Internet Technologies

Introduction to PHP

University of Cyprus
Department of Computer Science

What we've learned so far

Describes the content and structure of the page

+

Describes the appearance and style of the page

+

Client-side scripting language that allows user to interact with the page (input data, show, hide elements, send data via AJAX call to backend etc.)

Web page content is still static, no dynamic, customized response for each request …

… so we need a server-side scripting language

Introduction to PHP

- PHP originally stood for “**P**ersonal **H**ome **P**age”
- Now, PHP stands for the recursive acronym for “**P**HP: **H**ypertext **P**reprocessor”
- Widely-used open-source general-purpose scripting language
- Easy to use (C-like and Perl-like syntax)
- Especially suited for web development
- Can be embedded into HTML

Introduction to PHP

- PHP is a **server-side** scripting language
- That means: **PHP scripts are executed on the server**
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all web servers used today (Apache, Nginx, IIS, etc.)
- PHP supports connection to many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is free to download and use – from the official PHP resource https://www.php.net/

What is a PHP file?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code or only PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- The client will receive the results of running that PHP code, but does not know what the underlying code was
- **PHP file extension is .php**

What can PHP do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete and close files on the server
- PHP can collect form data (sent via JavaScript from browser to server)
- PHP can send/receive cookies
- PHP can create, update, delete data in your database
- PHP can be used to control user access
- PHP can encrypt data

Basic PHP syntax

- A PHP script can be embedded/placed anywhere in the HTML document
- The PHP code is enclosed in special start and end processing instructions **<?php** and **?>** that allow you to jump into and out of "PHP mode."

```
<!DOCTYPE html>
<html>
  <head>
   <title>EPL 425</title>
   <link rel="stylesheet" href="style.css" />
   <script src="script.js"></script>
  </head>
  <body>
     <?php echo "Hi, I'm a PHP script!"; ?>
  </body>
</html>
```

simplePhpFile.php

PHP file request

WEB SERVER with PHP engine (PHP interpreter)

PHP file is parsed and interpreted by PHP interpreter

```
<!DOCTYPE html>
<html>
  <head>
   <title>EPL 425</title>
  </head>
  <body>
     <?php
       echo "Hello";
       echo "<br/>";
       echo "<em>I am here</em>";
     ?>
  </body>
</html>
```

PHP FILE ON SERVER: index.php

```
<!DOCTYPE html>
<html>
  <head>
   <title>EPL 425</title>
  </head>
  <body>
       Hello<br/><em>I am here</em>
  </body>
</html>
```

1 **User requests index.php file**

PHP file request

- This program is extremely simple and you really did not need to use PHP to create a page like this.
- All it does is display: `Hello I am here` using the PHP `echo` statement.
- Think of PHP file as a normal HTML file which happens to have a set of special tags available to the programmer that do a lot of interesting things.

PHP Comments

- In PHP, we use // or # to make a single-line comment or /* and */ to make a large comment block.

```
<?php
	// This is a single line comment

	# This is also a single line comment

	/* This is a multiple
	lines comment block */
?>
```

PHP Variables

- Variables are used for storing values, like text strings, numbers or arrays.
- When a variable is declared, it can be used over and over again in your script.
- **All variables in PHP start with a $ sign symbol**, followed by the name of the variable**.**
- The correct way of declaring a variable in PHP:

```
<?php
$var_name = value;
?>
```

PHP Variables

```php
<?php
$txt = "Hello World!";
$x = 16;
?>
```

- In PHP, a variable does not need to be declared before adding a value to it.
- In the example above, you see that you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables

- A variable name must start with a letter or an underscore "_" -- not a number
- A variable name can only contain alpha-numeric characters, underscores (`a-z`, `A-Z`, `0-9`, and `_`)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore (`$my_string`) or with capitalization (`$myString`)
- Variable names are case-sensitive (`$age` and `$AGE` are two different variables)

PHP Output Variables

- The PHP echo statement is often used to output data to the screen.
  - Both scripts will produce the same output

```php
<?php
	$txt = "EPL425";
	echo "I love $txt!";
?>
```

```php
<?php
	$txt = "EPL425";
	echo "I love" . $txt . "!";
?>
```

The concatenation operator (.) is used to concatenate two string values

PHP Operators

Arithmetic Operators	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Assignment Operators	Is the same as
=	x=y
+=	x=x+y
-=	x=x-y
*=	x=x*y
/=	x=x/y
.=	x=x.y
%=	x=x%y

String concatenation

PHP Operators

Comparison Operators	Description
==	is equal to
!=	is not equal to
<>	is not equal to
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to

Logical Operators	Description
&&	AND
||	OR
!	NOT

PHP Case Sensitivity

- In PHP, all keywords (e.g. `if`, `else`, `while`, `echo`, etc.) classes, functions and user-defined functions are NOT case-sensitive

```php
<?php
      ECHO "Hello World<br/>";
      echo "Hello World<br/>";
      EcHo "Hello World<br/>";
?>
```

Hello World
Hello World
Hello World

PHP Case Sensitivity

- BUT all variable names are case-sensitive

```php
<?php
   $color = "red";
   echo "My car is " . $color . "<br/>";
   echo "My house is " . $COLOR . "<br/>";
   echo "My boat is " . $coLOR . "<br/>";
?>
```

My car is red

NOTICE Undefined variable: COLOR on line number 4
My house is

NOTICE Undefined variable: coLOR on line number 5
My boat is

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

Global and Local Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can ONLY be accessed outside a function:

```php
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

NOTICE Undefined variable: x on line number 6

Variable x inside function is:

Variable x outside function is: 5

PHP The global Keyword

- The `global` keyword is used to access a global variable from within a function.
- To do this, use the `global` keyword before the variables (inside the function):

```php
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

PHP GLOBALS array

- PHP also stores all global variables in an array called `$GLOBALS[index]`.
- The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
- The example above can be rewritten like this:

```php
<?php
$x = 5;
$y = 10;
function myTest() {
    $GLOBALS['y'] =
$GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- To do this, use the `static` keyword when you first declare the variable
- **Note:** The variable is still local to the function.

```php
<?php
function myTest() {
      static $x = 0;
      echo $x;
      $x++;
}
myTest();
myTest();
myTest();
?>
```

```
0
1
2
```

PHP echo and print Statements

- `echo` and `print` are more or less the same. They are both used to output data to the screen.
- The `echo` statement can be used with or without parentheses: `echo` or `echo()`.
- The `print` statement can be used with or without parentheses: `print` or `print()`.
- `echo` has no return value while `print` has a return value of 1.
- `echo` can take multiple parameters (although such usage is rare) while `print` can take one argument.
- `echo` is marginally faster than `print`.

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP String

- A string can be any text inside quotes.
- You can use single or double quotes.

```php
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

Hello world!
Hello world!

PHP Integer and Float

- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- A float (floating point number) is a number with a decimal point or a number in exponential form.

```php
<?php
$x = 5985;
var_dump($x);
$y = 10.365;
var_dump($y);
?>
```

The PHP var_dump() function returns the data type and value

```
int(5985) float(10.365)
```

PHP Arrays

- In PHP, there are three kind of arrays:
  - Numeric array - An array with a numeric index
  - Associative array - An array where each ID key is associated with a value
  - Multidimensional array - An array containing one or more arrays

PHP Numeric arrays

- There are two methods to create a numeric array:

```php
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
echo "<br/>Array size:" . count($cars);
echo "<br/>" . $cars[0];
?>
```

index is automatically assigned
(the index starts at 0)

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
Array size:3
Volvo
```

```php
<?php
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

index is manually assigned

PHP Associative Arrays

- With an associative array, each ID key is associated with a value.

```php
<?php
$ages = array("Peter"=>32, "John"=>25, "Natalie"=>29);
print_r($ages);
?>
```

```
Array ( [Peter] => 32 [John] => 25 [Natalie] => 29 )
```

The PHP `print_r()` function prints human-readable information about a variable or array

- Alternative associative array creation:

```php
<?php
$ages["Peter"] = 32;
$ages["John"] = 25;
$ages["Natalie"] = 29;
?>
```

PHP Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on.

```php
<?php
$families = array("Griffin" => array("Peter", "Lois", "Meghan"),
                  "Smith" =>  array("John"),
                  "Brown" => array("Arnold", "Molly"));
echo "<br/>" . $families["Brown"][1];  // outputs Molly
?>
```

PHP Array Functions

`array()` – create new array

`is_array(array)` – checks whether the variable is an array. Returns TRUE if the variable is an array, and FALSE otherwise

`in_array(needle, array, strict)` – searches for needle an array

`array_merge(array1, array2, array3, ...)` – merges two or more arrays

`array_keys(array, value, strict)` – fetches all the keys (indexes) with the specified value from an array

`array_values(array)` – fetches all the values from an array

`array_key_exists(key, array)` – checks if a key (index) is in array

`array_push(array, value1, value2, ...)` – inserts an element to the end of an array (you can add one value, or as many as you like)

`array_pop(array)` – deletes and returns the last element of an array

`array_map(myfunction, array1, array2, ...)` – apply a function to every single array element, and return an array with the new results

array_unique()

array_slice()

array_diff()

array_search()

array_reverse()

array_unshift()

PHP Objects

```php
<?php
class my_class
{
    function print_msg()
    {
        echo "Hello world.";
    }
}

$obj = new my_class; // use new statement to create an object
$obj->print_msg();
?>
```

PHP Anonymous Objects

- `StdClass` is PHP's generic empty class
- Useful for anonymous objects
- `StdClass` be considered as an alternative to associative array (without quoting all keys)

```php
<?php
$object = new StdClass;
$object->name = 'Peter';
$object->age = 32;
print_r($object);
?>
```

```
stdClass Object ( [name] => Peter [age] => 32 )
```

PHP NULL

- The special NULL value is used to represent empty variables in PHP.
- A variable of NULL value is a variable without any data.

```php
<?php
$a = NULL;
var_dump($a);
echo "<br>";
$b = "Hello World!";
$b = NULL;
var_dump($b);
?>
```

```
NULL
NULL
```

PHP Resource

- A resource is a special variable, holding a reference to an external resource.
- Resource variables typically hold special handlers to opened files and database connections.

```php
<?php
// Open a file for reading
$handle = fopen("note.txt", "r");
var_dump($handle);
echo "<br>";
// Connect to MySQL database server with default setting
$link = mysqli_connect("localhost", "root", "letmein");
var_dump($link);
?>
```

PHP Conditional Statements

- if
- if ... else
- if ... elseif ... else
- switch

```php
<?php
$d = date("D"); // returns
current day
if($d == "Fri") {
  echo "Have a nice weekend!";
}
elseif ($d == "Sun") {
  echo "Have a nice Sunday!";
}
else {
  echo "Have a nice day!";
}
?>
```

PHP Conditional statements

```php
<?php
switch(date("D")) {
   case "Fri":
      echo "Have a nice weekend!";
      break;
   case "Sun":
      echo "Have a nice Sunday!";
      break;
   default:
      echo "Have a nice day!";
}
?>
```

PHP Loops

- `while` - loops through a block of code while a specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- `for` - loops through a block of code a specified number of times
- `foreach` - loops through a block of code **for each element in an array**

PHP Loops (examples)

```php
<?php
$i=1;
while($i<=5) {
  echo "The number is $i<br/>";
  $i++;
}
?>
```

```php
<?php
$i=0;
do {
  $i++;
  echo "The number is $i<br/>";
} while($i<5);
?>
```

The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

PHP Loops (examples)

```php
<?php
for($i=1; $i<=5; $i++) {
  echo "The number is $i<br/>";
}
?>
```

The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

```php
<?php
$x = array("Bob", "Mary", "John");
foreach($x as $name) {
  echo $name . "<br/>";
}
?>
```

Bob
Mary
John

PHP Functions

```php
<?php
function function_name([parameters list]opt) {
   ...
}
?>
```

- Not allowed to overload the name of an existing function;
- Function names aren’t case-sensitive;
- To each parameter can be assigned a default value;
- Arguments can be passed by value or by reference (& in front of param)
- It’s possible using a variable number of parameters

PHP Functions examples

```php
<?php
function sum($x, $y) {
  return $x + $y;
}
echo sum(5,7);
?>
```

Call by value

Call by reference

```php
<?php
function goodbye(&$greeting) {
  $greeting = "See you later";
}
$myVar = "Hi there";
goodbye($myVar);
echo $myVar; // Displays "See you later"
?>
```

PHP Filesystem Functions

- fopen() – opens file or URL
- fclose() – closes an open file
- file_exists() – checks whether or not a file or directory exists
- feof() – checks if the "end-of-file" (EOF) has been reached for an open file
- fgets() – returns a line from an open file
- file() – reads a file into array
- file_get_contents() – reads a file or URL into a string
- fwrite() – writes to an open file
- filesize() – returns the file size

https://www.php.net/manual/en/ref.filesystem.php

PHP Filesystem Functions Example

```php
<?php
$myfile = fopen("file.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

die() is equivalent to exit()

Outputs a message and terminates the current script

PHP JSON Functions

- `json_decode($json, $assoc)` — takes a JSON encoded string and converts it into an appropriate PHP type
  - Usually returns arrays (`$json = '[4,5,6,7]'`) or StdClass objects (`$json='{"name":"John"}'`)
  - $assoc is boolean; if TRUE returned objects are converted to associative arrays.
- `json_encode($value, $flags)` — Returns a string containing the JSON representation of the supplied value
  - $flags are some constants that enable arbitrary checks e.g. JSON_NUMERIC_CHECK encodes numeric strings as numbers
- `json_last_error_msg()` — Returns the error string of the last json_encode() or json_decode() call
- `json_last_error()` — Returns the last error occurred

PHP JSON Functions examples

```php
<?php
$json_str =
'{"dictionary":{"English":["One","January"],"French":["Une","
Janvier"]}}';
$json_obj = json_decode($json_str);
print_r($json_obj->dictionary->English);
?>
```

```
Array ( [0] => One [1] => January )
```

```php
<?php
$json_str =
'{"dictionary":{"English":["One","January"],"French":["Une","
Janvier"]}}';
$json_obj = json_decode($json_str, true);
print_r($json_obj["dictionary"]["English"]);
?>
```

