

Adv Web Dev Arch

Lecture 5

Amir Amintabar, PhD

AJAX Calls

高级Web开发架构讲

座第5讲

阿米尔·阿明塔巴尔, 博士

AJAX调用

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

课程大纲（构建典型 Web API、AJAX 简介、数据库入门）

- 复习
- - AJAX 请求
 - 浏览器使用 HTTP 方法向服务器发送请求
- - AJAX 请求演示
- - 向远程服务器发起 AJAX 请求
- —— 下周自学内容
- —— 使用关系型数据库（增删改查操作）
- —— 使用 XAMPP 套件安装程序
- —— 使用 Node.js 连接 MySQL 数据库
- —— 在远程服务器上部署 Node.js 和 MySQL

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, timeout, 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 testWithModule.js > ...
1 const mo = require('./modules/math');
2 let r = 10;
3 console.log(`Hello YOURNAME. The area is`);
4
5
```

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

复习

- 第 1 周：面向多种设备开发 Web 应用时所面临的一般性挑战。复习资料（逾 120 张幻灯片）已提供，用于复习 HTML、CSS、JavaScript 等内容，并学习如何创建对象等知识。
- 第 2 周：我们定义了“Web 应用架构”这一术语，随后以小组讨论形式探讨了 Web 开发中各类架构类型，并学习了本地存储（Local Storage）相关知识。
- 第 3 周：进一步介绍 JavaScript，包括超时机制（timeout）、执行顺序、事件循环（Event Loop）、变量提升（hoisting）、var 与 let 的区别、Web API 的构成原理、RESTful API 入门、微服务（Microservices），以及 GET 与 POST 方法的对比。
- 第 4 周：在 JavaScript 中，每个函数都会返回某个值；同时介绍了 Node.js 模块机制。

数学.js
同样适用

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

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

1

AJAX calls

1

AJAX 调用

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

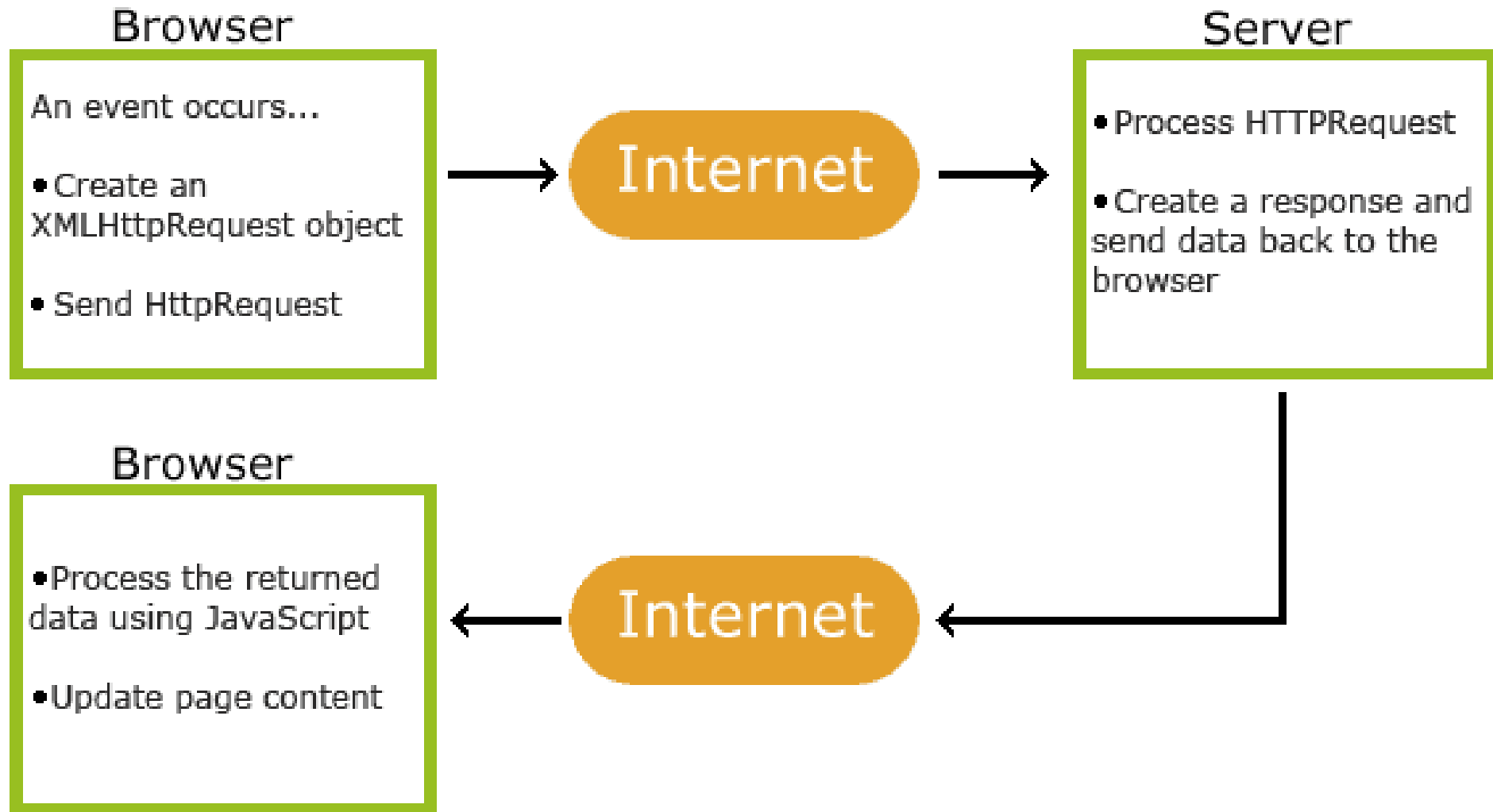
**截至目前，我们都是通过服务器发送请求
通过浏览器的地址栏！**

如果我们希望在 HTML 页面内部而非浏览器地址栏中向服务器发送请求，该怎么办？
页面内而非浏览器地址栏？
例如，单击一个按钮即可向服务器发送某些数据？

您可以通过表单来实现这一点
or
AJAX 调用

- AJAX allows sending request to server

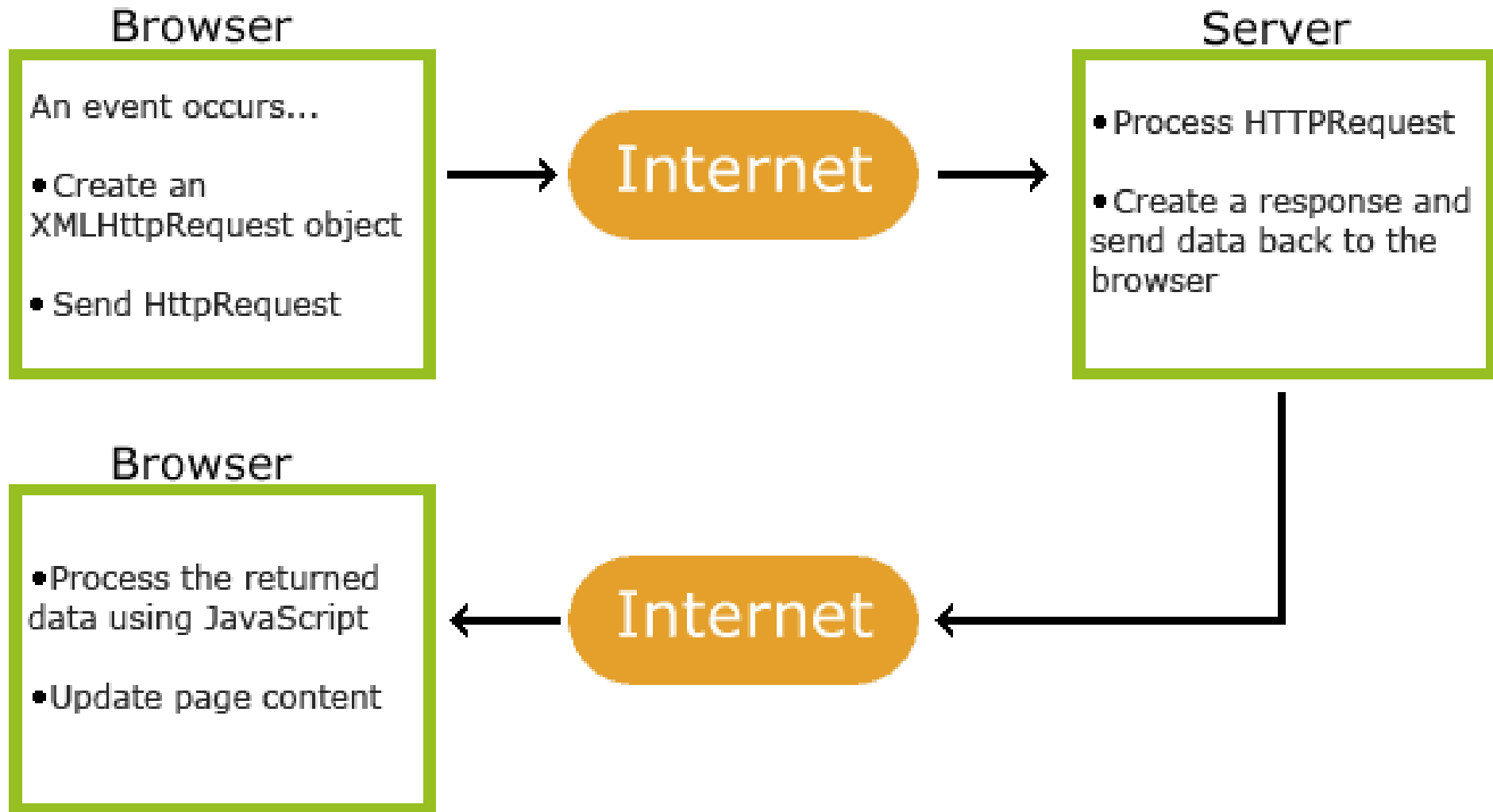
How AJAX Works



https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp

- AJAX 允许向服务器发送请求

How AJAX Works



https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp

- 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();

- 我们在 **客户端** 是这样实现的：
- **1- 创建一个 XMLHttpRequest 对象：**
 - （方式与创建 Date 对象类似）
 - **声明** xhttp = **新建** 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)

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

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

问： 由谁决定返
回何种状态码？

status of the XMLHttpRequest :
0 (未发送)： XHR 对象已创建， 但尚未调用
open() 方法。
1 (已打开)： 已调用 open() 方法。
2 (已接收响应头_)： 已调用 send() 方法， 且响应的头
部信息已可用。
3 (加载中)： 正在接收响应。随着数据陆
续到达，responseText 属性将被持续更新。
4 (已完成)： 操作已结束， 请求已成功完成
(状态码为 2xx) 或发生了错误。

3 (加载中) 状态码可能被多次更新为该状态， 前提是服务器正以
多个数据块的形式发送较大体积的响应。

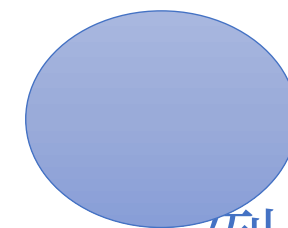
http response type your server script
返回值:
200: "OK"
3xx: 通常用于重定向目的4xx: 客户端
错误 (您的请求存在某些问题
403: "Forbidden"
404: "Not Found" etc)
5xx: server error (something wrong with the
server)

- **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();`

- **3- 打开 AJAX 请求**
- `xhttp.open(方法, URL, 异步或同步);`
 - 方法类型可以是 GET、POST 等
 - URL 是我们希望处理该请求的文件地址
- 那么，还剩下什么呢?
- 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



AX 调用
示例

Demo 1

sending AJAX call without any arguments !

演示 1

正在发送不带任何参数的
AJAX 请求！

```
<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);
```

```
<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>
```

客户端: ajax.html

您也可以在浏览器地址栏中直接输入该 URL。由于这是一个 GET 请求,

服务器: 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);
```

A blue oval label with the text "Demo 2" inside, tilted at an angle.

