Internet Technologies

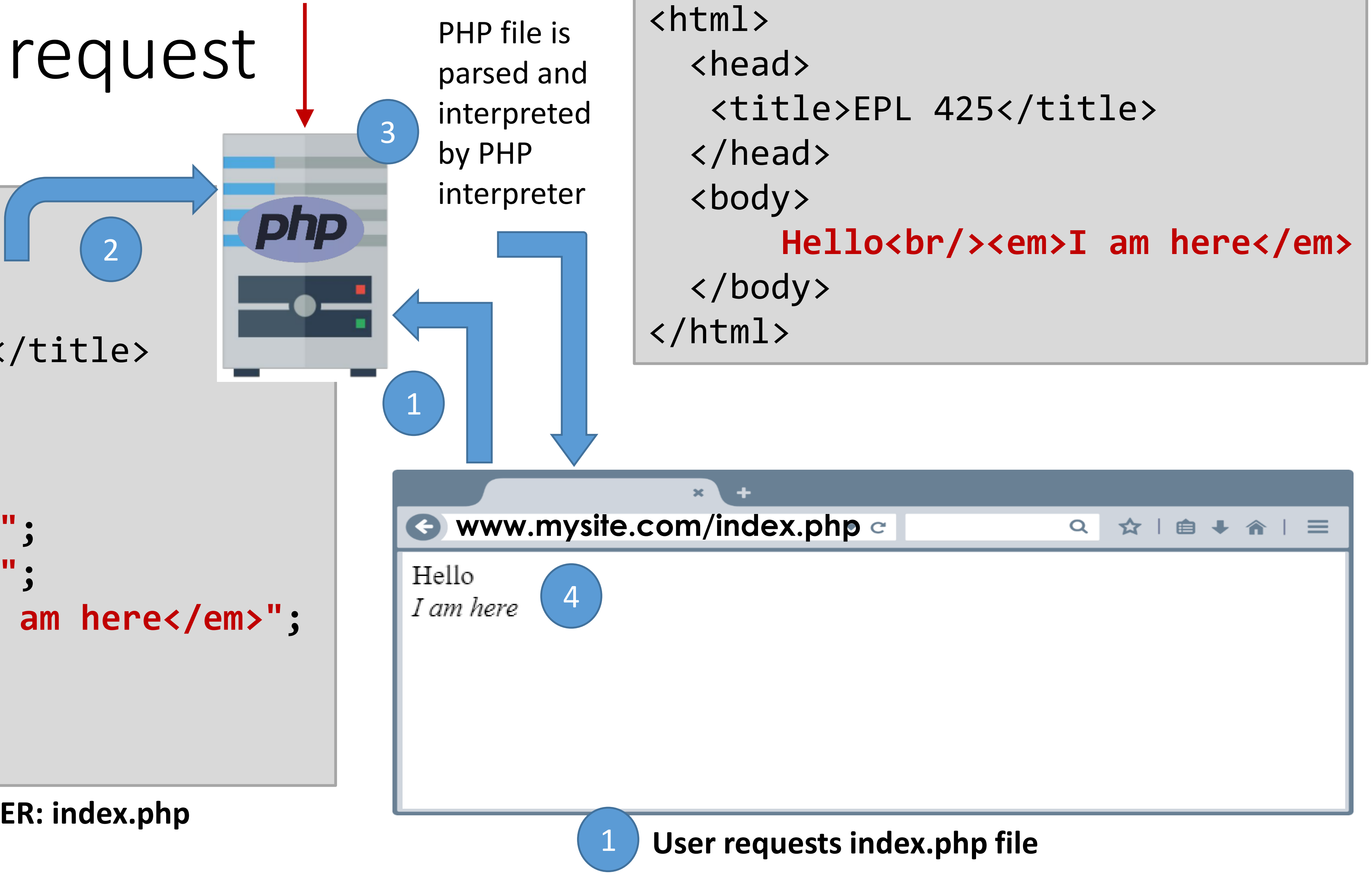
Introduction to PHP

University of Cyprus
Department of Computer Science

Introduction to PHP

- PHP is a **server-side** scripting language
- **PHP scripts are executed on the server**
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all web servers (Apache, Nginx, IIS, etc.)
- PHP can:
    - collect form data (sent via JavaScript from browser to server)
    - create, update, delete data from database
        - Supports connection to many databases (MySQL, Oracle, PostgreSQL, etc.)
    - send/receive cookies
    - create, open, read, write, delete and close files on the server

PHP file request


WEB SERVER with
PHP engine (PHP interpreter)
php
1
2
3
PHP file is parsed and interpreted by PHP interpreter
www.mysite.com/index.php
Hello
I am here
4
1
User requests index.php file

```
<!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>
```

PHP data structures

- Arrays and objects

- 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

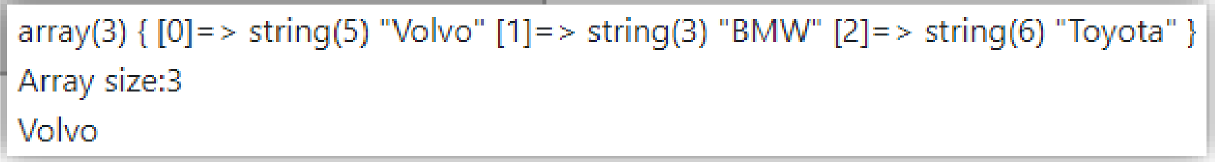
PHP Numeric arrays

All PHP variables start with $

- 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
```

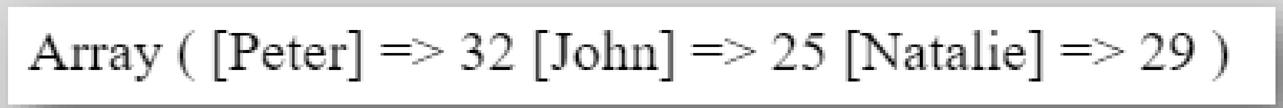
```php
<?php
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

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
$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

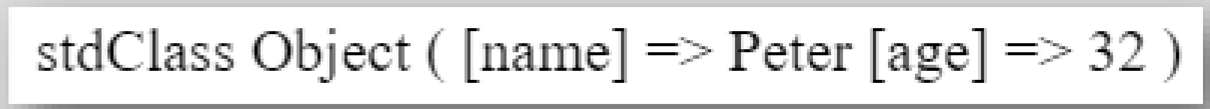
```php
<?php
class my_class
{
    function print_msg()
    {
        echo "Hello world.";
    }
}

$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 to collect data from HTML forms

- Data transfer from browser to server is activated via:
  - HTML form submission (without JavaScript)
  - JavaScript (submit form or XMLHttpRequest/Fetch API)
- Data travels across the Internet on top of HTTP messages:
  - GET messages
  - POST messages
- Data received/processed to server using PHP script

GET method

- Sends data appended to the request URL

Request URL:

$$\texttt{https:/}\underbrace{\texttt{/www.test.com/}}_{\text{Web server domain name}}\underbrace{\texttt{index.php?}}_{\text{Filename that will receive user information}}\underbrace{\texttt{key1=value1\&key2=value2}}_{\text{Data to be sent to web server}}$$

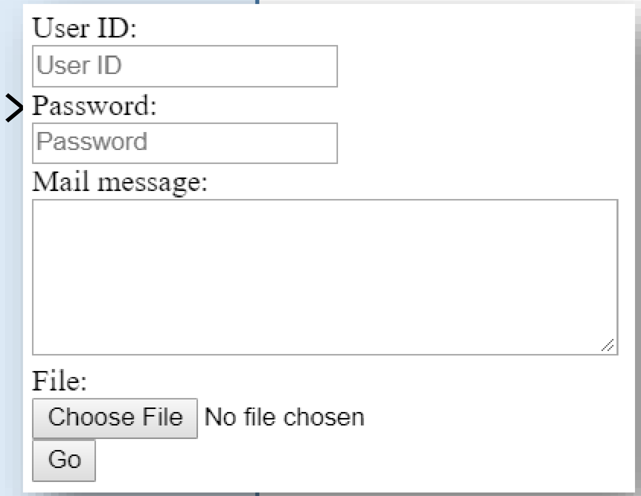
- Data has to be URL encoded prior sending to server (special characters e.g. # or spaces are replaced with a % followed by two hexadecimal digits)
- In request URL, the filename and the encoded data are separated by the ? character, followed by name/value pairs
- Name/value pairs are joined with equal signs (=) and different pairs are separated by the ampersand (&)

Browser
Sends GET msg via HTML form submission

form_get.html

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>
```


User ID:

User ID

Password:

Password

Mail message:

File:

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 reloads and is redirected to action_page.php**.

Web server
Receives GET msg using PHP

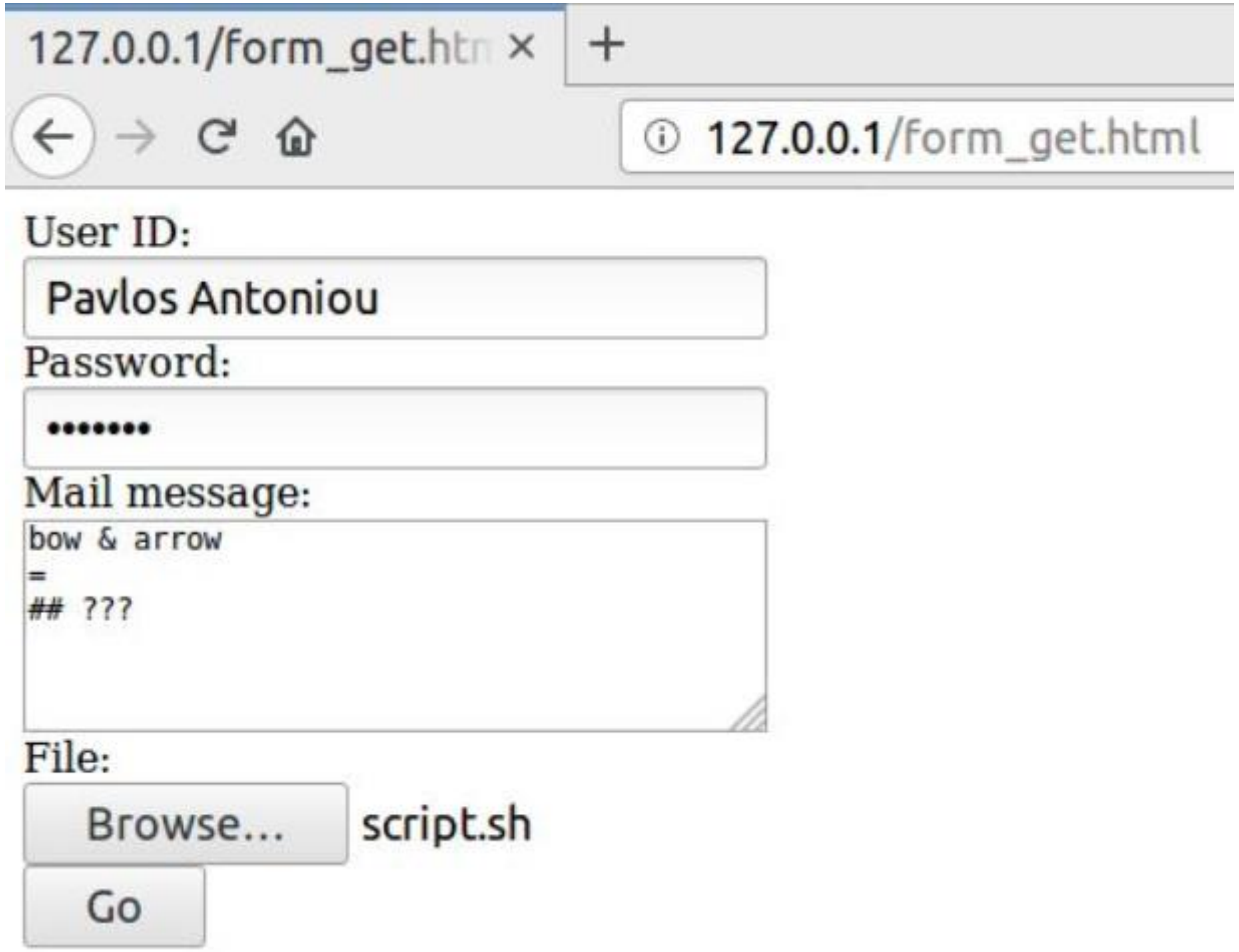
action_page.php

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/>";
}
```

PHP provides **$_GET** array to access all user information send via GET method
$_SERVER array provides information of request headers, paths, script location, etc.

Before form submission



127.0.0.1/form_get.htn ×

127.0.0.1/form_get.html

User ID:

Pavlos Antoniou

Password:

••••••••

Mail message:

```
bow & arrow
=
## ???
```

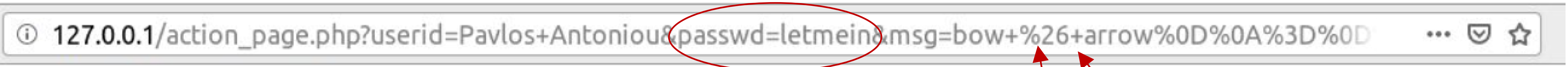
File:

Browse... script.sh

Go

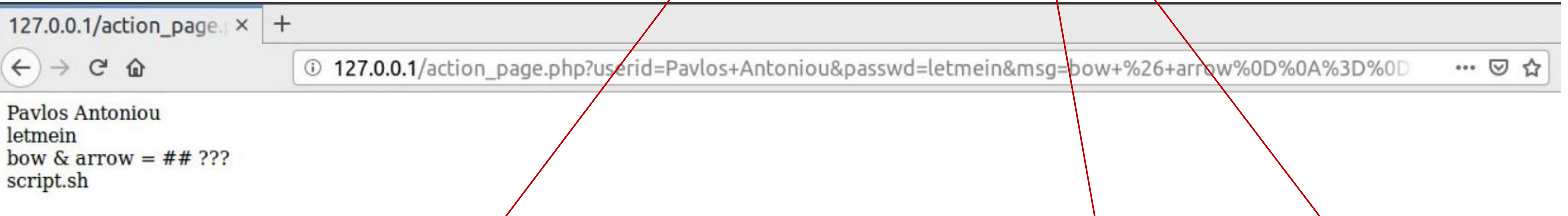
After form submission

- Form data are shown in browser address bar