```
Array ( [0] => One [1] => January )
```

PHP JSON Functions examples

```php
<?php
$json_str = json_encode(
    array(
        "dictionary" => array(
            'English' => array(
                'One',
                'January'
            ),
            'French' => array(
                'Une',
                'Janvier'
            )
        )
    )
);
echo $json_str;     // {"dictionary":{"English":["One","January"],"French":["Une","Janvier"]}}
?>
```

Exercise 1

- Create a PHP file that gets the weather forecast for the UCY campus using the [OpenWeatherForecast API] and prints the following info on the screen:
- Hint: Use file_get_contents() to get data via sending an HTTP GET message to the API

dt	temp
1554811200	19.71
1554822000	16.74
1554832800	15.82
1554843600	15.12
1554854400	13.94
1554865200	12.66
1554876000	15.33
1554886800	17.08
1554897600	15.3
1554908400	15.3
1554919200	14.48
.	
.	
.	

Exercise 2

- Add html code and bootstrap library to create the following table:

Use table-striped and table-hover

Datetime	Temperarure
1554811200	19.56
1554822000	16.62
1554832800	15.75
1554843600	15.08
1554854400	13.94
1554865200	12.66
1554876000	15.33
1554886800	17.08
1554897600	15.3
1554908400	15.3

Data transfer from browser to web server

- Data transfer from browser to server is activated via:
  - HTML form submission
  - JavaScript (submit form or XMLHttpRequest)
- Data travels across the Internet on top of HTTP messages:
  - GET messages
  - POST messages
- Data received/processed to server using PHP script

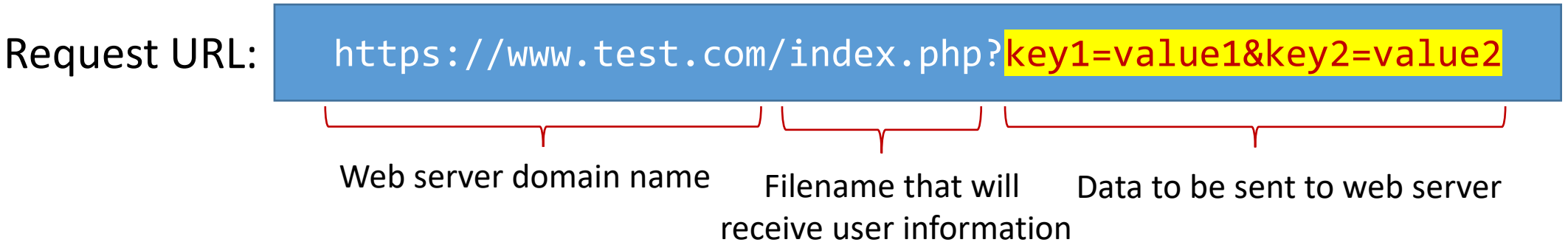
PHP Superglobal arrays

- Super global arrays are predefined (associative) arrays and can be accessed anywhere in the page, without using the keyword global

$GLOBALS	Contains all global scope variables.
$_SERVER	Contains information of headers, paths, script location. $_SERVER['SERVER_ADDR']: IP address of the host server $_SERVER['REQUEST_METHOD']: The request method used to access the page (such as POST) $_SERVER['REMOTE_ADDR']: The IP address from where the sent the request
$_GET	Contains all key-value pairs which are sent within the URL.
$_POST	Contains variables passed via the HTTP POST method when using *application/x-www-form-urlencoded* or *multipart/form-data*
$_COOKIE	Contains cookies data which the server send to client and is stored on client machine in form of a file.
$_FILES	Contains information of the file which we upload with help of HTTP POST method.
$_SESSION	All the session variables are stored in this array.
$_REQUEST	By default this array contains the contents of $_GET, $_POST, and $_COOKIE.

GET method

- Sends data appended to the request URL



- Data has to be URL encoded prior sending to server (next slide)
- In request URL, the filename and the encoded data are separated by the ? character, followed by name/value pairs
- Name/value pairs are joined with equal signs (=) and different pairs are separated by the ampersand (&)

URL encoding

- Convert URL string into valid URL format.
- Normally performed to convert data passed via html forms, because such data may contain special characters, which could either:
  1. have special meanings e.g. "#" character needs to be encoded because it has a special meaning of that of an html anchor.
  2. is not a valid character for an URL e.g. *space* character needs to be encoded because is not allowed on a valid URL format. URL encoding normally replaces a *space* with a plus (+) sign or with %20.
  3. could be altered during transfer e.g. "~" might not transport properly across the internet.
- URL encoding replaces unsafe ASCII characters with a % (percent encoding) followed by two hexadecimal digits.

GET method

- Produces a long URL request string (since information is appended)
- URL request string length is limited (more info).
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- Useful for form submissions where a user want to bookmark the result.
- GET request is typically sent via submitting HTTP form or via JavaScript
- The PHP provides **`$_GET`** associative array to access all user information send via GET method.

Send/receive data

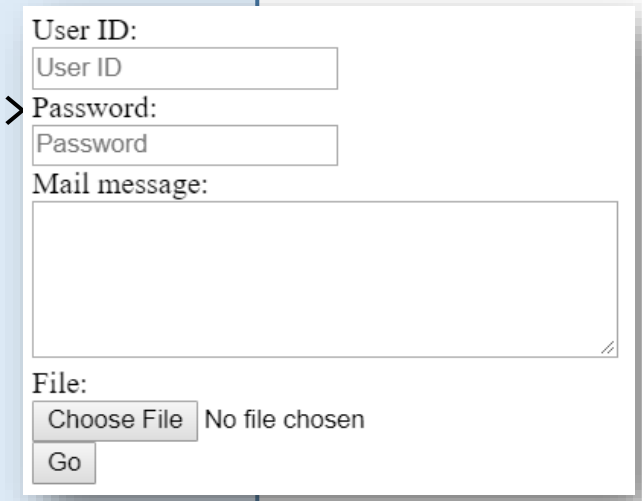
- Send data using:
  - GET via HTML form submission
  - GET via JavaScript (submit form or XMLHttpRequest)
- and receive information using PHP file

Browser
Sends GET msg via HTML form submission

form_get.html

HTML

```
<form action="./action_page.php" method="get">
   User ID:<br/>
   <input type="text" name="userid" placeholder="User ID"/><br/>
   Password:<br/>
   <input type="password" name="passwd" placeholder="Password"/>
   Mail message:<br/>
   <textarea name="msg" rows="5" cols="40"></textarea><br/>
   File:<br/>
   <input type="file" name="txtfile"/><br/>
   <button type="submit">Go</button>
</form>
```



When button is clicked, form data are appended to the URL in key/value pairs: action_page.php?userid={value}&passwd={value}&msg={value}&txtfile={value} and then a GET message is sent. After form submission (via GET), **the page is redirected to action_page.php**.

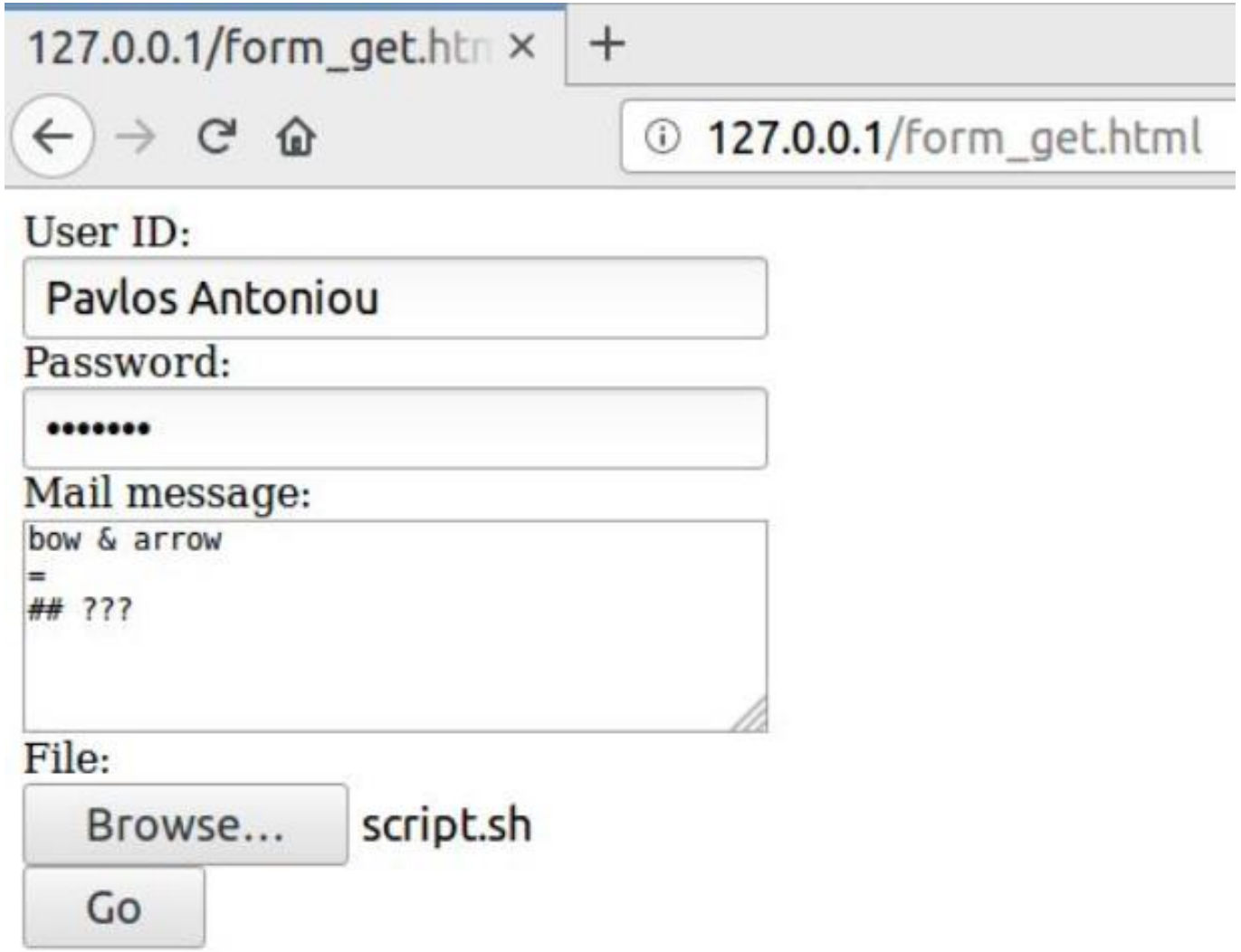
Web server
Receives GET msg using PHP

action_page.php

PHP

```php
// Check if GET request was received.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'GET') == 0) {
        echo $_GET["userid"] . "<br/>";
        echo $_GET["passwd"] . "<br/>";
        // url decode string
        echo urldecode($_GET["msg"]) . "<br/>";
        echo $_GET["txtfile"] . "<br/>";
}
```

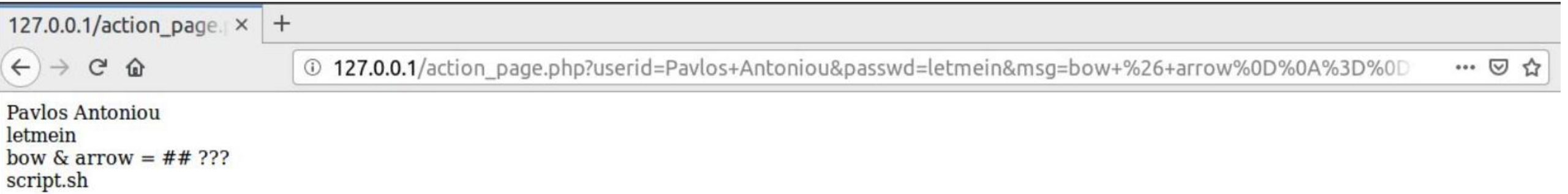
Before form submission



After form submission

- Form data are shown in browser address bar

- Browser **is redirected** to action_page.php



After form submission

- GET message as captured by Wireshark

```
GET /action_page.php?userid=Pavlos+Antoniou&passwd=letmein&msg=bow+%26+arrow%0D%0A%3D%0D%0A
%23%23+%3F%3F%3F&txtfile=script.sh HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/form_get.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

  - Password is sent in clear text within URL
  - Special characters (&, =, #, ?) are percent (%) encoded e.g. %26 is &, space → +
  - Filename is sent, but file contents not. See console warning below:

⚠ Form contains a file input, but is missing method=POST and enctype=multipart/form-data on the form_get.html form. The file will not be sent.

Browser
Sends GET msg via JavaScript

form_get_javascipt.html

HTML

```
<form>
   User ID:<br/>
   <input type="text" name="userid" id="userid" placeholder="User ID"/><br/>
   Password:<br/>
   <input type="password" name="passwd" id="passwd" placeholder="Password"/><br>
   Mail message:<br/>
   <textarea name="msg" id="msg" rows="5" cols="40"></textarea><br/>
   File:<br/>
   <input type="file" name="txtfile" id="txtfile"/><br/>
   <button type="button">Go</button>
</form>
```

When button clicked, JavaScript function can be called to send GET message. 2 options available:

1. set form action (e.g. action_page.php) and then submit form
2. collect form data (using id of each field), and then create XMLHttpRequest object to send GET msg to action_page.php.