Demo 2

sending AJAX call with passing arguments !

A blue oval label with the text "Demo 2" inside, tilted at an angle.

正在发送 AJAX 请求并传递参数
参数！

```
<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

```
<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>
```

客户端: ajax.html

The XMLHttpRequest Object

Change Content

The XMLHttpRequest Object

Hello John

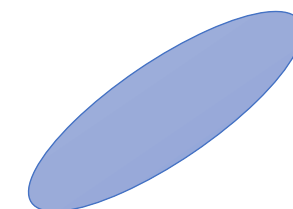
```
let http = require('http');
let url = require('url');
http.createServer(function (req, res) {
  let q = url.parse(req.url, true);
  console.log(q.query); //returns '?name=John'
  res.writeHead(200, {"Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"});
  res.end('Hello ' + q.query["name"]);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

服务器: app.js



AJAX call to remote server

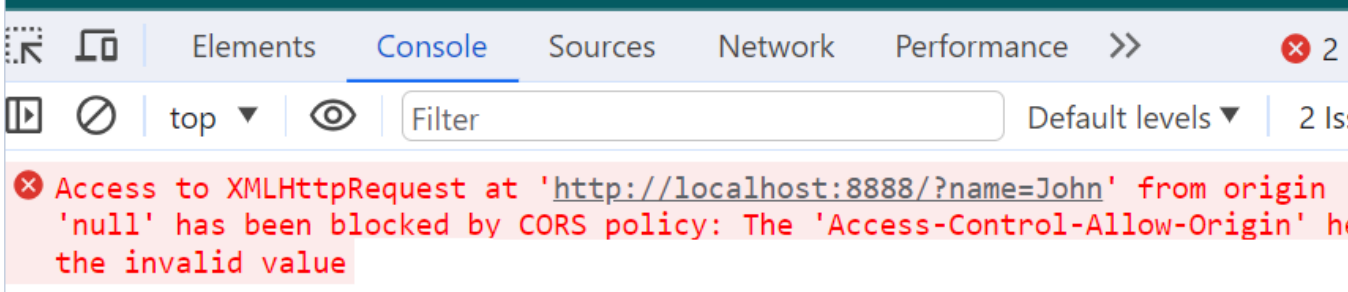
Configuring CORS



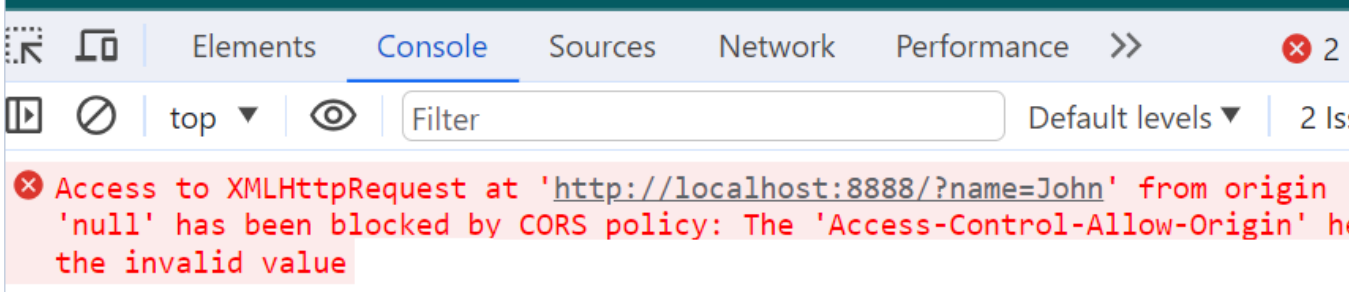
向远程服务器发起 AJAX 请求

配置 CORS

- Who has seen a similar message?

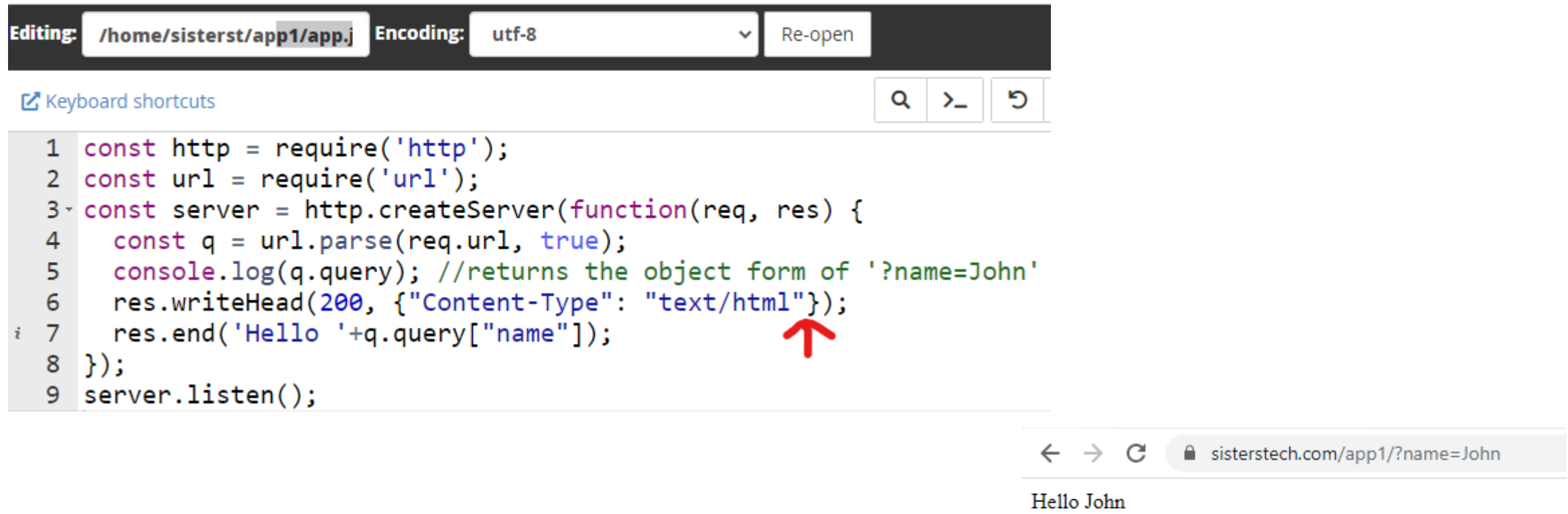


- 谁见过类似的消息？



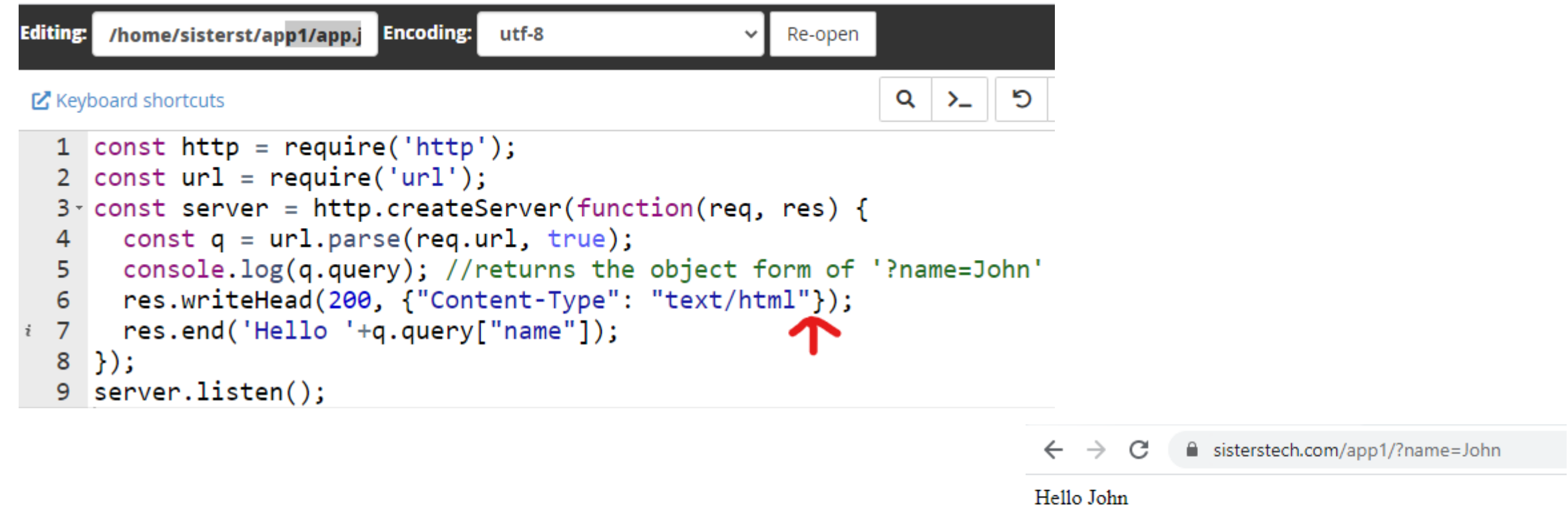
Demo 3

- Last time we tested this:



演示 3

- 上次我们测试了此功能：

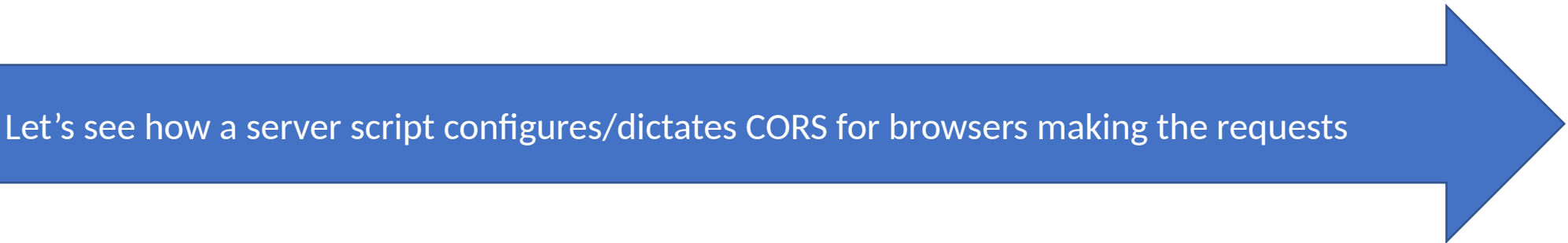


CORS – when a browser sends a req to a server

- 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**



CORS——当浏览器向服务器发送请求时 ^q向服务器发送请求时

- CORS 是一种安全机制，**由 Web 浏览器实现**，并由所访问的服务器**在响应浏览器发起的 AJAX 请求时，通过响应头进行配置**¹
- CORS（跨源资源共享）
- 默认情况下²，Web 浏览器强制执行一种名为“同源策略”（SOP）的安全策略，这意味着包含 AJAX 请求的 HTML 文件必须与响应这些请求的服务端脚本托管在同一源³上。
- 浏览器将阻止²跨源请求，除非**被访问的服务器**明确允许该行为

1: 请求主要通过 AJAX 发起（在大多数浏览器中，也可通过表单或地址栏中的 URL 发起）； 2: 这是大多数浏览器的预期行为； 3: 源（origin）应由域名、协议和端口三者共同定义（HTTP 默认端口为 80，HTTPS 默认端口为 443）
4: 当托管在源 A（originA）中的网页需要跨源向另一源（源 B，originB）中的服务器发送请求时



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)