```
127.0.0.1/action_page.php?userid=Pavlos+Antoniou&passwd=letmein&msg=bow+%26+arrow%0D%0A%3D%0D
```

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



```
127.0.0.1/action_page.php?userid=Pavlos+Antoniou&passwd=letmein&msg=bow+%26+arrow%0D%0A%3D%0D
```

```
Pavlos Antoniou
letmein
bow & arrow = ## ???
script.sh
```

- 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:

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. Function collects form data (using id of each field), and then creates XMLHttpRequest/Fetch API object to send GET msg to action_page.php.

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);
```

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

Send GET msg via JavaScript – Fetch API

JS

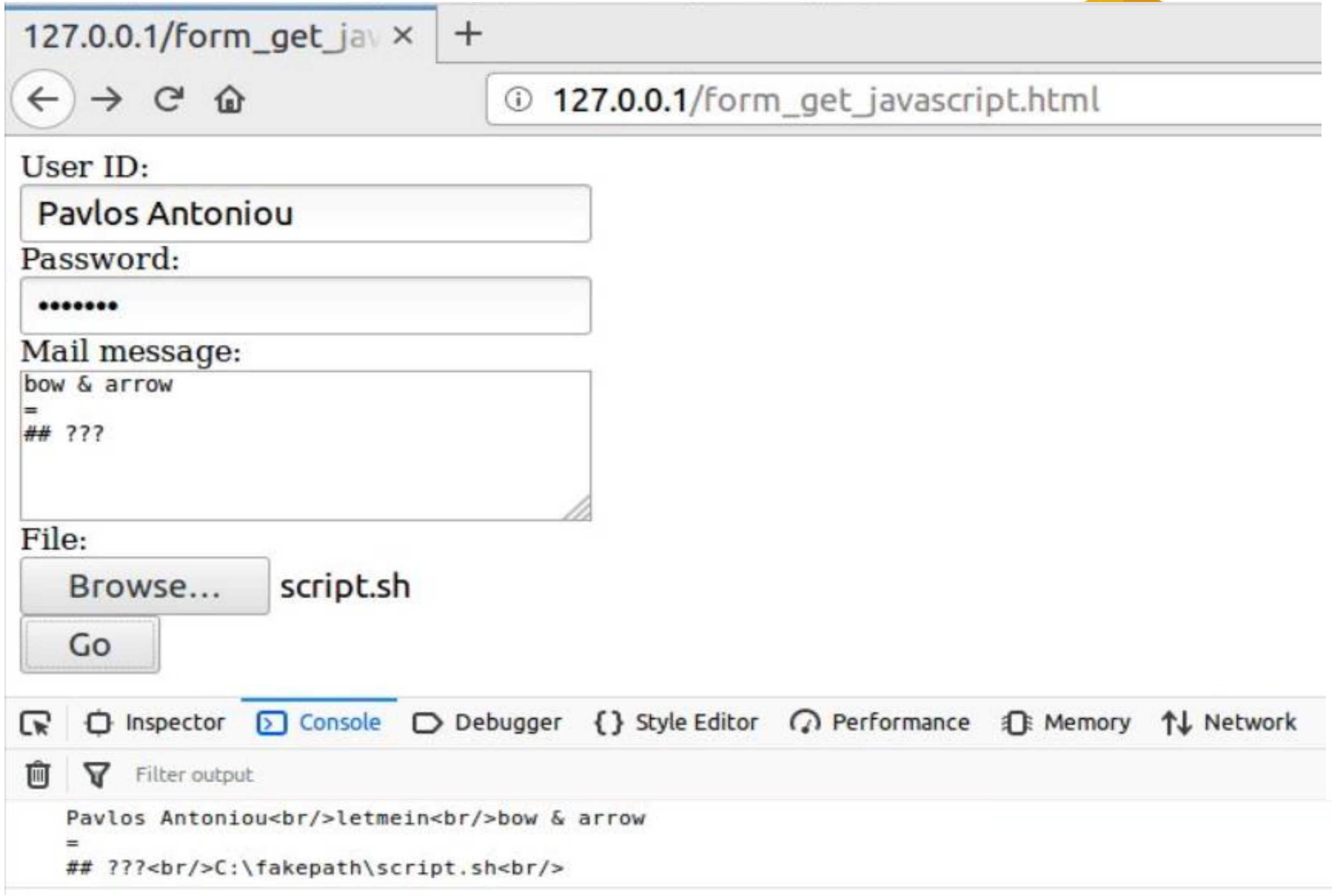
```
function onClick() {
    const userid = document.querySelector('#userid').value;
    const passwd = document.querySelector('#passwd').value;
    const msg = encodeURIComponent(document.querySelector('#msg').value);
    const txtfile = document.querySelector('#txtfile').value;

    fetch('action_page.php?userid='+userid+'&passwd='+passwd+'&msg='+msg+'&txtfile='+txtfile, {
      method: "GET"
    })
    .then(
      response => { // handle the response
          console.log(data);
        } // end of response
    ) // end of then
    .catch( error => { // handle the error
        console.log('error: ', error);
    });
}
const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

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



127.0.0.1/form_get_ja ×

127.0.0.1/form_get_javascript.html

User ID:

Pavlos Antoniou

Password:

•••••••

Mail message:

```
bow & arrow
=
## ???
```

File:

Browse... script.sh

Go

```
Pavlos Antoniou<br/>letmein<br/>bow & arrow
=
## ???<br/>C:\fakepath\script.sh<br/>
```

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

Browser
Sends POST msg via HTML form submission

form_post.html

```
<form action="./action_page.php" method="post"                    HTML
      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 reloads and is redirected to action_page.php** (file contents are not sent).

Web server
Receives POST msg using PHP

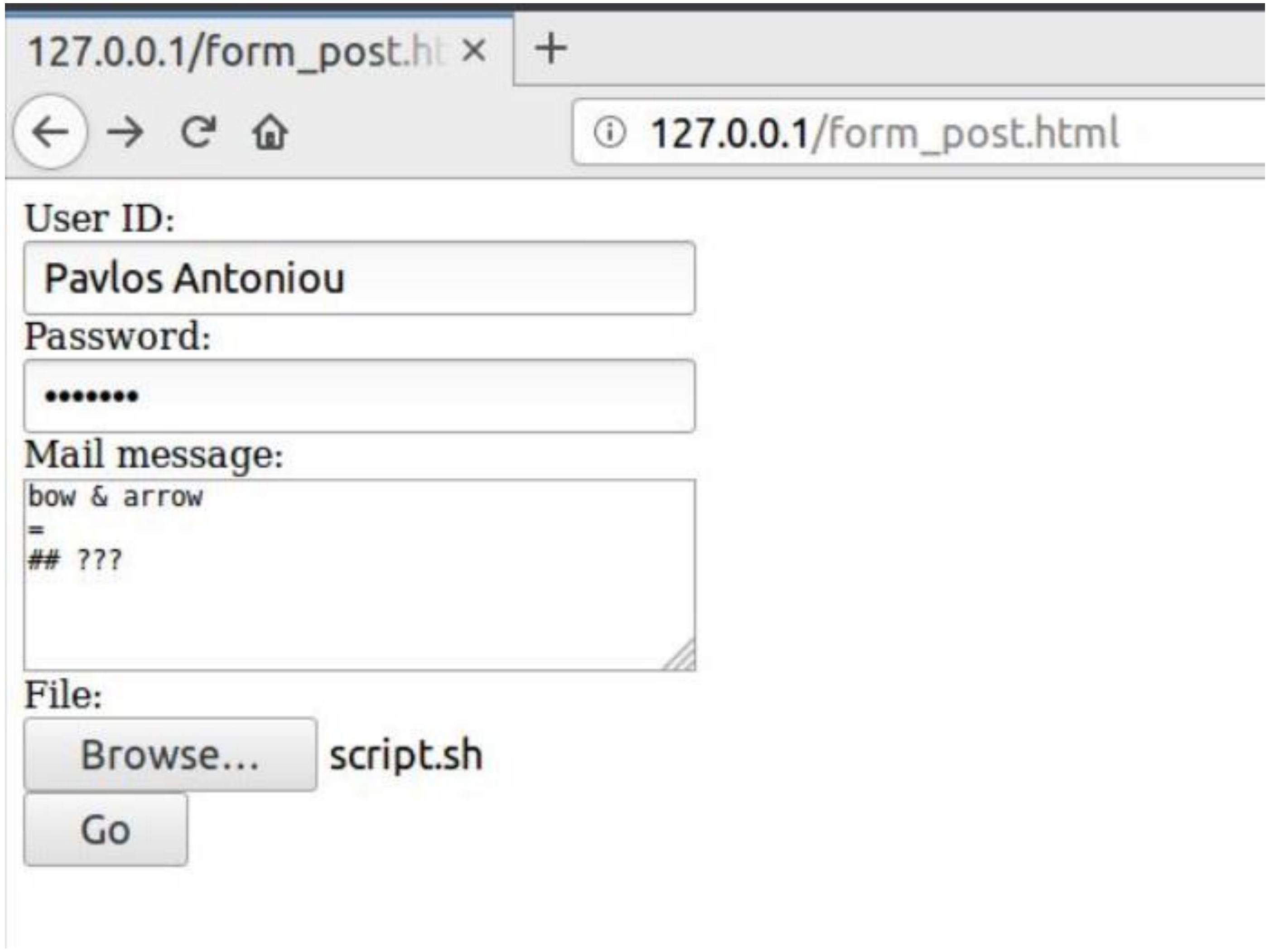
action_page.php

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 POST method

Before form submission



User ID:

Pavlos Antoniou

Password:

•••••••

Mail message:

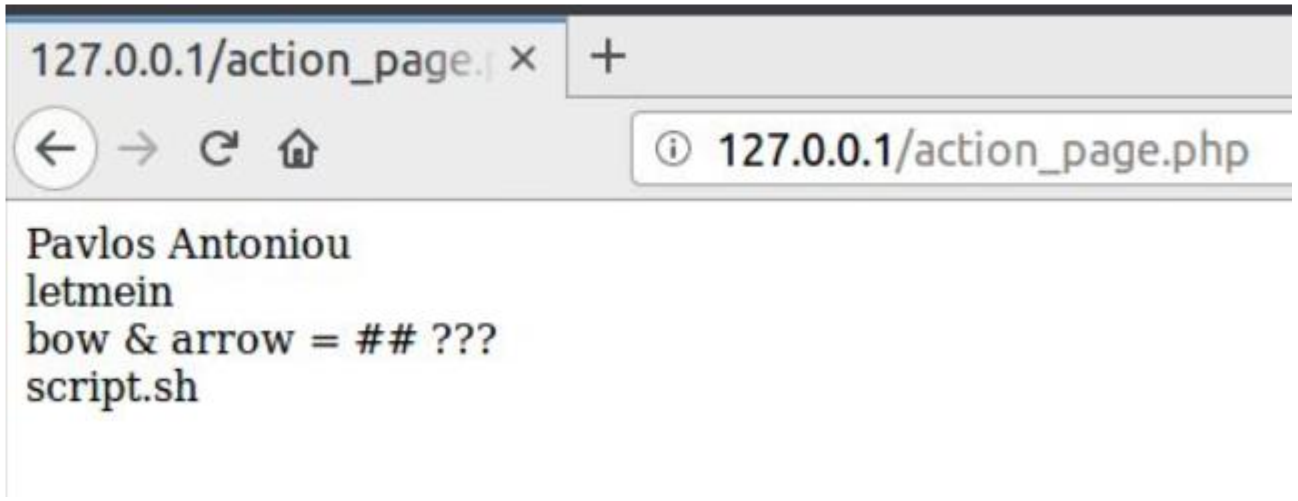
```
bow & arrow
=
## ???
```

File:

script.sh

After form submission

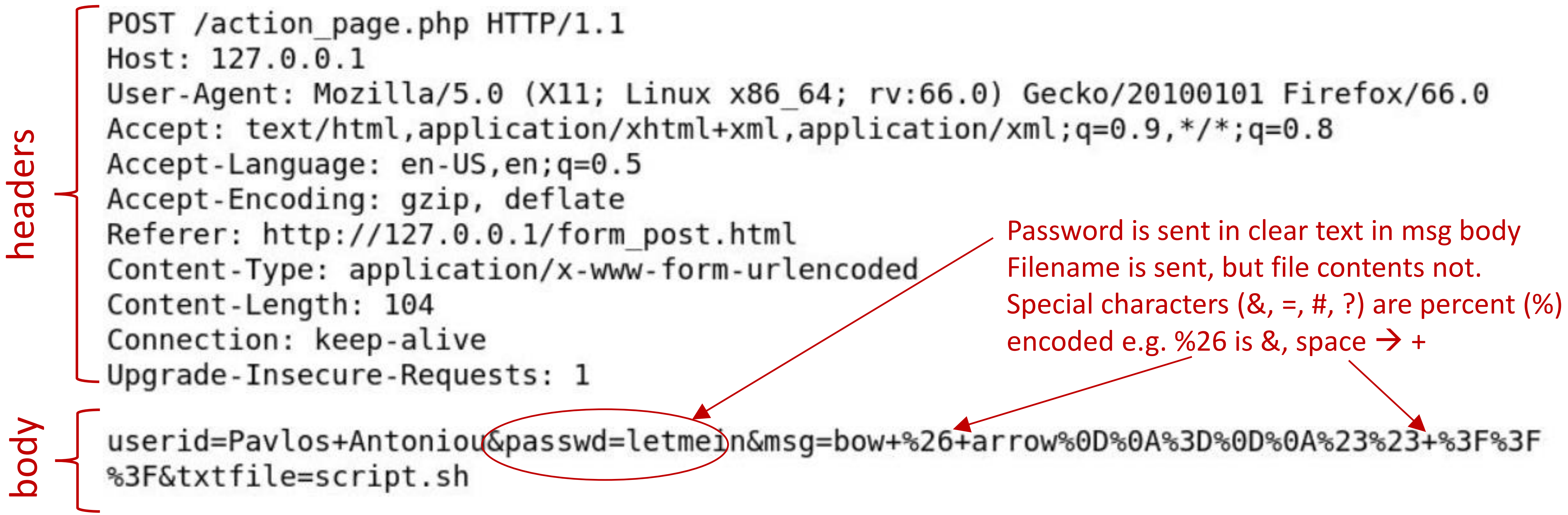
- Form data are NOT shown in browser address bar
- Browser **is redirected** to action_page.php



```
Pavlos Antoniou
letmein
bow & arrow = ## ???
script.sh
```