Send GET msg via JavaScript – Submit form

form_get_javascipt.js

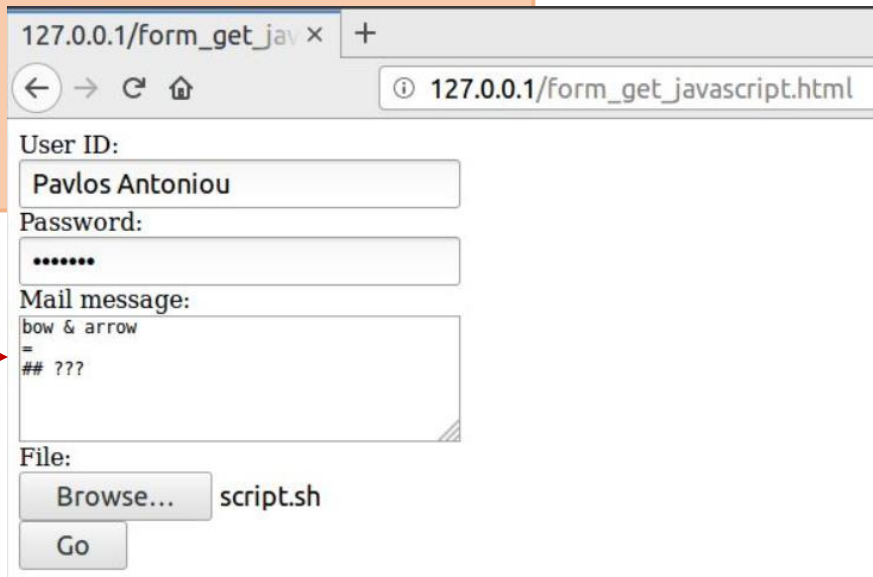
JS

```
function onSubmit() {
        // access form element
        const form = document.querySelector('form');
        // set action on form
        form.action = '/action_page.php';
        // submit form
        // this has the same effect as clicking button of type="submit"
        form.submit();
}

const button = document.querySelector('button');
button.addEventListener('click', onSubmit);
```

Option 1

Before button click →

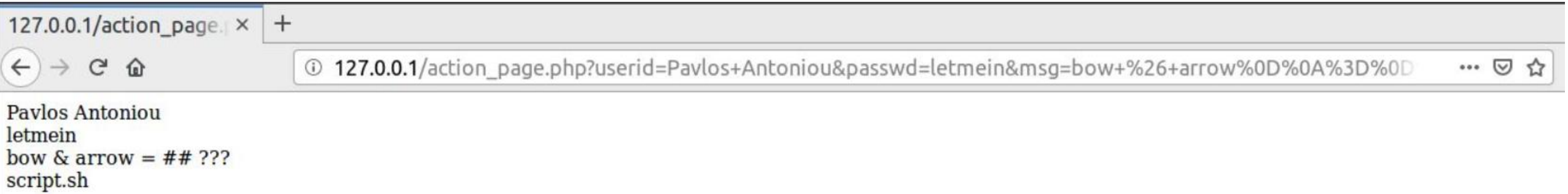


After button click

- The same action_page.php is used to collect data from GET msg
- Form data are shown in browser address bar

Option 1

- Browser **is redirected** to action_page.php



We experience the same behavior as submitting form without JavaScript

After form submission

- GET message as captured by Wireshark

```
GET /action_page.php?userid=Pavlos+Antoniou&passwd=letmein&msg=bow+%26+arrow%0D%0A%3D%0D%0A
%23%23+%3F%3F%3F&txtfile=script.sh HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/form_get.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Option 1

- Password is sent in clear text within URL
- Special characters (&, =, #, ?) are percent (%) encoded e.g. %26 is &, space → +
- Filename is sent, but file contents not. See console warning below:

⚠ Form contains a file input, but is missing method=POST and enctype=multipart/form-data on the form_get.html form. The file will not be sent.

Send GET msg via JavaScript – XMLHttpRequest

JS

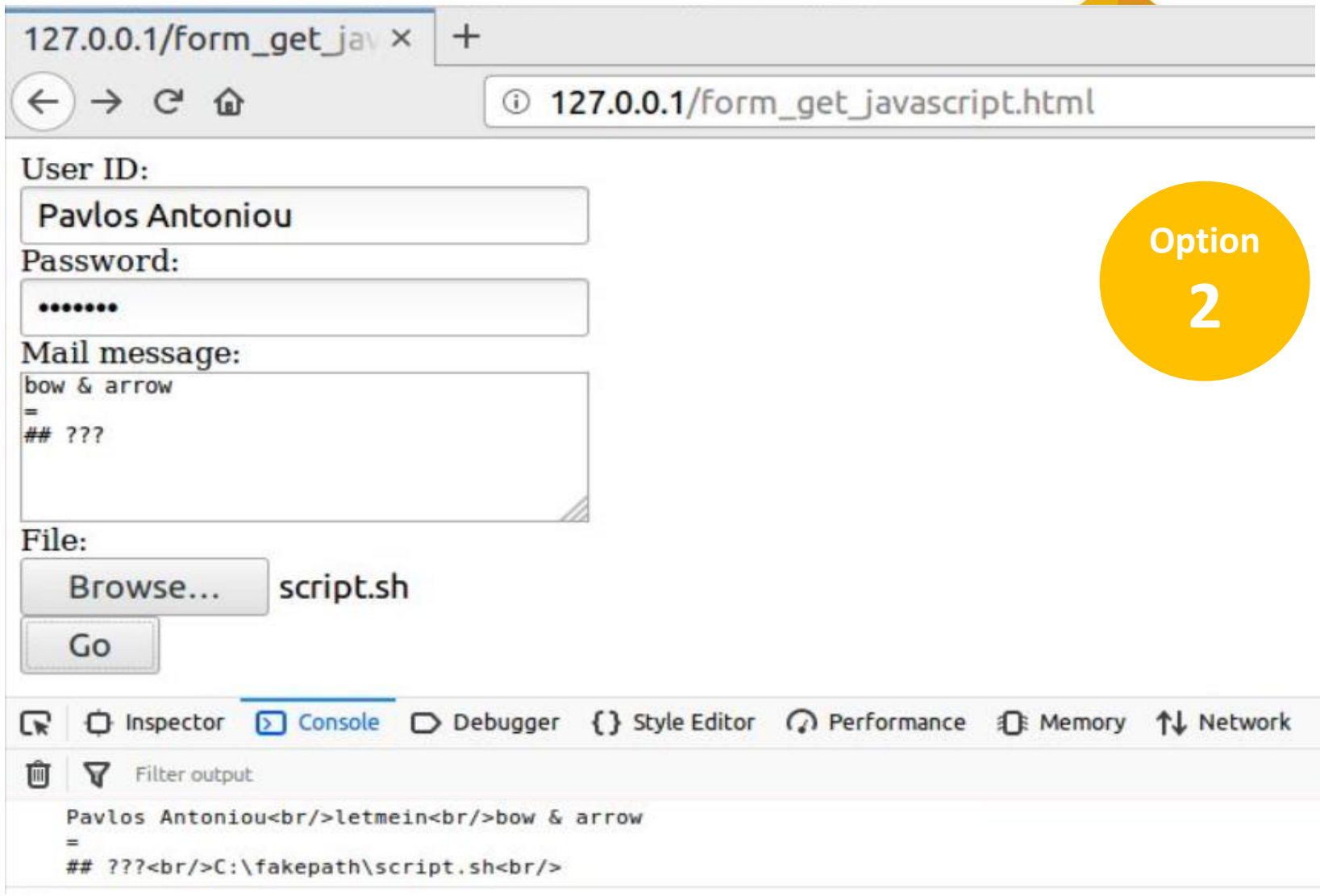
```
function onClick() {
    // Set up our HTTP request
    var xhr = new XMLHttpRequest();
    // Setup our listener to process completed requests
    xhr.onreadystatechange = function () {
        // Only run if the request is complete
        if (xhr.readyState !== 4) return;
        // Process our return data
        if (xhr.status >= 200 && xhr.status < 300) {
                console.log(xhr.responseText);
        } else {
                console.log('error', xhr);
        }
    };
    const userid = document.querySelector('#userid').value;
    const passwd = document.querySelector('#passwd').value;
    const msg = encodeURIComponent(document.querySelector('#msg').value);
    const txtfile = document.querySelector('#txtfile').value;
    xhr.open('GET', 'action_page.php?userid='+userid+'&passwd='+passwd+'&msg='+msg+'&txtfile='+txtfile);
    xhr.send();
}
const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

Option 2

Percent encoding data of the textarea since data may contain special characters. Potentially, all input values can be percent encoded.

After button click

- Form data are NOT shown in browser address bar
- Browser is NOT redirected to action_page.php
- Web page does not reload after AJAX call



Option 2

After button click

- GET message as captured by Wireshark

```
GET /action_page.php?%20userid=Pavlos%20Antoniou&passwd=letmein&msg=bow%20%26%20arrow%0A%3D
%0A%23%23%20%3F%3F%3F&txtfile=C:\fakepath\script.sh HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/form_get_javascript.html
Connection: keep-alive
```

Option 2

- Password is sent in clear text
- Special characters (space, &, =, #, ?) are percent encoded e.g. %20 is space, %26 is &
- Filename is sent, but file contents not
  - Filename is C:\fakepath\script.sh. Some browsers have a security feature that prevents JavaScript from knowing files' local full path so the server will not be aware of local machine's filesystem.

POST method

- POST method transfers data via HTTP request body
- POST method does not have any restriction on data size to be sent.
- Form submissions with POST cannot be bookmarked.
- POST method can be used to send **ASCII** as well as **binary data**.
- POST method can be used to **upload files**.
- The type of the body of the request is indicated by the `Content-Type` header.
- A POST request is typically sent via submitting HTTP form or via JavaScript

POST method via HTML form submission

- When submitting HTML form, `Content-type` is selected by putting the adequate string in the `enctype` attribute of the `<form>` element or the `formenctype` attribute of the `<input>` or `<button>` elements:
  - `application/x-www-form-urlencoded`: the keys and values are URL encoded in key-value tuples separated by '&', with a '=' between the key and the value. Non-alphanumeric characters in both keys and values are percent encoded: this is the reason why **this type is not suitable to use with binary data** (use multipart/form-data instead)
  - `multipart/form-data`: each value is sent as a block of data ("body part"), with a user agent-defined delimiter ("boundary") separating each part. The keys are given in the Content-Disposition header of each part. **Used for uploading files.**
  - `text/plain`: send data as plain text (human readable), can be avoided. See here.

POST method via Javascript

- When the POST request is sent via a method other than an HTML form — like via XMLHttpRequest/Fetch API — the body can take any type e.g. `application/json` since the developer is responsible for encoding information in the appropriate type

Send/receive data

- Send data using:
  - POST via HTML form submission
  - POST JavaScript (submit form or XMLHttpRequest)
- and receive information using PHP file

Browser
Sends POST msg via HTML form submission

form_post.html

HTML

```
<form action="./action_page.php" method="post"
      enctype="application/x-www-form-urlencoded">
   User ID:<br/>
   <input type="text" name="userid" placeholder="User ID"/><br/>
   Password:<br/>
   <input type="password" name="passwd" placeholder="Password"/><br>
   Mail message:<br/>
   <textarea name="msg" rows="5" cols="40"></textarea><br/>
   File:<br/>
   <input type="file" name="txtfile"/><br/>
   <button type="submit">Go</button>
</form>
```

When button is clicked, form data are converted to a string of key/value pairs: userid={value}&passwd={value}&msg={value}&txtfile={value} which is then placed on the body of the POST message to be sent. After form submission (via POST), **the page is redirected to action_page.php**.

Web server
Receives POST msg using PHP

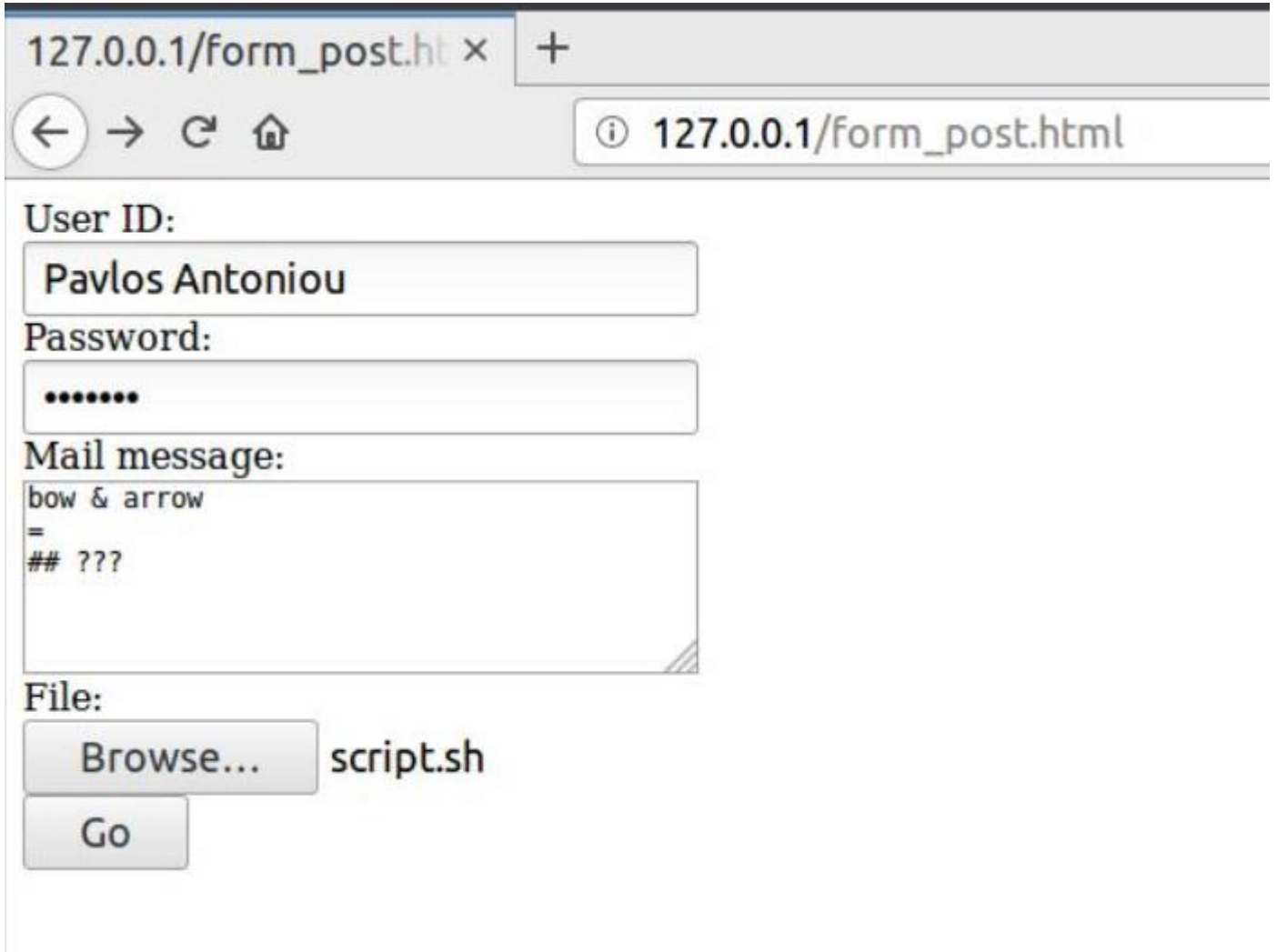
action_page.php

PHP

```php
// Check if GET request was received.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'POST') == 0) {
        echo $_POST["userid"] . "<br/>";
        echo $_POST["passwd"] . "<br/>";
        // url decode string
        echo urldecode($_POST["msg"]) . "<br/>";
        echo $_POST["txtfile"] . "<br/>";
}
```