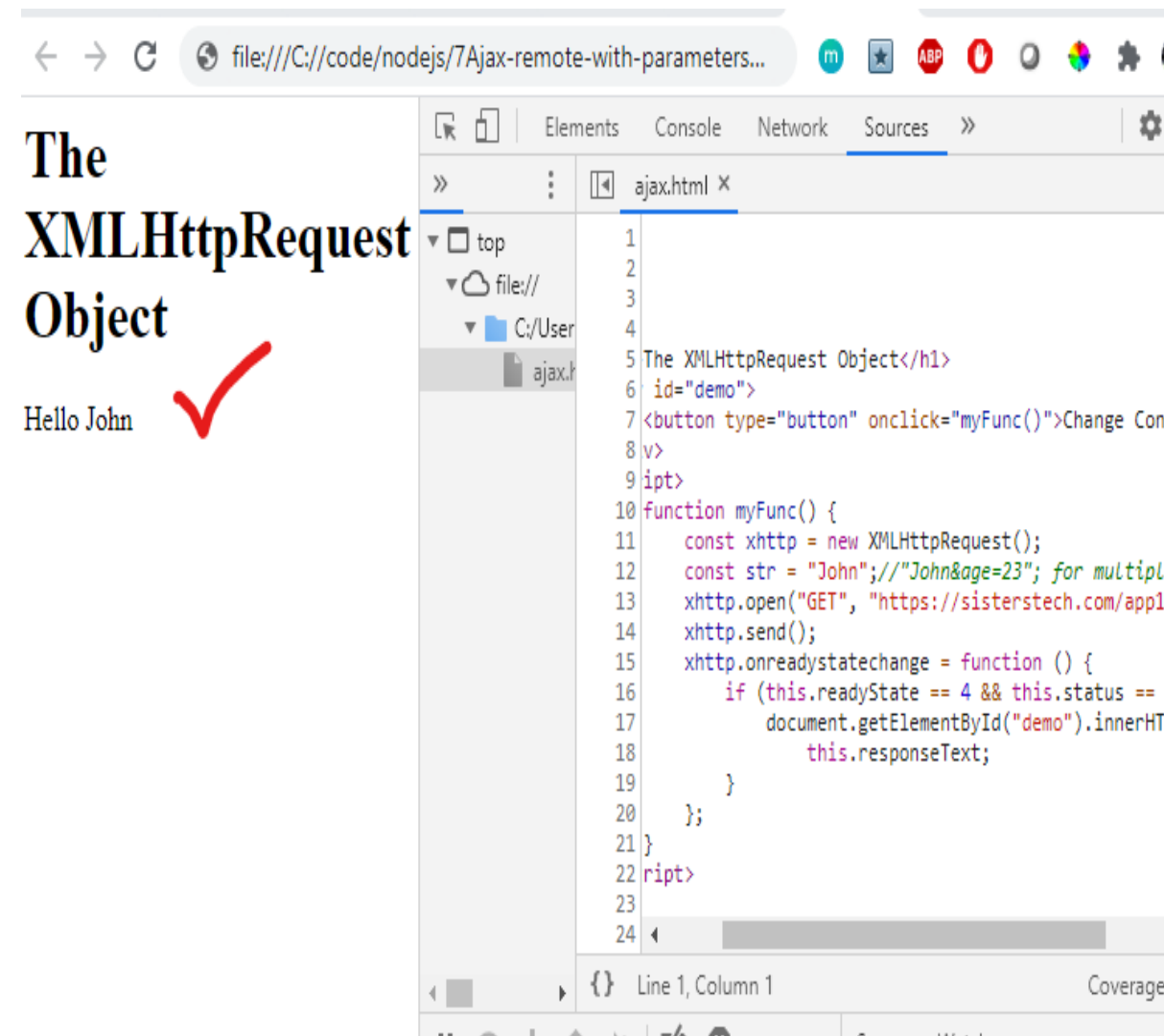
```
Editing: /home/sisterst/app1/app.j Encoding: utf-8 Re-open
Keyboard shortcuts

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 '?'
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)

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 ipt>
10 function myFunc() {
11   const xhttp = new XMLHttpRequest();
12   const str = "John";//"John&age=23"; for multipl
13   xhttp.open("GET", "https://sisterstech.com/appl
14   xhttp.send();
15   xhttp.onreadystatechange = function () {
16     if (this.readyState == 4 && this.status == .
17       document.getElementById("demo").innerHTI
18         this.responseText;
19   }
20   };
21 }
22 ript>
24
```

请在服务器端脚本中配置浏览器的跨域资源共享（CORS）！

- 现在，我们向 `writehead ()` 方法额外传递一个参数，该参数用于创建与客户端的连接。
- 其中，**"Access-Control-Allow-Origin"** 告诉浏览器哪些源（origin）可以向该服务器发送请求。
- 在本例中，我们选择了 **"*"**，表示**允许所有源!** 全球所有源均可向该服务器发送请求
- 服务器
既然我们已声明 **"Access-Control-Allow-Origin": "*"**

服务器端 (Node.js)

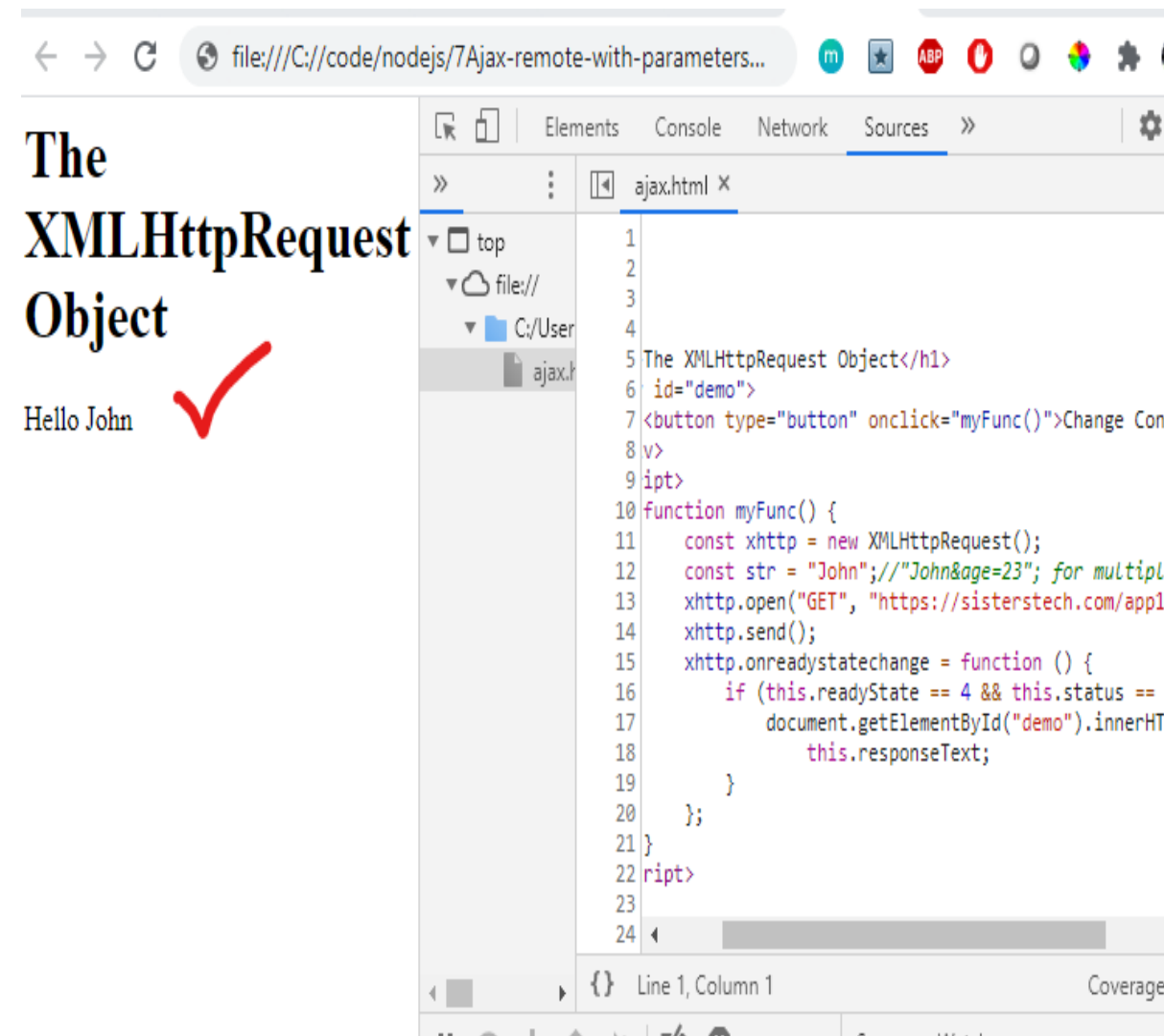
```
Editing: /home/sisterst/app1/app.j Encoding: utf-8 Re-open
Keyboard shortcuts

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 '?'
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
```

客户端 (HTML、JavaScript)

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 ipt>
10 function myFunc() {
11   const xhttp = new XMLHttpRequest();
12   const str = "John";//"John&age=23"; for multipl
13   xhttp.open("GET", "https://sisterstech.com/appl
14   xhttp.send();
15   xhttp.onreadystatechange = function () {
16     if (this.readyState == 4 && this.status == .
17       document.getElementById("demo").innerHTI
18         this.responseText;
19   }
20   };
21 }
22 ript>
24
```

Q: What do these two do?

Access-Control-Allow-Methods

Access-Control-Allow-Headers

问：这两项分别有何作用？

Access-Control-Allow-Methods（允许的跨域请求方法）

访问控制允许的请求头

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

问：什么是**预检请求（Preflight Request）**？它与 HTTP 方法 OPTIONS 有何关联？

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

问：服务器端应如何实现 HTTP 方法 OPTIONS（即服务器在收到 OPTIONS 请求时应返回哪些内容）？

- 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?

- 问： 我们能否向任意 URL 发送请求，并期望 Postman 返回响应，而完全不受 CORS 设置的影响？

- 该陈述由 ChatGPT 3.5 生成： “CORS 允许托管在不同域名上的 Web 应用程序之间进行安全通信，同时仍能控制哪些域名有权访问。 ”
- 问： 上述陈述存在什么问题？

Ajax call using
POST

Demo 4

3

使用 POST 方法
发起 Ajax 请求

演示 4

3

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

客户端 POST

```
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>
```

对于 POST 请求，查询字符串位于请求体中。

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 });
```

POST 服务器端

- 了解数据如何
- 从 POST 请求中
- 分块提取 (*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

演示 4

使用 POST 和 GET
方法发起 Ajax 请求

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/";
```

客户端：POST 和 GET （ajaxPostGet.html）

- 部分变量初始化

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

客户端：POST 和 GET (ajaxPostGet.html)

- “application/x-www-form-urlencoded”
- 内容类型用于描述表单数据，该数据
- 以单一数据块的形式发送至 HTTP 消息体中
- HTTP 头部是关于通过 HTTP 请求所发送数据的一段信息，它向接收该请求的服务器说明：
 - 所封装数据的类型、格式及所用语言。（如果您还记得的话，服务器也会在响应消息的头部中添加类似的信息。）
- 一个 HTTP 请求中可包含多个请求头，每个请求头均由一个“名称”和一个“值”组成。

```
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  }
```

对于 POST 请求，查询数据位于请求体中。

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);
```

服务器端：POST 和 GET
(serverPostGet.js)