After form submission

- POST message as captured by 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.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
Connection: keep-alive
Upgrade-Insecure-Requests: 1

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 → +

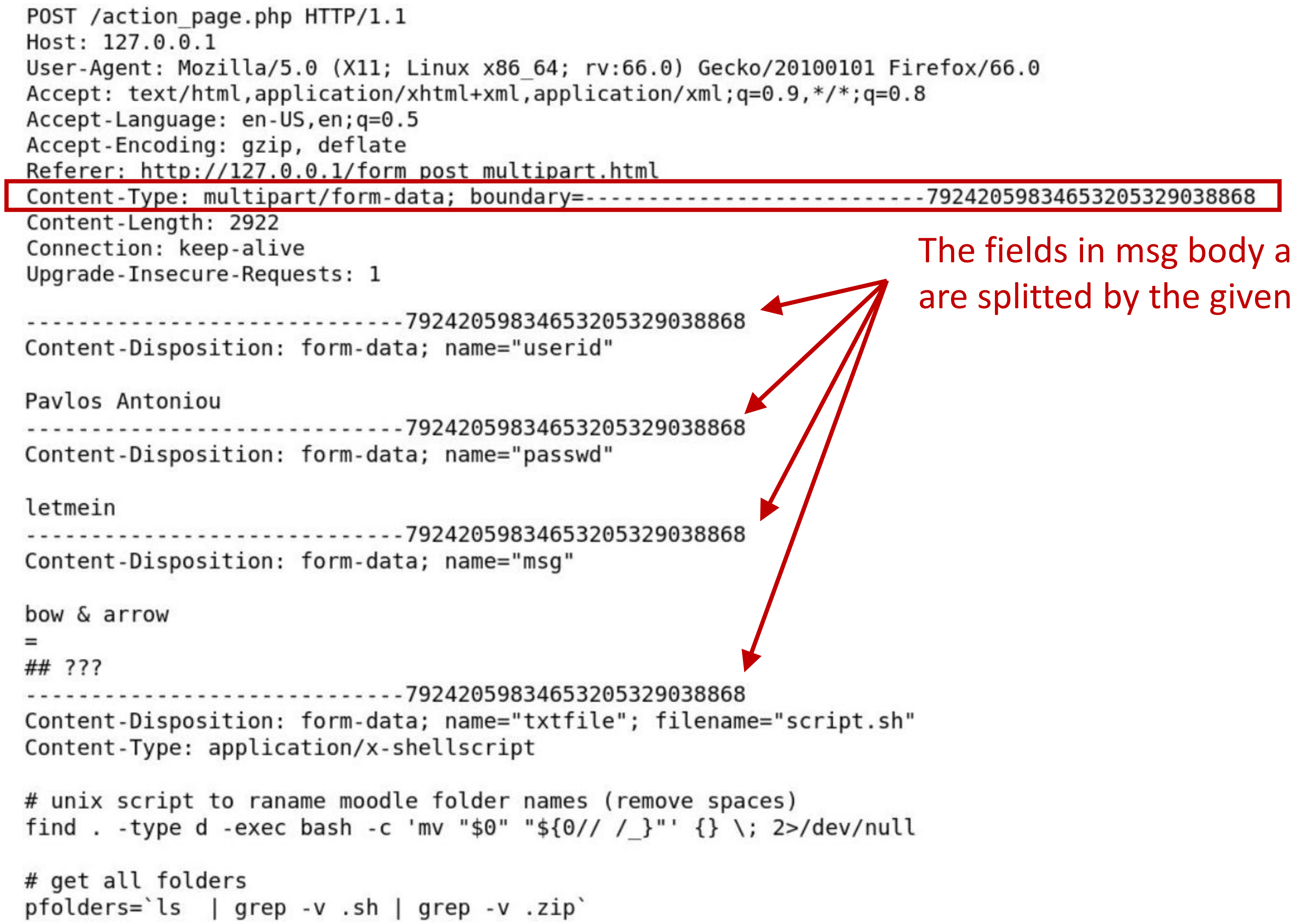
Browser
Sends POST msg via HTML form submission

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=---------------------------79242059834653205329038868
Content-Length: 2922
Connection: keep-alive
Upgrade-Insecure-Requests: 1

-----------------------------79242059834653205329038868
Content-Disposition: form-data; name="userid"

Pavlos Antoniou
-----------------------------79242059834653205329038868
Content-Disposition: form-data; name="passwd"

letmein
-----------------------------79242059834653205329038868
Content-Disposition: form-data; name="msg"

bow & arrow
=
## ???
-----------------------------79242059834653205329038868
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

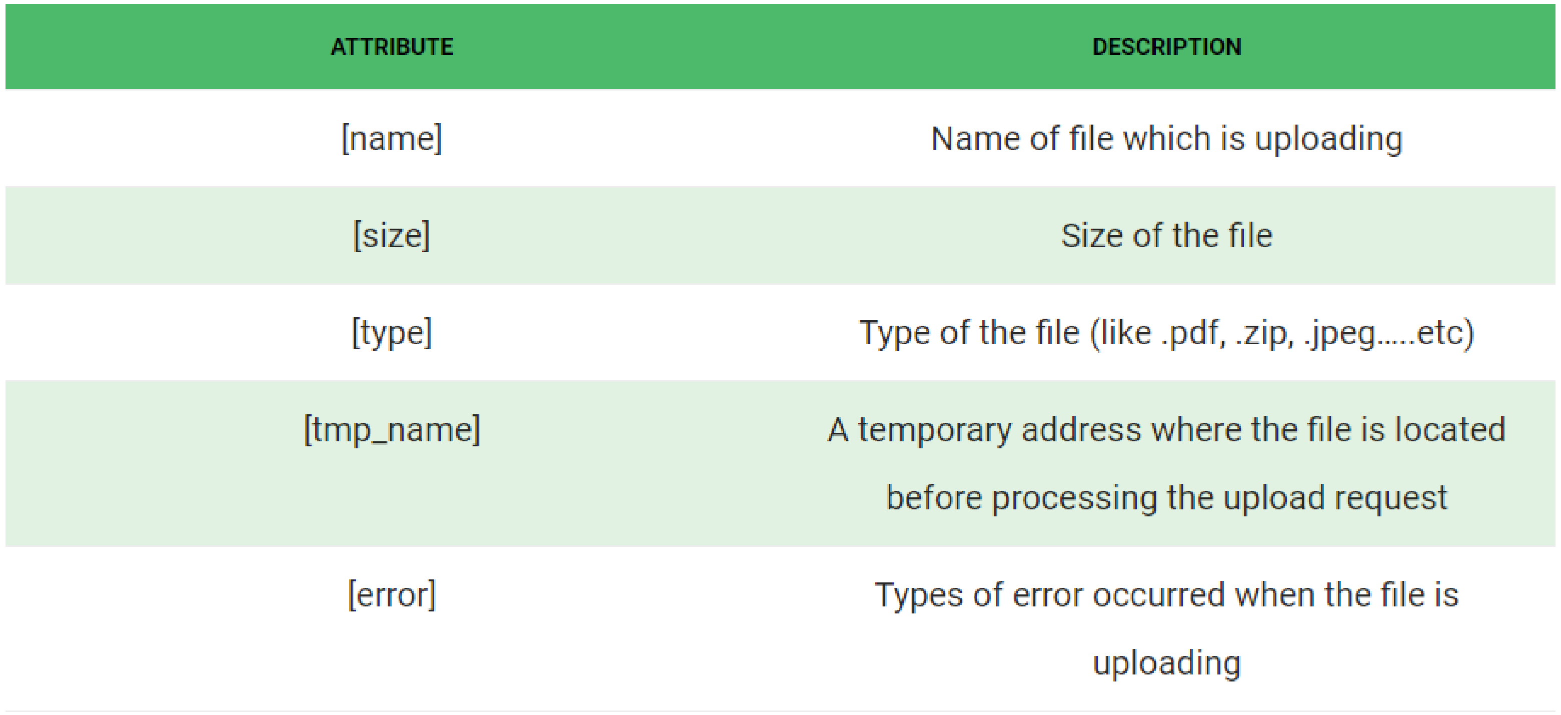
action_page.php

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

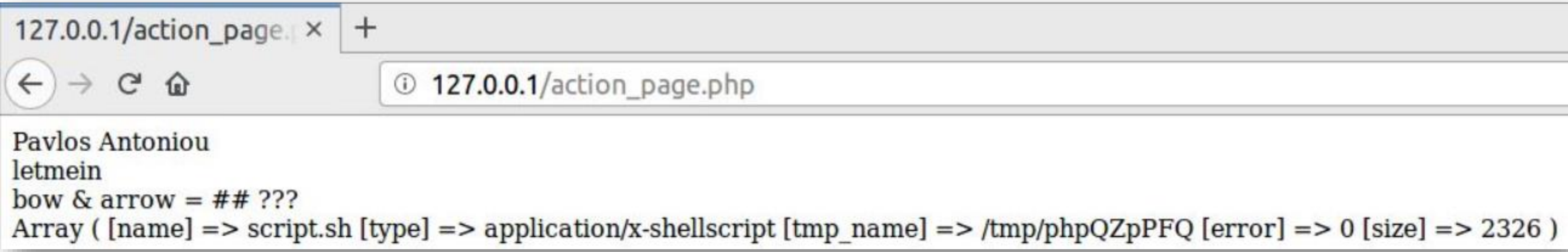


ATTRIBUTE	DESCRIPTION
[name]	Name of file which is uploading
[size]	Size of the file
[type]	Type of the file (like .pdf, .zip, .jpeg.....etc)
[tmp_name]	A temporary address where the file is located before processing the upload request
[error]	Types of error occurred when the file is uploading

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:



```
Pavlos Antoniou
letmein
bow & arrow = ## ???
Array ( [name] => script.sh [type] => application/x-shellscript [tmp_name] => /tmp/phpQZpPFQ [error] => 0 [size] => 2326 )
```

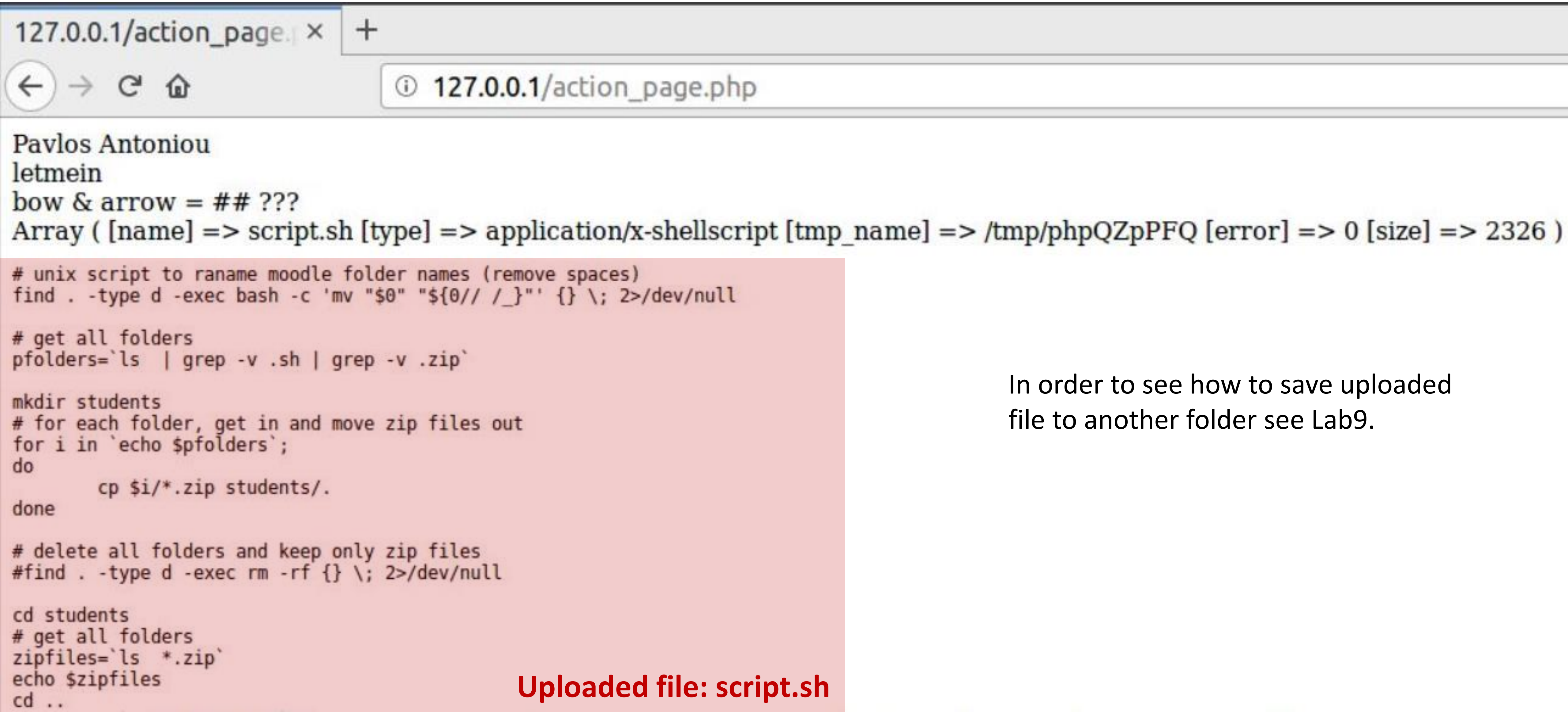
Update php to print uploaded file contents

action_page.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.



```
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

In order to see how to save uploaded file to another folder see Lab9.

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. Function collects form data (using id of each field), and then creates XMLHttpRequest/Fetch API 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 – 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.

Send POST msg via JavaScript – Fetch API

JS

```
function onClick() {
      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;

      fetch('action_page.php', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
      })
      .then(
        response => { // handle the response
          // Parse response as JSON (no need to call JSON.parse())
          response.json().then(
            data => {
                  console.log(data);
                }
          );
        } // end of response
      )
      .catch( error => { // handle the error
          console.log('Error: ', error);
      });
}
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.