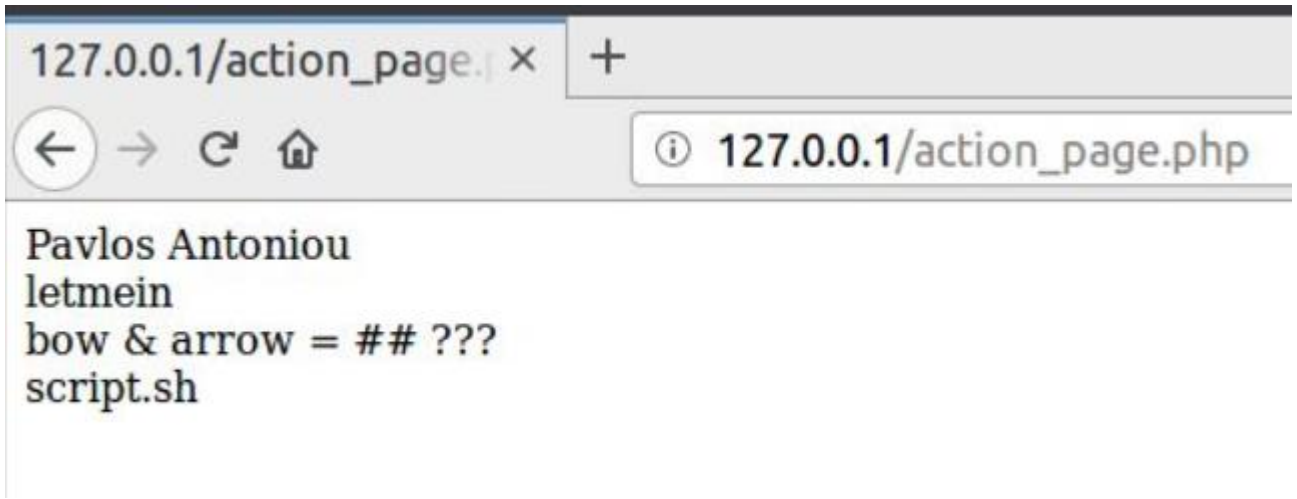
PHP provides **`$_POST`** array to access all user information send via GET method

Before form submission



After form submission

- Form data are NOT shown in browser address bar
- Browser **is redirected** to action_page.php



After form submission

- POST message as captured by Wireshark

headers

```
POST /action_page.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/form_post.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

body

```
userid=Pavlos+Antoniou&passwd=letmein&msg=bow+%26+arrow%0D%0A%3D%0D%0A%23%23+%3F%3F
%3F&txtfile=script.sh
```

Password is sent in clear text in msg body
Filename is sent, but file contents not.
Special characters (&, =, #, ?) are percent (%) encoded e.g. %26 is &, space → +

Browser
Sends POST msg via HTML form submission

form_post_multipart.html

```HTML
<form action="./action_page.php" method="post" enctype="multipart/form-data">
   User ID:<br/>
   <input type="text" name="userid" placeholder="User ID"/><br/>
   Password:<br/>
   <input type="password" name="passwd" placeholder="Password"/><br>
   Mail message:<br/>
   <textarea name="msg" rows="5" cols="40"></textarea><br/>
   File:<br/>
   <input type="file" name="txtfile"/><br/>
   <button type="submit">Go</button>
</form>
```

When button is clicked, form data are placed on the body of the POST message as parts (see next slide). After form submission (via POST), **the page is redirected to action_page.php**.

POST message in Wireshark

```
POST /action_page.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/form_post_multipart.html
Content-Type: multipart/form-data; boundary=---------------------------792420598346532053290388 68
Content-Length: 2922
Connection: keep-alive
Upgrade-Insecure-Requests: 1

-----------------------------792420598346532053290388 68
Content-Disposition: form-data; name="userid"

Pavlos Antoniou
-----------------------------792420598346532053290388 68
Content-Disposition: form-data; name="passwd"

letmein
-----------------------------792420598346532053290388 68
Content-Disposition: form-data; name="msg"

bow & arrow
=
## ???
-----------------------------792420598346532053290388 68
Content-Disposition: form-data; name="txtfile"; filename="script.sh"
Content-Type: application/x-shellscript

# unix script to raname moodle folder names (remove spaces)
find . -type d -exec bash -c 'mv "$0" "${0// /_}"' {} \; 2>/dev/null

# get all folders
pfolders=`ls  | grep -v .sh | grep -v .zip`
```

The fields in msg body are placed in separated parts which are splitted by the given boundary string

Web server
Receives POST msg using PHP

action_page.php

PHP

```php
// Check if GET request was received.
if(strcasecmp($_SERVER['REQUEST_METHOD'], 'POST') == 0) {
        echo $_POST["userid"] . "<br/>";
        echo $_POST["passwd"] . "<br/>";
        // url decode string
        echo urldecode($_POST["msg"]) . "<br/>";
        if(isset($_SERVER["CONTENT_TYPE"])) {
                $contentType = $_SERVER["CONTENT_TYPE"];
                $contentType = explode("; ", $contentType)[0];
        }
        else
                $contentType = "";
        if(strcasecmp($contentType, 'multipart/form-data') == 0)
                print_r($_FILES["txtfile"]) . "<br/>";
        else
                echo $_POST["txtfile"] . "<br/>";
}
```

$_FILES superglobal array

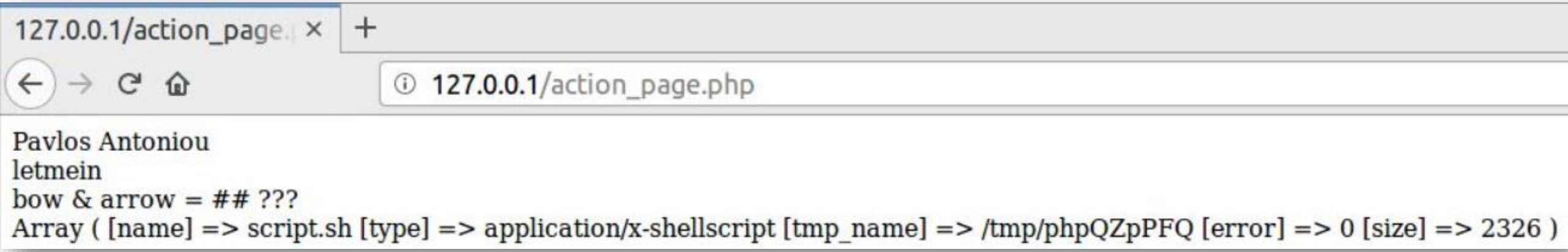
- `$_FILES` is a 2D associative global array of items which are being uploaded by via HTTP POST method and holds the attributes of files such as

ATTRIBUTE	DESCRIPTION
[name]	Name of file which is uploading
[size]	Size of the file
[type]	Type of the file (like .pdf, .zip, .jpeg.....etc)
[tmp_name]	A temporary address where the file is located before processing the upload request
[error]	Types of error occurred when the file is uploading

Files will, by default be stored in the server's default temporary directory (e.g. in /tmp), unless another location has been given with the upload_tmp_dir directive in php.ini. The server's default directory can be changed by setting the environment variable TMPDIR in the environment in which PHP runs.

After form submission

- Form data are NOT shown in browser address bar
- Browser is redirected to action_page.php
- Uploaded file information is shown:



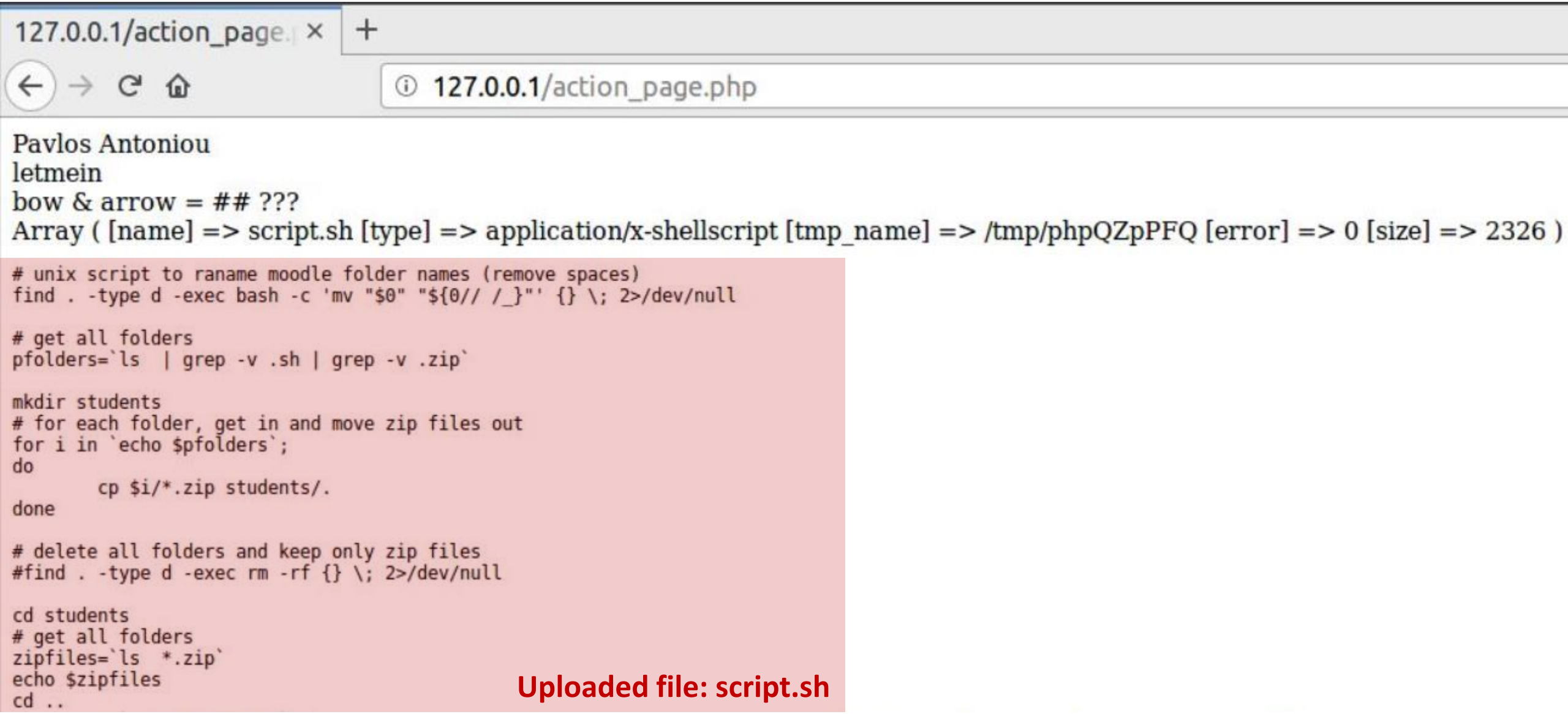
Update php to print uploaded file contents

action_page.php

PHP

```php
// Check if POST request was received.
if(strcasecmp($_SERVER[REQUEST_METHOD'], 'POST') == 0) {
        echo $_POST["userid"] . "<br/>";
        echo $_POST["passwd"] . "<br/>";
        // url decode string
        echo urldecode($_POST["msg"]) . "<br/>";
        if(isset($_SERVER["CONTENT_TYPE"])) {
                $contentType = $_SERVER["CONTENT_TYPE"];
                $contentType = explode("; ", $contentType)[0];
        }
        else
                $contentType = "";
        if(strcasecmp($contentType, 'multipart/form-data') == 0) {
                print_r($_FILES["txtfile"]) . "<br/>";
                echo "<pre>" . file_get_contents($_FILES["txtfile"]["tmp_name"]) . "</pre>";
        }
        else
                echo $_POST["txtfile"] . "<br/>";
}
```

The <pre> tag defines preformatted text. Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.


127.0.0.1/action_page.
127.0.0.1/action_page.php
Pavlos Antoniou
letmein
bow & arrow = ## ???
Array ([name] => script.sh [type] => application/x-shellscript [tmp_name] => /tmp/phpQZpPFQ [error] => 0 [size] => 2326)
unix script to raname moodle folder names (remove spaces)
find . -type d -exec bash -c 'mv "$0" "${0// /_}"' {} \; 2>/dev/null
# get all folders
pfolders=`ls | grep -v .sh | grep -v .zip`
mkdir students
for each folder, get in and move zip files out
for i in `echo $pfolders`;
do
cp $i/*.zip students/.
done
delete all folders and keep only zip files
#find . -type d -exec rm -rf {} \; 2>/dev/null
cd students
# get all folders
zipfiles=`ls *.zip`
echo $zipfiles
cd ..
Uploaded file: script.sh

Update php to save uploaded file on disk

action_page.php

PHP

```
...
        if(strcasecmp($contentType, 'multipart/form-data') == 0) {
                echo print_r($_FILES["txtfile"]) . "<br/>";
                echo "<pre>" . file_get_contents($_FILES["txtfile"]["tmp_name"]) . "</pre>";
                // save uploaded file
                $uploaddir = '/var/www/html/uploads/';
                $uploadfile = $uploaddir . basename($_FILES['txtfile']['name']);
                echo $uploadfile . "<br/>";
                if (move_uploaded_file($_FILES['txtfile']['tmp_name'], $uploadfile))
                        echo "File is valid, and was successfully uploaded.<br/>";
                else
                        echo "Possible file upload attack!<br/>";
        }
...
```

Note: The `$uploaddir` **must exist and be owned by the apache server user** in order to be able to save a file in there (see more on the next slide)

Prerequisites for file uploading

- Show apache user:

```
ps -ef | egrep '(httpd|apache2|apache)' | grep -v `whoami`
| grep -v root | head -n1 | awk '{print $1}'
```

```
> www-data
```

- Create uploads folder and give ownership to apache user (www-data):

```
sudo mkdir /var/www/html/uploads
```

```
sudo chown -R www-data:www-data /var/www/html/uploads
```

Browser
Sends POST msg via JavaScript

form_post_javascipt.html

HTML

```
<form>
   User ID:<br/>
   <input type="text" name="userid" id="userid" placeholder="User ID"/><br/>
   Password:<br/>
   <input type="password" name="passwd" id="passwd" placeholder="Password"/><br>
   Mail message:<br/>
   <textarea name="msg" id="msg" rows="5" cols="40"></textarea><br/>
   File:<br/>
   <input type="file" name="txtfile" id="txtfile"/><br/>
   <button type="button">Go</button>
</form>
```

When button clicked, JavaScript function can be called to send POST msg. 2 options available:

1. set form action (e.g. action_page.php) and enctype and then submit form
2. collect form data (using id of each field), and then create XMLHttpRequest object to send POST msg to action_page.php (object can be converted to JSON string and placed in body).

Send POST msg via JavaScript – Submit form

form_post_javascipt.js

JS

```
function onSubmit() {
        // access form element
        const form = document.querySelector('form');
        // set action on form
        form.action = "/action_page.php";
        // set enctype on form
        form.enctype = "multipart/form-data";
        // submit form
        // this has the same effect as clicking button of type="submit"
        form.submit();
}

const button = document.querySelector('button');
button.addEventListener('click', onSubmit);
```

Option 1

We experience the same behavior as submitting form without JavaScript

Send GET msg via JavaScript – XMLHttpRequest

JS

```
function onClick() {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function () {
        if (xhr.readyState !== 4) return;
        if (xhr.status >= 200 && xhr.status < 300) {
                console.log(xhr.responseText);
        } else {
                console.log('error', xhr);
        }
    };

    xhr.open('POST', 'action_page.php');
    xhr.setRequestHeader("Content-Type", "application/json");
    const data = {};
    data.userid = document.querySelector("#userid").value;
    data.passwd = document.querySelector("#passwd").value;
    data.msg = encodeURIComponent(document.querySelector("#msg").value);
    data.txtfile = document.querySelector("#txtfile").value;
    xhr.send(JSON.stringify(data));
}
const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

- If data is to be sent as JSON string, set Content-Type
- Create JavaScript object.
- Set object properties.
- Convert object to JSON string and send.

