

Adv Web Dev Arch

Lecture 5

Amir Amintabar, PhD

AJAX Calls

Outline (Building a Typical Web API, AJAX, intro to DB)

- Review
- - AJAX calls
 - Browser uses http methods to send request to server
- - AJAX calls demos
- - AJAX call to remote server
- --- self study for next week
- - Working with a Relational Database (CRUD)
- - Using XAMPP package installer
- - Connecting to mySQL DB using nodejs
- - Hosting Node js, MySQL on remote server

review

- Week 1: **general challenges when developing web app for various devices.** review material (over 120 slides) provided to review HTML, CSS, JS) create objects etc
- Week 2: we defined the term web app architecture, then group-discussed various architecture types in web development, learned about local storage
- Week 3: more intro to JS, setTimeout, order of execution, event loop, hoisting, var vs let, Anatomy of WebAPIs, intro to RESTful APIs, Microservices, GET vs POST
- Week 4: in js every function returns something nodejs modules

math.js
Works too

```
JS testWithModules.js > ...
1  const mo = require('./modules/math');
2  let r = 10;
3  console.log(`Hello YOURNAME. The area is: ${mo.area * r * r}`);
4
5

JS math.js
modules > JS math.js > area > area
1  const PI = 3.1416;
2  exports.area = function (r)
3  {
4    return r*r*PI;
5  };
```

1

AJAX calls

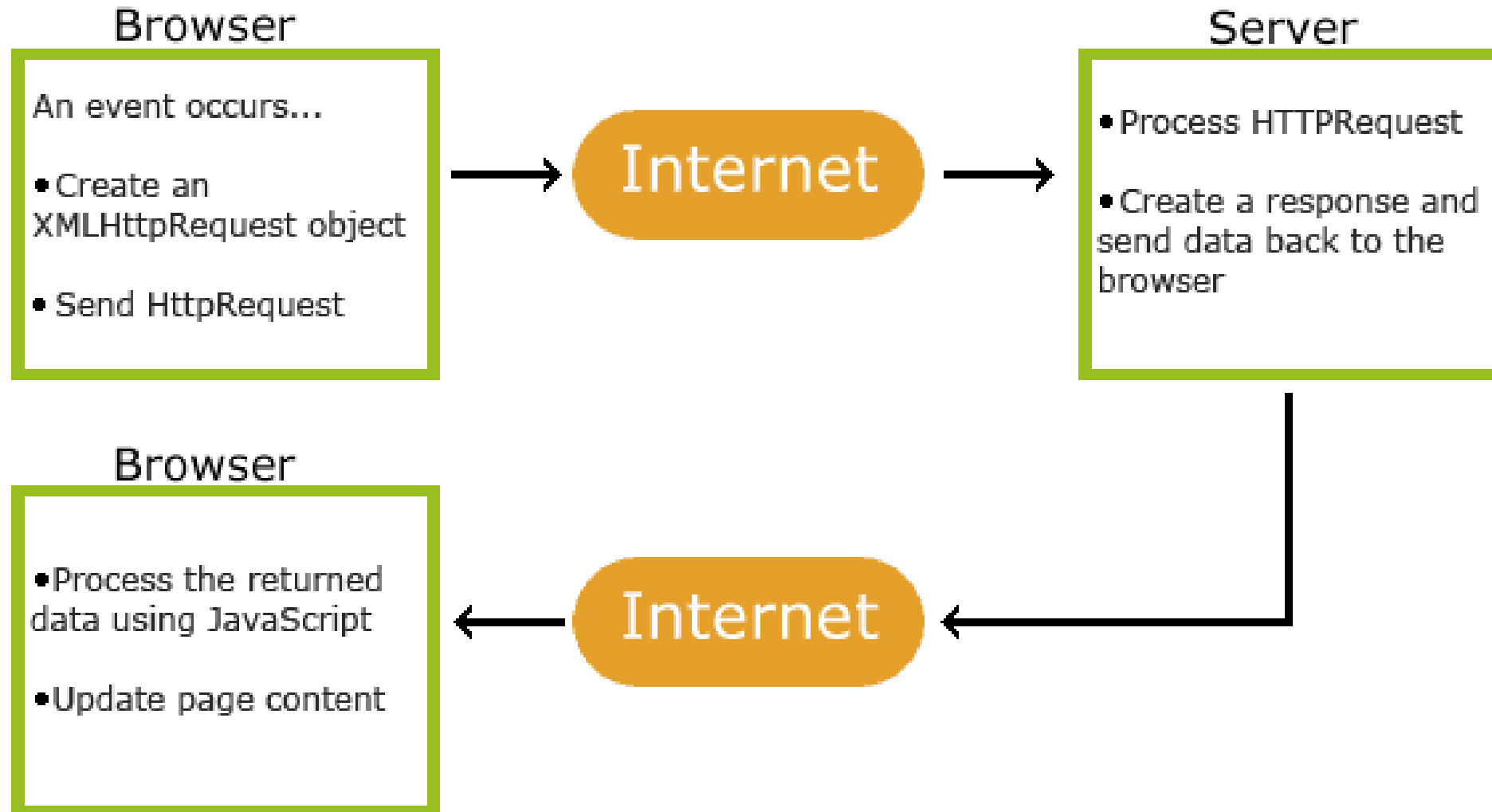
So far we sent our requests to the server
via the **browsers' address bar!**

What if we wanted send requests to servers from within our HTML
pages and not browser address bar?
e.g. to press a button and send something to the server?

You could do that either via forms
or
AJAX calls

- AJAX allows sending request to server

How AJAX Works



- Here is how we do it at the **client side** :
- **1- Create an XMLHttpRequest Object:**
 - (just like the way we created Date object)
 - **let** xhttp = **new** XMLHttpRequest();

- **2- specify the function that handles response :**

```
xhttp.onreadystatechange = function () {  
    if (this.readyState == 4 && this.status == 200) {  
        document.getElementById("demo").innerHTML =  
            this.responseText;  
    }  
};
```

Q: Who determines
what status code to
return?

status of the XMLHttpRequest :

0 (UNSENT): The XHR object has been created, but open() has not been called yet.

1 (OPENED): open() has been called.

2 (HEADERS_RECEIVED): send() has been called, and the headers of the response are available.

3 (LOADING): The response is being received. As data comes in, the.responseText property is updated.

4 (DONE): The operation is complete, and either the request has been successfully completed (status code 2xx) or an error occurred.