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);
```

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.

Update php to collect data in POST msg body

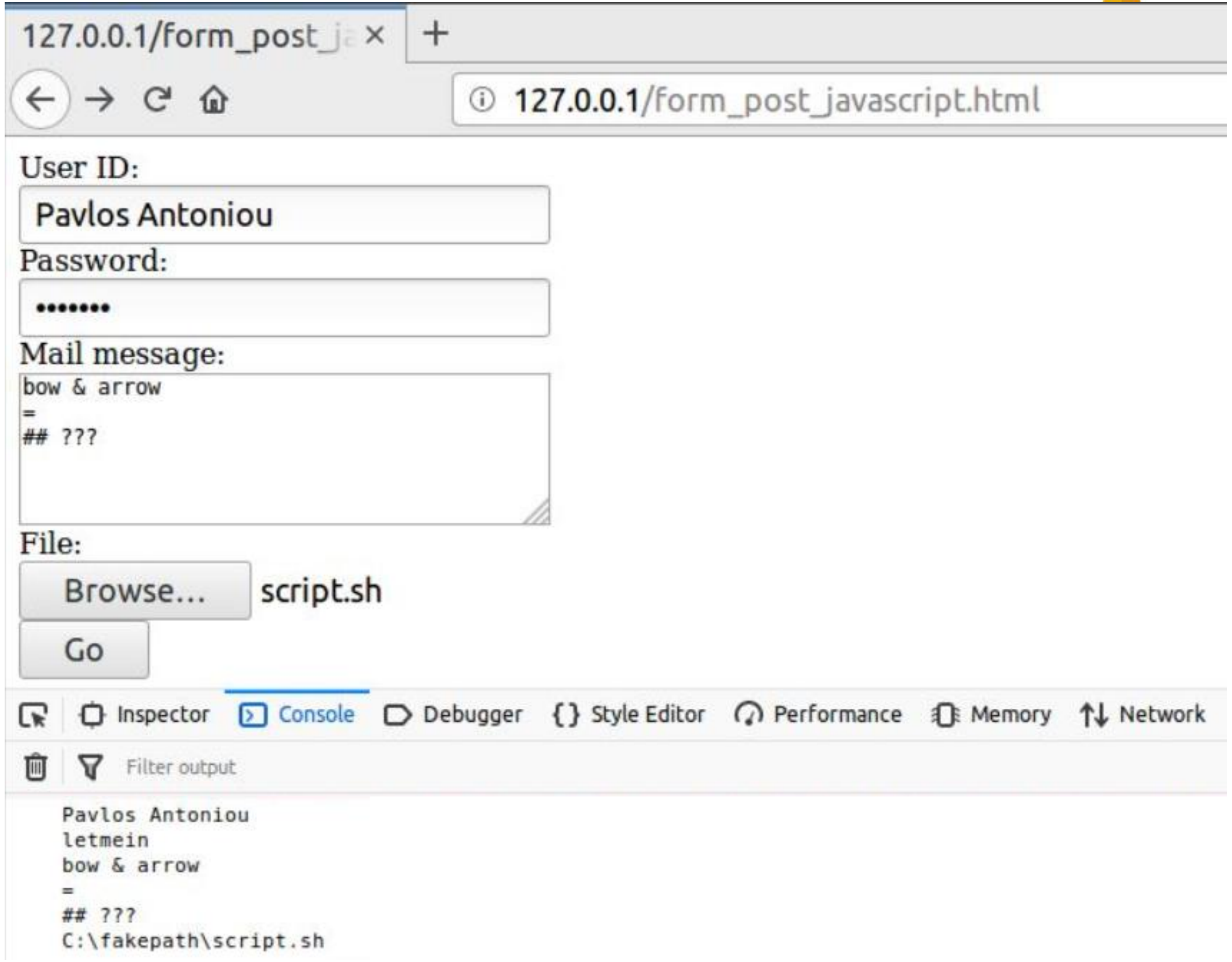
action_page.php

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



127.0.0.1/form_post_javascript.html

User ID: Pavlos Antoniou

Password: •••••••

Mail message:

```
bow & arrow
=
## ???
```

File: Browse... script.sh

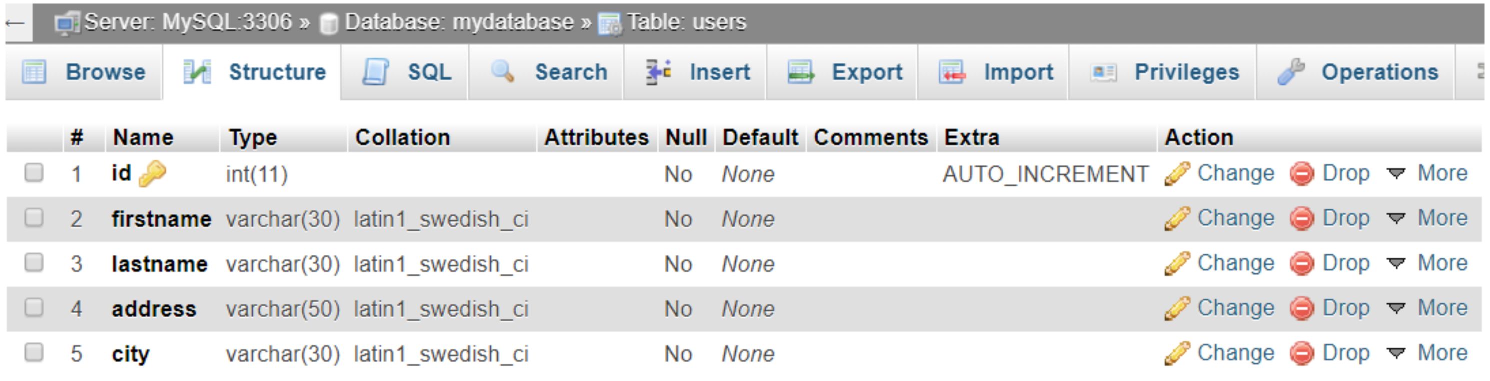
Go

```
Pavlos Antoniou
letmein
bow & arrow
=
## ???
C:\fakepath\script.sh
```

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)



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

- 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
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " .
    mysqli_error($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
    $conn = mysqli_connect("localhost", "root", "") or die("Could not connect: " .
    mysqli_error($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 while)

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

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

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

```php
$users = array();                                                                            PHP
while($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
```

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)

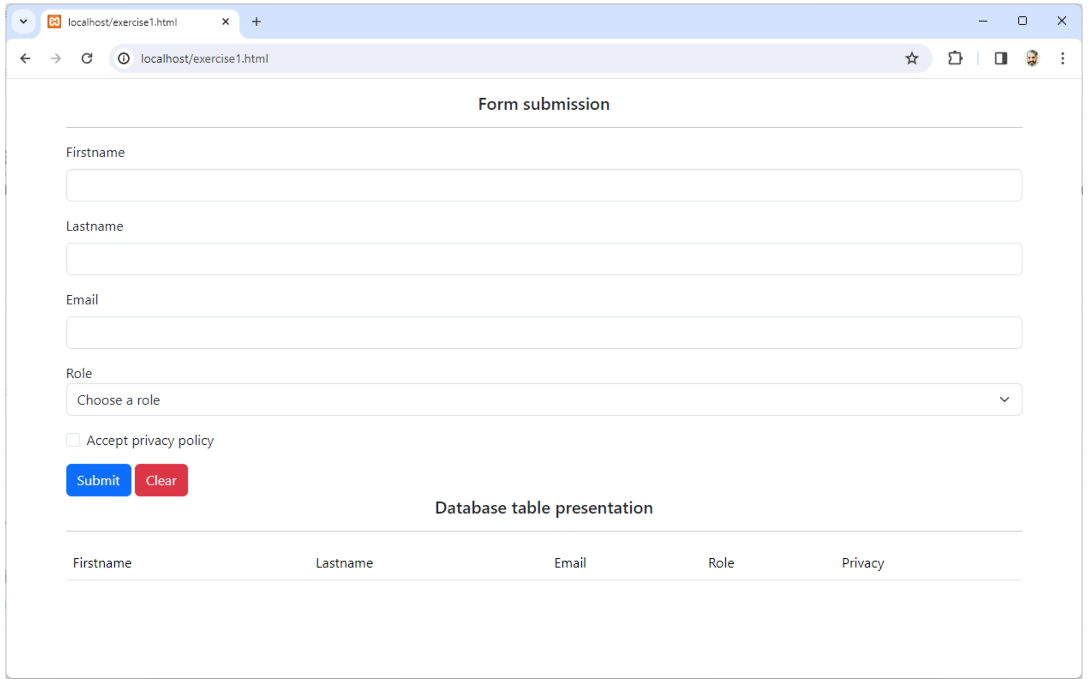
Set header and response code

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 1

- Use the exercise1.html and exercise1.js given in course website to finalize the implementation of a user registration system. Place both files under the C:\xampp\htdocs folder and access the web app via http://localhost/exercise1.html



localhost/exercise1.html

Form submission

Firstname

Lastname

Email

Role

Choose a role

Accept privacy policy

Submit Clear

Database table presentation

Firstname	Lastname	Email	Role	Privacy

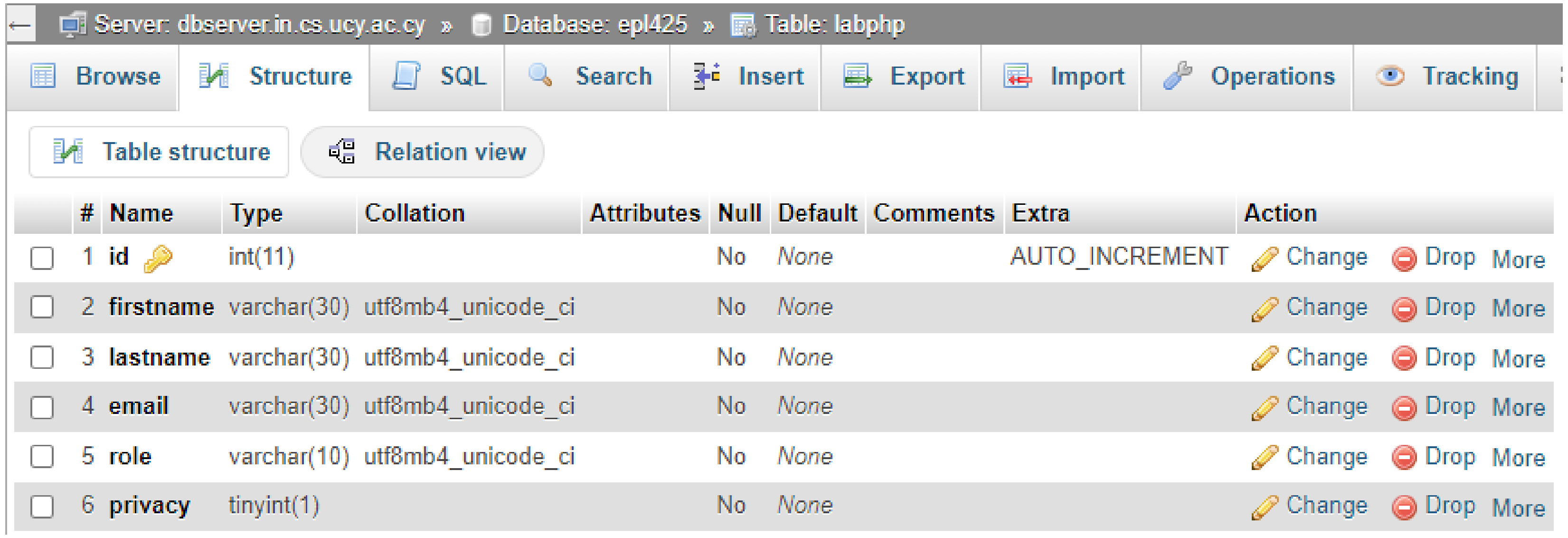
Exercise 1

- The web app features a bootstrap-powered form to collect user data (firstname, lastname, email, role, acceptance of privacy policy)
- When the submit button is clicked, sendData() function is called and sends form data via AJAX call (Fetch API, POST message) as JSON string to exercise1.php file which stores data into a database
- On the bottom of the webpage there is a table that displays user information. Data is retrieved by the receiveData() function via AJAX call (Fetch API, GET message) to exercise1.php. This function is called (a) everytime the webpage is loaded and (b) after data submission.
- HTML + JavaScript files are complete. No need to modify.

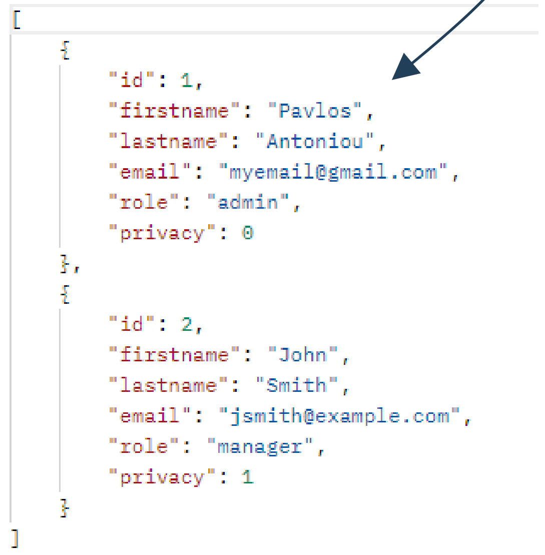
Exercise 1 – What to implement

- Create exercise1.php file (in the same folder) which accepts:

1. GET message to SELECT all user data from `labphp` table of `epl425` database (connect to dbserver.in.cs.ucy.ac.cy using username & password given in HW1) and return as an array of JSON objects along with 200 OK message having Content-Type: application/json header



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	id	int(11)			No	None		AUTO_INCREMENT	Change	Drop	More
☐	2	firstname	varchar(30)	utf8mb4_unicode_ci		No	None			Change	Drop	More
☐	3	lastname	varchar(30)	utf8mb4_unicode_ci		No	None			Change	Drop	More
☐	4	email	varchar(30)	utf8mb4_unicode_ci		No	None			Change	Drop	More
☐	5	role	varchar(10)	utf8mb4_unicode_ci		No	None			Change	Drop	More
☐	6	privacy	tinyint(1)			No	None			Change	Drop	More