Option 2

Update php to collect data in POST msg body

- **$_POST** can be used to obtain data when Content-Type is set to *application/x-www-form-urlencoded* or *multipart/form-data*
- How to get JSON data from POST body if Content-Type is application/json?
  - `php://input` - is a read-only stream that allows us to read raw data from the request body. It returns all the raw data after the HTTP-headers of the request, regardless of the content type.
  - `file_get_contents()` function to read a file (stream) into a string.
  - `json_decode()` function to convert JSON string into a PHP variable that may be an array or an object.

PHP

```php
// Takes raw data from the request body
$json = file_get_contents('php://input');

// Converts it into a PHP object
$data = json_decode($json);
```

Update php to collect data in POST msg body

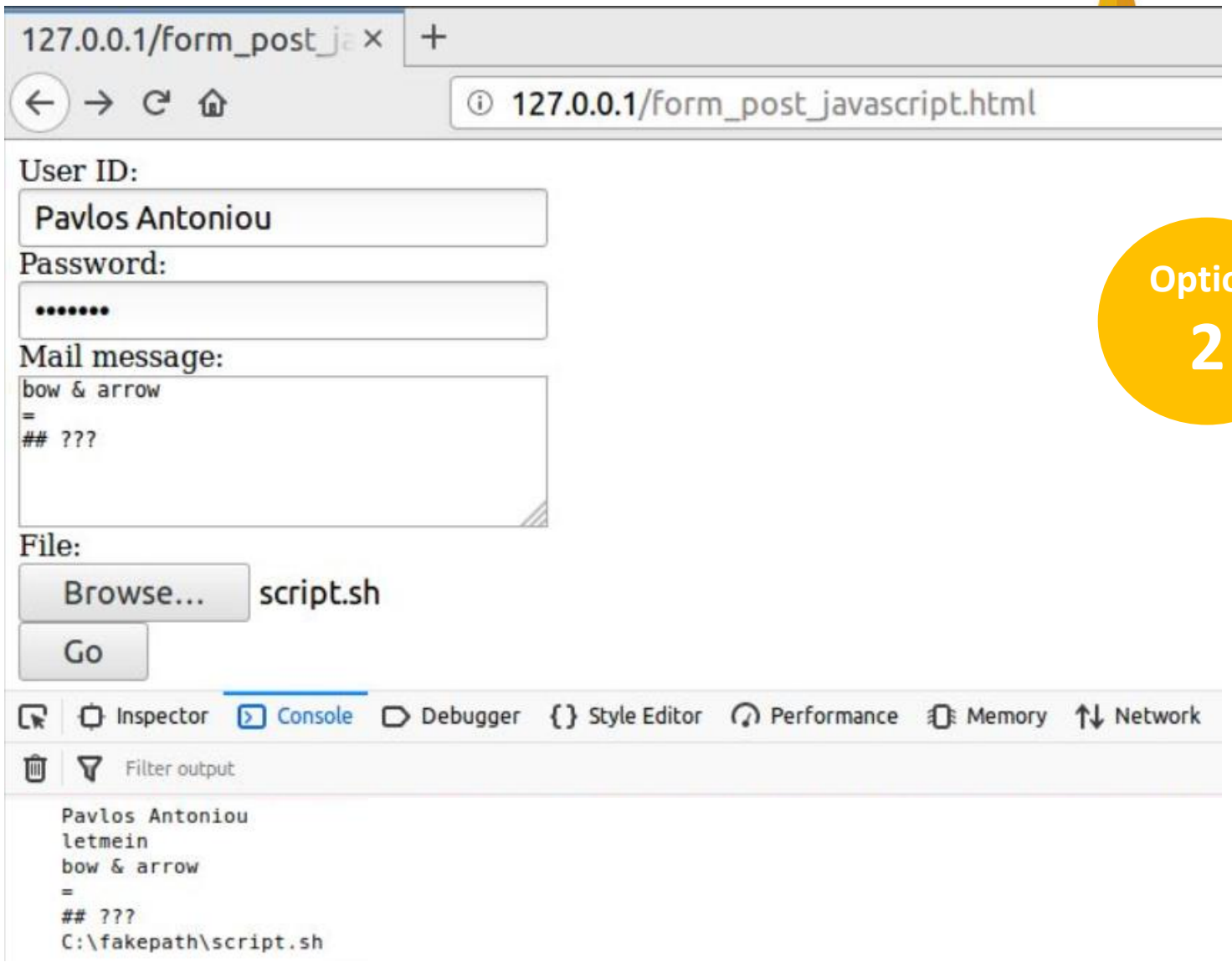
action_page.php

PHP

```
if(strcasecmp($_SERVER["CONTENT_TYPE"], "application/json") == 0) {
        $json = trim(file_get_contents("php://input"));
        $data = json_decode($json);
        // access properties of PHP object
        echo $data->userid . "\n";
        echo $data->passwd . "\n";
        echo urldecode($data->msg) . "\n";
        echo $data->txtfile . "\n";
}
```

After button click

- Form data are NOT shown in browser address bar
- Browser is NOT redirected to action_page.php
- Web page does not reload after AJAX call

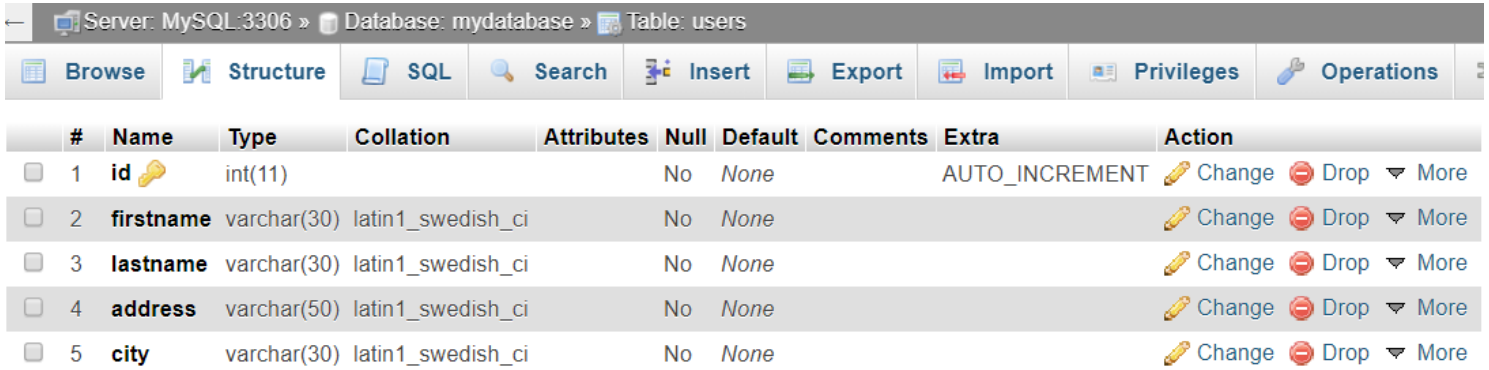


Option 2

Basic PHP MySQL functions

See on APPENDIX how to create MySQL DB & the following Table in phpMyAdmin.

- Connect to MySQL server
- Select a database
- Run a query
- Use results of query
- Close the connection (disconnect from MySQL server)



- Original functions start with `mysql_`
- Improved version from php5 starts with `mysqli_`

Connect to MySQL server

- **mysqli_connect(server, username, password)**
  - server default is the string "localhost" if mysql is installed on the same machine; otherwise url of the mysql server must be used (e.g. dbserver.in.cs.ucy.ac.cy in HW1)
  - username is a string for the user name (e.g. student in HW1)
  - password is a string for the password
- E.g. for WAMP/MAMP/XAMPP with default username (root) & password:

```php
<?php
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " .
    mysqli_error($conn));
    echo "Successful Connection";
    mysqli_close($conn);
?>
```

Error messages and closing connection

- **mysqli_error(connection)**
  - Returns an error string or error number (connection is optional - with last opened connection used if none supplied)
  - Empty string is returned if there is no error.
- **mysqli_close(connection)**
  - Closes the database connection to release allocated resources

Select a database

- **mysqli_select_db(connection , name)**
  - Select a database given by the string name (e.g. epl425 in HW1)
  - The connection variable is required

PHP

```php
<?php
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " .
    mysqli_error($conn));
    mysqli_select_db($conn , "mydatabase") or die ("db will not open" . mysqli_error($conn));
    echo "Database Connected";
    mysqli_close($conn);
?>
```

Run a query

- **mysqli_query(connection , query)**
  - query is a string for the MySQL query (in SQL)
  - semicolon (;) should NOT be used to terminate query
  - query uses valid SQL command

PHP

```php
<?php
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " .
    mysqli_error($conn));
    mysqli_select_db($conn , "mydatabase") or die ("db will not open" . mysqli_error($conn));
    $query = "SELECT * FROM users";
    $result = mysqli_query($conn, $query) or die("Invalid query");
    echo "Successful Query";
    mysqli_close($conn);
?>
```

Parsing results from MySQL

- **mysqli_num_rows(result)**
  - returns number of rows from a select query
- **mysql_fetch_row(result)**
  - each call returns the next row as an numerical array, keys start from 0
- **mysql_fetch_assoc(result)**
  - each call returns the next row as an associative array, table column names are the keys storing corresponding value
- **mysql_fetch_array(result)**
  - each call returns an array with both the contents of mysql_fetch_row and mysql_fetch_assoc merged into one. It will both have numeric and string keys which will let you access your data in whatever way you'd find easiest.
- **mysql_fetch_object(result)**
  - each call returns the next row as an object

Examples (using for loop)

```php
$num = mysqli_num_rows($result);
for($i=0; $i<$num; $i++) {
    $row = mysqli_fetch_row($result);
    echo $row[0] . " " . $row[1] . " " . $row[2] . " " . $row[3] . " " . $row[4] . "<br/>";
}
```

PHP

```php
$num = mysqli_num_rows($result);
for($i=0; $i<$num; $i++) {
    $row = mysqli_fetch_assoc($result);
    echo $row['id'] . " " . $row['firstname'] . " " . $row['lastname'] . " " .
$row['address'] . " " . $row['city'] . "<br/>";
}
```

PHP

```php
$num = mysqli_num_rows($result);
  for($i=0; $i<$num; $i++) {
    $row = mysqli_fetch_object($result);
    echo $row->id . " " . $row->firstname . " " . $row->lastname . " " . $row->address . " "
. $row->city . "<br/>";
}
```

PHP

Examples (using for while)

```php
while($row = mysqli_fetch_row($result)) {
    echo $row[0] . " " . $row[1] . " " . $row[2] . " " . $row[3] . " " . $row[4] . "<br/>";
}
```

```php
while($row = mysqli_fetch_assoc($result)) {
    echo $row['id'] . " " . $row['firstname'] . " " . $row['lastname'] . " " .
$row['address'] . " " . $row['city'] . "<br/>";
}
```

```php
while($row = mysqli_fetch_object($result)) {
        echo $row->id . " " . $row->firstname . " " . $row->lastname . " " . $row->address .
" " . $row->city . "<br/>";
}
```

```php
$users = array();
while($row = mysqli_fetch_assoc($result)) {  # instead of printing data
        array_push($users, $row);            # create a PHP array to store all rows
}                                            # an export it as json
echo json_encode($users,JSON_NUMERIC_CHECK); # this is a more structured way of exposing data
```

Other MySQL functions

- **mysqli_affected_rows(result)**
  - used after INSERT, UPDATE, or DELETE query to return the number of rows affected
- **mysqli_free_result(result)**
  - frees the memory associated with the result

Set header and response code

- **header(string)** is used to send a raw HTTP header e.g. “Content-type: application/json”
  - must be called before any actual output is sent
- **http_response_code(code)** is used to set the HTTP response code e.g. 404 (Not Found), 400 (Bad Request), 301 (Moved Permanently), etc
  - By default, the return response code is 200 (OK)

Set header and response code

PHP

```php
<?php
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " . mysqli_error($conn));
    mysqli_select_db($conn , "mydatabase") or die ("db will not open" . mysqli_error($conn));
    $query = "SELECT * FROM users WHERE userid=4";
    $result = mysqli_query($conn, $query) or die("Invalid query");
    if (mysqli_num_rows($result) > 0) {
        header('Content-Type: application/json;');
        http_response_code(200);
        $users = array();
        while($row = mysqli_fetch_assoc($result)) {
                array_push($users, $row);
        }
        echo json_encode($users);
    } else {
        header('Content-Type: application/json;');
        http_response_code(404);
        $reply['status'] = 'fail';
        $reply['message'] = 'data not found in db';
        echo json_encode($reply, JSON_NUMERIC_CHECK);
    }
?>
```