3(LOADING) code be updated multiple times to this status if the server is sending a large size of response in multiple chunks

http response type your server script returns:

200: "OK"

3xx: generally used for redirection purposes

4xx: client error (something wrong with your request

403: "Forbidden"

404: "Not Found" etc)

5xx: server error (something wrong with the server)

- **3**- Open AJAX request
- `xhttp.open(method, url, Async or sync);`
 - Method type could be GET or POST etc
 - url is the url of the file we want to handle our request
- So what else is left?
- Everything is in place! We just send it and carry on (we don't wait if we sent it asynchronously)
- **4**-`xhttp.send();`

2

AJAX call demos

Demo 1

sending AJAX call without any arguments !

```
<div id="demo">
  <h1>The XMLHttpRequest Object</h1>
  <button type="button" onclick="myFunc()">
    Change Content</button>
</div>
<script>
  function myFunc() {
    const xhttp = new XMLHttpRequest();
    xhttp.open("GET", "http://localhost:8888/",
      true);
    xhttp.send();
    xhttp.onreadystatechange = function () {
      console.log("hello")
    };
  }
</script>
```

Client: ajax.html

You can also type this url in the browser address bar. Since it's a Get request

Server: app.js

```
const http = require('http');
http.createServer(function (req, res) {
  console.log("The server recived a request ");
  res.writeHead(200, {
    "Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"
  });
  res.end("server's response!");
}).listen(8888);
```



Demo 2

sending AJAX call with passing
arguments !

```

<h1>The XMLHttpRequest Object</h1>
<div id="demo">
  <button type="button" onclick="myFunc()">Change Content</button>
</div>
<script>
  function myFunc() {
    const xhttp = new XMLHttpRequest();
    const str = "John";//"John&age=23"; for multiple parameters
    xhttp.open("GET", "http://localhost:8888/?name=" + str, true);
    xhttp.send();
    xhttp.onreadystatechange = function () {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML =
          this.responseText;
      }
    };
  }
</script>

```

Client: ajax.html

The XMLHttpRequest Object

Change Content

The XMLHttpRequest Object

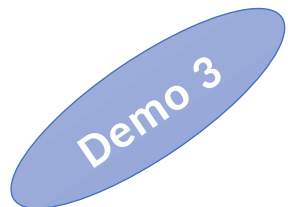
Hello John

```

let http = require('http');
let url = require('url');
http.createServer(function (req, res) {
  let q = url.parse(req.url, true);
  console.log(q.query); //returns '?name=John'
  res.writeHead(200, {"Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"});
  res.end('Hello ' + q.query["name"]);
}).listen(8888);

```

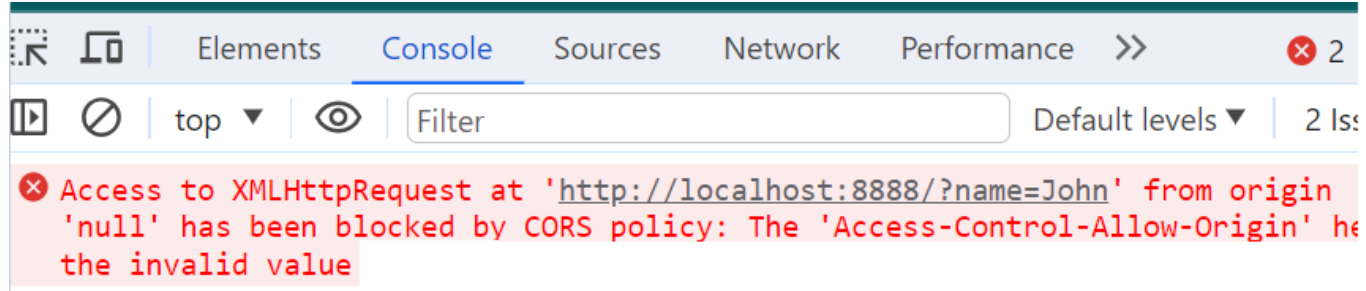
Server: app.js



AJAX call to remote server

Configuring CORS

- Who has seen a similar message?



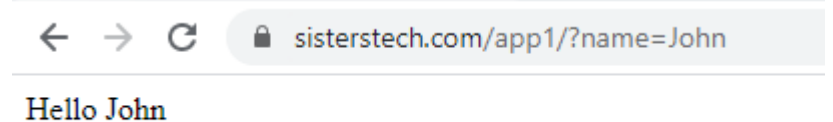
Demo 3

- Last time we tested this:

Editing: /home/sisterst/app1/app.j Encoding: utf-8 Re-open

[Keyboard shortcuts](#) 🔍 >_ ↺

```
1 const http = require('http');
2 const url = require('url');
3 const server = http.createServer(function(req, res) {
4   const q = url.parse(req.url, true);
5   console.log(q.query); //returns the object form of '?name=John'
6   res.writeHead(200, {"Content-Type": "text/html"});
7   res.end('Hello '+q.query["name"]);
8 });
9 server.listen();
```



CORS – when a browser sends a req to a server

- CORS is a security feature **implemented in web browsers** and is **configured by the contacted server** in the response's header to the AJAX requests made via the browser¹
- CORS (Cross-Origin Resource Sharing)
- By default², web browsers enforce a security policy called "Same-Origin Policy" (SOP) means the html files containing the AJAX requests have to be hosted in the same origin³ that also hosts the server side scripts, responding to those requests.
- **browsers will block² the Cross-Origin Requests unless the server being contacted explicitly allows it**

1: requests made mainly via AJAX, (also in most browsers via forms the url address bar).

2: expected behaviour for most browsers

3: An origin is (supposed to be) defined by the combination of domain, protocols and ports (80 is the default for HTTP, 443 for HTTPS)

4: When a web page hosted in **originA** needs to cross and send requests to a server in another origin, **originB**



Let's see how a server script configures/dictates CORS for browsers making the requests

Configure the browser CORS in your server side script!

- We now pass one more argument to the `writeHead()` method which creates connection back to the client.
- The **"Access-Control-Allow-Origin"** tells the browser what origins can send requests to this server
- In our case we chose **"*"** which means **everyone!** all origins around the world Can Send request to this server
- Since we said **"Access-Control-Allow-Origin": "*"**

server side (node js)

```
Editing: /home/sisterst/app1/app.js Encoding: utf-8 Re-open
Keyboard shortcuts

1 const http = require('http');
2 const url = require('url');
3 const server = http.createServer(function(req, res) {
4   const q = url.parse(req.url, true);
5   console.log(q.query); //returns the object form of '?'
6   res.writeHead(200, {
7     "Content-Type": "text/html",
8     "Access-Control-Allow-Origin": "*"
9   });
10  res.end('Hello ' + q.query["name"]);
11 });
12 server.listen();
13
```

Client side (HTML Javascript)

file:///C:/code/nodejs/7Ajax-remote-with-parameters...

The XMLHttpRequest Object

Hello John ✓

```
1
2
3
4
5 The XMLHttpRequest Object</h1>
6 id="demo">
7 <button type="button" onclick="myFunc()">Change Con
8 v>
9 <input>
10 function myFunc() {
11   const xhttp = new XMLHttpRequest();
12   const str = "John"; // "John&age=23"; for multipl
13   xhttp.open("GET", "https://sisterstech.com/app1
14   xhttp.send();
15   xhttp.onreadystatechange = function () {
16     if (this.readyState == 4 && this.status ==
17       document.getElementById("demo").innerHTM
18       this.responseText;
19   }
20   };
21 }
22 </input>
23
24
```

Line 1, Column 1 Coverage:

Q: What do these two do?

