```
<?php
```

```
$rest = substr("abcdef", 0, -1);
```

// returns "abcde"

```
$rest = substr("abcdef", 2, -1);
```

// returns "cdef"

```
$rest = substr("abcdef", 4, -4);
```

// returns "cd"

```
$rest = substr("abcdef", -3, -1);
```

// returns "def"

```
?>
```



EXAMPLE ON SUBSTR

```
<?php
```

```
echo substr('abcdef', 1); // bcdef
```

```
echo substr('abcdef', 1, 3); // bcd
```

```
echo substr('abcdef', 0, 4); // abcd
```

```
echo substr('abcdef', 0, 8); // abcdef
```

```
echo substr('abcdef', -1, 1); // f
```

```
// Accessing single characters in a string
```

```
// can also be achieved using "square brackets"
```

```
$string = 'abcdef';
```

```
echo $string[0]; // a
```

```
echo $string[3]; // d
```

```
echo $string[strlen($string)-1]; // f
```



SOME ADDITIONAL FUNCTIONS

strrev — Reverse a string.

strpos — Find the position of the first occurrence of a substring in a string or returns FALSE.

str_replace — Replace all occurrences of the search string with the replacement string

substr_replace — Replace text within a portion of a string



EXAMPLES

```
$email = "mghantous@liu.edu.lb";  
$newemail=str_replace("mghantous",  
"miladghantous", $email);
```

```
$nameend=strpos($email,'@');  
$newemail=substr_replace($email,"miladghantous",  
0,$nameend)
```



STRING COMPARISONS

`strcmp("Dan", "Don")` returns a value < 0

`strcmp("Don", "Dan")` returns positive value.

`strcmp("Don", "Don")` returns 0

Check out:

`Similar_text`, `levenshtein`, `strncmp`, `explode`,
`implode`, `strsplit`



PART 5

Array Functions



ASSOCIATIVE ARRAYS

- Instead of using indices (numbers) to access the cells of an array (A[1], A[6]), we can use more descriptive names.
- Example: T[first_name], T[Last_name]
- Think of a database tuple retrieved from a table: we access the attributes by number (first attribute is 0, second is 1, ...), or by name (→ Associative)

```
<?php
$paper['copier'] = "Copier & Multipurpose";
$paper['inkjet'] = "Inkjet Printer";
$paper['laser'] = "Laser Printer";
$paper['photo'] = "Photographic Paper";
echo $paper['laser'];
?>
```



ASSOCIATIVE ARRAYS (CONT.)

We can also do this:

```
<?php
$p1 = array("Copier", "Inkjet", "Laser", "Photo");

$p2 = array('copier' => "Copier & Multipurpose",
            'inkjet' => "Inkjet Printer",
            'laser' => "Laser Printer",
            'photo' => "Photographic Paper");

echo "p2 element: " . $p2['inkjet'] . "<br>";
?>
```



PRINT_R

print_r — Prints human-readable information about a variable

```
$v = array('a','b','c','d');
```

```
Array ( [0] => a [1] => b [2] => c [3] => d )
```

```
print_r($v);
```

```
$movies=array('title' =>  
'avatar', 'length'=>'160',  
'studio'=>'fox');
```

```
Array ( [title] => avatar [length] => 160  
[studio] => fox )
```

```
print_r($movies);
```



IS_ARRAY()

```
echo (is_array($fred)) ? "Is an array" : "Is not  
an array";
```

COUNT()

Counts the number of elements in an array.

```
echo count($fred); //single dimensional  
echo count($fred, 1); //multi-dimensional
```



EXPLODE()

take a string containing several items separated by a single character (or string of characters) and then, place each of these items into an array.

One handy example is to split a sentence up into an array containing all its words

```
<?php
$temp = explode(' ', "This is a sentence with
seven words");
print_r($temp);
?>
```

```
<?php
$temp = explode('***',
"A***sentence***with***asterisks");
print_r($temp);
?>
```

Array
(
[0] => This
[1] => is
[2] => a
[3] => sentence
[4] => with
[5] => seven
[6] => words
)

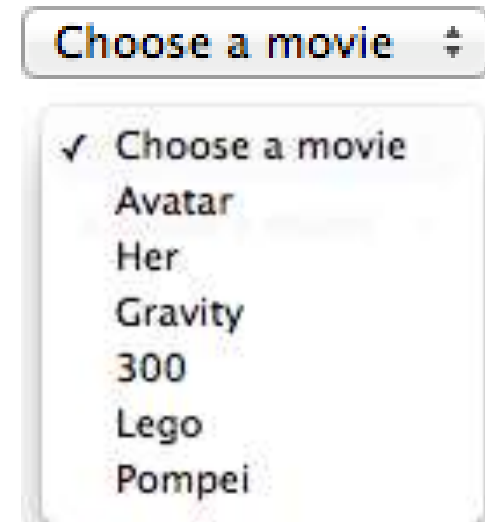


PRACTICE

- 1) Use echo to create an array containing several movies.
(in normal cases, this array is a database query answer, but here we will use an array)
- 2) Create a “SELECT” list populated by these movies.
- 3) Hint: the <select> syntax is the following:

<select>

```
<option value= -1>choose a movie</option>
<option value="value1">some value 1</option>
<option value="value2">some value 2</option>
<option value="value3">some value 3</option>
</select>
```



Choose a movie

- ✓ Choose a movie
- Avatar
- Her
- Gravity
- 300
- Lego
- Pompei



CENG420
Server side
Lecture 2

PHP

Handling user input



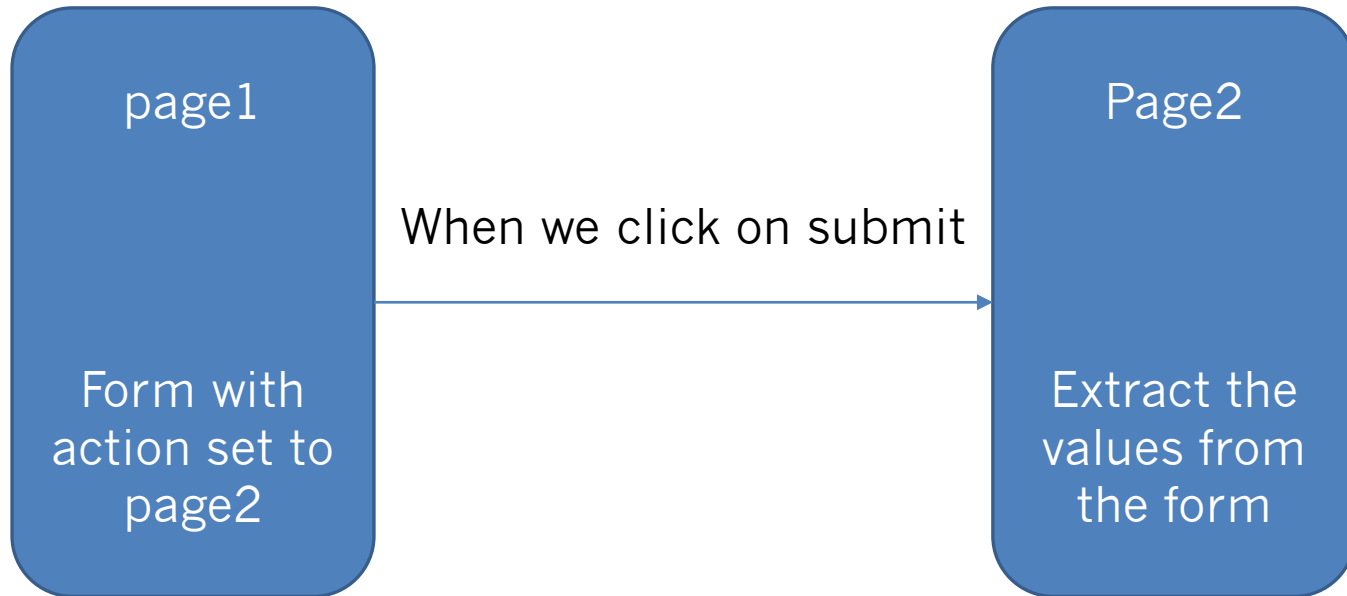
Why?

- The majority of websites have forms
- Users use forms to input values.
- Whether, you're filling an application online or you're commenting on facebook or posting to a forum.
- When the form is submitted, those values should be found somewhere and can be accessed!
- Hints: `$_POST`, `$_GET`, `$_FILES`,...

What really happens?

```
<form action="somepage.php" method="GET">
<input type="text" name="fname"/>
<input type="submit" value="next"/>
</form>
```

- Assume that the above html code is inside ***index.php*** or ***index.html*** page
- When the submit button is clicked, this form is sent to ***somepage.php*** (which is on the server)
- The code inside `somepage.php` should be able to extract the values that the user has input in the form. In our case, we have 1 input only (fname)



- Page 2 might contain html code or might not!
- It means, page 2 might be sent back to the client to see it or might do background jobs (like saving to DB) and then redirecting to another page that will be sent from the server back to client.

PHP helps you!

- It creates associative arrays ready for you to use when you submit a form.
- Access those arrays and read those values.
- **The index of those arrays are the “name” fields in the form**
- Some of them are editable → assign values instead of reading (covered later!)