Exercise 3

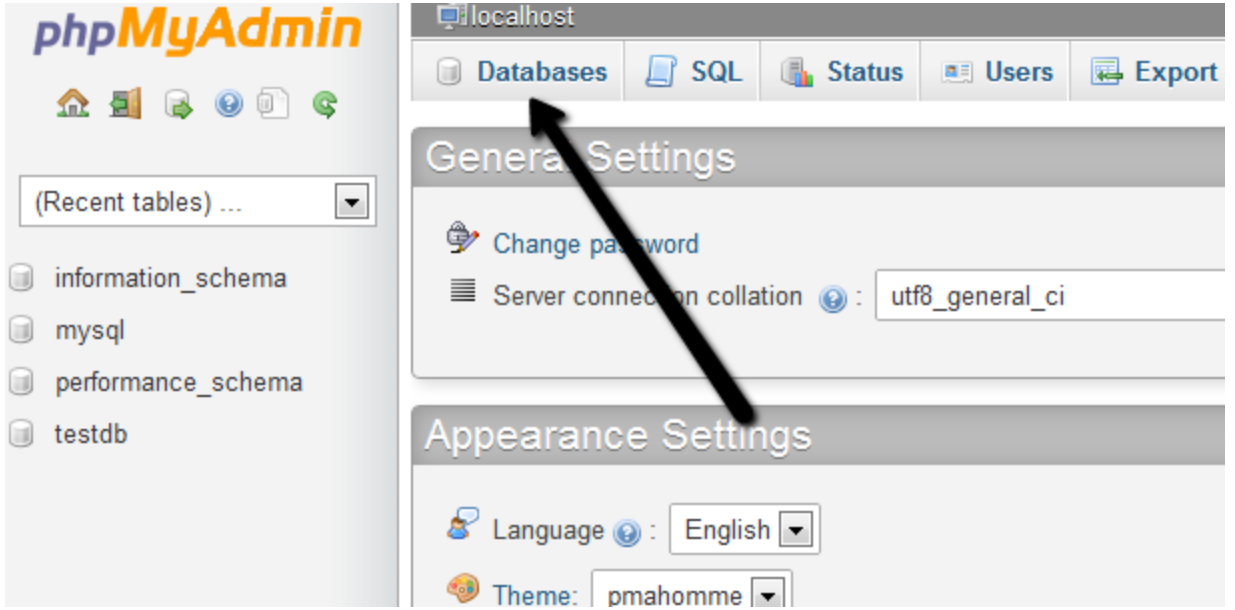
- Create a simple form to insert a row in table user (firstname, lastname, address, city) of the mydatabase (MySQL DB)
- Create a boostrap table that shows all information of the users table.

Set up MySQL DB & table using phpmyadmin

University of Cyprus
Department of Computer Science

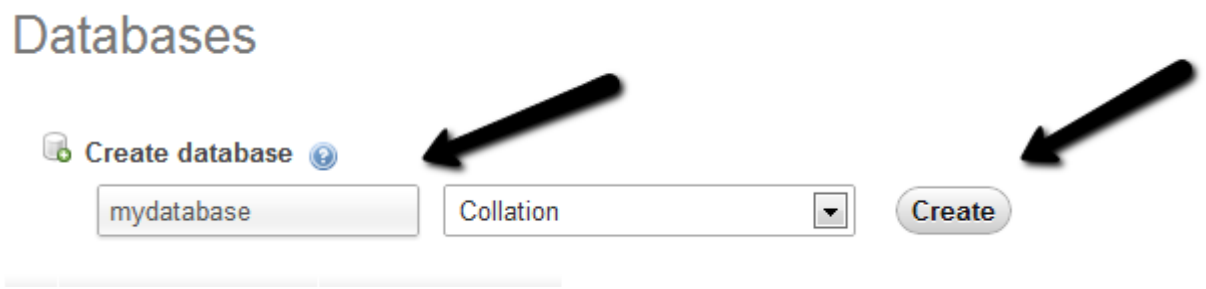
Create DB in phpMyAdmin - 1

- Browse to your phpMyAdmin URL using your Internet Web Browser
  - E.g. on local XAMPP: http://127.0.0.1/phpmyadmin
- From the main menu choose **Databases**



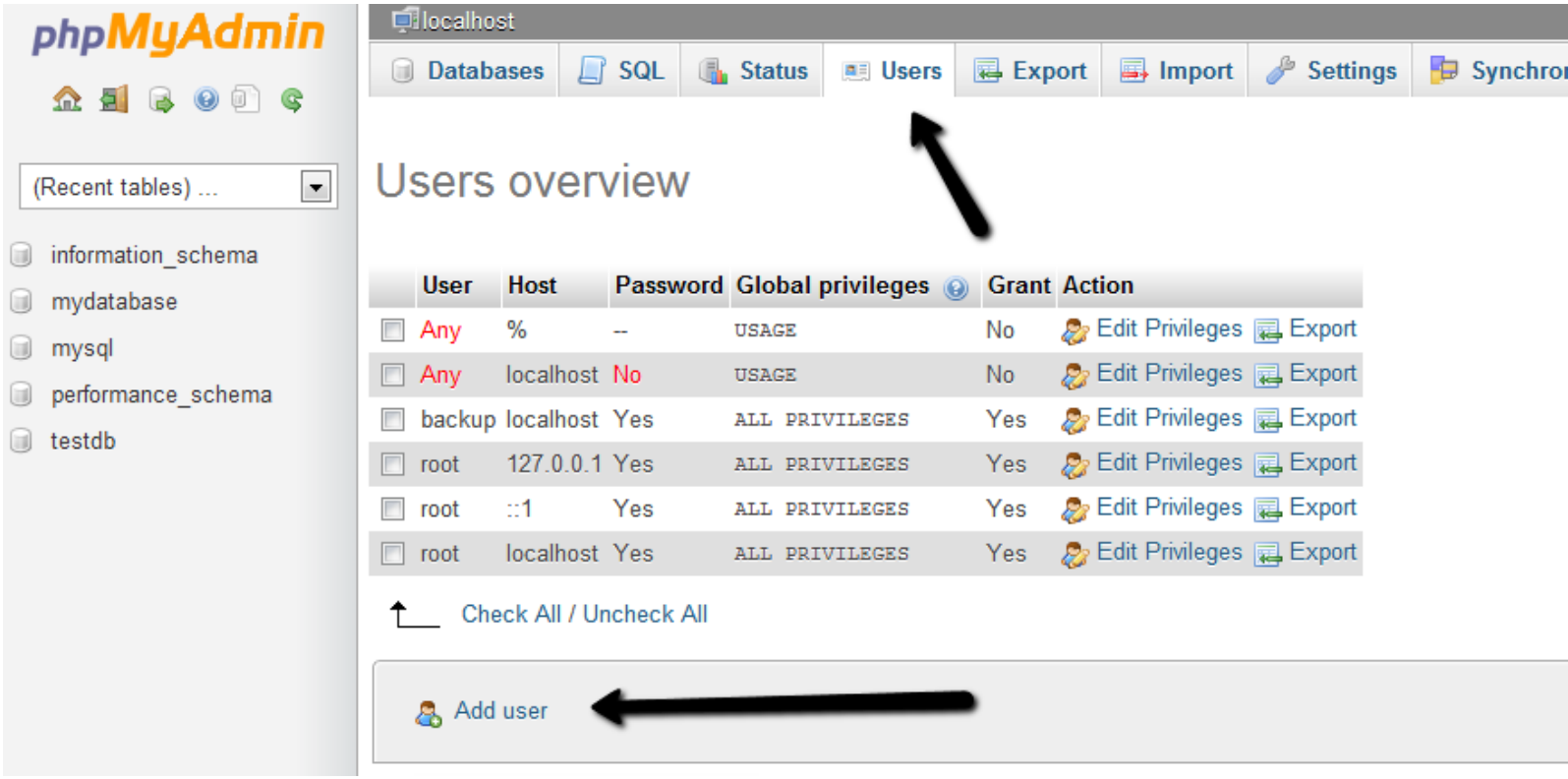
Create DB in phpMyAdmin - 2

- In the create database field type in a name for your database. Leave the collation drop down box if you wish to use the default MySQL schema collation. Click **Create**.



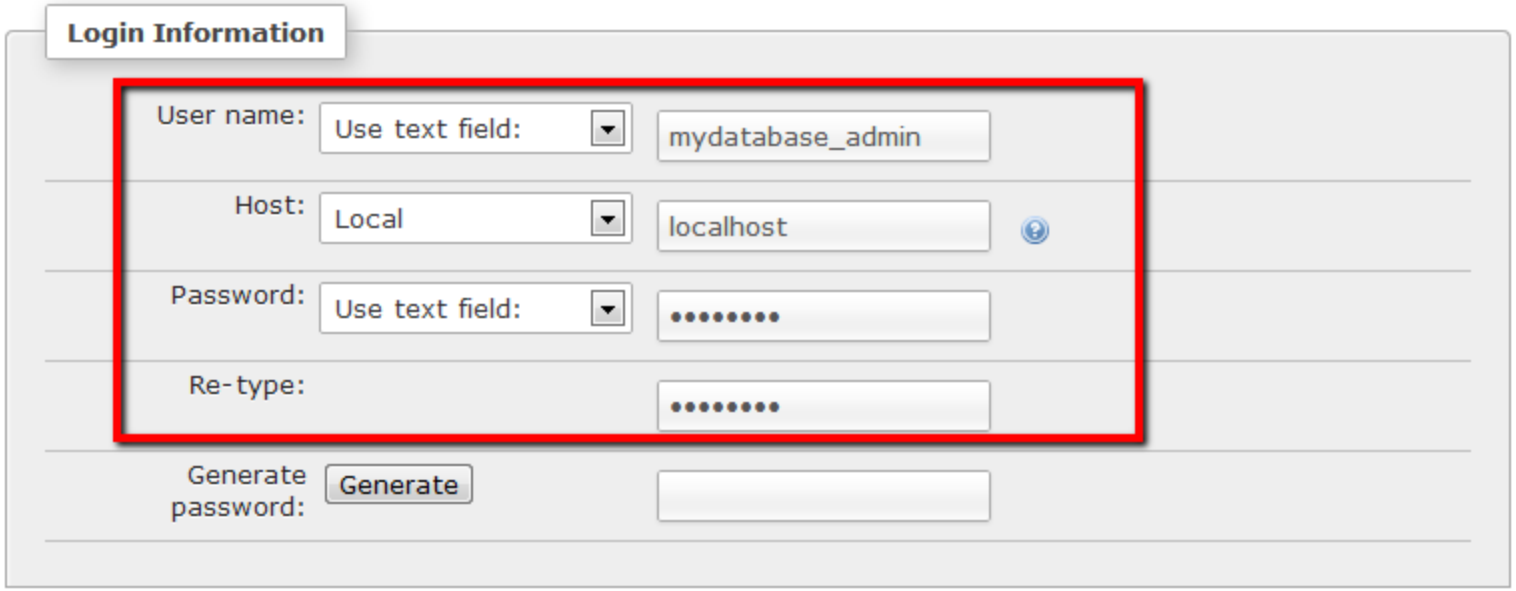
Create DB in phpMyAdmin - 3

- Your database will now be visible on the right hand side under the list of available databases. To setup a new user login to access this database, click on **Users** (or User account) in the main menu. Choose the **Add User** option under the list of available server users.



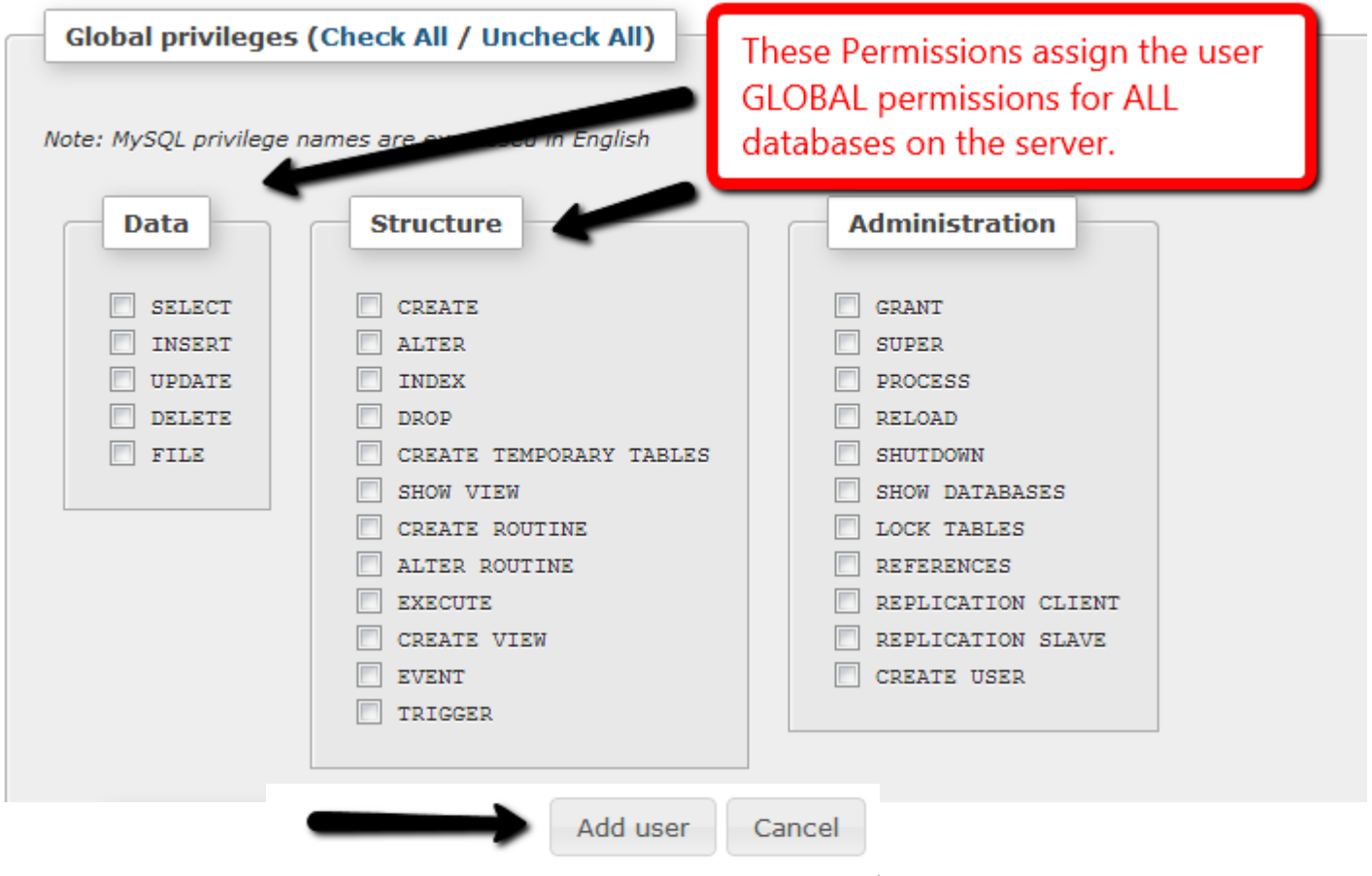
Create DB in phpMyAdmin - 4

- In the section titled **Login Information** - type in a **username**, **localhost** and a **password** in the fields as shown. Optionally you can press the **Generate** button to create a random password for you.



