ΕΠΛ425

Τεχνολογίες Διαδικτύου

(Internet Technologies)

Introduction to Back-End Development:

PHP & MySQL

Διδάσκων

Δρ. Χριστόφορος Χριστοφόρου

christophoros@cs.ucy.ac.cy

Goals

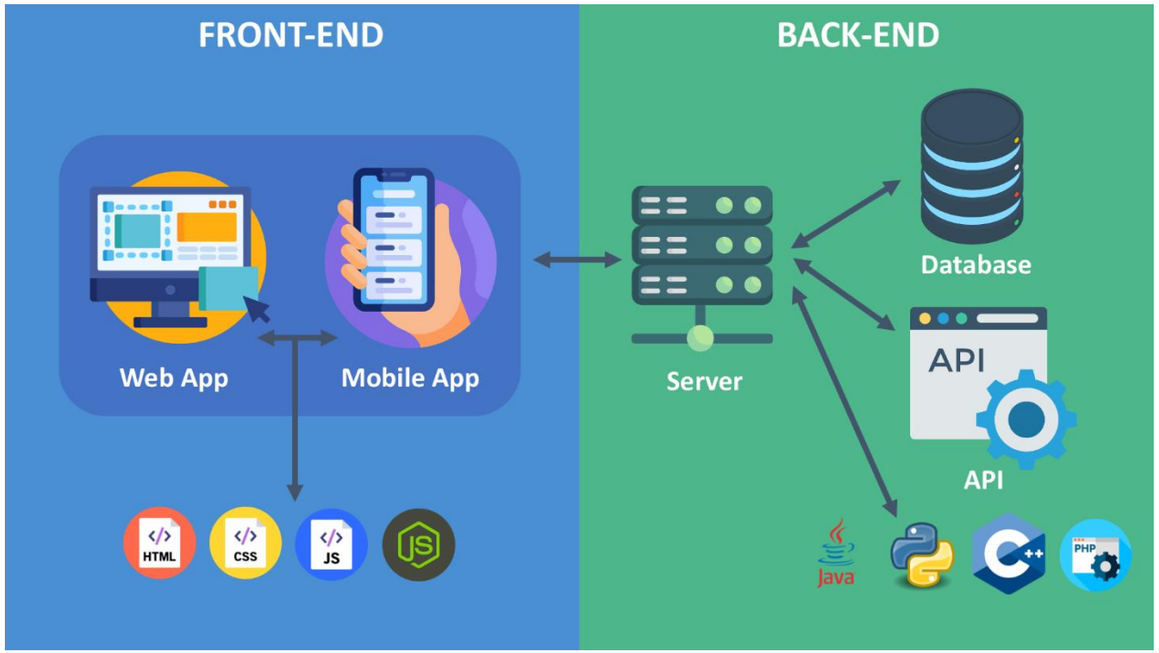
Introduction to Back-End Development:

- **PHP: Hypertext Preprocessor** for **server-side processing**, and **database access**.
- **MySQL** for **storing** and **managing** **data** in web applications.

Back-End vs Front-End Development

- **Front-end development** refers to the part of web development that **deals with the user interface** and **client-side scripting**, such as HTML, CSS, and JavaScript.

- **Back-end development** provides the **necessary functionality** to support the **creation of dynamic web pages** by the front-end.



Back-End Development

- Back-end development focuses on the server-side of web development. This includes:
  - **Server-side programming (processing)**, which involves executing code and performing tasks on the server, and
  - **Database management**, which involves **storing** and **retrieving data from a database**.
- Back-end development is also responsible for **user authentication**, such as login and registration systems.

Server Side Programming

- In web development, **server-side programming** plays a **crucial role** mainly in **processing user data**, and **communicating with a database**.
- Server-side programming is the **process** of **executing code** on a **web server** to **generate dynamic content** or **perform tasks** that cannot be accomplished with client-side scripting.
- There are **several programming languages** commonly used for server-side programming, each with its own strengths and weaknesses.