PHP associative arrays are called AutoGlobals

Array	Description
<code>\$_COOKIE</code>	An array of values passed to the current script as HTTP cookies
<code>\$_ENV</code>	An array of environment information
<code>\$_FILES</code>	An array of information about uploaded files
<code>\$_GET</code>	An array of values from a form submitted with the “get” method
<code>\$_POST</code>	An array of values from a form submitted with the “post” method
<code>\$_REQUEST</code>	An array of all the elements in the <code>\$_COOKIE</code> , <code>\$_GET</code> , and <code>\$_POST</code> arrays
<code>\$_SERVER</code>	An array of information about the Web server that served the current script
<code>\$_SESSION</code>	An array of session variables that are available to the current script
<code>\$GLOBALS</code>	An array of references to all variables that are defined with global scope

How to extract them?

```
<form action="somepage.php" method="GET">  
<input type="text" name="fname"/>  
<input type="submit" value="next"/>  
</form>
```

- If the method of the form is GET, then the values input by the user are saved into the **\$_GET** array
- If the method was POST, the values are saved into the **\$_POST** array
- In our case, we can say `$n= $_GET['fname'];`

In brief

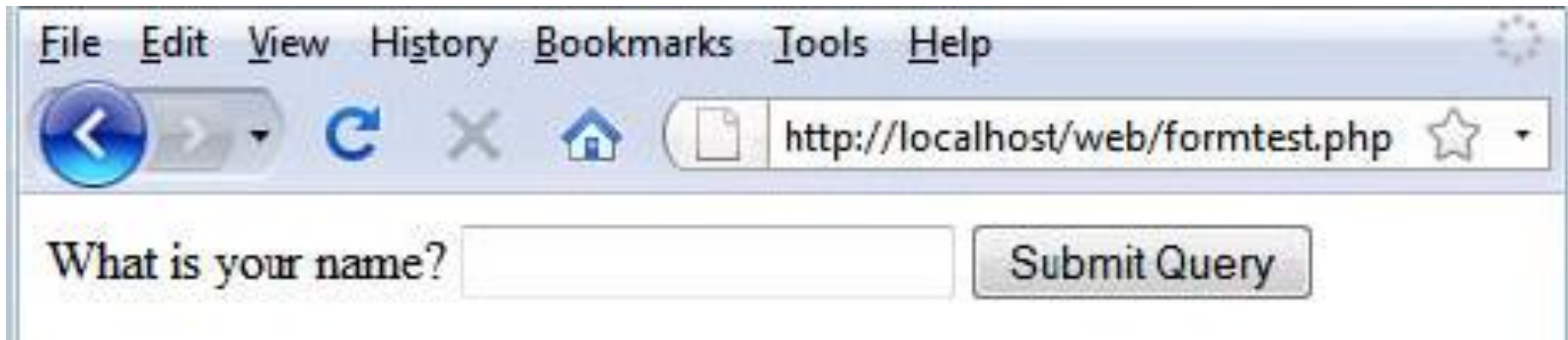
- Sometimes, we need to pass values from a page to another but it isn't any element of the form!
- Use “**hidden**” types and assign values to them:

```
<input type="hidden" name="delete" value="yes" />
```

The function isset()

- Can be used to test if the user has inputted anything that led to filling the `$_POST` or `$_GET` or `$_FILES`,.....
- EXAMPLE:
- `if(isset($_POST['delete']) && isset($_POST['fname']))`

A simple example, revisited!



```
echo <<<_END
```

```
<html> <head> <title>Form Test</title> </head>  
<body>
```

```
<form method="post" action="formtest2.php">
```

```
What is your name?
```

```
<input type="text" name="your_name" />
```

```
<input type="submit" value="Submit Query" />
```

```
</form> </body> </html>_END;
```




What if we want to show the name the user enters on the same page, and keep the form?

- Add a small PHP code to read this value from the `$_POST` array.
- Of course, we need to check if this value is set (i.e. if user enters anything) to make sure it's not first time we navigate to page or if we click on submit without entering anything.
- Solution next.