```
[
    {
        "id": 1,
        "firstname": "Pavlos",
        "lastname": "Antoniou",
        "email": "myemail@gmail.com",
        "role": "admin",
        "privacy": 0
    },
    {
        "id": 2,
        "firstname": "John",
        "lastname": "Smith",
        "email": "jsmith@example.com",
        "role": "manager",
        "privacy": 1
    }
]
```

Exercise 1 – What to implement

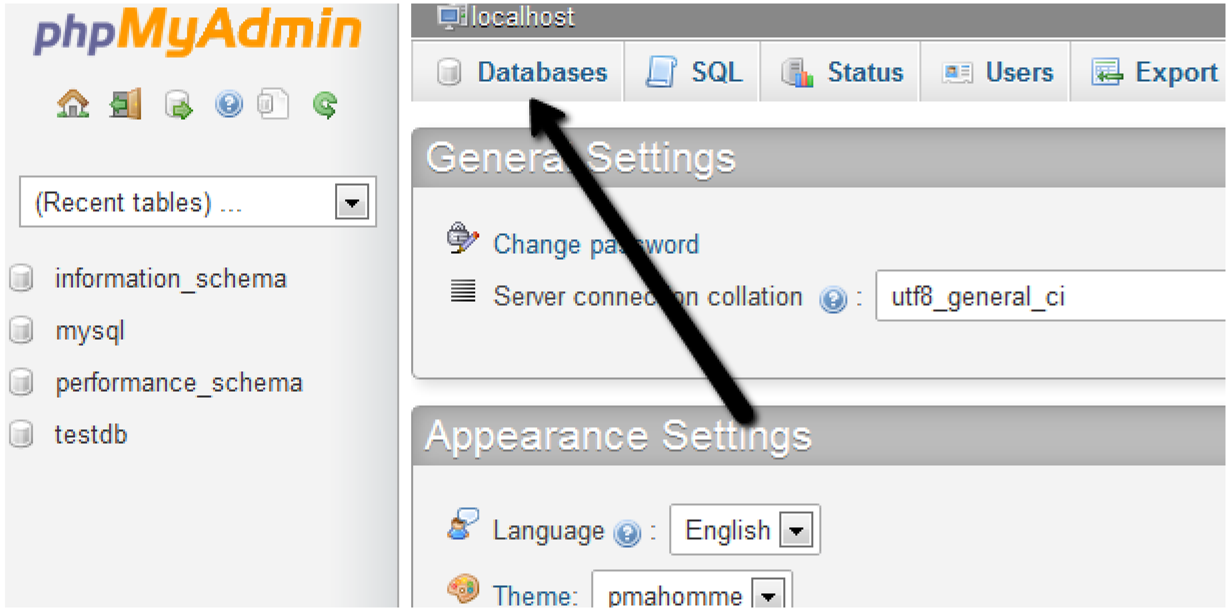
2. POST message with user data (JSON) and INSERTS data into `labphp` table – after successful insertion 201 Created message is returned with Content-Type: application/json header

Set up MySQL DB & table using phpmyadmin

University of Cyprus
Department of Computer Science

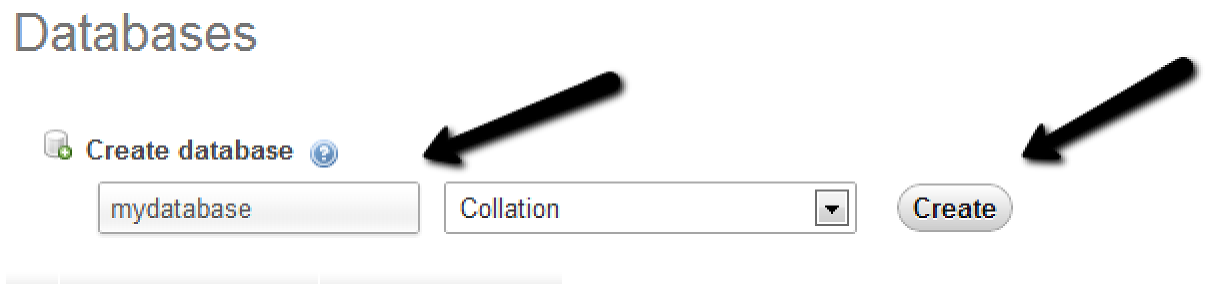
Create DB in phpMyAdmin - 1

- 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**



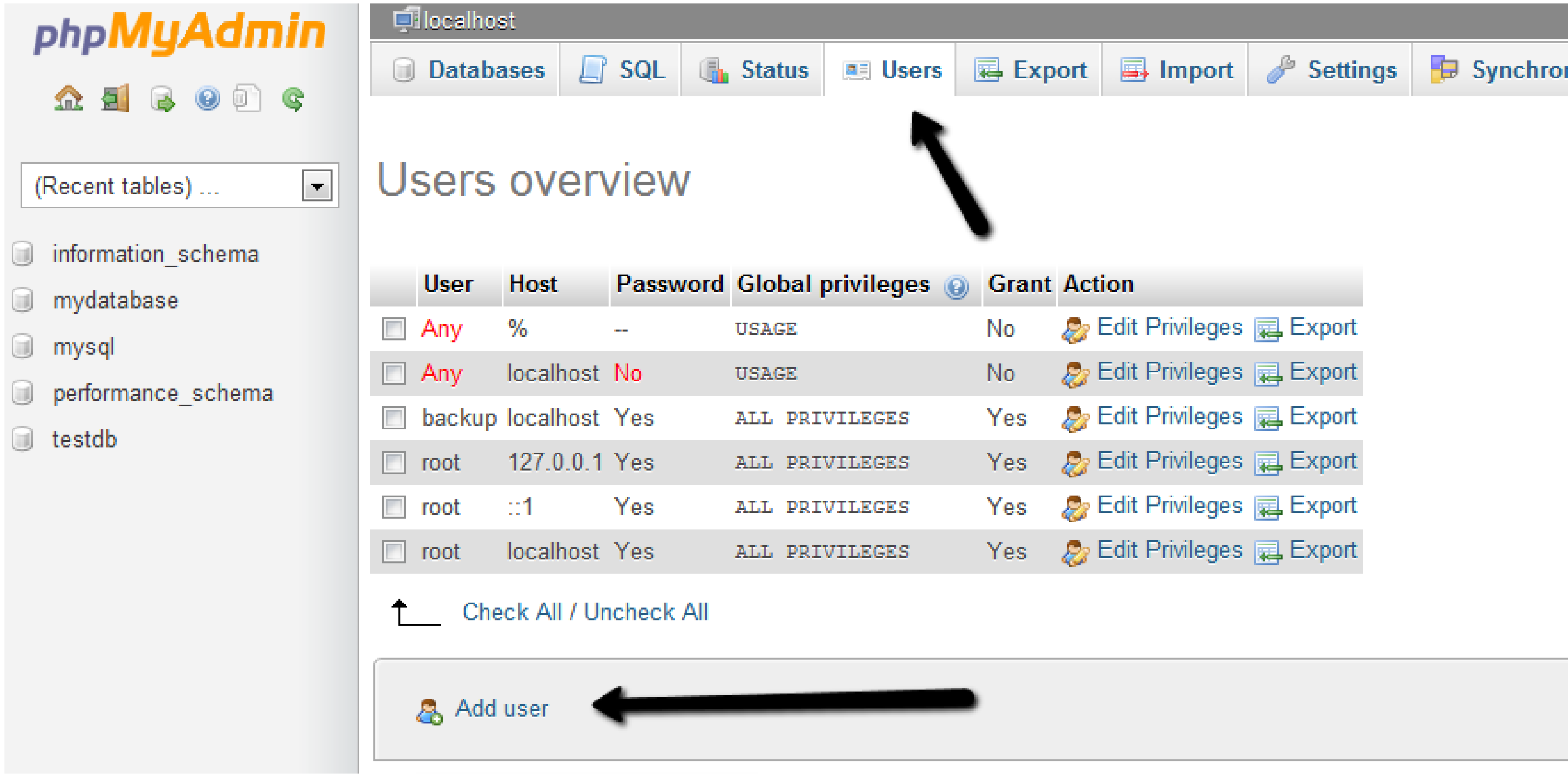
Create DB in phpMyAdmin - 2

- 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

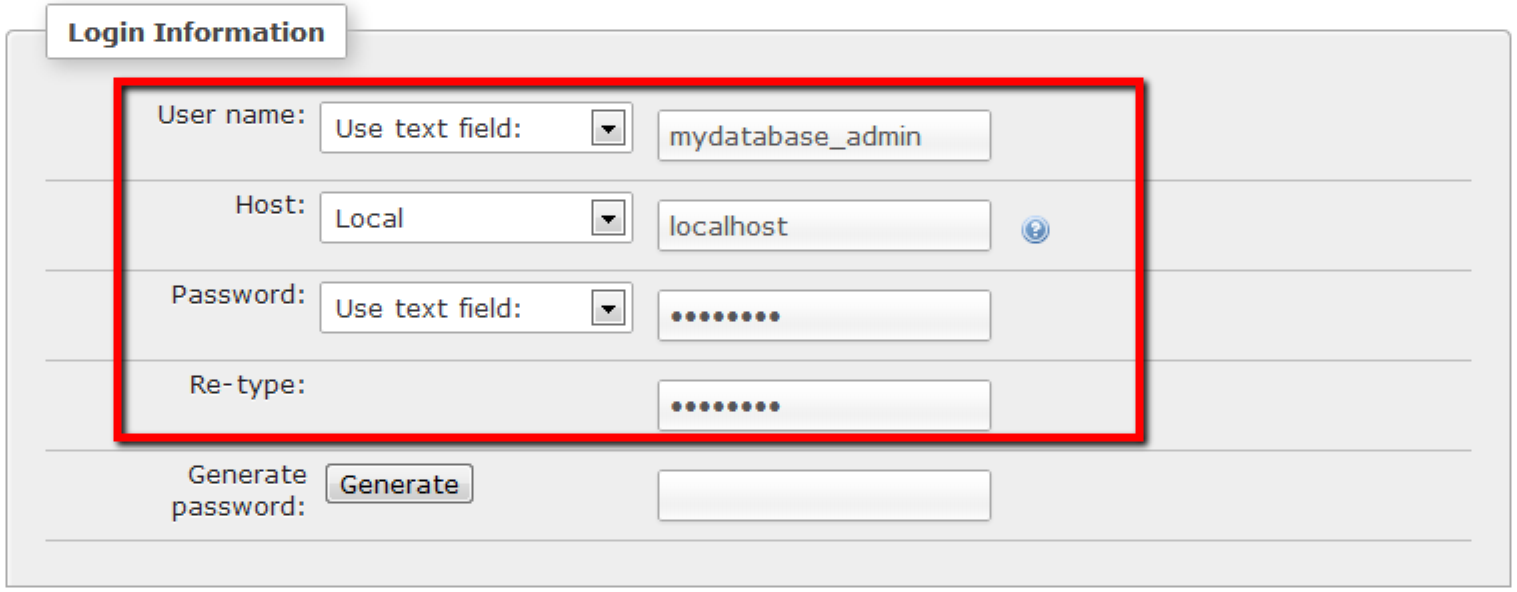
- Your database will now be visible on the right hand side under the list of available databases. To setup a new user login to access this database, click on **Users** (or User account) in the main menu. Choose the **Add User** option under the list of available server users.



	User	Host	Password	Global privileges	Grant	Action
☐	Any	%	--	USAGE	No	Edit Privileges Export
☐	Any	localhost	No	USAGE	No	Edit Privileges Export
☐	backup	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	::1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export

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.

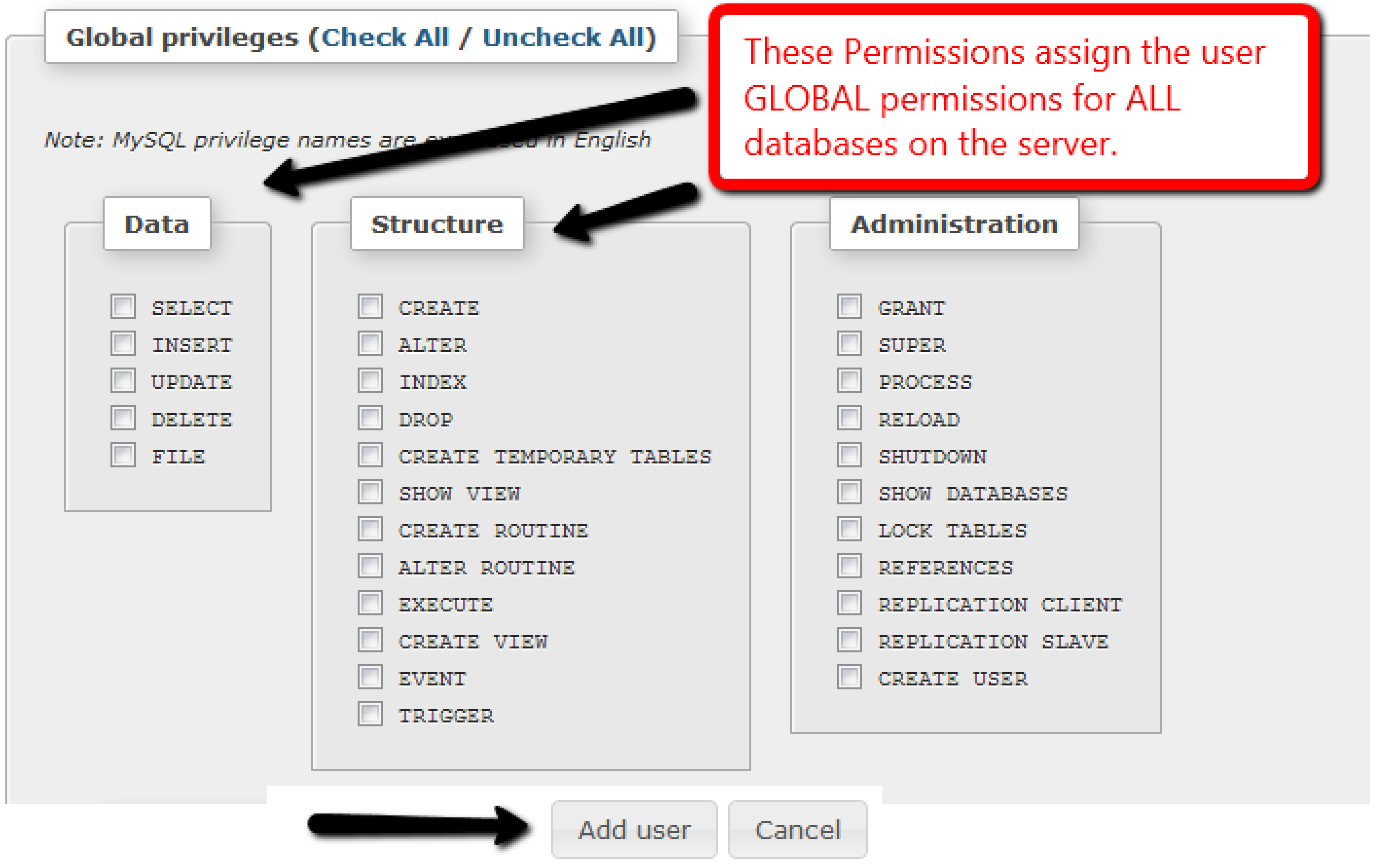


Login Information

User name:	Use text field:	mydatabase_admin
Host:	Local	localhost
Password:	Use text field:	•••••••••
Re-type:		•••••••••
Generate password:	Generate	

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.



Global privileges (Check All / Uncheck All)

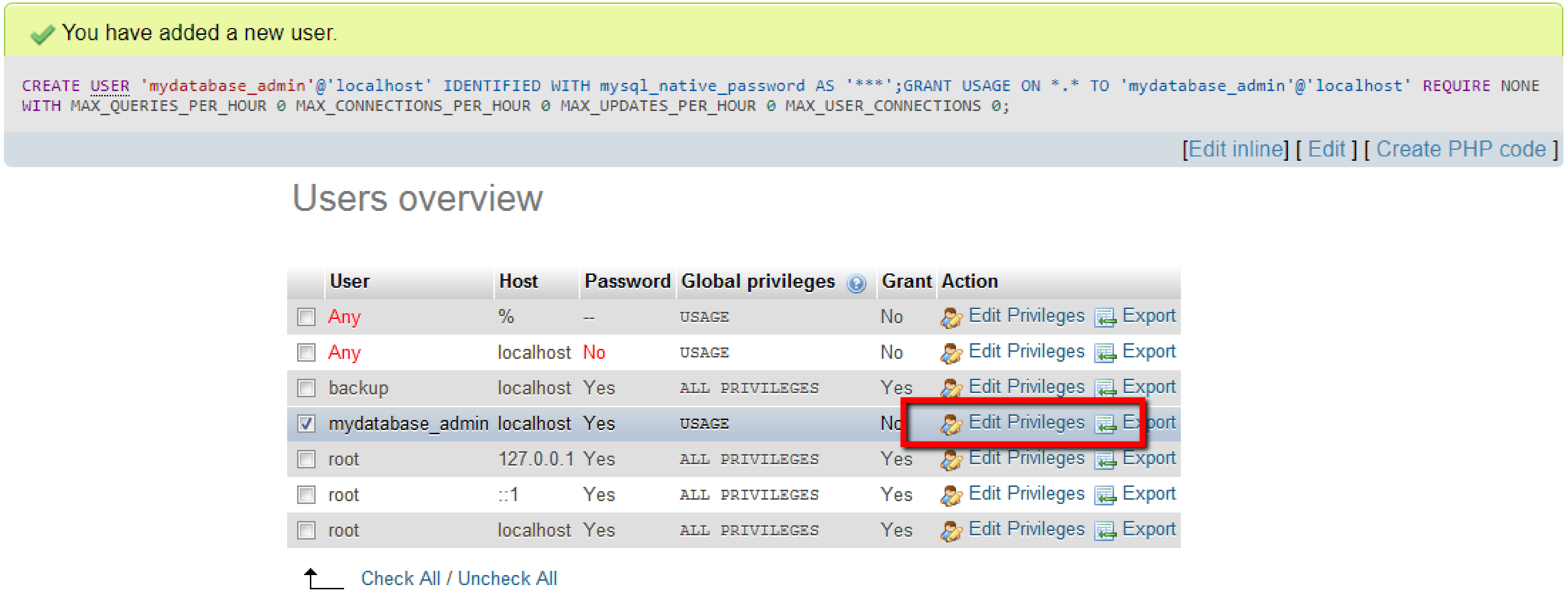
Note: MySQL privilege names are expressed in English

These Permissions assign the user GLOBAL permissions for ALL databases on the server.

Data	Structure	Administration
☐ SELECT	☐ CREATE	☐ GRANT
☐ INSERT	☐ ALTER	☐ SUPER
☐ UPDATE	☐ INDEX	☐ PROCESS
☐ DELETE	☐ DROP	☐ RELOAD
☐ FILE	☐ CREATE TEMPORARY TABLES	☐ SHUTDOWN
	☐ SHOW VIEW	☐ SHOW DATABASES
	☐ CREATE ROUTINE	☐ LOCK TABLES
	☐ ALTER ROUTINE	☐ REFERENCES
	☐ EXECUTE	☐ REPLICATION CLIENT
	☐ CREATE VIEW	☐ REPLICATION SLAVE
	☐ EVENT	☐ CREATE USER
	☐ TRIGGER	

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.



✓ You have added a new user.