Access-Control-Allow-Methods

Access-Control-Allow-Headers

Q: What is a **Preflight Request** and how is it relevant to the http method OPTIONS

Q: How the http method OPTIONS should be implemented at the server side (what should the server return upon an OPTIONS request?)

- Q: Can we send requests to any url and expect respond using Postman regardless of the CORST setting ?
- This statement has been generated by chatGPT 3.5.
“CORS allows for secure communication between web applications hosted on different domains while still maintaining control over which domains have access. ”
- Q: what seems to be wrong with the above statement?

Demo 4

3

Ajax call using
POST

POST client side

```
3  <head>
4      <script>
5          function sendRequest() {
6              const name = document.getElementById("name").value;
7              const xhr = new XMLHttpRequest();
8              xhr.open("POST", "http://localhost:3000");
9              xhr.send("name=" + name);
10             xhr.onload = function () {
11                 let response = xhr.responseText;
12                 document.getElementById("result").innerHTML = response;
13             };
14         }
15     </script>
16 </head>
17
18 <body>
19     <input type="text" id="name" placeholder="Enter your name">
20     <button onclick="sendRequest()">Send Request</button>
21     <div id="result"></div>
22 </body>
```

For POST, the
query string sits
in the body of the
request

POST server side

- See how data is
- extracted from the
- POST request *in chunks*

```
1  const http = require("http");
2  const server = http.createServer(function(req, res) {
3    // check if the request method is POST
4    if (req.method === "POST") {
5      if (req.headers["access-control-request-method"]) {
6        res.setHeader("Access-Control-Allow-Origin", "*");
7        res.setHeader("Access-Control-Allow-Methods", "POST");
8        res.end();
9      } else {
10       let query = "";
11       req.on("data", function(chunk) {
12         query += chunk;
13       });
14       req.on("end", function() {
15         let params = new URLSearchParams(query);
16         let name = params.get("name");
17         res.setHeader("Content-Type", "text/plain");
18         res.setHeader("Access-Control-Allow-Origin", "*");
19         res.write("Hello " + name);
20         res.end();
21       });
22     }
23   });
24 // listen on port 3000
25 server.listen(3000, function() {
26   console.log("Server is running on port 3000");
27 });
```


Demo 4

Ajax call using both
POST and GET
methods

Client side: POST and GET (ajaxPostGet.html)

- Some variable initializations

The XMLHttpRequest Object

no response yet!

```
ajaxPostGet.html > html > body > script
1  <html>
2  <head>
3  </head>
4  <body>
5      <h1>The XMLHttpRequest Object</h1>
6      <button type="button" onclick="post()">POST</button>
7      <button type="button" onclick="getAll()">GET</button>
8
9      <div id="demo">
10         no response yet!
11     </div>
12     <script>
13         const xhttp = new XMLHttpRequest();
14         const endPointRoot = "http://localhost:8888/API/v1/";
15         let params = "?name=John&age=23";// or a json string
16         let resource = "patients/";
```

Client side: POST and GET (ajaxPostGet.html)

- The 'application/x-www-form-urlencoded'
- content type describes form data that is
- sent in a single block in the HTTP message body
- A header is a piece of information about the data sent via HTTP request. It tells the server receiving the request what type of data is enclosed, its formatting, the language used. (If you remember, The server also puts something similar in the head part of the response)
- More than one header can be put on a HTTP request; each in a 'name' and a 'value' pair

```
17  /** send() accepts an optional parameter which lets you specify the request's body;
18  this is primarily used for requests such as POST, PUT. If the request method is GET
19  or HEAD, the body parameter is ignored and the request body is set to null.
20  */
21  function post() {
22      xhttp.open("POST", endPointRoot + resource, true);
23      xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
24      xhttp.send(params);
25      xhttp.onreadystatechange = function () {
26          if (this.readyState == 4 && this.status == 200) {
27              document.getElementById("demo").innerHTML =
28                  this.responseText;
29          }
30      };
31  }
32  function getAll() {
33      const url = endPointRoot + resource + params;
34      xhttp.open("GET", url, true);
35      xhttp.send();
36      xhttp.onreadystatechange = function () {
37          if (this.readyState == 4 && this.status == 200) {
38              document.getElementById("demo").innerHTML =
39                  this.responseText;
40          }
41      };
42  }
```

For POST, the query sits in the body of the request

Server side: POST and GET (serverPostGet.js)

- Note the differences between handling a GET request and a POST request
- See how data is extracted from the POST request *in chunks*

```
serverPostGet.js > http.createServer() callback
1  const http = require('http'); const url = require('url');
2  const GET = 'GET'; const POST = 'POST';
3  const endPointRoot = "/API/v1/";
4  http.createServer(function (req, res) {
5      res.writeHead(200, {
6          "Content-Type": "text/html",
7          //all origins can send request to this server
8          // is this really a good practice?
9          "Access-Control-Allow-Origin": "*",
10         "Access-Control-Allow-Methods": "*"
11     });
12     console.log(req.headers);
13     if (req.method === GET) {
14         const q = url.parse(req.url, true);
15         // here you fetch some data from DB
16         res.end(`Hello ${q.query["name"]}.
17         GET all request is recvied!`);
18     }
19     if (req.method === POST && req.url === endPointRoot + 'patients/') {
20         let body = "";
21         /* req.on('data' gets called when something has been read
22         from the stream */
23         req.on('data', function (chunk) {
24             if (chunk != null) { // you need to check this if
25                 body += chunk; // data may come in multiple chunks
26             }
27         });
28         /*req.on('end' gets called when stream has ended
29         ( all data from cleint has arrived to server)*/
30         req.on('end', function () {
31             let q = url.parse(body, true);
32             res.end(`Hello: ${q.query.name}, we got your POST request`);
33         });
34     }
35 }
36 ).listen(8888);
```



Question!

In the context of HTTP methods and Cross-Origin Resource Sharing (CORS), what is a "preflight request,"?

- Disclaimer: It's the students' responsibility to write their own code.
- Codes provided in lecture notes are just samples to get students a head start .

Working with a Relational Database



Content from this point onward are to be self studied for the next week . . .