Solution

```
<?php // formtest2.php

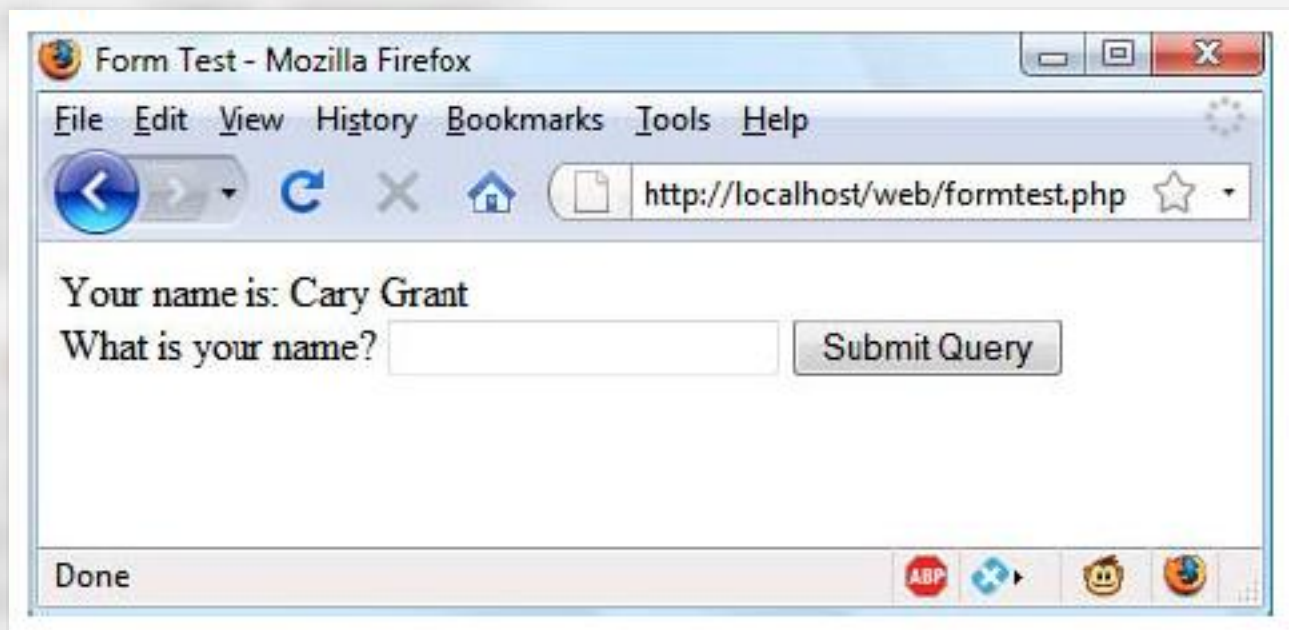
if (isset($_POST['your_name']))
    $n = $_POST['your_name'];
else
    $n = "(Not entered)";

echo <<<_END
<html> <head> <title>Form Test</title> </head>
<body>

Your name is: $n <br/>
<form method="post" action="formtest2.php">
What is your name?
<input type="text" name="your_name" />
<input type="submit" value="Submit Query" />

</form> </body> </html> _END; ?>
```

Output if the user enters “Cary Grant” then clicks on submit





1

Input Types

Text Boxes

```
<input type="text" name="name"  
size="size" maxlength="length"  
value="value" />
```

Text Area

```
<textarea name="name" cols="width"  
rows="height" wrap="type">  
This is some default text.  
</textarea>
```

TextArea (cont)

Table 11-1. The wrap types available in a textarea input

Type	Action
off	Text does not wrap and lines appear exactly as the user types them.
soft	Text wraps but is sent to the server as one long string without carriage returns and line feeds.
hard	Text wraps and is sent to the server in wrapped format with soft returns and line feeds.

Checkboxes

- This is what we are used to:

Vanilla

```
<input type="checkbox" name="ice" value="Vanilla"/>
```

Chocolate

```
<input type="checkbox" name="ice" value="Chocolate"/>
```

Strawberry

```
<input type="checkbox" name="ice" value="Strawberry"/>
```

What if the user click on several checkboxes?

- He/she is allowed of course to check several checkboxes.
- If you want to limit him/her to 1 choice, use radio buttons (old stuff!)
- What if we access `$_POST['ice']` now? (or `$_GET['ice']` if the method was GET?)

Vanilla ☐ Chocolate ☒ Strawberry ☒

Slight modification to the html code: add []

Vanilla

```
<input type="checkbox" name="ice[]" value="Vanilla" />
```

Chocolate

```
<input type="checkbox" name="ice[]" value="Chocolate" />
```

Strawberry

```
<input type="checkbox" name="ice[]" value="Strawberry" />
```



Now, we can do this:

```
$ice = $_POST['ice']
```

- What do you think the variable `$ice` hold now?
- (note that `$ice` could also be `$x`, `$y`,... we dont have to name it the same as the name of the checkboxes!!!)
- `$ice` now is an array! We can access `$ice[0]` and `$ice[1]` for example if the user has clicked on 2 boxes.

The possible values of the variable \$ice

One value submitted	Two values submitted	Three values submitted
<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>
<code>\$ice[0] => Chocolate</code>	<code>\$ice[1] => Chocolate</code>	<code>\$ice[1] => Chocolate</code>
<code>\$ice[0] => Strawberry</code>		<code>\$ice[2] => Strawberry</code>
	<code>\$ice[0] => Vanilla</code>	
	<code>\$ice[1] => Strawberry</code>	
	<code>\$ice[0] => Chocolate</code>	
	<code>\$ice[1] => Strawberry</code>	

Submit button

- 2 options: either an input with type=submit (covered before)
- Or an image:

```
<input type="image"  
name="submit" src="image.gif"  
>
```



2

Sanitizing inputs

Caution

- The first thing to remember is that regardless of what constraints you have placed in an HTML form to limit the types and sizes of inputs:
 - ***it is a trivial matter for a hacker to use their browser's View Source feature to extract the form and modify it to provide malicious input to your website.***



That's why

- We need to **sanitize** the inputs from the users:
- Inputs might contain escape characters, html tags, javascript commands, slashes that might interfere with the website functionality or might reveal some database tables!