- 注意处理 GET 请求与 POST 请求之间的区别
- 查看如何从 POST 请求中以 分块方式 提取数据

```
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,"?

问题!

在 HTTP 方法和跨域资源共享（CORS）的背景下，“预检请求”（preflight request）指的是什么？

- 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 .

- 免责声明： 编写自身代码是学生的责任。
- 讲义中提供的代码仅作为示例， 旨在帮助学生快速入门。

Working with a Relational Database



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

使用关系型 数据库



从本处开始的内容需在接下来的一周内自主学习……

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**elete 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

CRUD

- 本周我们将讨论如何将 Node.js 连接到关系型数据库（例如 MySQL 服务器，其免费开源版本称为 MariaDB）
- 我们将学习如何执行 SQL 查询，这些查询可用于
- **C**创建表或数据库
- **R**从数据库中读取数据
- **U**更新数据库中的记录
- **D**删除某条记录或删除某个表
- 所有这些针对数据库的操作，简称为 CRUD。
- 本讲主要聚焦于 Node.js 部分，假设 SQL 语句的编写已在你们第二学期的数据库课程中涵盖。

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

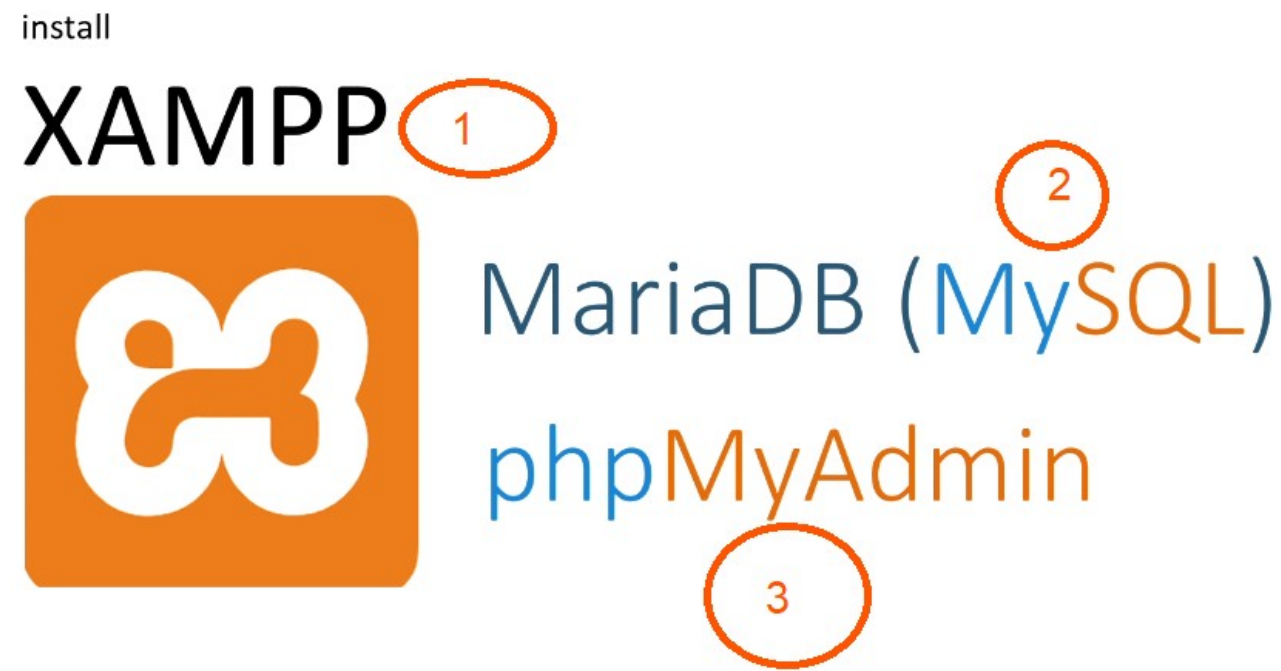
简介
to
连接数据库
使用 Node.js
在您的 **PC** 上

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

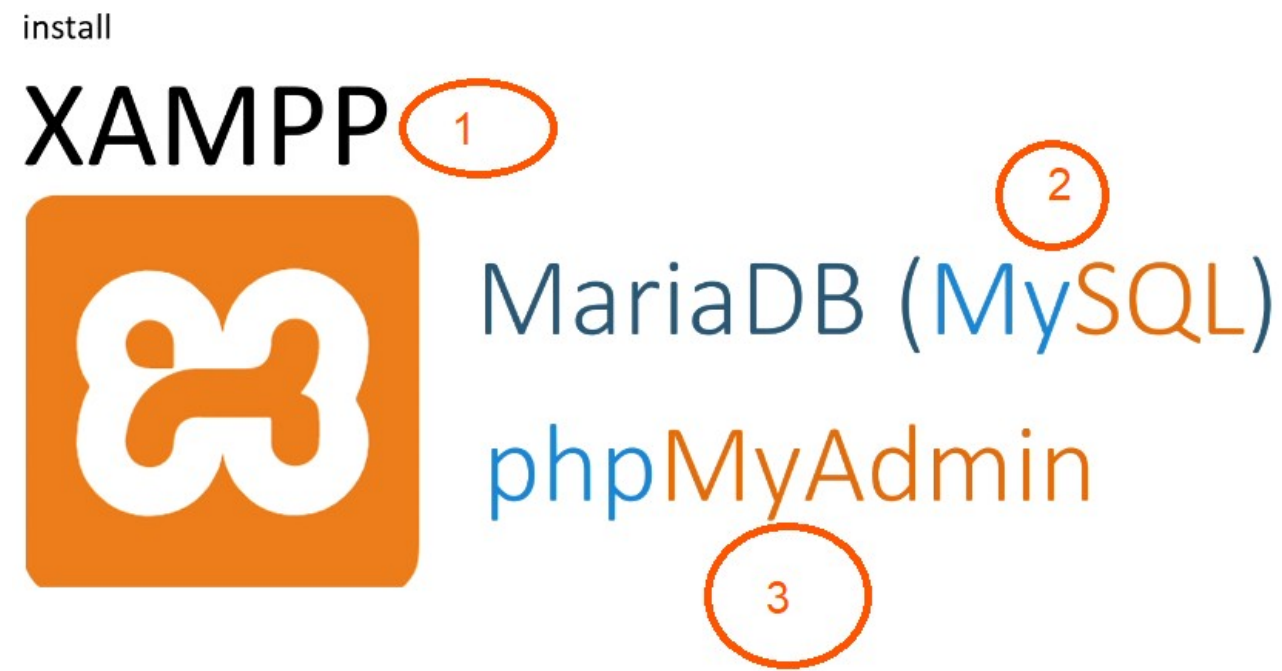


使用 XAMPP 软件包安装程序

- 您需要一个能够为您执行 SQL 语句的数据库引擎
- 以及一个数据库管理工具（例如类似编辑器的应用程序），您可借此编写 SQL 语句并直观地查看执行结果

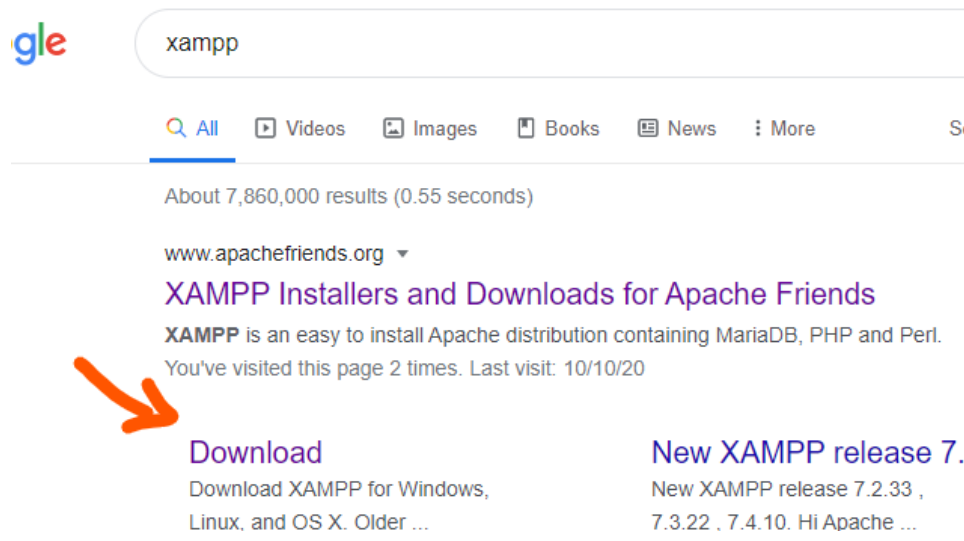
- 1- XAMPP 同时安装这两者
- 2- 数据库引擎（MySQL）