Server Side Programming

- Some of the most popular server-side programming languages include: **PHP, Ruby on Rails, Python, Java, Node.js,** and **C#**
- The **choice of language** often depends on the **requirements of the project**, the **skills of the development team**, and the *availability of resources and support**.
- **For server side programming we will use PHP!**

* By "availability of resources and support," we mean the availability of tools, libraries, frameworks, documentation, and community support that are necessary to develop and maintain a software project using a particular programming language. For example, if a development team is considering using a **relatively new programming language** that does not have a large user base or a well-established community, they may have difficulty finding libraries, frameworks, or other tools that they need for their project. This could result in **additional development time** and cost, or a **lower-quality** end product.

Databases in Web Development

- **Databases** are an essential part of web development, **providing** a **structured** and **organized way** to **store**, **manage**, and **retrieve data**.
- There are several popular database management systems (DBMS) used in web development, each with its own strengths and weaknesses. Some of the most popular DBMS include:
  - **MySQL**: A popular open-source SQL database management system used by many web applications, including WordPress, Drupal, and Joomla.

Databases in Web Development

- **PostgreSQL**: Another popular open-source SQL database management system known for its scalability, reliability, and data integrity.
- **MongoDB**: A popular open-source NoSQL document database used by many web applications, including Forbes, eBay, and LinkedIn.
- Other popular DBMS include **Oracle**, **Microsoft SQL Server**, and **Redis.**

- **We will use MySQL!**

Server Side Programming with PHP

- PHP is a popular server-side programming language used for web development to create dynamic web pages.
- It was first released in 1995 and has since become one of the most widely used languages for web development, **with over 80% of all websites on the internet using PHP** in some form.
- PHP is known for its **ease of use**, **flexibility***, and **scalability**, making it a **popular choice** for building web applications of **all sizes** and **complexity**.

*PHP can be **integrated with many other technologies** and **languages**, such as HTML, CSS, JavaScript, SQL databases, and other web services. PHP can also run on a **variety of platforms** and **web servers**, including Windows, Linux, and macOS, making it a versatile choice for web development projects.

Server Side Programming with PHP

- **PHP syntax** is based on C and includes several common programming elements, such as **variables**, **data types**, **operators**, **control structures**, and **functions**.
- **PHP code** is **included** in **<?php .... ?>** tags, and the default **file extension** for PHP files is **.php**

myphp.php

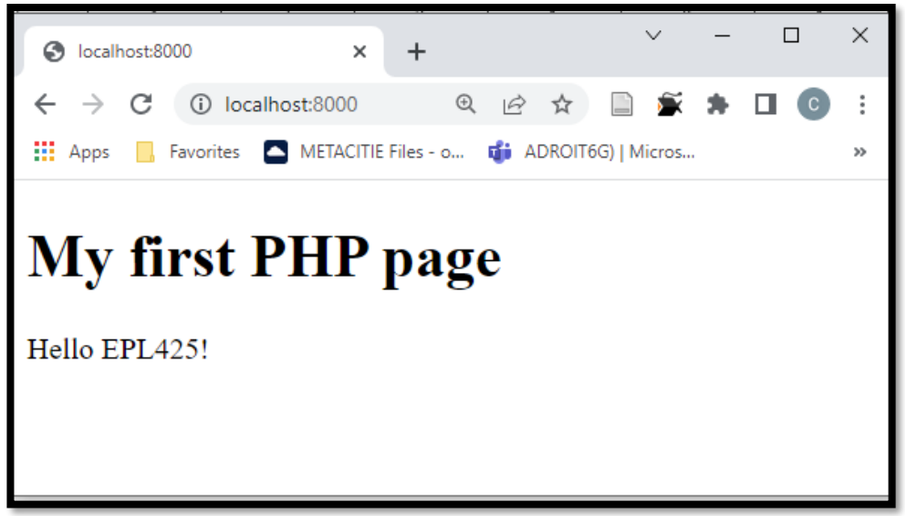
```
<?php
  // Write your PHP code here
?>
```