Some useful functions

```
<?php
$str = "Is your name O'reilly?";

// Outputs: Is your name O'reilly?
echo stripslashes($str);
?>
```

```
<?php
$str = "A 'quote' is <b>bold</b>";

// Outputs: A 'quote' is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);
```

```
<?php
$text = '<p>Test paragraph.</p><!-- Comment --> <a href="#fragment">Other text</a>';
echo strip_tags($text);
// outputs: Test paragraph. Other text
```


solution

```
function sanitizeString($var)
{
    $var = stripslashes($var);
    $var = htmlentities($var);
    $var = strip_tags($var);
    return $var;
}
```

```
function sanitizeMySQL($var)
{
    $var = mysqli_real_escape_string($var);
    $var = sanitizeString($var);
    return $var;
}
?>
```

Now we can do this:


```
$variable = sanitizeString($_POST['user_input']);
```



3

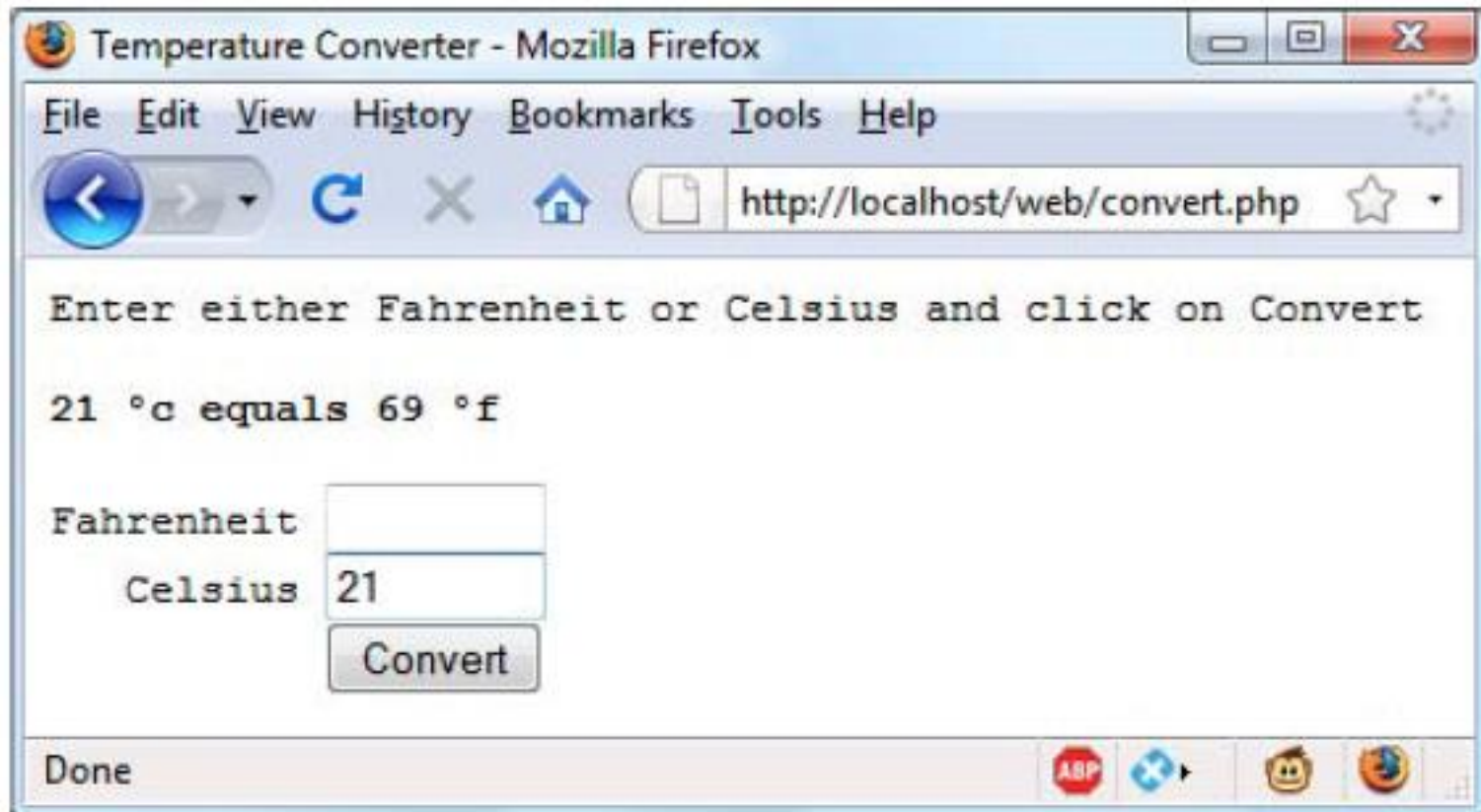
Example

```
<?php // convert.php
$f = $c = "";
if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);
if ($f != '')
{
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f °F equals $c °C";
}
elseif($c != '')
{
    $f = intval((9 / 5) * $c + 32);
    $out = "$c °C equals $f °F";
}
else $out = "";
```



```
echo <<<_END
<html><head><title>Temperature Converter</title>
</head><body><pre>
Enter either Fahrenheit or Celsius and click on Convert
<b>$out</b>
<form method="post" action="convert.php">
Fahrenheit <input type="text" name="f" size="7" />
Celsius <input type="text" name="c" size="7" />
<input type="submit" value="Convert" />
</form></pre></body></html>
_END;
```

```
function sanitizeString($var)
{
$var = stripslashes($var);
$var = htmlentities($var);
$var = strip_tags($var);
return $var;
}
?>
```