- 3- 以及数据库管理工具 (phpMyAdmin)

5



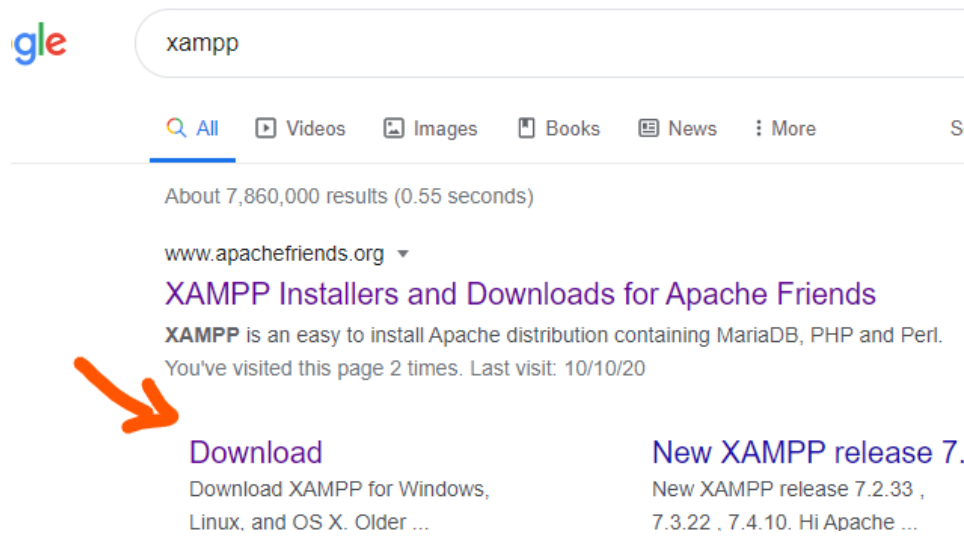
Setting up DB server on your PC

- 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 databale with your partner
- phpMyAdmin needs PHP engine to run
- XAMPP is an application package installer that installs all of those for us



在您的电脑上设置数据库服务器

- 我们需要安装 MySQL 数据库服务器。它是一种关系型数据库引擎，用于执行我们的 SQL 查询。
- 此外，我们还需要 phpMyAdmin——一款基于 Web 的数据库管理工具。由于它是基于 Web 的，因此您可远程使用，或与合作伙伴共享数据库。
- phpMyAdmin 的运行依赖 PHP 引擎。
- XAMPP 是一个应用软件包安装程序，可为我们自动安装上述所有组件。



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

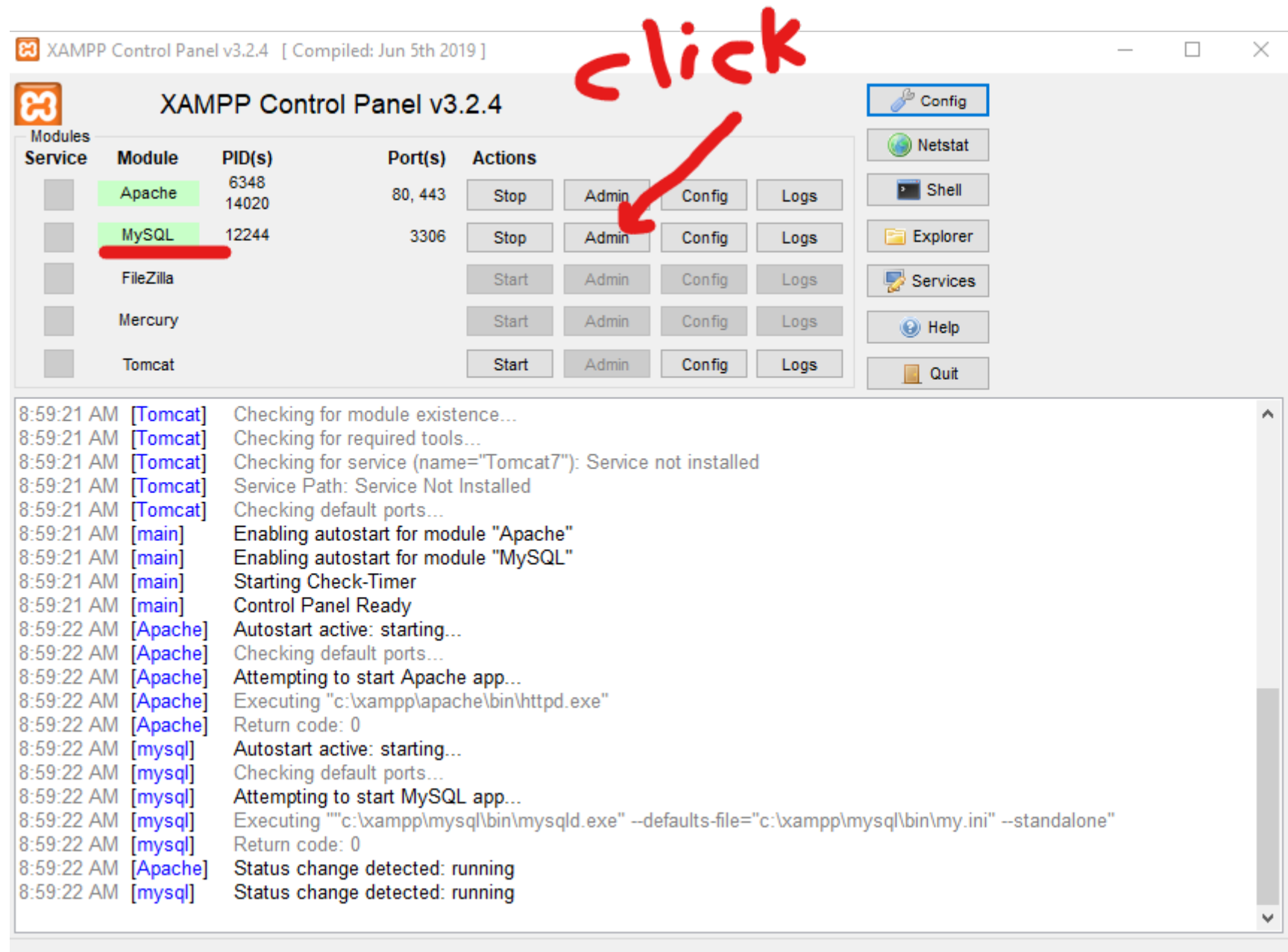


- 1: 单击 “admin” 将跳转至 phpMyAdmin
- 2— 单击 “shell” 将跳转至 MySQL 命令行界面



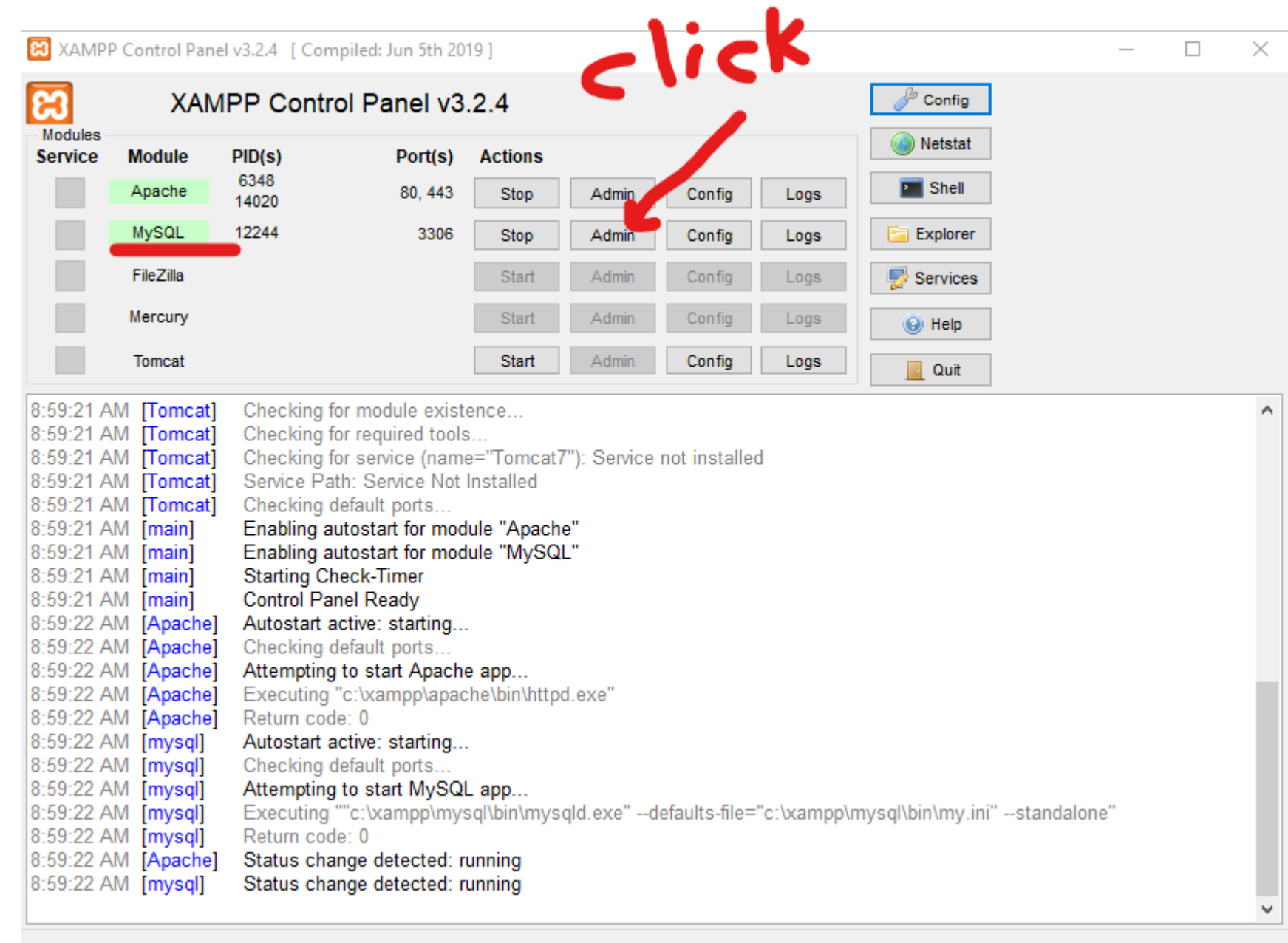
2- goto phpMyAdmin and to create a DB and a table

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

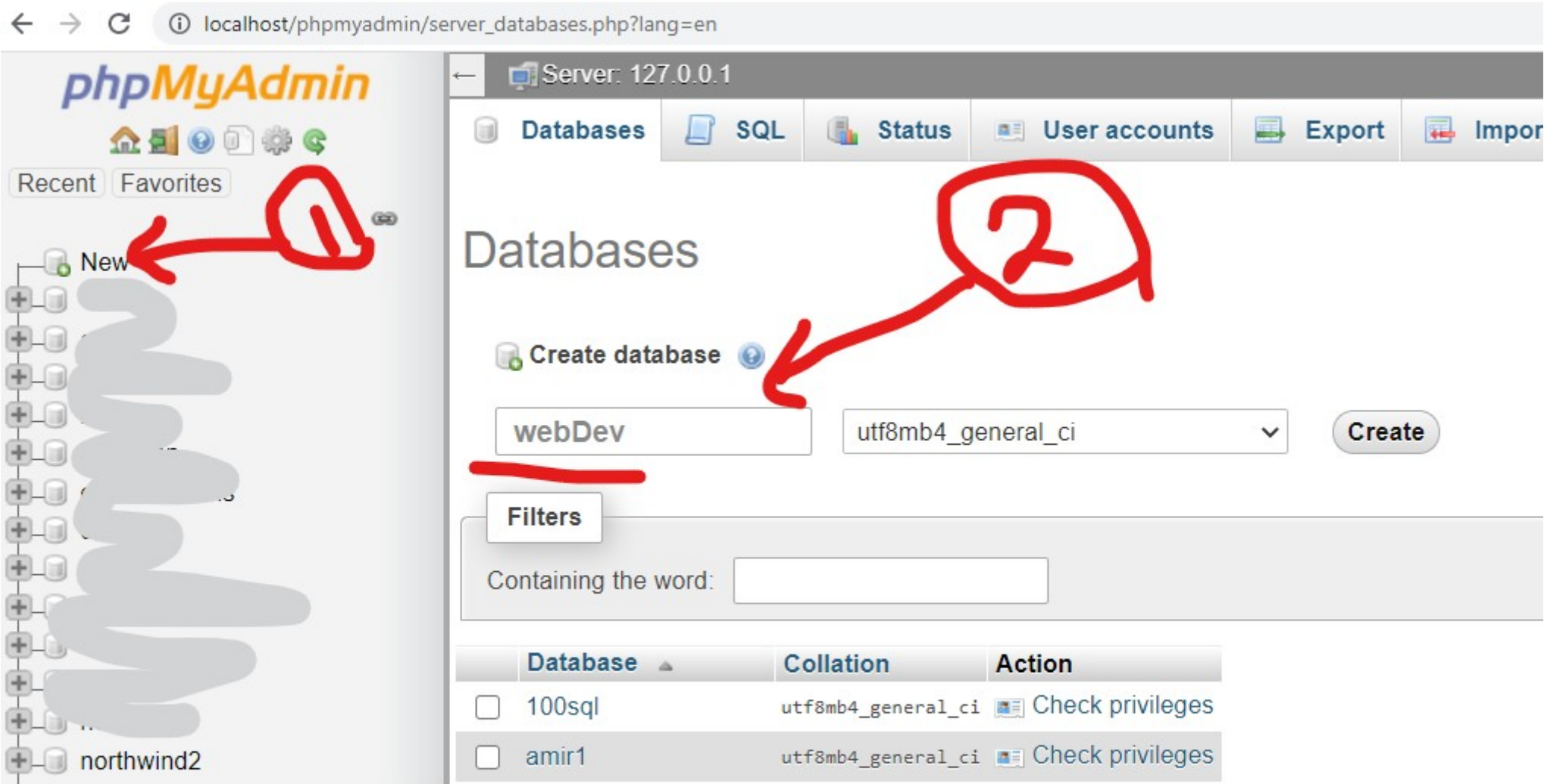


2. 进入 phpMyAdmin 并创建数据库及数据表

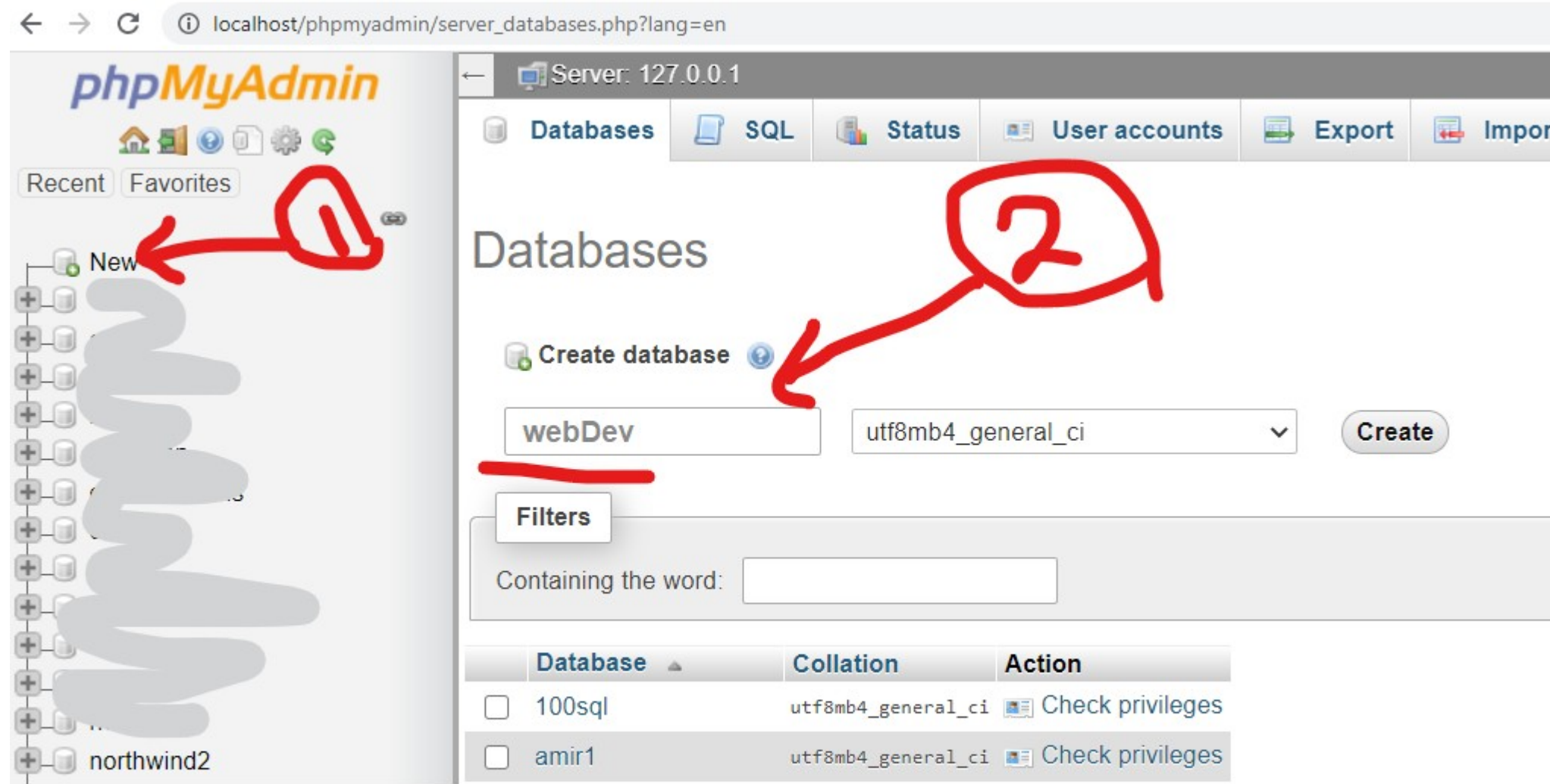
- 单击该链接即可加载
- <http://localhost/phpmyadmin/>
- (请确保 MySQL 正在运行, XAMPP)



3- create a DB in phpmyadmin



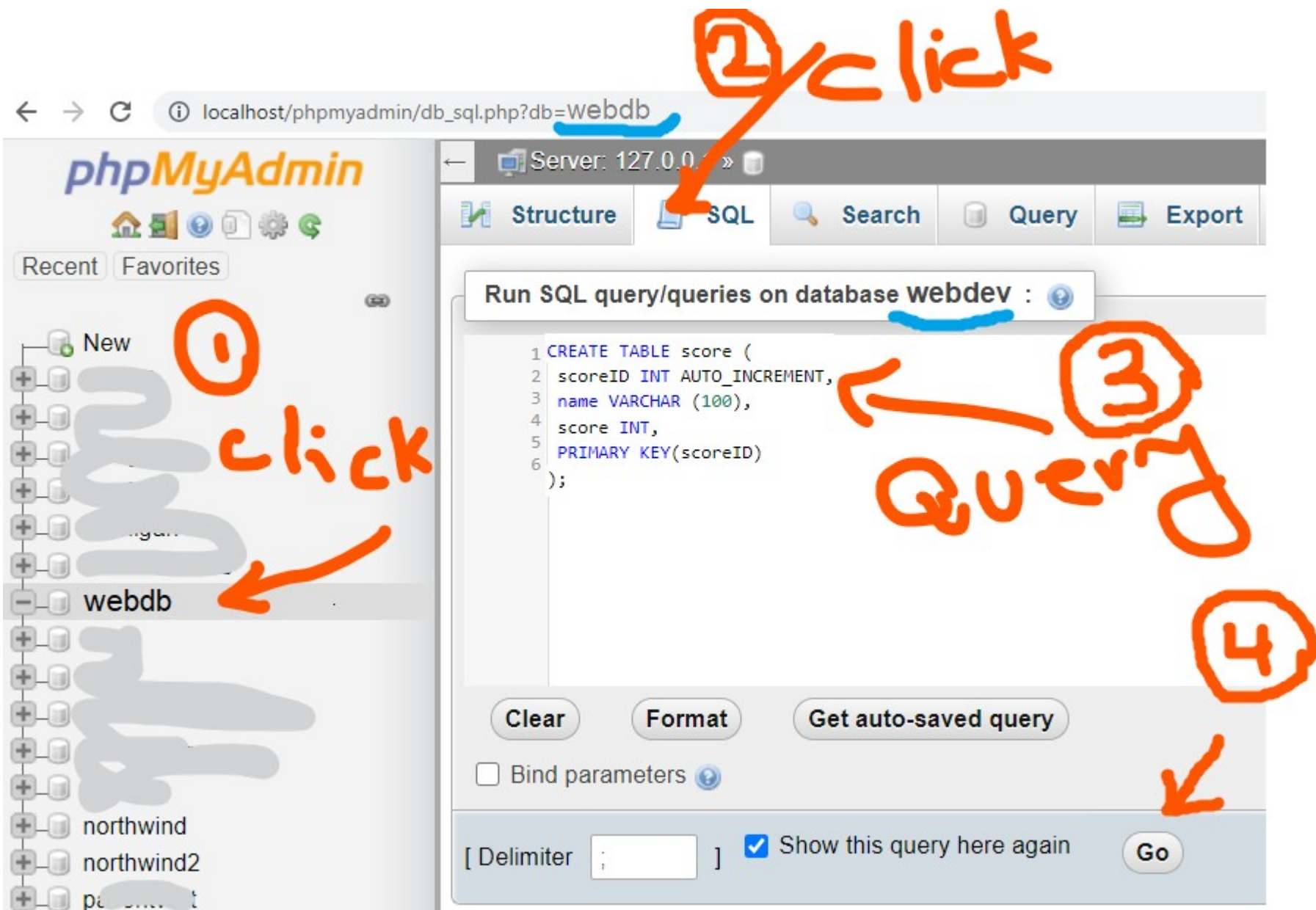
3. 在 phpMyAdmin 中创建数据库



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)  
);
```