CRUD

- This week we will discuss how to connect node js to a relational DB such as MySQL server (the free open source version is called MarinaDB)
- We will learn how to run SQL queries which can be used to
- **C**reate a table or DB
- **R**ead from a DB
- **U**ppdate records on a DB
- **D**eleate a record or Drop a table
- All of this operation with DB all referred as CRUD in brief
- In this lecture we focus on the node js part mostly assuming writing SQL statement is covered in your term 2 DB course

Introduction
to
Connecting to DB
using node js
on your **PC**

Using XAMPP package installer

- You need a DB engine that runs SQL statements for you
- And a DB administrative tool like an editor on which you could write your SQL statements and see result visually
- 1- XAMPP installs both
- 2- The DB engine (mySQL)
- 3- And DB admin tool (phpMyAdmin)

5

install

XAMPP 1

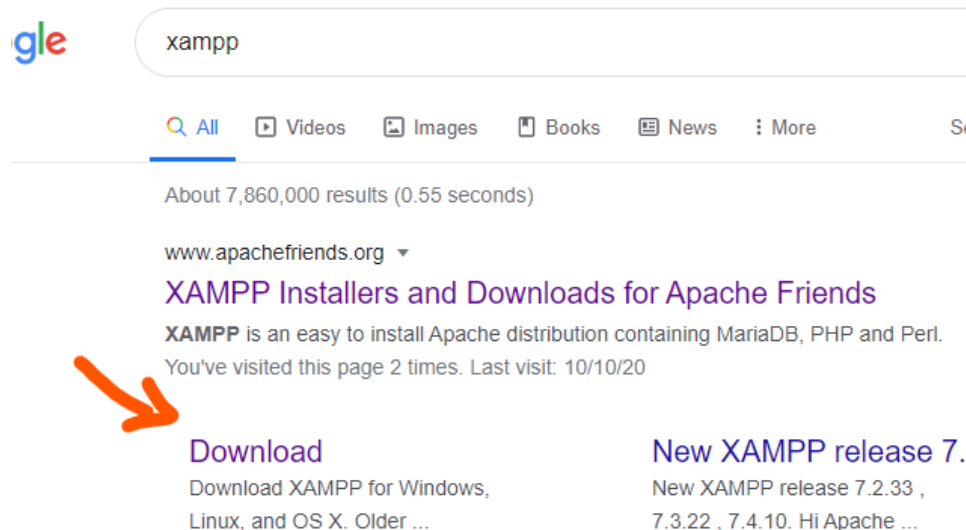


MariaDB (MySQL) 2

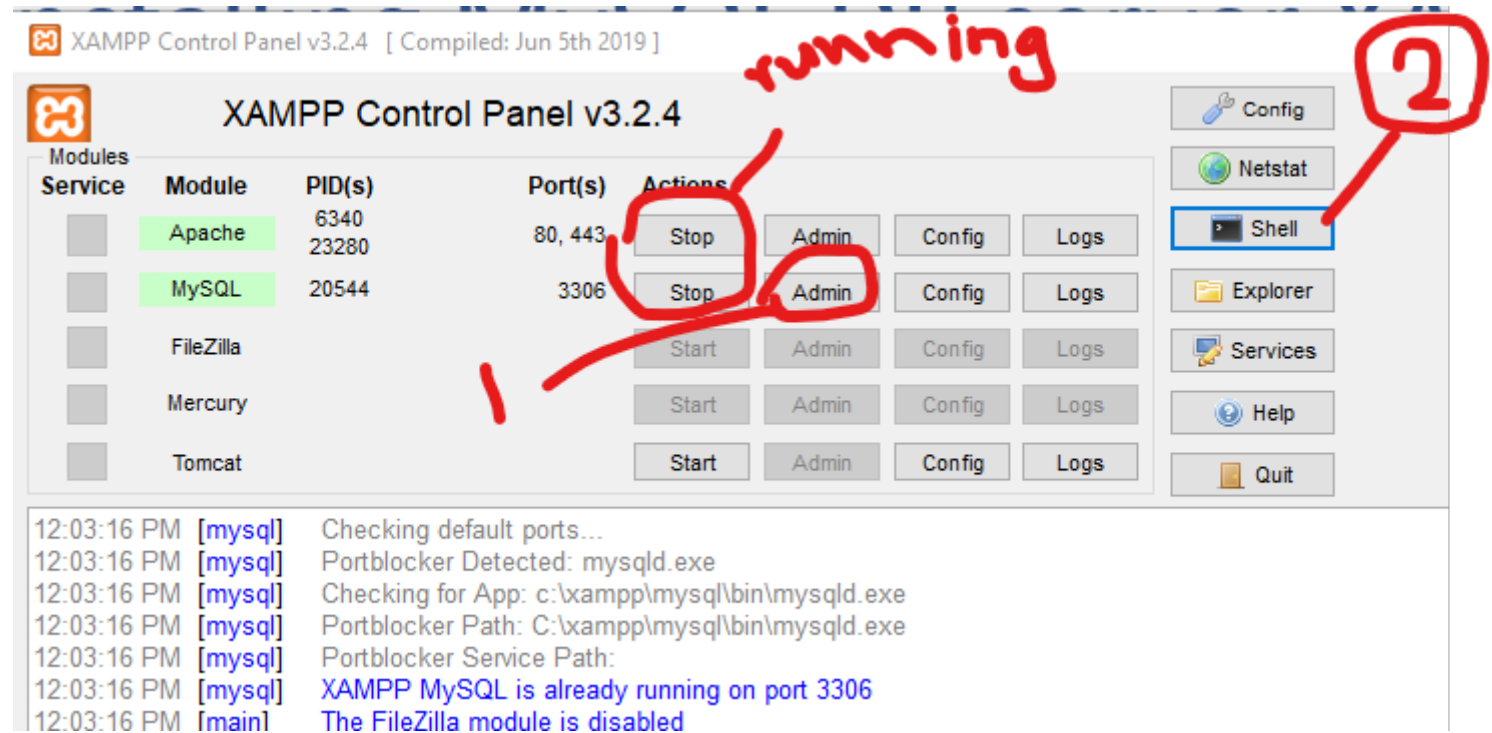
phpMyAdmin 3

Setting up DB server on your PC

- We need to install mySQL DB server. It's a relational DB engine which runs our SQL queries
- As well we need phpMyAdmin which is a web based database administrative tool. It is web based therefore you can use it remotely or share a database with your partner
- phpMyAdmin needs PHP engine to run
- XAMPP is an application package installer that installs all of those for us