Some tips and tricks (1)

- When you are embedding php within html, it's a common **MISTAKE** to write this

```
<p> the value is <?php $v ?> </p>
```

- This will not print the variable \$v as you did not ask php to print it. it should be

```
<p> the value is <?php echo $v; ?> </p>
```

- A **quick** way to do this also:

```
<p> the value is <?=$v?> </p>
```

Some tips and tricks (2)

- When your php script receives the form, we usually read the form values using `$_GET[]` and `$_POST[]`
- There's a quick way to get those values using their "name" fields in the form
- just do this:
`EXTRACT($_POST); // or`
`EXTRACT($_GET)`
- then you have all the values in variables having the same name as the "name" fields in html.

Example on extract

- Assume the form looks like this

```
<form action='second.php' method='GET'>  
  value 1 <input type='text' name='v1'/>  
  value 2 <input type='text' name='v2'/>  
  <input type='submit' value='go'/>  
</form>
```

value 1 value 2

Example on extract (cont.)

value 1 value 2

- in the page second.php

```
<?php
```

```
extract($_GET);  
echo "Value 1 is $v1 and value 2 is $v2";
```

```
?>
```



directly use the names as variables

GET Trick

- sometimes, we want to send a value or some values to another page without having a form. We can do this using a link and **manually doing** a GET request.

```
<a href='somepage.php?x=2&y=3'> go </a>
```

Now, in somepage.php, we can extract those values using `$_GET['x']` and `$_GET['y']`

Lab session/ example

- Write an index page containing a form that has 3 input fields and a submit button.
- When we click on submit, we are navigated to another page containing a select list whose options are the values entered by the user in page 1.