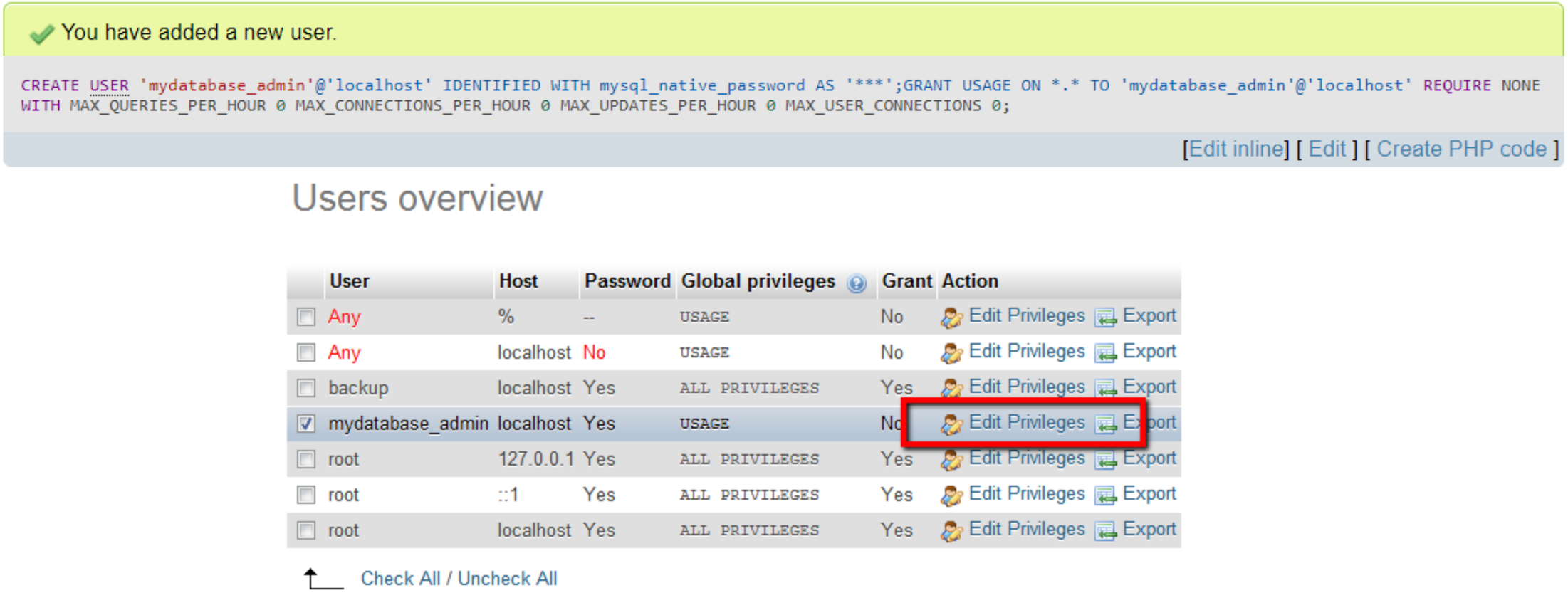
Create DB in phpMyAdmin - 5

- The section that relates to the users GLOBAL privileges are privileges you want to assign to this user which apply to **ALL databases** on the server. It is recommended that you do **NOT** assign these permissions unless you know exactly what you are doing. It is far more secure to assign separate user logins to each piece of software or website that will require access to only a *particular database*. Therefore press **Add User** (or Go) button.



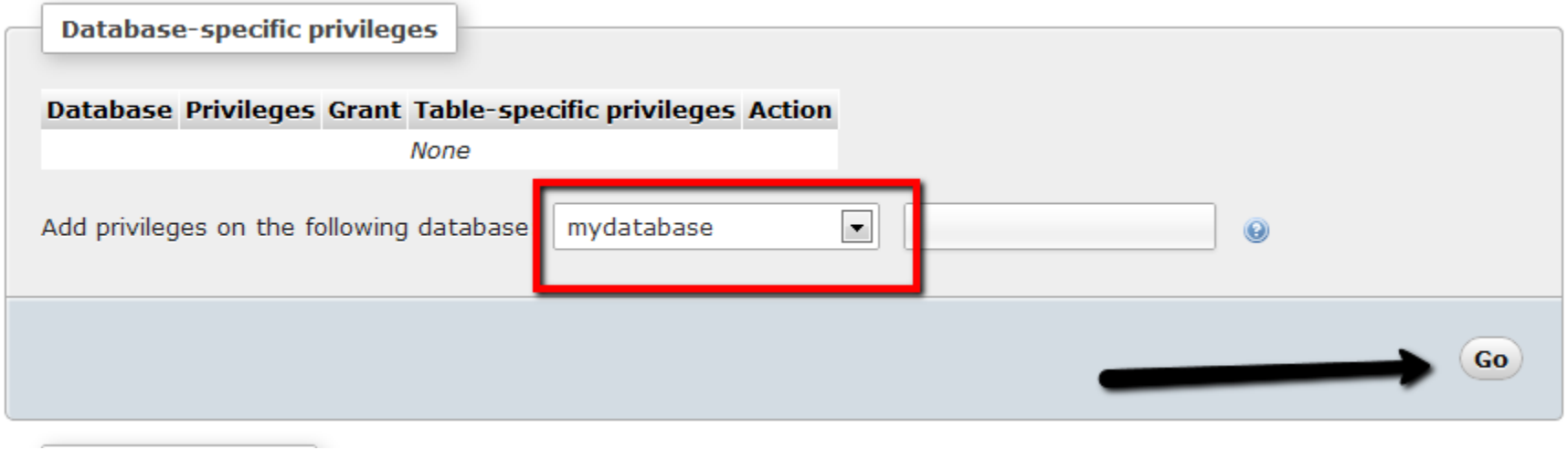
Create DB in phpMyAdmin - 6

- After the user is created, you can see it listed on the Users page. Click **Edit Privileges** to assign access to a specific database.



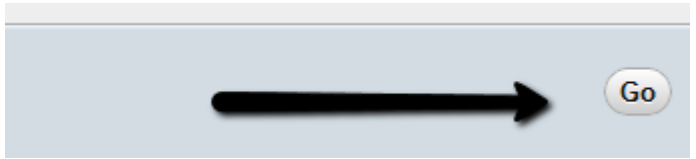
Create DB in phpMyAdmin - 7

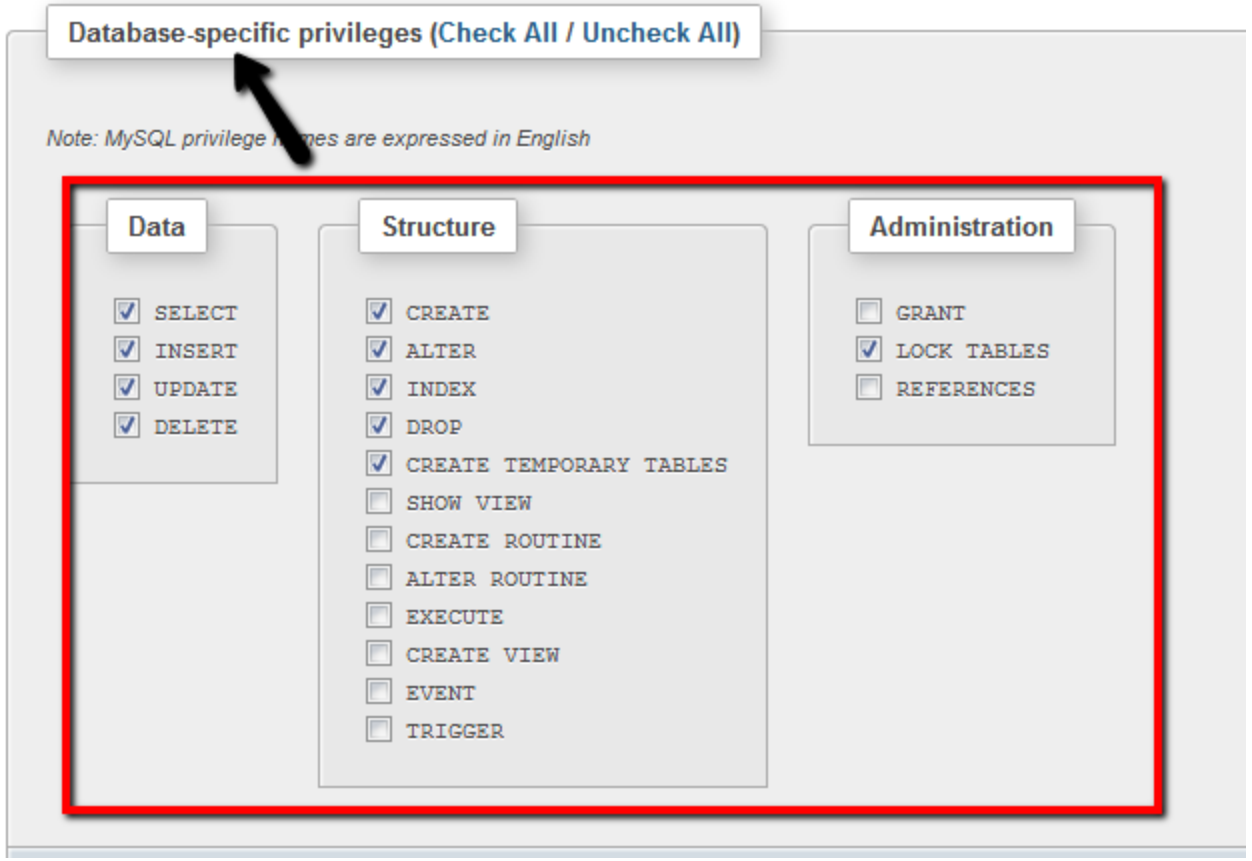
- Once again leave the Global Privileges section **BLANK**. Select the tab titled **Database**. Choose the **database** you want the user to be able to access from the list, and click **GO**.

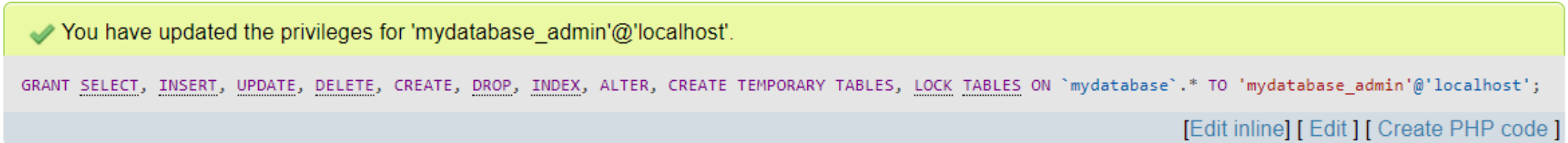


Create DB in phpMyAdmin - 8

- Assign the permissions as shown to provide the user with access to the given database. The selected permissions are recommended for compatibility with most modern web-based software apps
- Click **GO** after selecting the relevant privileges.







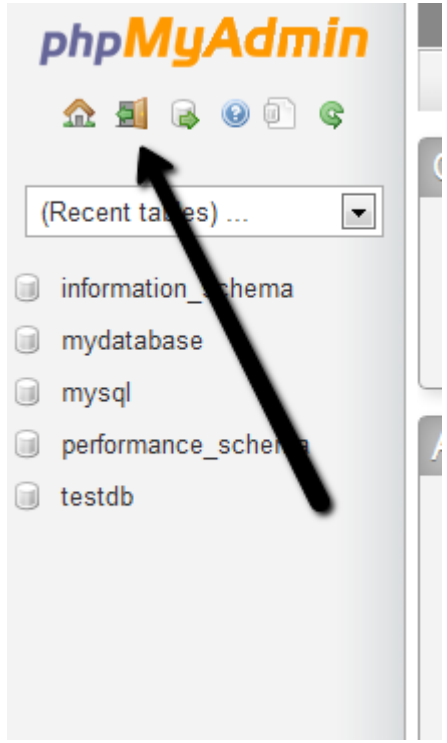
Create DB in phpMyAdmin - 9

- If you click on the users Edit Privileges option now, you will see that new privileges for the specific database are now listed as belonging to the user.

Database-specific privileges

Database	Privileges	Grant	Table-specific privileges	Action
mydatabase	SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,INDEX,ALTER,CREATE TEMPORARY TABLES,LOCK TABLES	No	No	Edit privileges Revoke

- Click the Logout option in the top left corner, and test your new user login with phpMyAdmin.