Server Side Programming with PHP

- PHP scripting code **can be embedded** in HTML code and **run on the server before being sent** to the **client's browser**.

index.php

```
<!DOCTYPE html>
<html>
<body>

    <h1>My first PHP page</h1>

    <?php
    echo "Hello EPL425!";
    ?>

</body>
</html>
```



Variables in PHP

- A variable is defined using the **$** symbol **followed by** the **variable name**.
- PHP supports several data types, including **integers**, **floats**, **strings**, **booleans**, **arrays**, and **objects**.
- Here are some examples of defining and using variables in PHP:

```php
<?php
$name = "Chris"; // String Variable
$age = 25;       /* Integer Variable */
$balance = 3.14;
$is_valid = true;
$numbers = array(1, 2, 3);
$person = new Person($name, $age);
?>
```

- Statements end with a semicolon ;
- Comments can be added using double slashes // or with a block comment /* … */

Variables in PHP

- At the right side is an example of how the class Person can be defined in PHP.

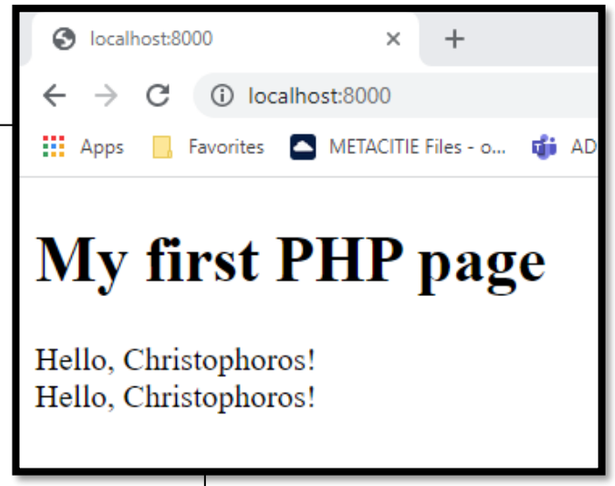
```php
<?php
class Person
{
    // Properties
    public $fName;
    public $age;

    //constructor function
    function __construct($fName, $age)
    {
        $this->fName = $fName;
        $this->age = $age;
    }
}
?>
```

Outputting Data in PHP

- In PHP, data can be output to the browser using the **echo** keyword. Here are some examples of outputting data in PHP:

index.php

```
<!DOCTYPE html>
<html>
<body>
    <h1>My first PHP page</h1>

    <?php
    echo "Hello, Christophoros! <br>"; // outputs "Hello, Christophoros!"

    // Note: In PHP, the . operator is used for string concatenation.
    $name = "Christophoros";
    echo "Hello, " . $name . "!"; // outputs "Hello, Christophoros!"
    ?>

</body>
</html>
```



Control Structures in PHP

If/elseif/else statement

```php
<?php
$age = 25;
if ($age >= 65) {
    echo "You are wise.";
} elseif ($age >= 18)  {
    echo "You are an adult.";
} else {
    echo "You are not an adult.";
}

?>
```

while Loop

```php
<?php
$i = 0;
while ($i < 10) {
    echo $i;
    $i++;
}
?>
```

for Loop

```php
<?php
for ($i = 0; $i < 10; $i++) {
    echo $i;
?>
```

Classes and Objects in PHP

- **Objects** in PHP are used to create instances of classes.
- A class is a blueprint for creating objects, and it contains **properties** and **methods**.
- To **access** the object's **properties** or **invoke** its **methods** we use the **arrow operator ->**
- At the next slide there is an example of defining and using a class in PHP

Objects in PHP

```php
<?php
class Person {
    public $name;
    public $age;

    public function __construct($name, $age) {
        $this->name = $name;
        $this->age = $age;
    }

    public function say_hello() {
        echo "Hello, my name is " . $this->name . " and I am " . $this->age . " years old.";
    }
}

$person = new Person("Chris", 44);
$person->say_hello(); // outputs "Hello, my name is Chris and I am 44 years old."
?>
```

Superglobals in PHP

- Superglobals in PHP are **built-in variables** (thus they start with $) that are **always available** in any scope throughout a PHP script.
- They are always **prefixed** with an **underscore _ character**, and they are **always uppercase**.
- Superglobals are commonly used in web applications for **handling user input**, **session management**, **server information** **and more**.