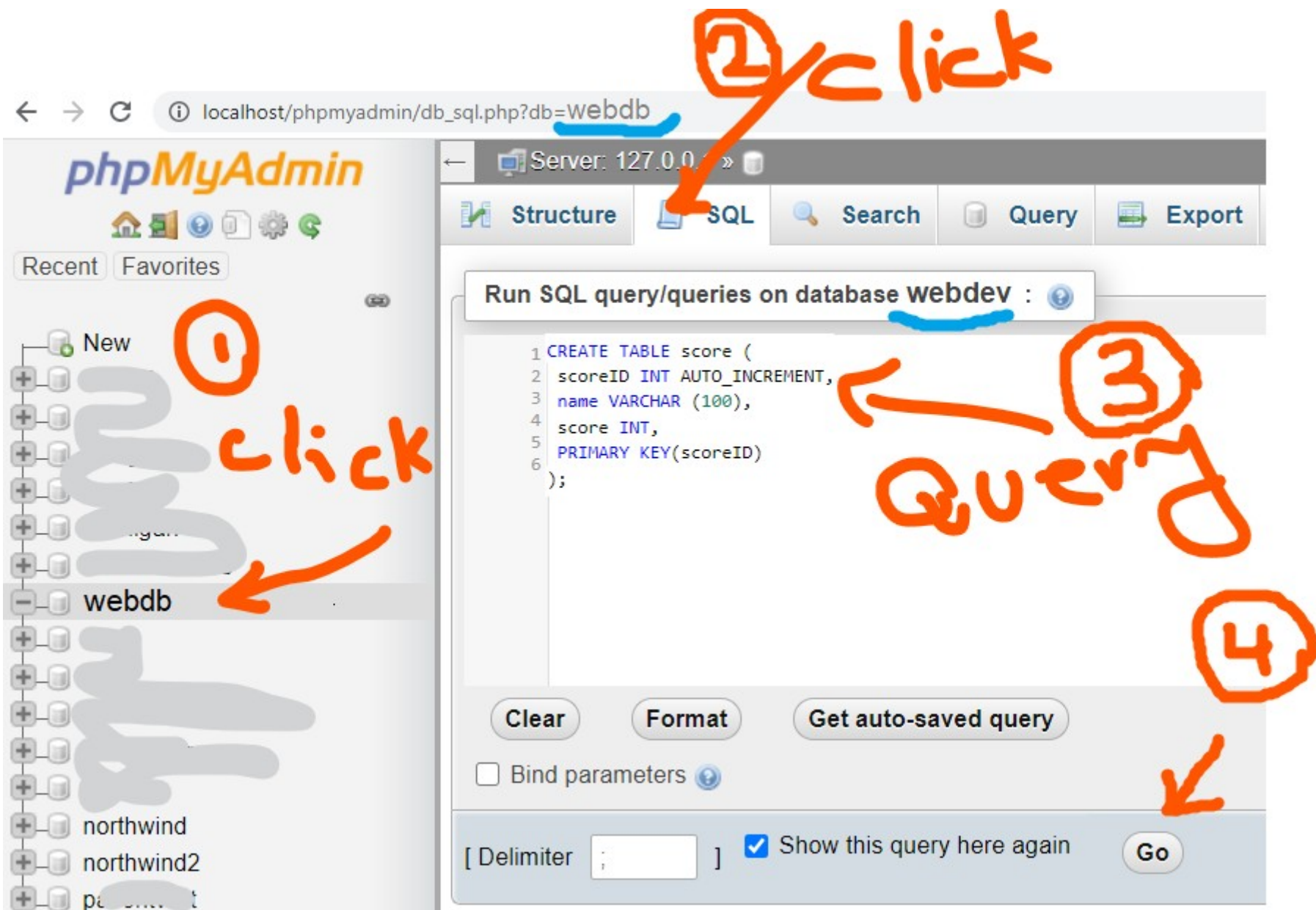
- Note: It is ok to have column names with the same name as the table name, both are score
- the Now our DB is ready!