CENG420
Server side
Lecture 3

PHP

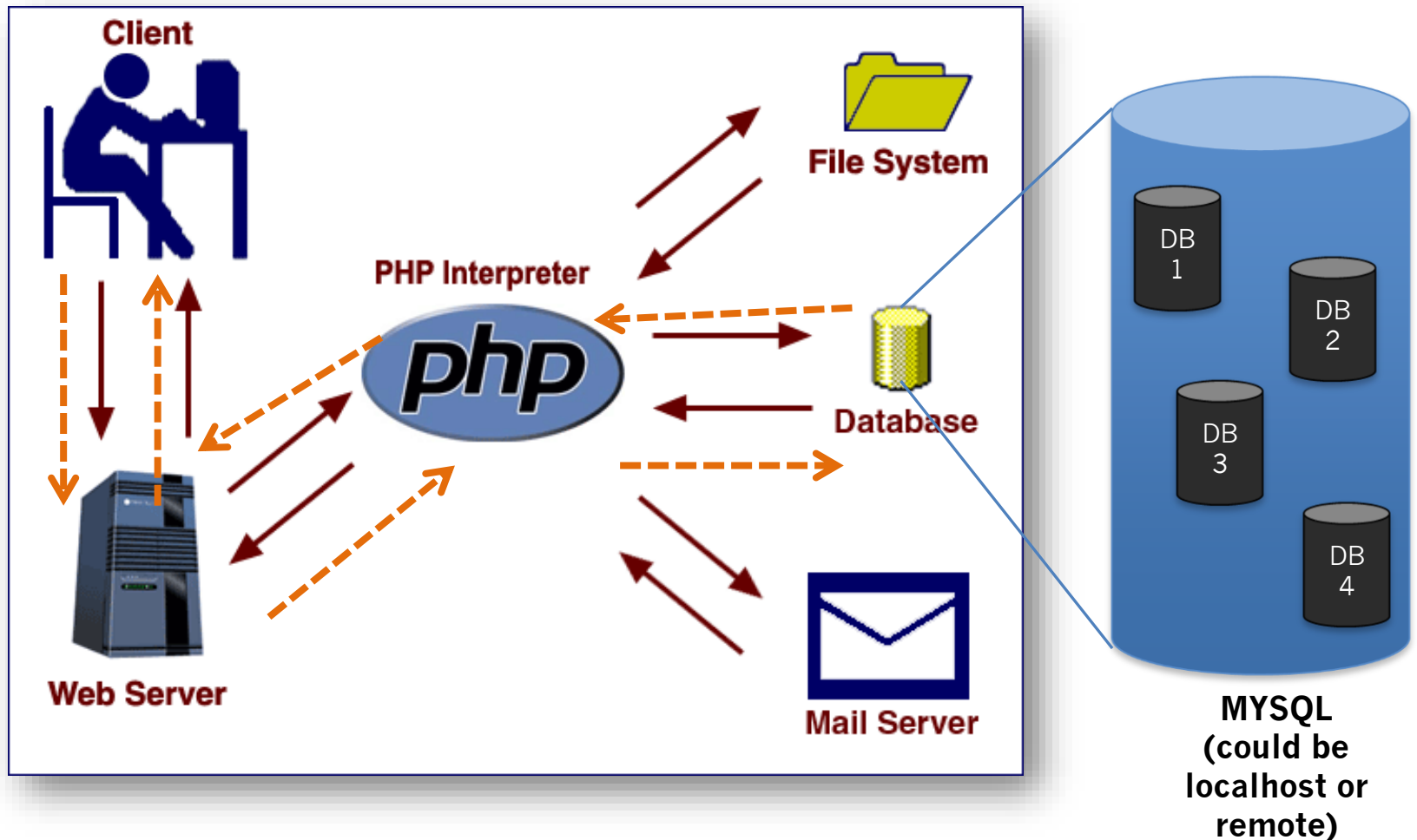
MYSQL integration



Why?

- Most websites use a database in the backend
- Databases usually hold data about:
 - Usernames and their login info
 - Website contents that might change by just changing the database contents
 - Any additional info the website might need

How they are really interconnected?



The process

- 1. Connect to MySQL.
- 2. Select the database to use.
- 3. Build a query string.
- 4. Perform the query.
- 5. Retrieve the results and output them to a web page.
- 6. Repeat Steps 3 through 5 until all desired data has been retrieved.
- 7. Disconnect from MySQL.

Connecting to Mysql

- All we need is:
- The **hostname** (where the mysql DB reside)
- The **username** and the **password** set by the administrator
- The database name

Returns an object representing the connection to the MySQL server

```
<?php
$con =
mysqli_connect("localhost","my_user","my_password","my_db");

// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```


Example

```
<?php
```

```
$db_hostname = 'localhost';
```

```
$db_database = 'school';
```

```
$db_username = 'root';
```

```
$db_password = 'root'; // your username and passwords might be  
different on your xampp (usually username is “root”, password is  
“”)
```

```
$con =
```

```
mysqli_connect($db_hostname,$db_username,$db_password,$db_d  
atabase);
```

```
// Check connection
```

```
if (mysqli_connect_errno())
```

```
{
```

```
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
```

```
}
```

```
?>
```

What if your website contains several pages, each is trying to access the DB?

- Do we have to repeat this code over and over?
- We don't have to if we write it ONCE, save it in a file, let's say, called connection.php,
- And then just include it in any page that needs it
- We will use the keyword **require_once**
- Or you can use **include**

(code written before saved into
connection.php)

```
<?php

$db_hostname = 'localhost';
$db_database = 'school';
$db_username = 'root';
$db_password = 'root';
$con = mysqli_connect($db_hostname,$db_username,$db_password,$db_database);
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```

Then from now on, in *each* page:

```
<?php
require_once 'connection.php' // or include
'connection.php'
// all other php code
?>
```

Building and executing a query

- To execute queries, php uses **`mysqli_query(connection, StringQuery)`**
- **Example:**

```
<?php
require_once 'connection.php'

$query = "SELECT * FROM classics";
$result = mysqli_query($con,$query);

if (!$result) die ("Database access failed: " .
mysqli_error());
?>
```

The result of the query is returned into the resource \$result or FALSE if failure

\$con is from the connection.php

Fetching Results

- From now on, you can use the resource returned by `mysqli_query()` to do many things.
- One of the things you can do is:

```
$r = mysqli_num_rows($result)
```

- This will return the number of rows returned by the query

Fetching Results

- Another thing we can do is to fetch results:
- Using: `$row = mysqli_fetch_row($result)`
- This will return an array containing the current row, and then places the pointer on the next row.
- We can now access any attribute by: `$row[0]`, `$row[1]`,... and so on depending on the number of attributes

Example

```
<?php
require_once 'connection.php'
$query = "SELECT * FROM classics";
$result = mysqli_query($con, $query);
$r = mysqli_num_rows($result)
```

```
for ($j = 0 ; $j < $r ; ++$j)
{
    $row = mysqli_fetch_row($result); ←
```

```
    echo 'Author: ' . $row[0] . '<br />'; ←
    echo 'Title: ' . $row[1] . '<br />';
    echo 'Category: ' . $row[2] . '<br />';
    echo 'Year: ' . $row[3] . '<br />';
    echo 'ISBN: ' . $row[4] . '<br /><br />';
}
?>
```

Do we need to memorize the index of each attribute?