```
CREATE USER 'mydatabase_admin'@'localhost' IDENTIFIED WITH mysql_native_password AS '***';GRANT USAGE ON *.* TO 'mydatabase_admin'@'localhost' REQUIRE NONE
WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

[Edit inline] [Edit] [Create PHP code]

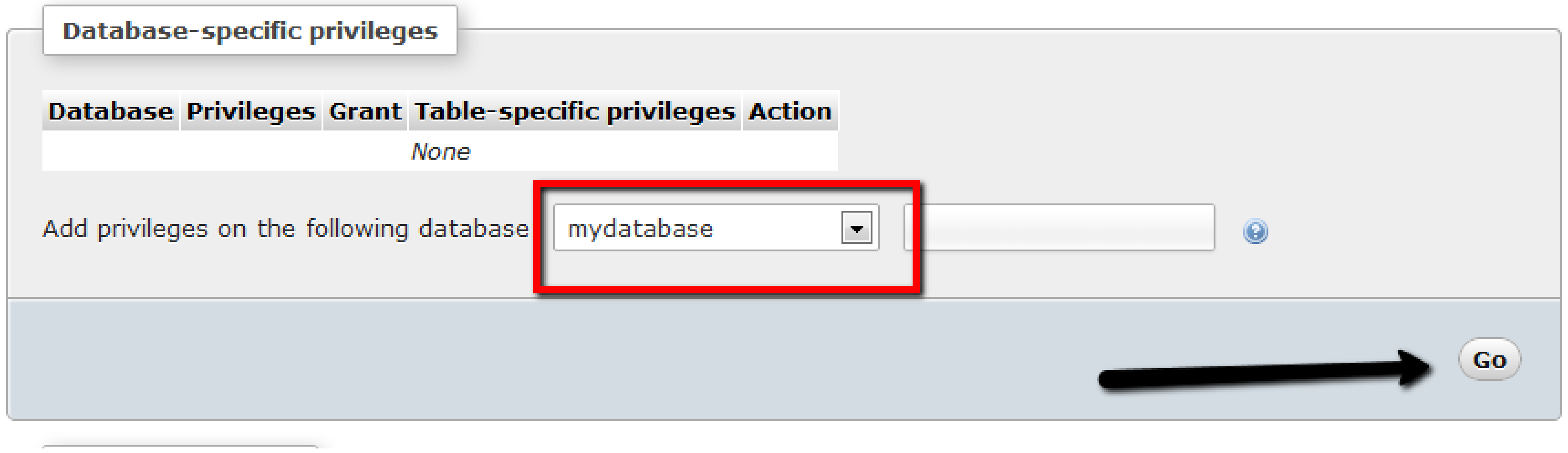
Users overview

	User	Host	Password	Global privileges	Grant	Action
☐	Any	%	--	USAGE	No	Edit Privileges Export
☐	Any	localhost	No	USAGE	No	Edit Privileges Export
☐	backup	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
✓	mydatabase_admin	localhost	Yes	USAGE	No	Edit Privileges Export
☐	root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	::1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
☐	root	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export

Check All / Uncheck All

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**.



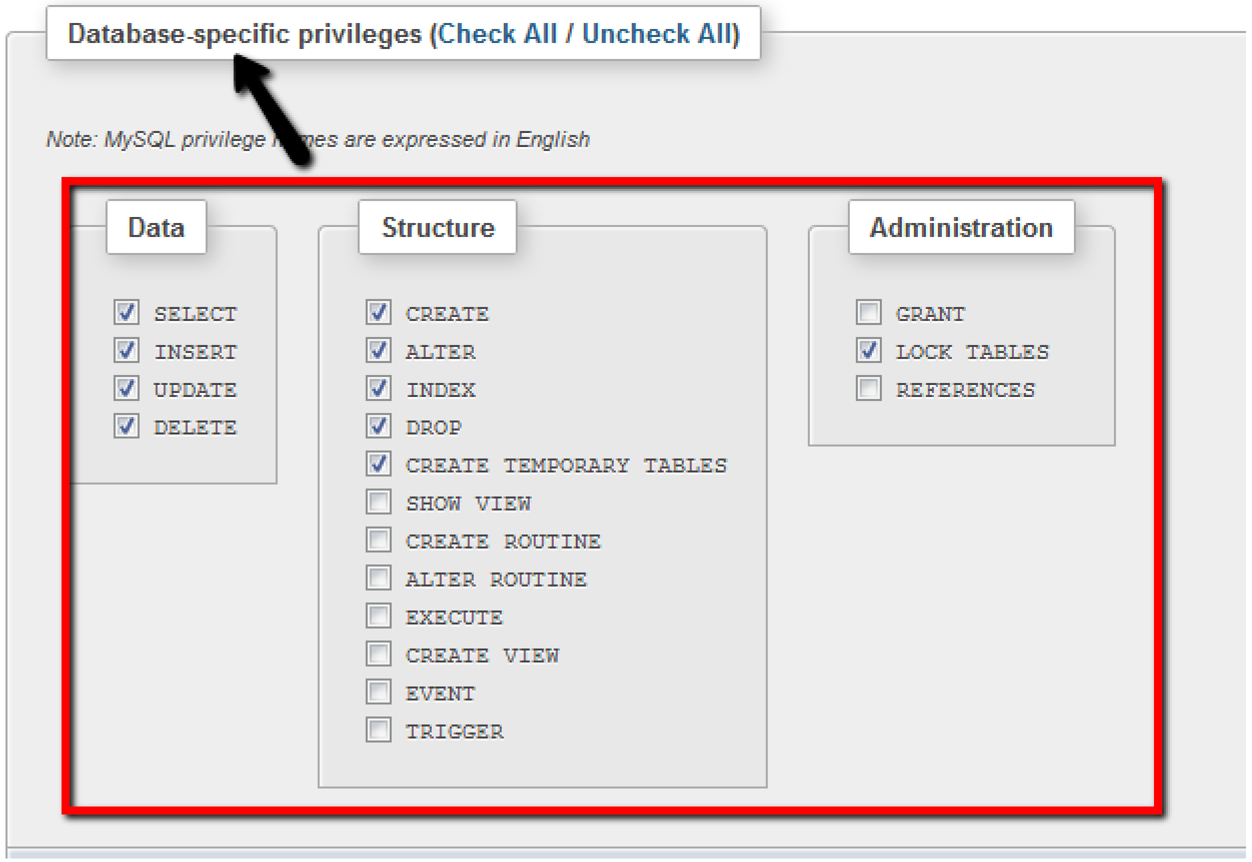
Database-specific privileges

Database	Privileges	Grant	Table-specific privileges	Action
None				

Add privileges on the following database: mydatabase

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.