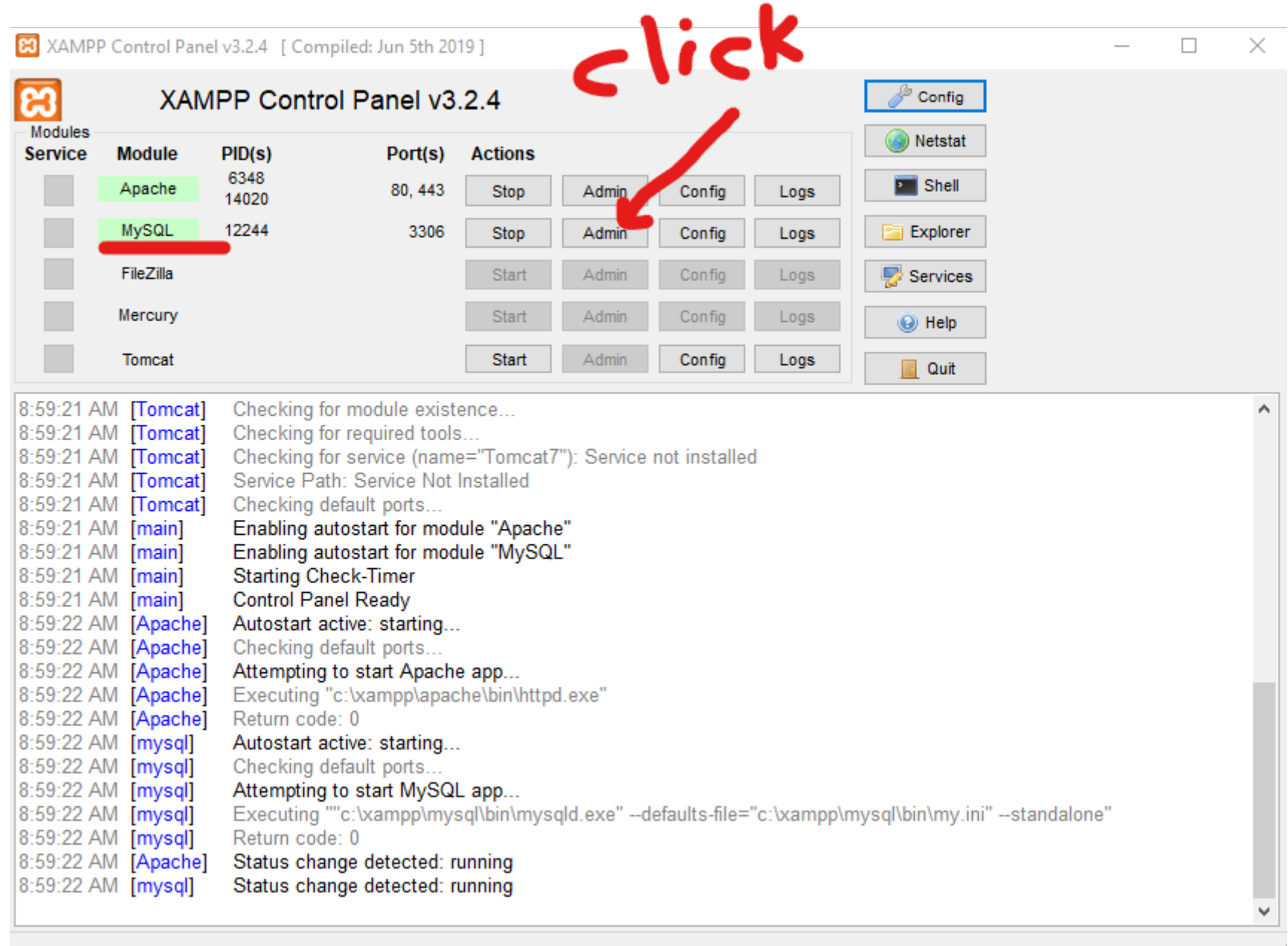


- 1: clicking on admin will take you to phpMyAdmin
- 2- Clicking on shell will take you to the mySQL command line

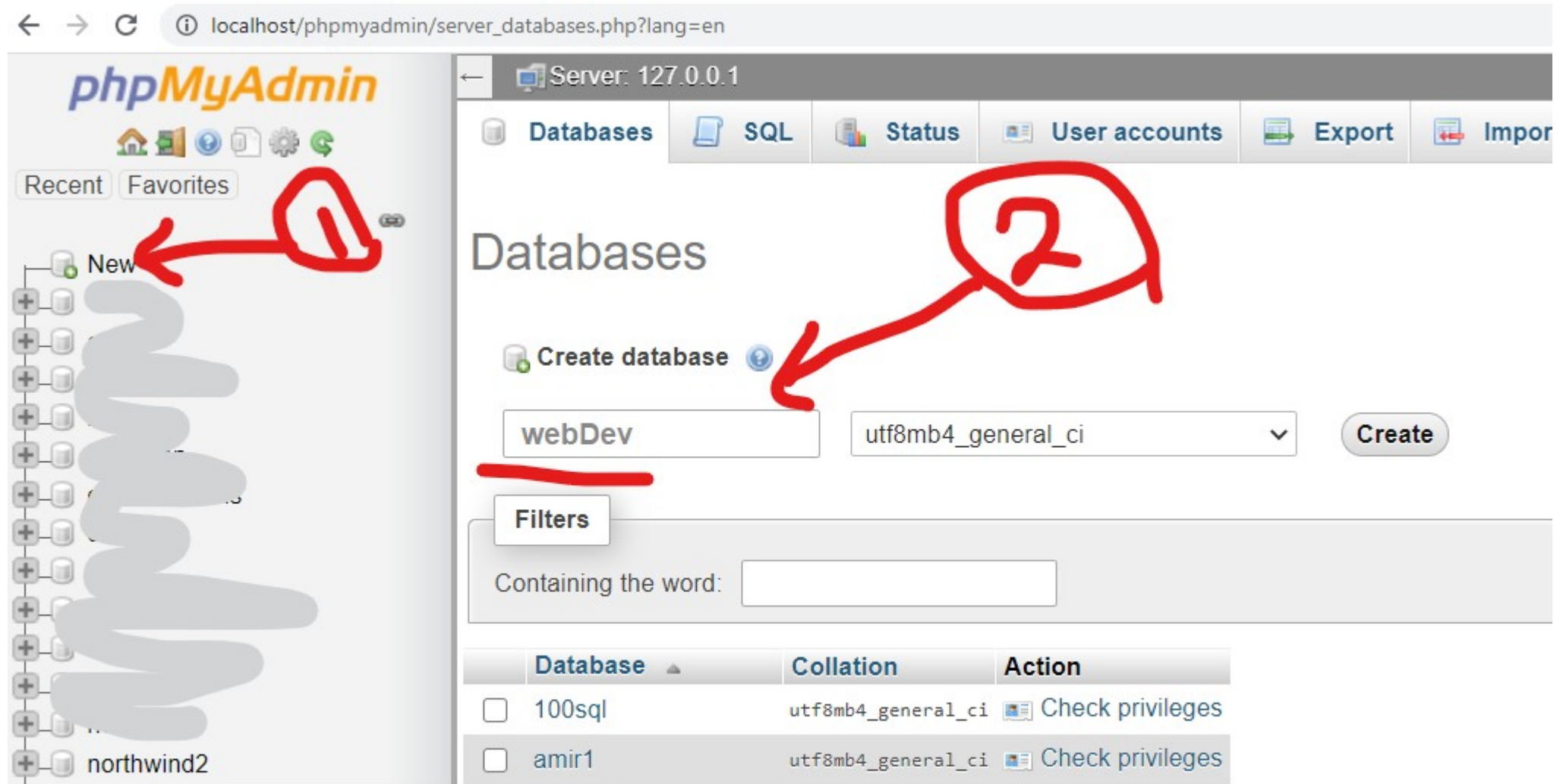


2- goto phpMyAdmin and to create a DB and a table

- Clicking on that link Loads
- <http://localhost/phpmyadmin/>
- (Make sure MySQL is running on XAMPP)