- No!
- We can use
`$row= mysqli_fetch_assoc($result)`
- This will return an associative array so we can refer to attributes by their names instead of numbers. `$row['title']` for example

Closing a connection

- **mysqli_close(\$con)**
- Note we use the \$con to close the connection. → this was the handle returned by mysqli_connect

Additional features

- You can simply create, add, delete, update tables in your database from PHP
- Use same syntaxes as before, just change the query!!

The AUTO_INCREMENT field

- This is used by **mysql** when creating a table, usually with the ID attribute so the user doesn't need to worry about assigning IDs manually
- With the insertion of a new row, this field is incremented automatically
- PHP can access and read this value using: **mysqli_insert_id()**

Auto_increment (cont.)

- When inserting a new row, leave the auto_increment field NULL

```
$query = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";
```

```
$result = mysqli_query($con, $query);
```

```
echo "The Insert ID was: " . mysqli_insert_id();
```

Mysql_real_escape_string

- Use this when reading values inputted from users, before saving into database or using them into queries to avoid sql injection.

```
<?php
$user = mysql_fix_string($_POST['user']);
$pass = mysql_fix_string($_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND
pass='$pass'";

function mysql_fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysql_real_escape_string($string);
}
?>
```



Example

Practice example

- Create a single page that does several things with the database “SCHOOL”
- School has 3 tables:
 - PROFESSOR (ssn, pname, status, salary)
 - COURSE (code, cname, credits)
 - TAUGHT (code, semester, ssn)

The table contents

	code	cname	credits
	405	micro	3
	410	security	3
	415	comm	3
	440	database	5
	647	Android	3

code	semester	ssn
405	fa12	4
410	fa12	5
410	sp12	2
415	fa12	3
440	fa12	1
440	sp10	2
440	sp11	1


SSN	PNAME	STATUS	SALARY
1	Sullivan	full	3000
2	Mark	associate	2000
3	Greg	associate	2000
4	Paul	assistant	1000
5	Nancy	full	4000

Create 2 pages till now

- Connection.php that establishes the connection to the db and can be included at the top of any other page:

connection.php

```
<?php
$db_hostname = 'localhost';
$db_database = 'school';
$db_username = 'root';
$db_password = 'root';
$con =
mysqli_connect($db_hostname,$db_username,$db_password,$db_database);
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```



The second page is used to display the form required to add the professor:
`db_example.php`


- We have 2 options:
- Option 1: set the action of the form to a new page, let's say `add.php`
- Option 2: set the action to the same page which is `db_example.php`
- Both will have similar programming!!
For now, we choose option 2. look next.

```
<?php
```

```
require_once 'connection.php';
if ( isset($_POST['ssn']) && isset($_POST['name']) && isset($_POST['status']) &&
isset($_POST['salary']) )
{
    $ssn= $_POST['ssn'];
    $name= $_POST['name'];
    $status= $_POST['status'];
    $salary= $_POST['salary'];
    // add values to the db to the table professor
    $sql_add_query = "INSERT INTO PROFESSOR VALUES('$ssn','$name','$status','$salary')";
    if(mysqli_query($con, $sql_add_query) === FALSE) die("Could not add the new professor");}
?>
```

```
<html>
<body>
    <form class="s" method="POST" action="DB_example.php">
        <p>Insert a new professor</p>
        <table>
            <tr>
                <td>SSN</td><td><input type="text" name="ssn"/></td></tr> <tr>
                <td>Name</td><td><input type="text" name="name"/></td></tr>
<tr><td>Status</td><td><input type="text" name="status"/></td> </tr>
            <tr><td>Salary</td><td><input type="text" name="salary"/></td> </tr>
<tr><td></td><td><input type="submit" value="Add professor"/></td></tr>

</table></form></body></html>
```

- 
- change the index.php to display all the professors in a table in html. **Process:**
 - run a query to get all professors (select *)
 - Read the query results and organize them into a <table>

solution (some details omitted)

```
<?php require_once 'connection.php';  
$query = "SELECT * FROM professors";  
$result = mysqli_query($con, $query);  
$r=  mysqli_num_rows($result); ?>
```

```
<table border="1">  
<tr><td>ssn</td><td>name</td>  
<td>status</td><td>salary</td><td>delete</td>  
</tr>  
<?php  
for($j=0; $j<$r; $j++){  
$fetched_row = mysqli_fetch_assoc($result);  
$ssn = $fetched_row['ssn'];  
$name = $fetched_row['name'];  
$status = $fetched_row['status'];  
$salary = $fetched_row['salary'];  
echo "<tr><td>$ssn</td><td>$name</td><td>$status</td><td>$salary</td>"  
  . "<td><a href='delete.php?proffssn=$ssn'>delete</a></td>"  
  . "</tr>";  
  }??  
</table>
```

what do you think is this?



- Assignment: design the page
delete.php

It deletes the professor and then redirects the user back to the index page

To redirect a user, use:

```
header("location:index.php");
```