Database-specific privileges (Check All / Uncheck All)

Note: MySQL privilege names are expressed in English

Data

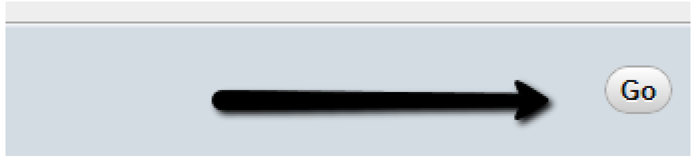
- [✓] SELECT
- [✓] INSERT
- [✓] UPDATE
- [✓] DELETE

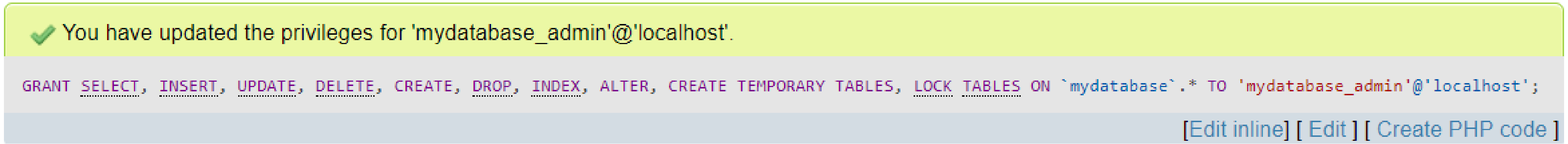
Structure

- [✓] CREATE
- [✓] ALTER
- [✓] INDEX
- [✓] DROP
- [✓] CREATE TEMPORARY TABLES
- [ ] SHOW VIEW
- [ ] CREATE ROUTINE
- [ ] ALTER ROUTINE
- [ ] EXECUTE
- [ ] CREATE VIEW
- [ ] EVENT
- [ ] TRIGGER

Administration

- [ ] GRANT
- [✓] LOCK TABLES
- [ ] REFERENCES





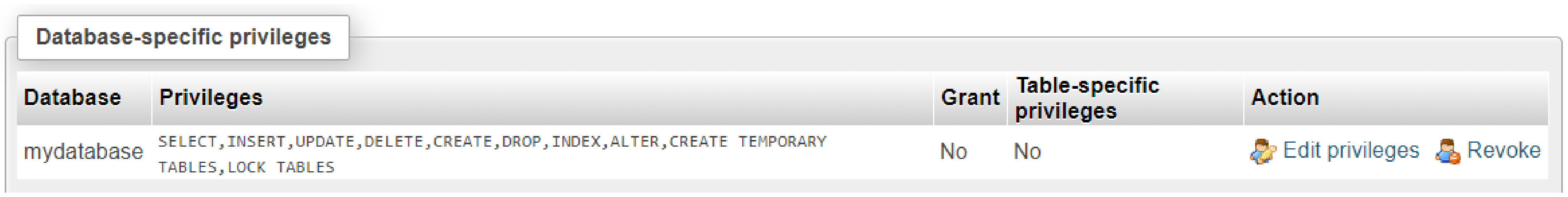
You have updated the privileges for 'mydatabase_admin'@'localhost'.