3- create a DB in phpmyadmin



The screenshot shows the phpMyAdmin interface for a local server (127.0.0.1). The left sidebar contains a 'New' button, which is circled in red and labeled with a red '1'. The main content area shows the 'Databases' tab, where a 'Create database' link is circled in red and labeled with a red '2'. Below this link, the database name 'webDev' is entered in a text field, and the collation 'utf8mb4_general_ci' is selected from a dropdown menu. A red underline is drawn under the 'webDev' text field. The 'Create' button is visible to the right of the form. Below the form, there is a 'Filters' section with a search box labeled 'Containing the word:'. At the bottom, a table lists existing databases:

Database	Collation	Action
<input type="checkbox"/> 100sql	utf8mb4_general_ci	Check privileges
<input type="checkbox"/> amir1	utf8mb4_general_ci	Check privileges

4- create a table in that DB to store scores

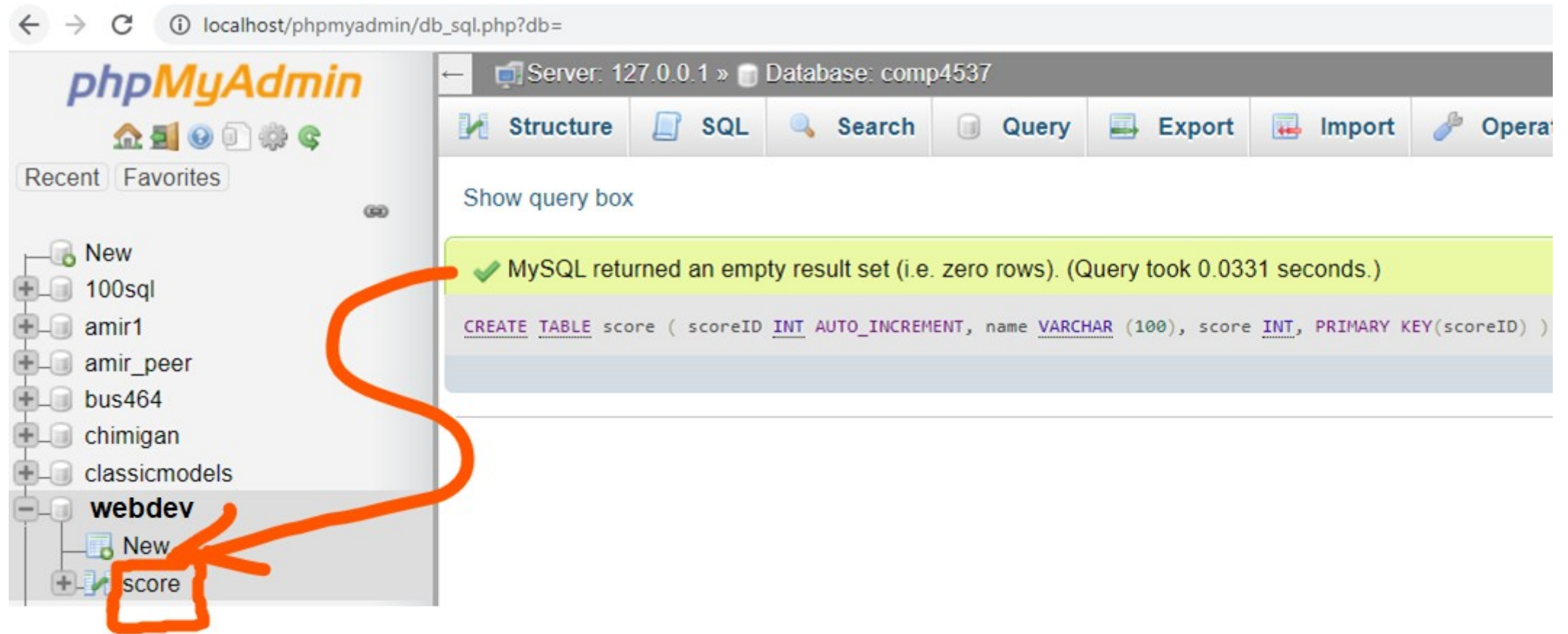
```
CREATE TABLE score (  
  scoreID INT AUTO_INCREMENT,  
  name VARCHAR (100),  
  score INT,  
  PRIMARY KEY(scoreID)  
);
```

- Note: It is ok to have column names with the same name as the table name, both are score
- the Now our DB is ready!

The screenshot shows the phpMyAdmin interface with the following elements and annotations:

- Database Selection:** In the left sidebar, the 'webdb' database is selected. An orange circle '1' and the word 'click' with an arrow point to this database.
- SQL Tab:** The 'SQL' tab is selected in the top navigation bar. An orange circle '2' and the word 'click' with an arrow point to this tab.
- Query Entry:** The SQL query is entered into the text area. An orange circle '3' and the word 'Query' with an arrow point to the query text.
- Execution:** The 'Go' button at the bottom right is highlighted with an orange circle '4' and an arrow.
- Interface Details:** The URL is 'localhost/phpmyadmin/db_sql.php?db=webdb'. The server is '127.0.0.1'. The query area includes buttons for 'Clear', 'Format', and 'Get auto-saved query', and a checkbox for 'Bind parameters'.

5- notice the confirmation message on creation of table



The screenshot shows the phpMyAdmin web interface in a browser. The address bar displays `localhost/phpmyadmin/db_sql.php?db=`. The interface includes a sidebar on the left with a tree view of databases. The 'webdev' database is expanded, showing a 'New' button and a 'score' table. The 'score' table is highlighted with an orange box, and an orange arrow points from it to the confirmation message in the main panel. The main panel shows the 'Structure' tab selected, with a toolbar containing 'Structure', 'SQL', 'Search', 'Query', 'Export', 'Import', and 'Operations'. Below the toolbar, a message box displays a green checkmark and the text: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0331 seconds.)'. Below this message, the SQL query used to create the table is shown: `CREATE TABLE score (scoreID INT AUTO_INCREMENT, name VARCHAR (100), score INT, PRIMARY KEY(scoreID))`.

localhost/phpmyadmin/db_sql.php?db=

phpMyAdmin

Server: 127.0.0.1 » Database: comp4537

Structure SQL Search Query Export Import Operations

Show query box

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0331 seconds.)