4. 在该数据库中创建一张表，用于存储分数。

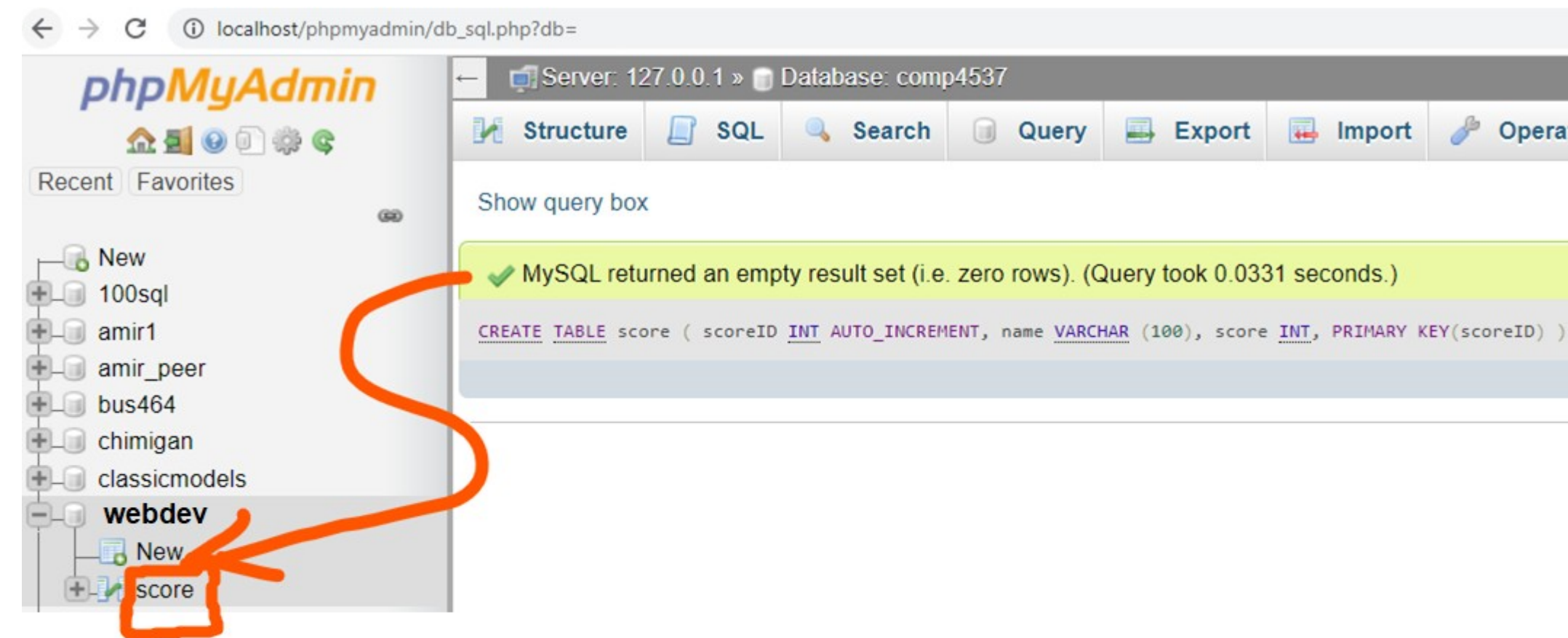
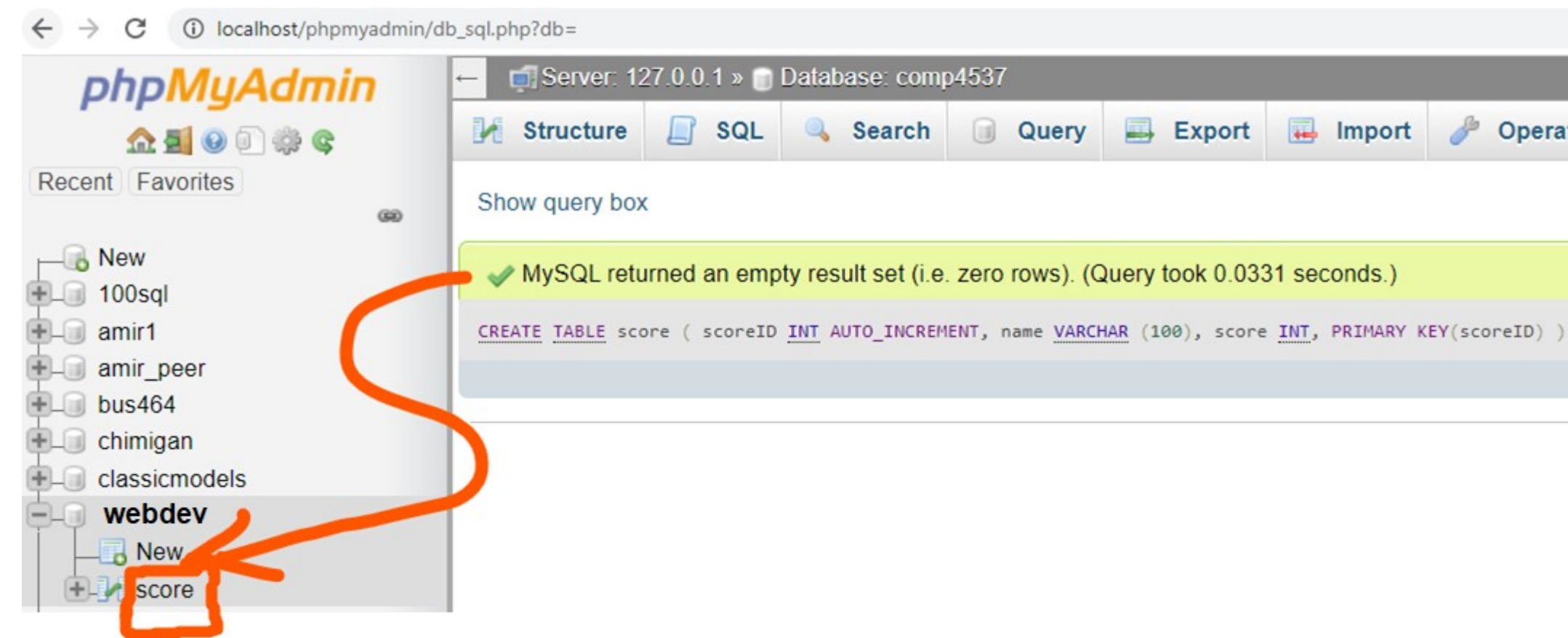
```
CREATE TABLE score (  
  scoreID INT AUTO_INCREMENT,  
  name VARCHAR (100),  
  score INT,  
  PRIMARY KEY(scoreID)  
);
```

- 注意：列名与表名相同是允许的，两者均可命名为 score
- 现在，我们的数据库已准备就绪！



5- notice the confirmation message on creation of table

5 — 注意创建表时显示的确认消息

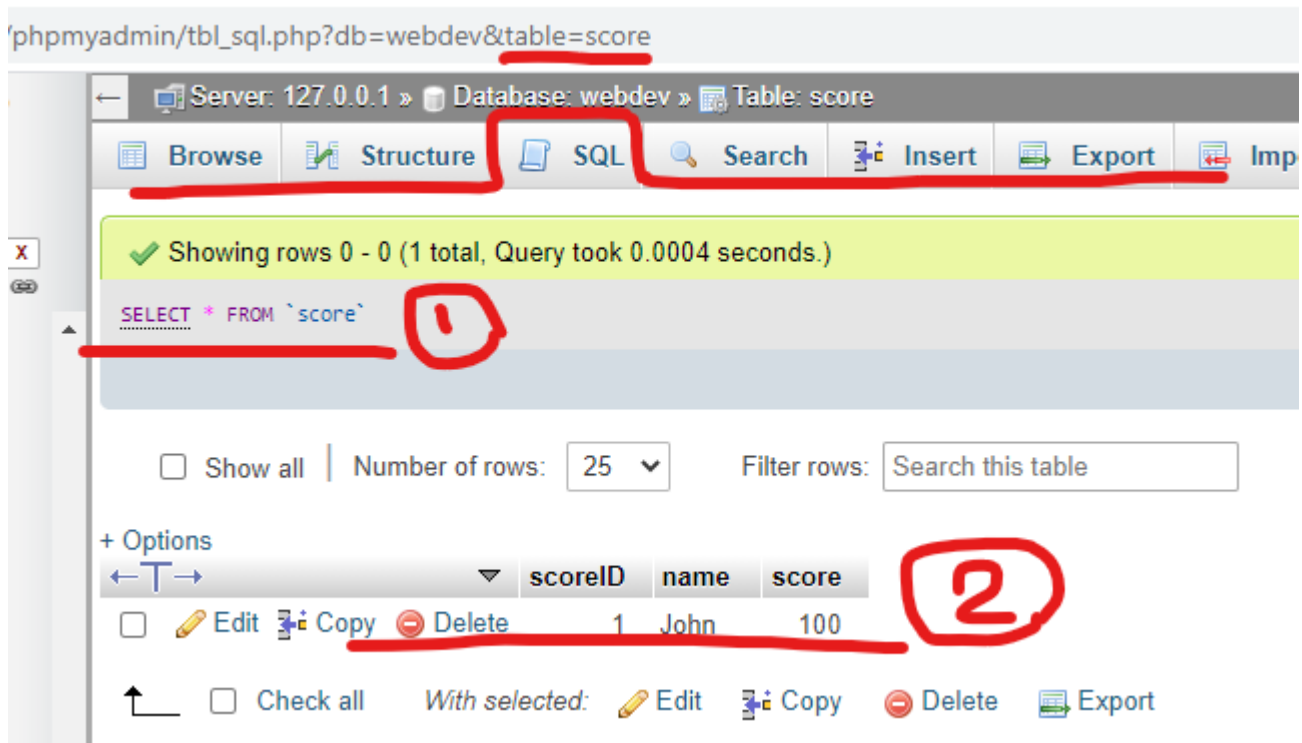


6- run few SQL statements on table you have created

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!

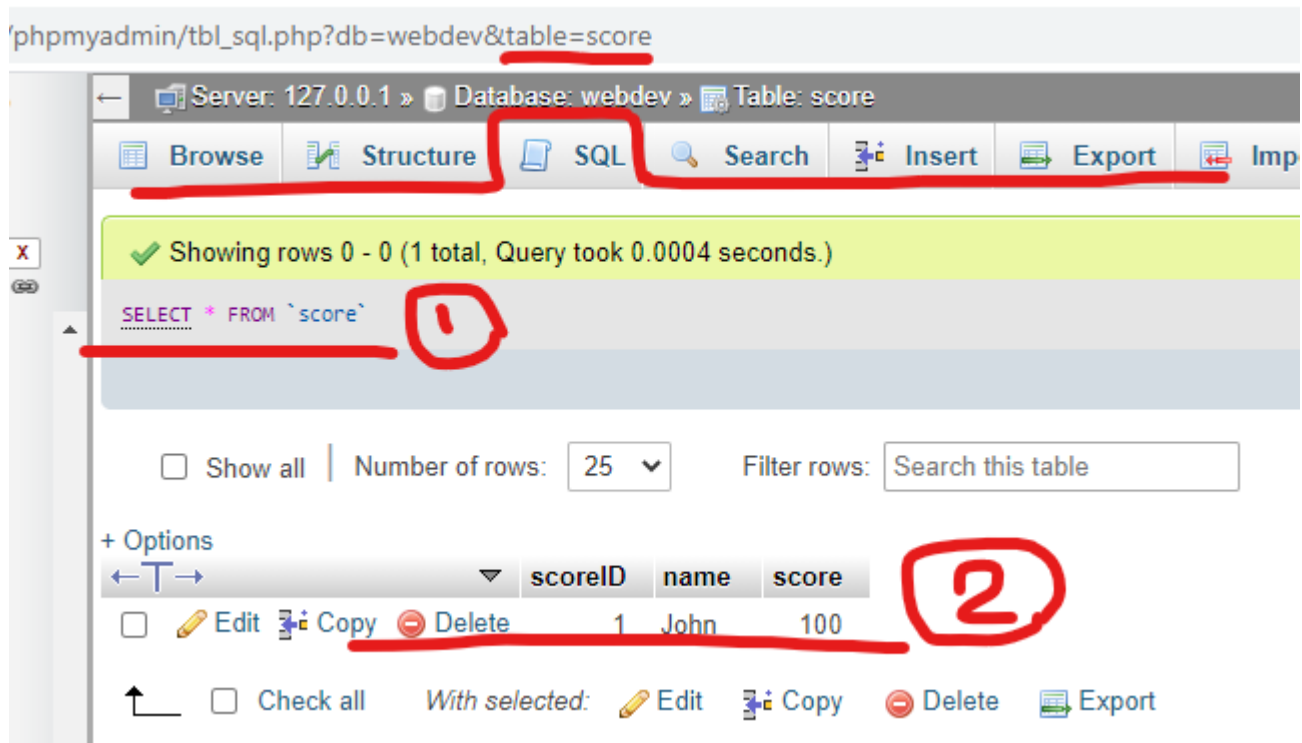


6- 在您创建的表上执行若干 SQL 语句

0- 确保已选中您所创建的数据库, 以便在该数据库上执行 SQL 查询

1- 输入您的查询语句 (SQL) tab

2- 点击“执行”按钮, 查询结果即会显示!