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES ON `mydatabase`.* TO 'mydatabase_admin'@'localhost';
```

[Edit inline] [Edit] [Create PHP code]

Create DB in phpMyAdmin - 9

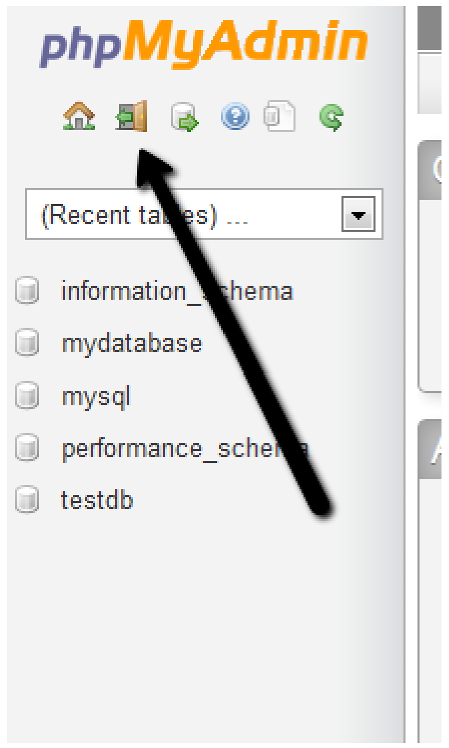
- If you click on the users Edit Privileges option now, you will see that new privileges for the specific database are now listed as belonging to the user.



Database-specific privileges

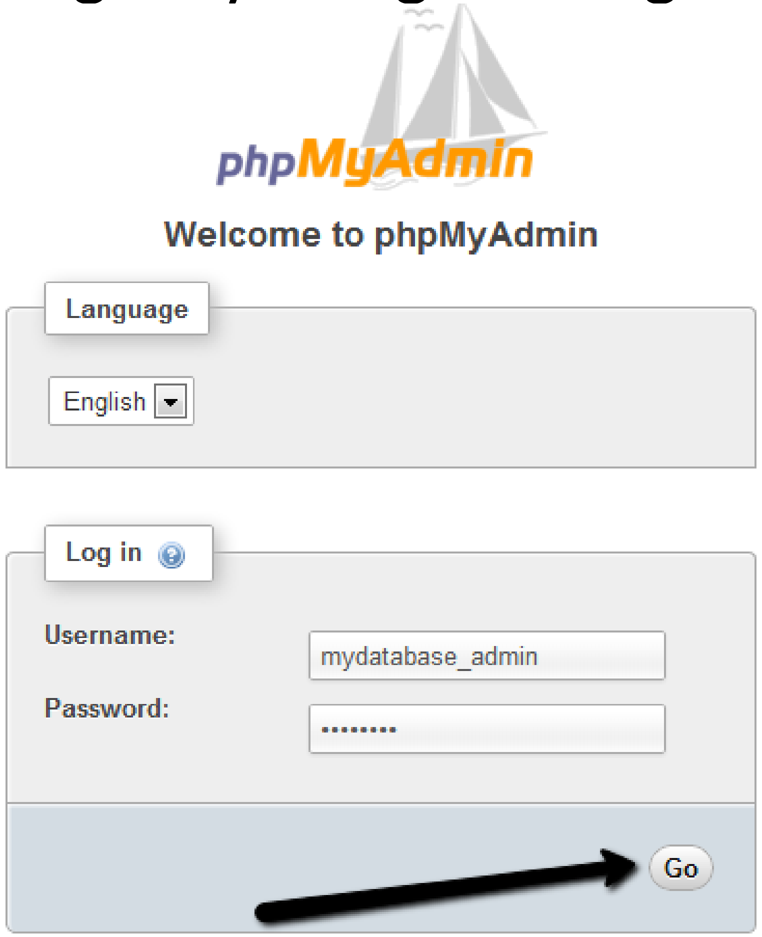
Database	Privileges	Grant	Table-specific privileges	Action
mydatabase	SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,INDEX,ALTER,CREATE TEMPORARY TABLES,LOCK TABLES	No	No	Edit privileges Revoke

- 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.

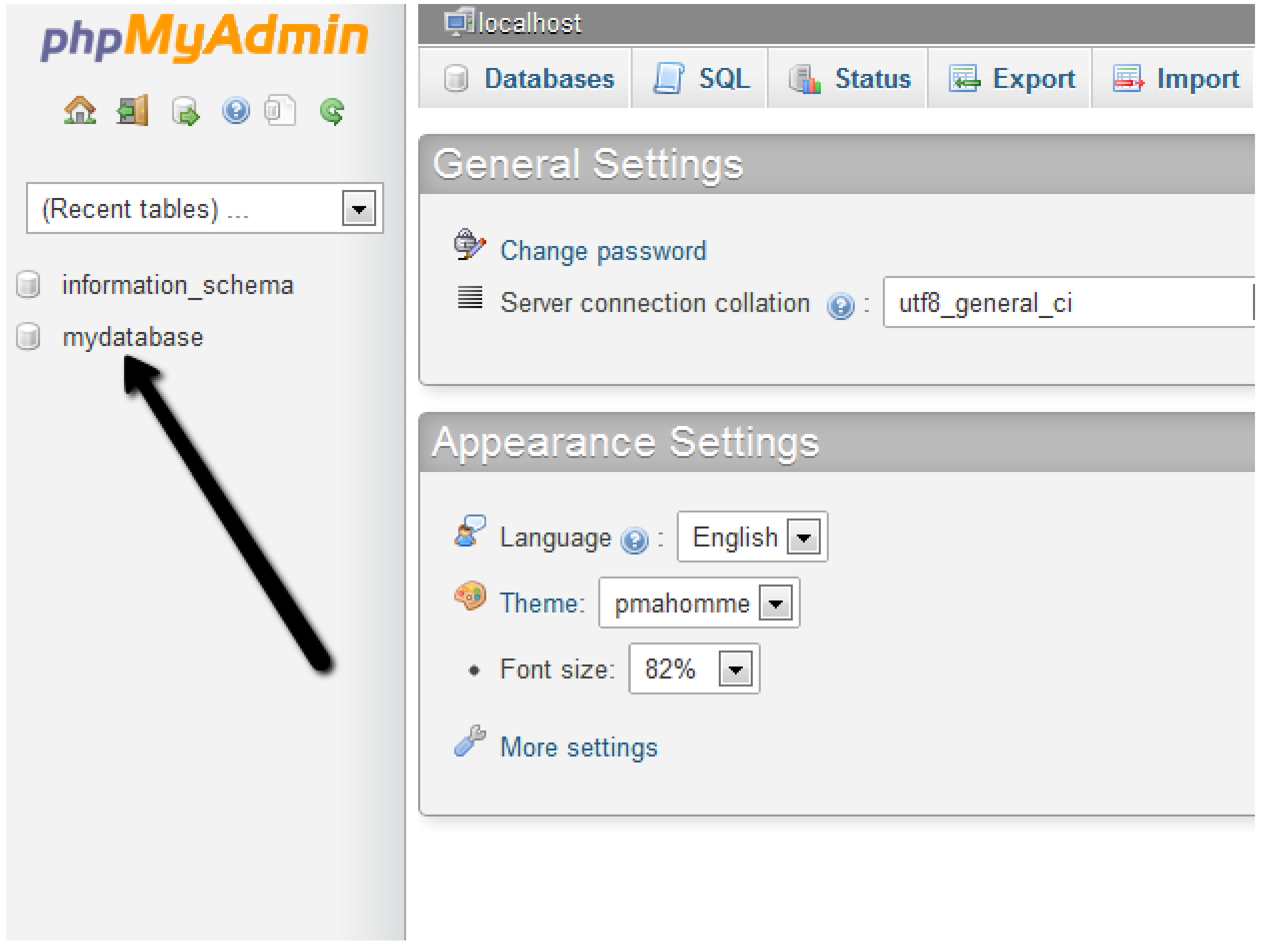


Username: mydatabase_admin

Password: ••••••••

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.



Server connection collation: utf8_general_ci

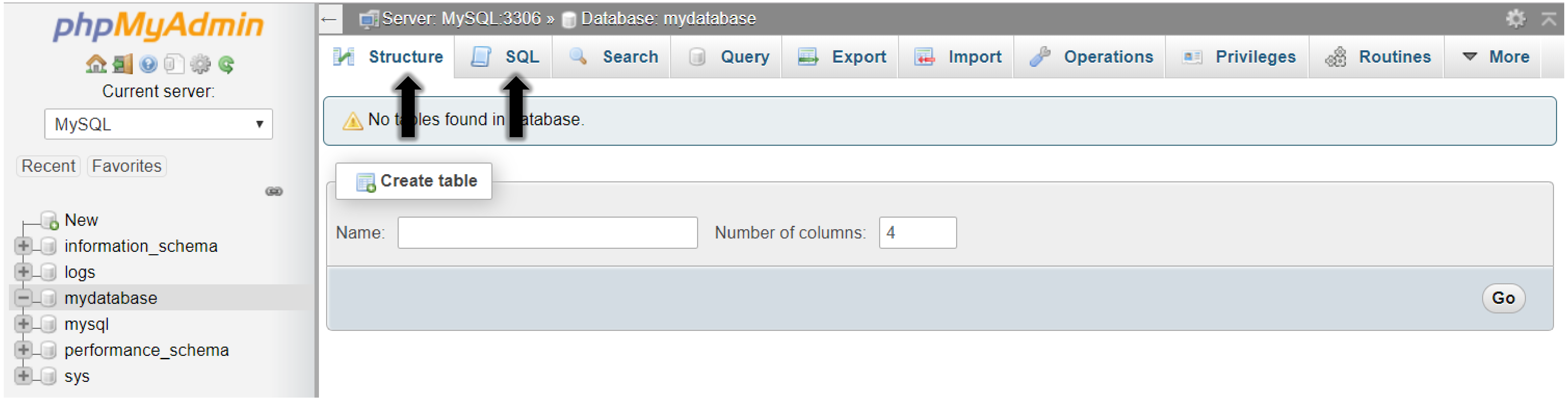
Language: English

Theme: pmahomme

Font size: 82%

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.



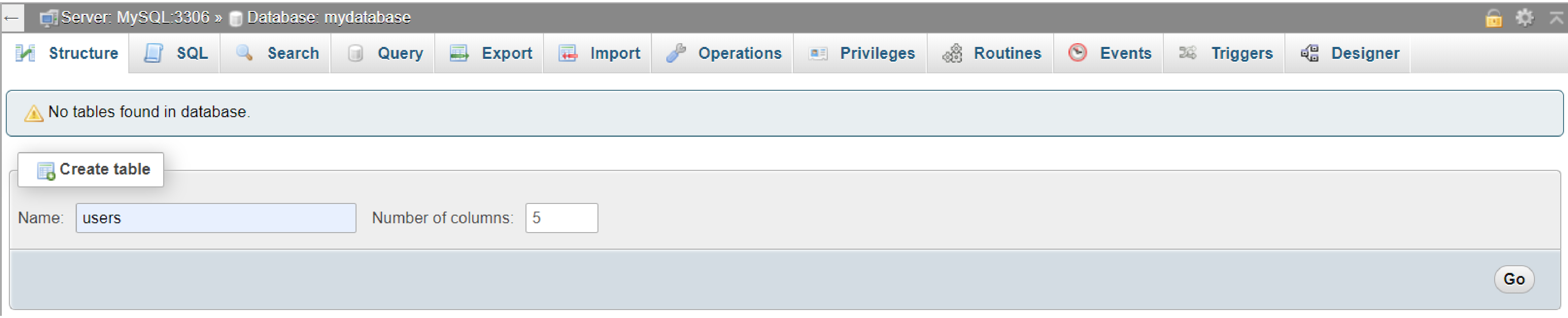
No t les found in atabase.

Name: Number of columns: 4

- 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**



No tables found in database.

Name: users

Number of columns: 5

Create Table in phpMyAdmin - 3

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

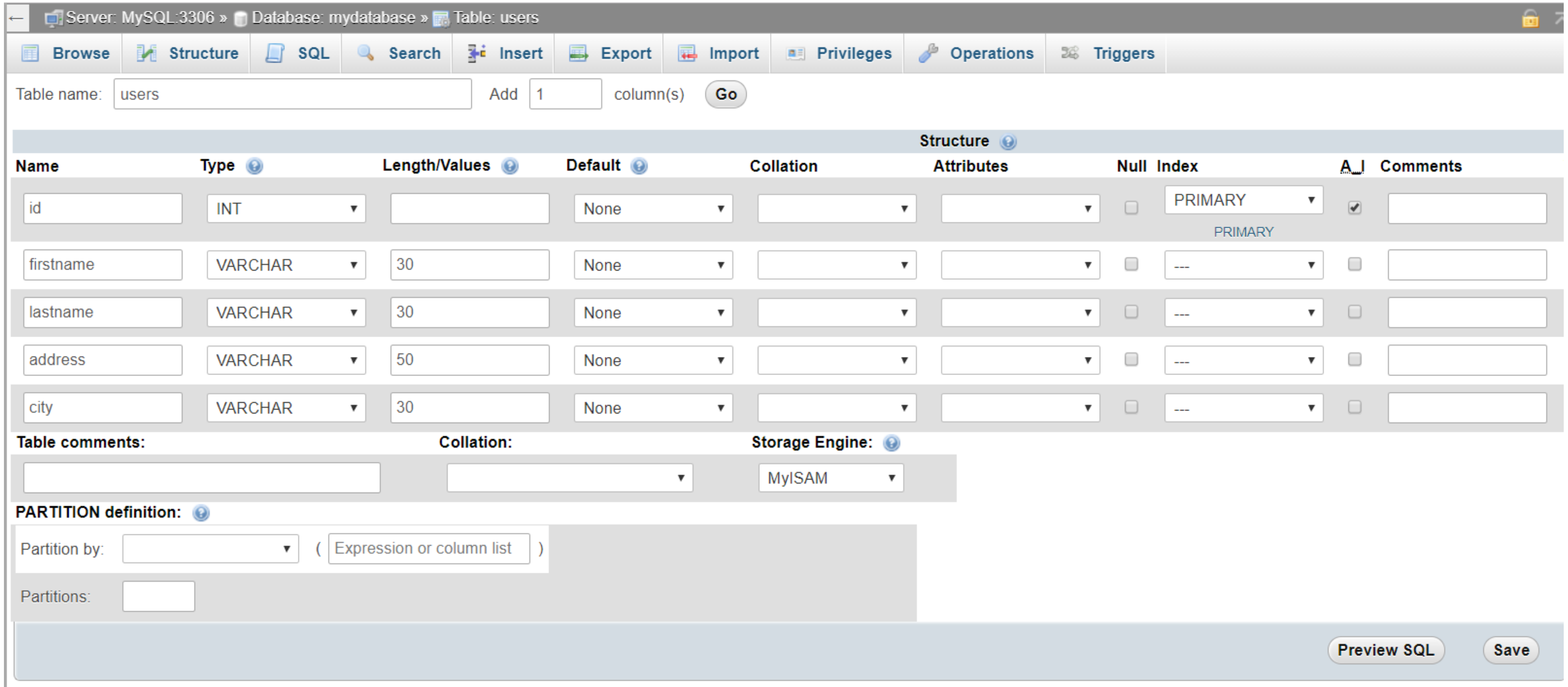


Table name: users   Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
id	INT		None			☐	PRIMARY (PRIMARY)	✓	
firstname	VARCHAR	30	None			☐	---	☐	
lastname	VARCHAR	30	None			☐	---	☐	
address	VARCHAR	50	None			☐	---	☐	
city	VARCHAR	30	None			☐	---	☐	

Table comments: Collation: Storage Engine: MyISAM

PARTITION definition:

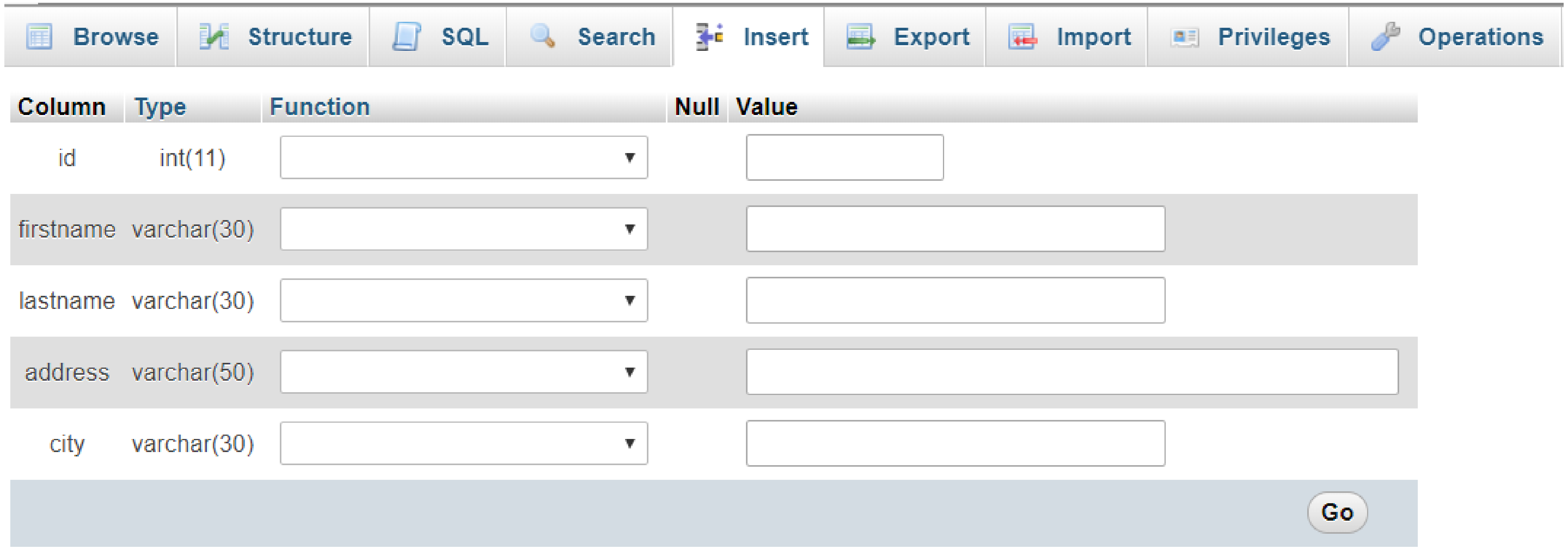
Partition by: (Expression or column list)

Partitions:

Preview SQL   Save

Create Table in phpMyAdmin - 4

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

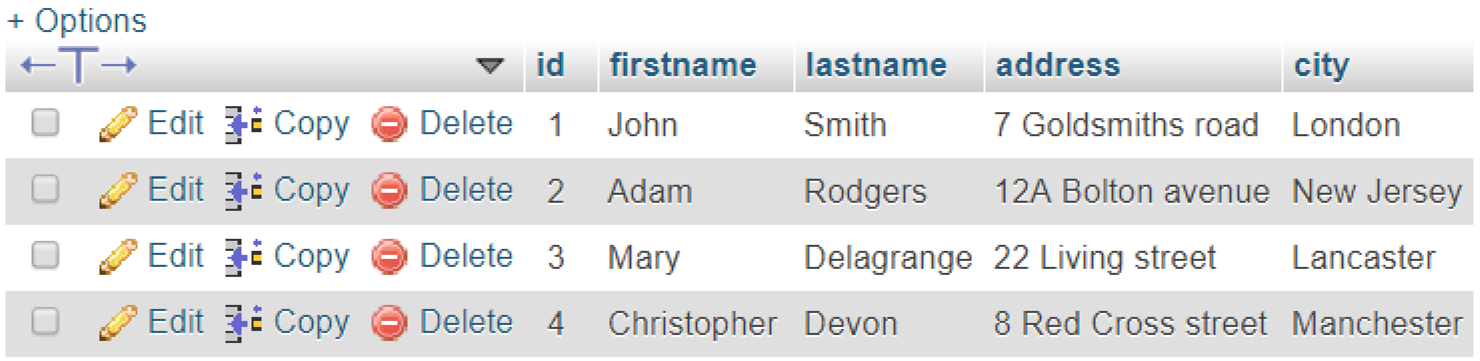


Column	Type	Function	Null	Value
id	int(11)			
firstname	varchar(30)			
lastname	varchar(30)			
address	varchar(50)			
city	varchar(30)			

  - using **SQL**

Create Table in phpMyAdmin - 5

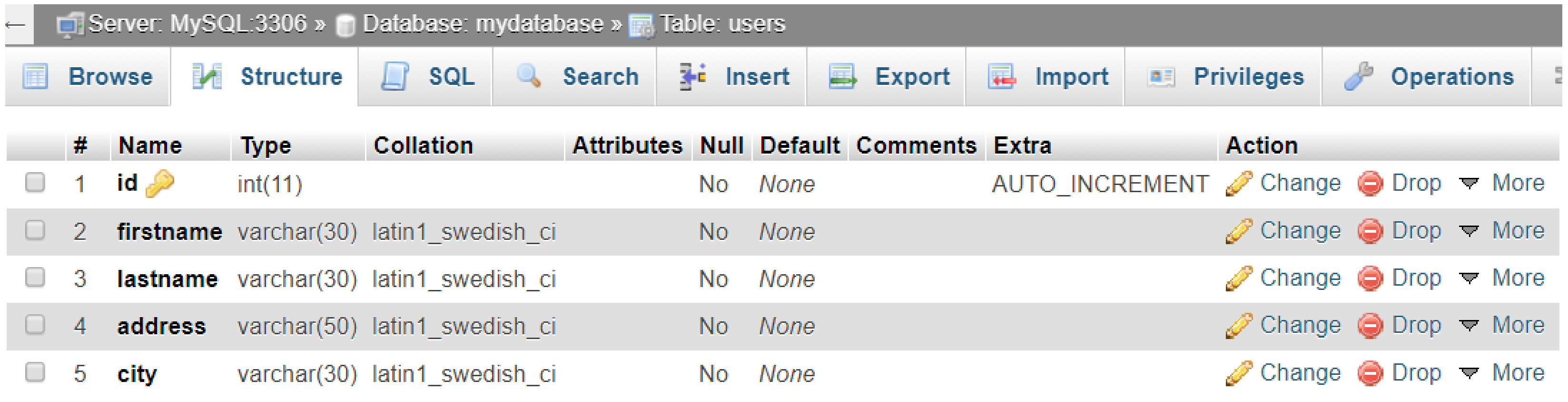
- Visit **Browse** to see all rows of the table



+ 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