`CREATE TABLE score (scoreID INT AUTO_INCREMENT, name VARCHAR (100), score INT, PRIMARY KEY(scoreID))`

Recent Favorites

New

100sql

amir1

amir_peer

bus464

chimigan

classicmodels

webdev

New

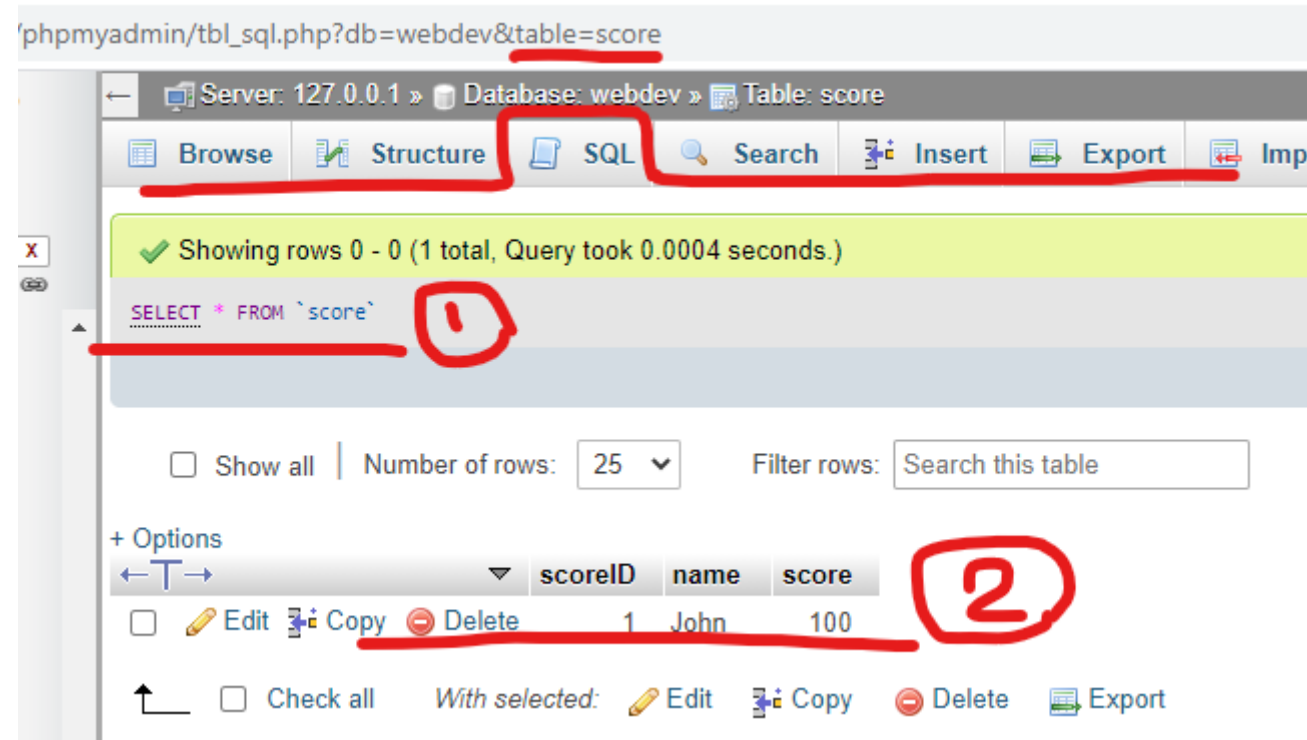
score

6- run few SQL statements on table you have created

0- make sure you have selected the DB you have created to run the SQL query on that

1- type down your query at SQL tab

2- press GO button and the result of your query appears!





Connecting to mySQL DB using nodejs

7. Connect with DB using node js

- Turns out, mysql is an external module and needs to be installed
- Here is how to
 - 1 at command prompt switch to the current directory of this file
 - 2 enter
>npm install mysql



```
JS db_simple.js > ...
3
4 const mysql = require("mysql");
5
6 // Create connection
7 const db = mysql.createConnection({
8   host: "localhost",
9   // ...
10 });
11
12 db.connect((err) => {
13   if (err) throw err;
14   console.log('Connected to MySQL database');
15 });
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
C:\Program Files\nodejs\node.exe .\db_simple.js
> Uncaught Error: Cannot find module 'mysql'
Debugger listening on ws://127.0.0.1:61995/a7d40b89-307e-4c54-a91b-931dad04061
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
internal/modules/cjs/loader.js:638
  throw err;
  ^

Error: Cannot find module 'mysql'
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:636:15)
    at Function.Module._load (internal/modules/cjs/loader.js:562:25)
    at Module.require (internal/modules/cjs/loader.js:692:17)
```

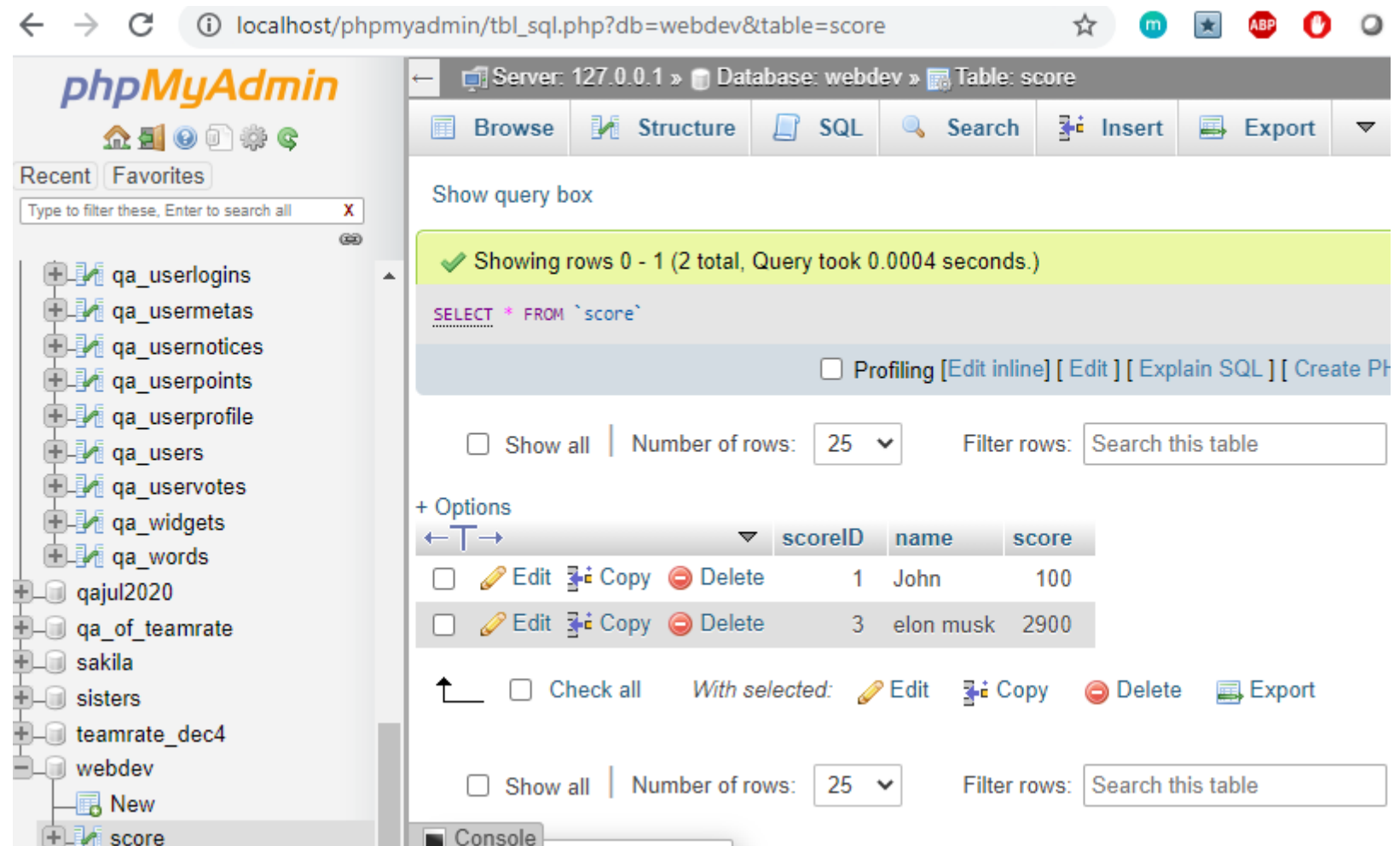
```
+ mysql@2.18.1
added 11 packages from 15 contributors and audited 11 packages in 1.094s
found 0 vulnerabilities
```