6

Connecting to mySQL DB using nodejs

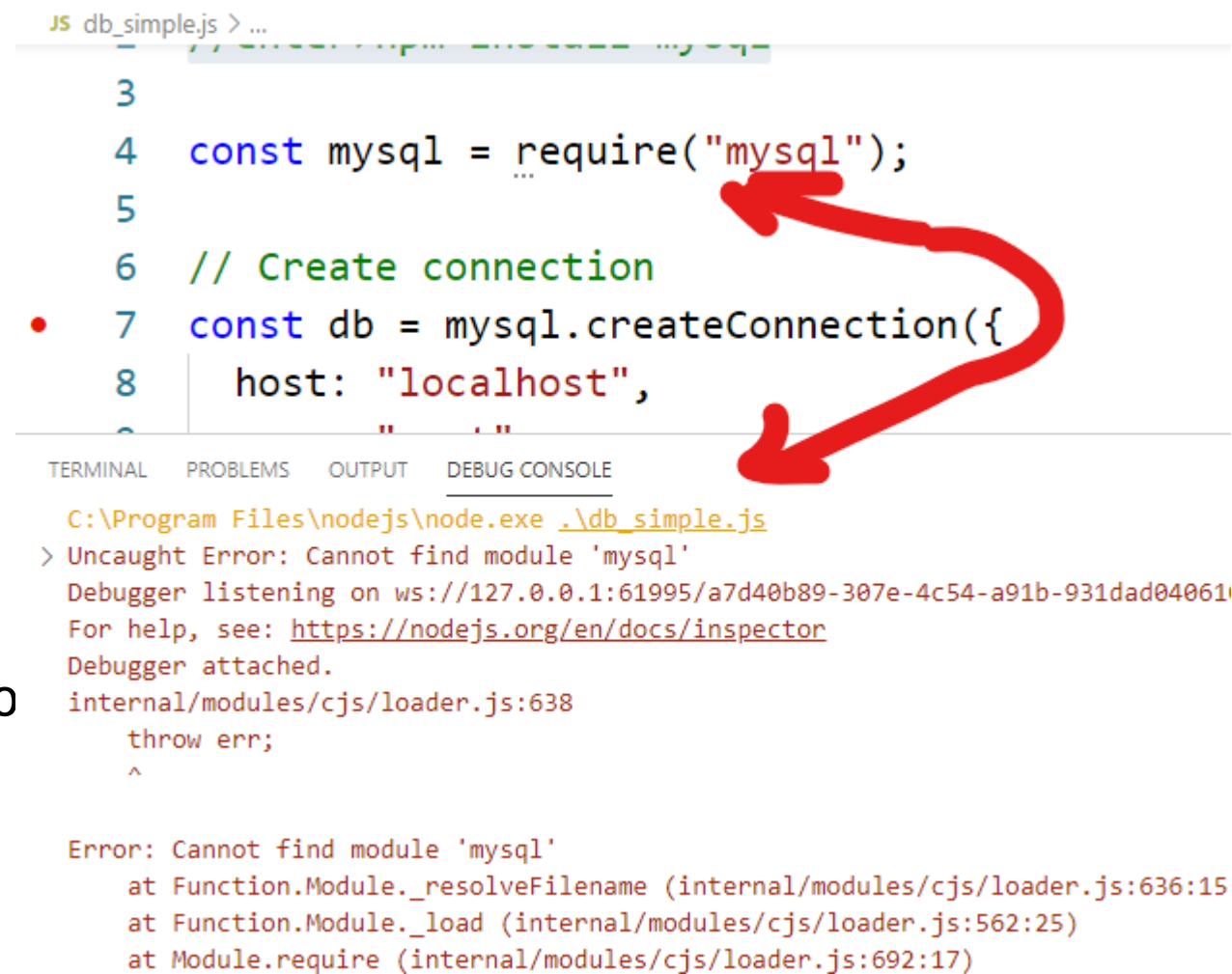
6

使用 Node.js 连接到 MySQL 数据库

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

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



```
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
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623

```

- 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");
  });
});
```

- 首先创建一个 MySQL

对象

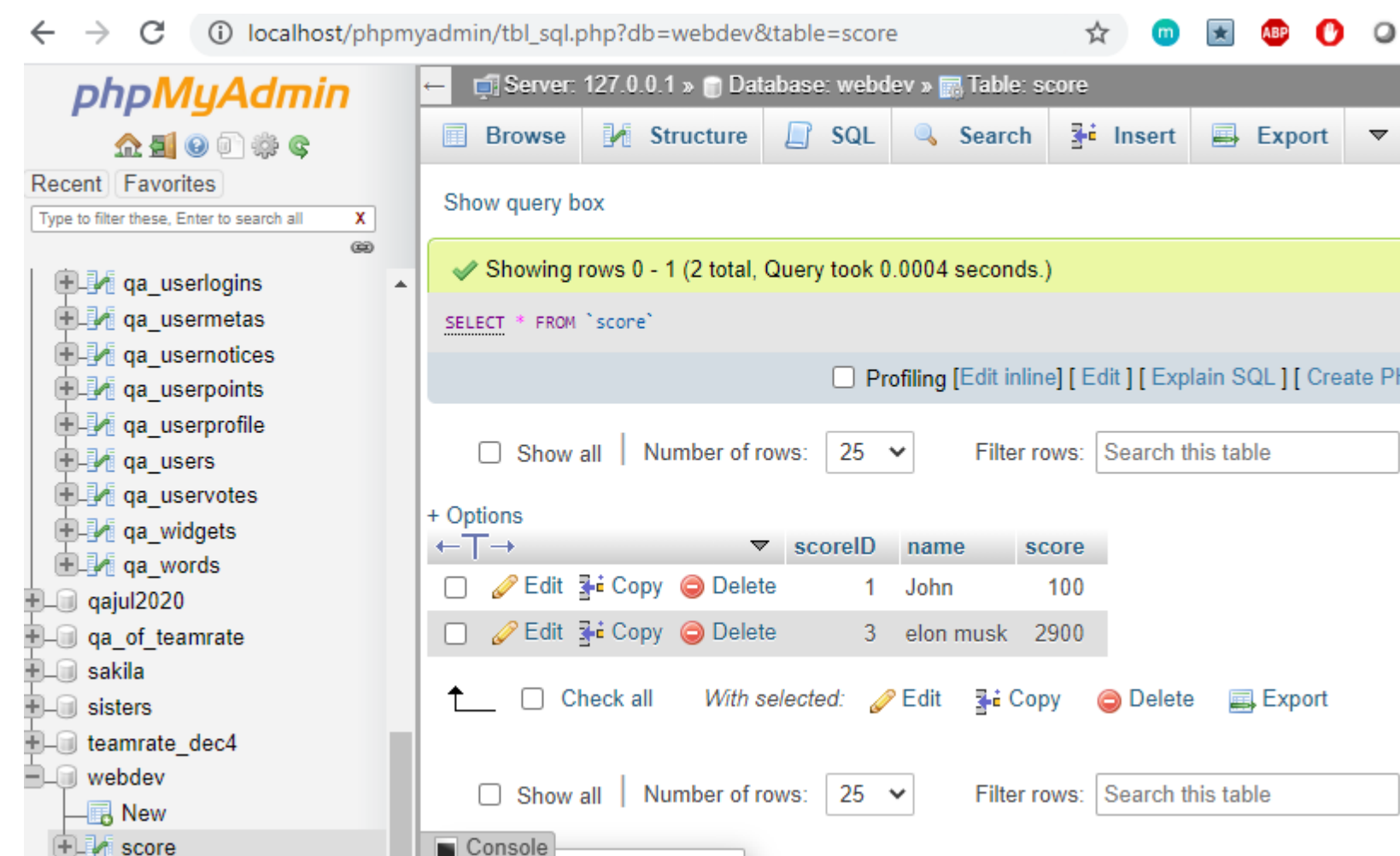
- 然后建立连接
- 并执行一条 SQL

语句

```
// 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");
  });
});
```

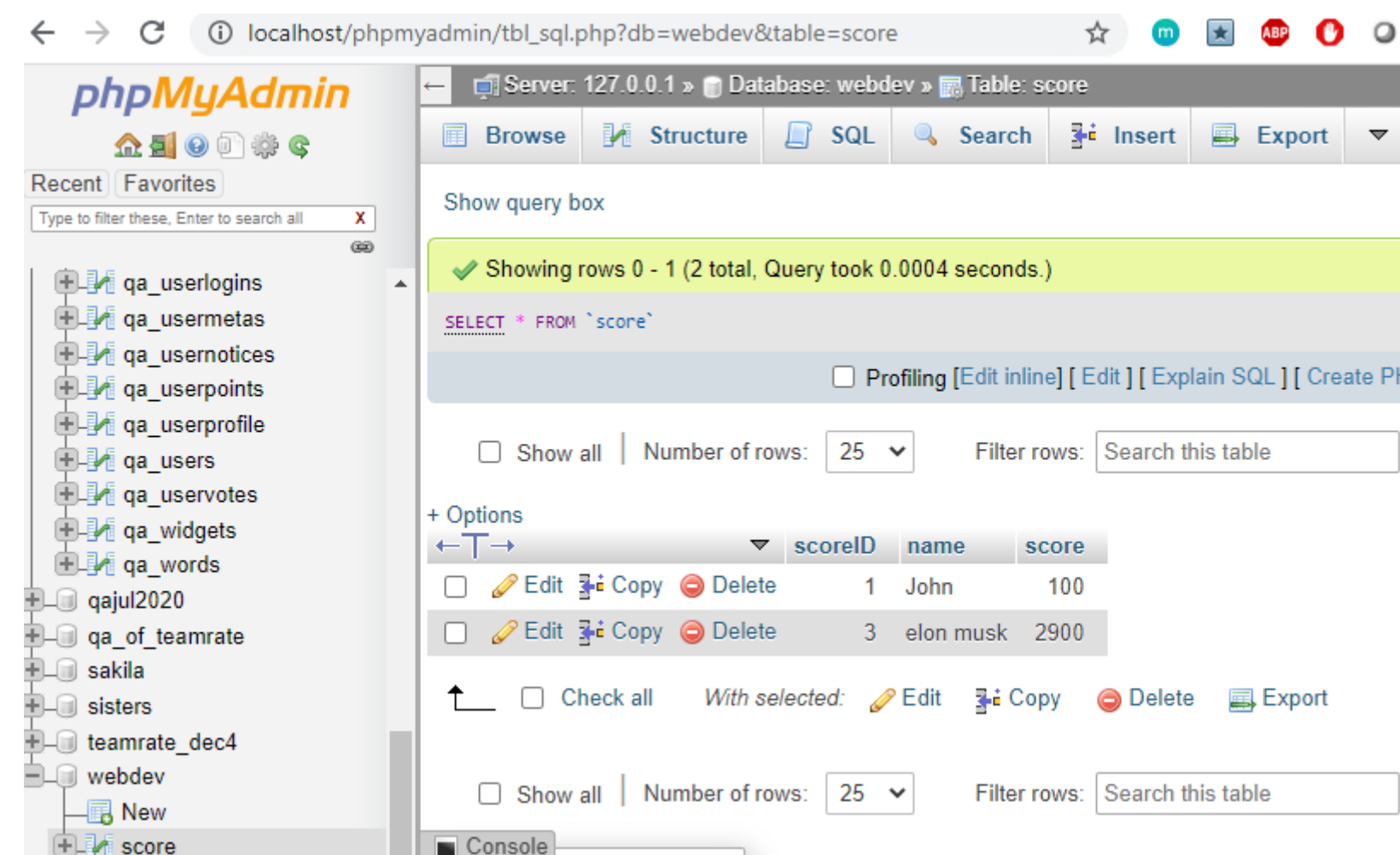
Check your table to verify the SQL ran successfully on node

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



请检查您的数据表，以确认 SQL 已在节点上成功执行。

- 如您所见
- 分数为
- 埃隆·马斯克为已录入我们的数据表中
- 成功！



Hosting Node js, MySQL on remote server

7



Please watch this video
Hosting nodejs MySql remotely CPanel Shared Hosting.mp4

在远程服务器上托管 Node.js 和 MySQL

7



请观看此视频
远程托管 Node.js 与 MySQL (cPanel 共享主机) .mp4

Beginner
's guide

Node js +
MySQL

cPanel

remote
server


Node.js +
MySQL


cPanel

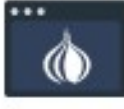
远程
服务器


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


如何在共享主机账户中托管 Node.js 和 MySQL


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

source

- chatGPT 3.5 used to generate partial code for this set

源代码

- ChatGPT 3.5 曾用于为此套代码生成部分代码