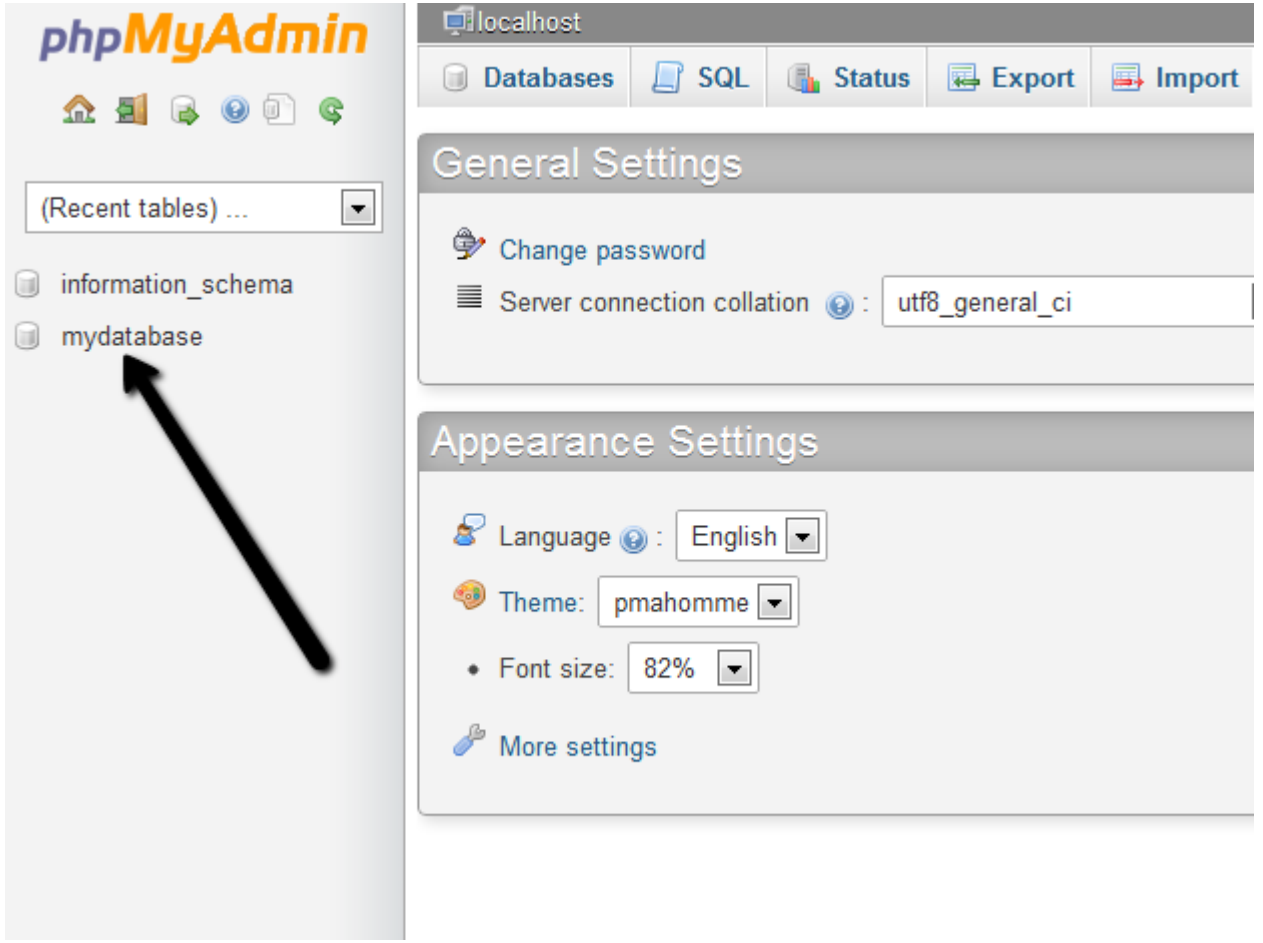


Create DB in phpMyAdmin - 10

- Test your new user login by using it to login to phpMyAdmin.

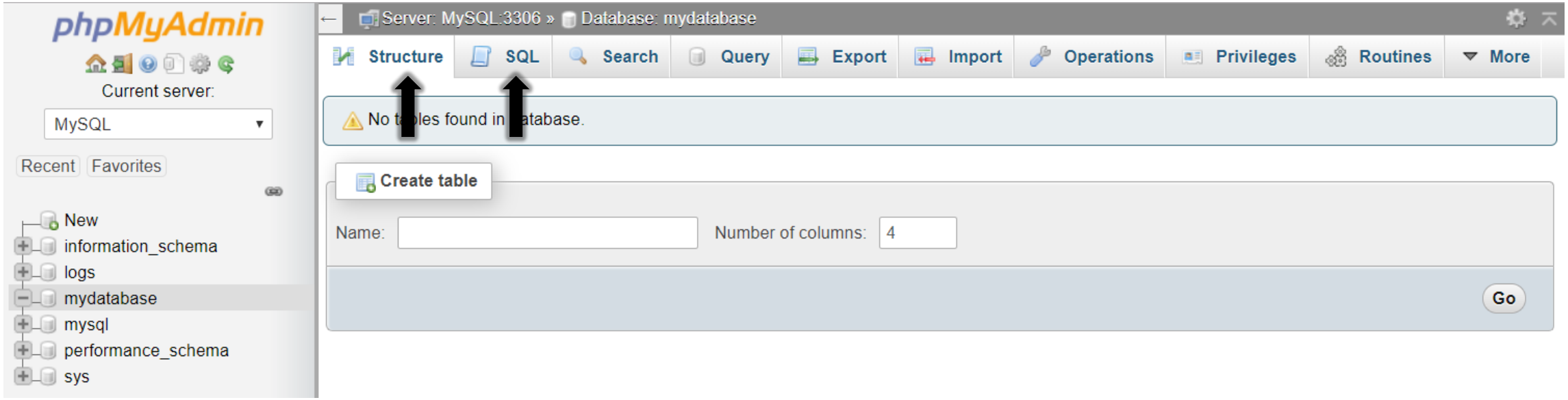
Create DB in phpMyAdmin - 11

- If you can only your new database in the list of schema's on the left then your new database and username is most likely ready for use.



Create Table in phpMyAdmin - 1

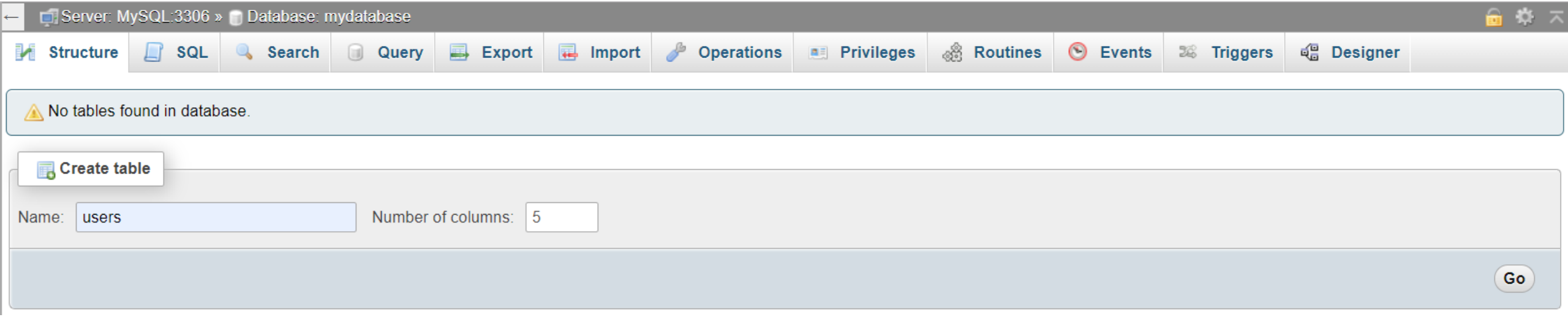
- Click on the database name in which under you create a table. After click on the database name you find a page like that.



- You have two options to create table
  - use **structure**
  - using **SQL**

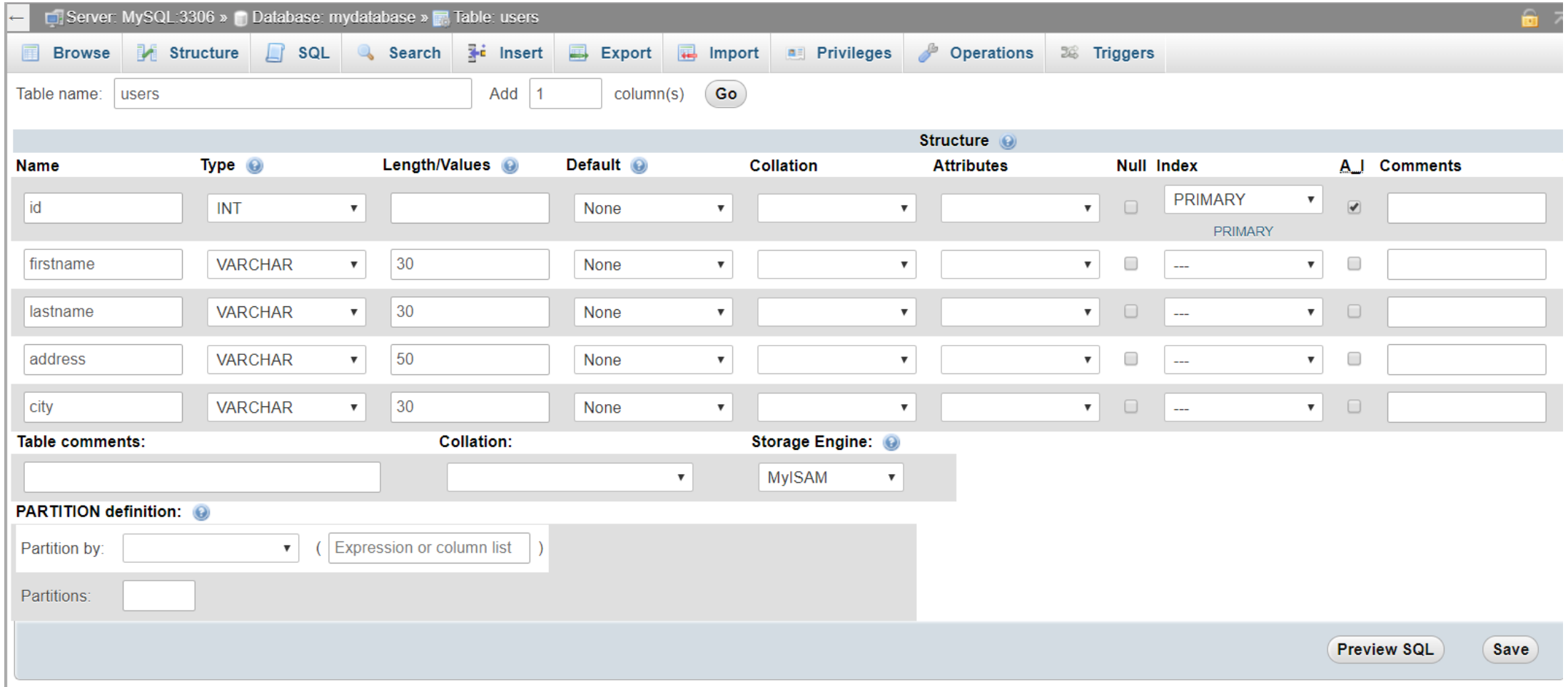
Create Table in phpMyAdmin - 2

- If you want to create a table by writing SQL Query simply click on the **SQL** button on the page and write your query and click on the go button.
- Else click **Structure**, provide the name of the table and the number of rows and then **Go**



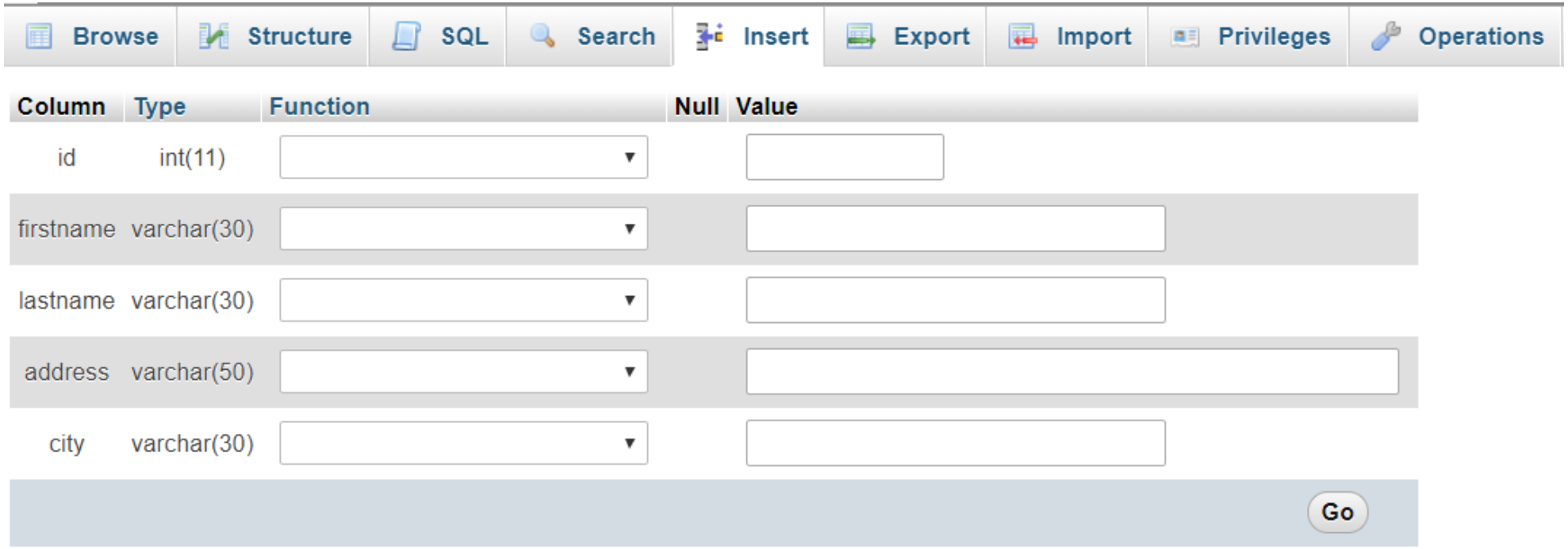
Create Table in phpMyAdmin - 3

- Provide the necessary information and click on **Save**



Create Table in phpMyAdmin - 4

- You have two options to insert data in table
  - use **Insert**



  - using **SQL**

Create Table in phpMyAdmin - 5

- Visit **Browse** to see all rows of the table

+ Options

	id	firstname	lastname	address	city
Edit Copy Delete	1	John	Smith	7 Goldsmiths road	London
Edit Copy Delete	2	Adam	Rodgers	12A Bolton avenue	New Jersey
Edit Copy Delete	3	Mary	Delagrange	22 Living street	Lancaster
Edit Copy Delete	4	Christopher	Devon	8 Red Cross street	Manchester

- Visit **Structure** to see all columns (and their types) of the table

Server: MySQL:3306 » Database: mydatabase » Table: users

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	firstname	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
3	lastname	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More
4	address	varchar(50)	latin1_swedish_ci		No	*None*			Change Drop More
5	city	varchar(30)	latin1_swedish_ci		No	*None*			Change Drop More