- First create a mysql object
- Then connect
- And run a SQL statement

```
// at command prompt switch to the current directory of this file
//enter>npm install mysql
const mysql = require("mysql");
// Create connection
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "webdev"
});
// Connect to MySQL to run SQL query
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  let sql = "INSERT INTO score(name, score) values ('elon musk', 2900)";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

Check your table to verify the SQL ran successfully on node

- As you can see
- The score of
- elon musk was entered in our table
- Successfully!



The screenshot shows the phpMyAdmin interface for a database named 'webdev'. The 'score' table is selected, and the 'Browse' tab is active. The table structure is shown as follows:

scoreID	name	score
1	John	100
3	elon musk	2900

The interface also displays the SQL query used to retrieve the data: `SELECT * FROM `score``. The status bar indicates that 2 rows are shown out of 2 total rows, and the query took 0.0004 seconds to execute.

Hosting Node js, MySQL on remote server



Please watch this video

Hosting nodejs MySql remotely CPanel Shared Hosting.mp4

Beginner
's guide

Node js + MySQL


cPanel


remote

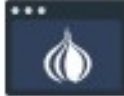
server


How to host nodejs MySQL in share hosting accounts


SOFTWARE


 WordPress Manager by Softaculous Apps Installer

 PHP PEAR Packages


 Perl Modules


 Setup Node.js App


 Select PHP Version

 Setup Python App


DATABASES


 phpMyAdmin


 MySQL® Databases


 MySQL® Database V


ADVANCED


 Terminal

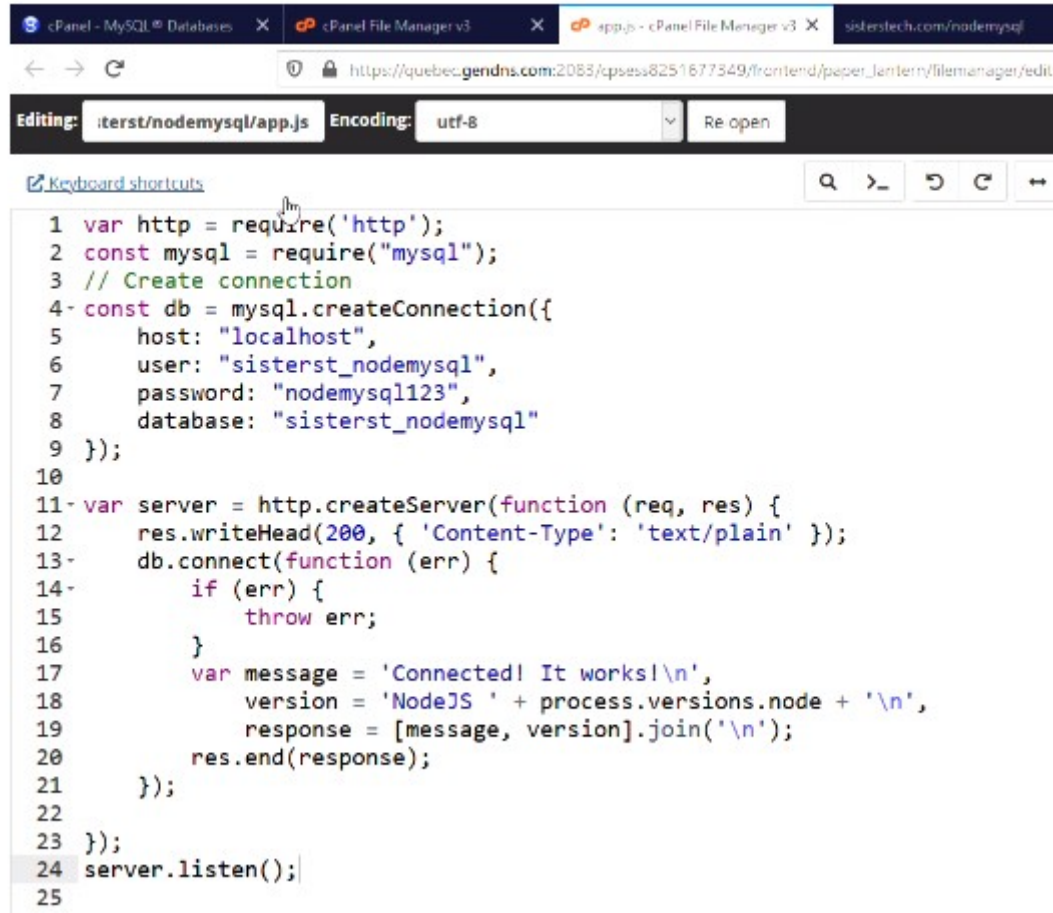
 Cron Jobs

 Track DNS

 Apache Handlers

 MIME Types

 Virus Scanner



The screenshot shows a web browser with three tabs: 'cPanel - MySQL® Databases', 'cPanel File Manager v3', and 'app.js - cPanel File Manager v3'. The address bar shows the URL 'https://quebec.gendns.com:2083/cpsess8251677349/frontend/paper_jantern/filemanager/edit'. The editor displays the following code:

```
1 var http = require('http');
2 const mysql = require("mysql");
3 // Create connection
4 const db = mysql.createConnection({
5   host: "localhost",
6   user: "sisterst_nodemysql",
7   password: "nodemysql123",
8   database: "sisterst_nodemysql"
9 });
10
11 var server = http.createServer(function (req, res) {
12   res.writeHead(200, { 'Content-Type': 'text/plain' });
13   db.connect(function (err) {
14     if (err) {
15       throw err;
16     }
17     var message = 'Connected! It works!\n',
18       version = 'NodeJS ' + process.versions.node + '\n',
19       response = [message, version].join('\n');
20     res.end(response);
21   });
22 });
23
24 server.listen();
25
```

source

- chatGPT 3.5 used to generate partial code for this set