Superglobals in PHP

- There are several superglobals in PHP, like:
  - **$_GET**: Contains variables passed to the current script via the URL parameters.
  - **$_POST**: Contains variables passed to the current script via HTTP POST method.

Example of using $_GET in PHP

- Also recall "HTML Forms …" Lecture….

```
<!-- Example form with GET method using the default submission -->
<form action="example.php" method="get">
    Name: <input type="text" name="name"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Submit">
</form>
```

```
<?php
$name = $_GET['name'];
$email = $_GET['email'];
// Then do what ever you want with the data
?>
```

Build in Functions in PHP

- PHP provides a set of pre-defined functions that can be used to perform various tasks, including:
  - **String manipulation** - functions to find and replace substrings, concatenate strings, trim whitespace, and more.
  - **Array manipulation** - functions to add, remove, and modify elements in an array, sort and search arrays, and more.
  - **File handling** - functions to read and write files, create and delete directories, and more.
  - **Database access** - functions to connect to databases, perform queries, and retrieve data.

Build in Functions in PHP

- **HTTP handling** - functions to retrieve and send HTTP requests, handle cookies and sessions, and more.
- **JSON handling** - functions to encode and decode JSON data.
- **Object-oriented programming** - functions to create and manipulate objects, work with classes and inheritance, and more.
- **Security** - functions to validate input, sanitize data, and protect against SQL injection and other security threats.

- These are just a few examples of the many built-in functions available in PHP.

Build in Functions in PHP

In this lecture the goal is to **connect** our PHP code **with a MySQL Database!!!**

Thus, we will get familiar first with the build-in functions that deals with Database access → *Functions to **connect** to **databases**, **perform queries**, and **retrieve data***

There are two approaches to access the Database:

- ***Procedural Approach***
- ***Object-Oriented Approach***

MySQL Database access build in Functions in PHP

Procedural Approach

mysqli_connect(): This function is used to **establish a new connection** to a MySQL **database**.

- It **takes four parameters**: the **hostname**, **username**, **password** (here we assume that we did not have a password), and **database name**.
- It **returns** an object representing the connection to the database server.
- For example:

```
$conn = mysqli_connect("localhost", "root", "", "mydb");
```

MySQL Database access build in Functions in PHP

Procedural Approach

mysqli_query(): This function is used to **execute a MySQL query** on a database connection.

- It takes two parameters: the **connection variable** and the **SQL query**. For example:

```
$result = mysqli_query($conn, "SELECT * FROM mytable");
```

MySQL Database access build in Functions in PHP

Procedural Approach

The **mysqli_query()** function **returns different results** based on the **type of query** executed:

- For **SELECT** queries:
  - **On success**: returns a **mysqli_result object** containing the returned **result set**.
  - **On failure**: returns **False**.
- For other query types (**INSERT**, **UPDATE**, **DELETE**, or **CREATE**):
  - **On success**: returns **True**.
  - **On failure**: returns **False**.

MySQL Database access build in Functions in PHP

Procedural Approach

mysqli_fetch_assoc(): This function is used to **retrieve a single row** from a **mysqli_result object** result set (`$result`) as an **associative array**.

- It takes one parameter: the **result set variable**. For example:

```
while ($row = mysqli_fetch_assoc($result)) {
  $firstname = $row["firstname"];
  $lastname = $row["lastname"];
  echo "User Details: $firstname $lastname";
}
```

MySQL Database access build in Functions in PHP

Procedural Approach

mysqli_num_rows(): This function is used to **get the number of rows** in a **mysqli_result object** result set.

- It takes one parameter: the **result set variable**. For example:

```
if (mysqli_num_rows($result) > 0) {
  // Output data of each row
  while ($row = mysqli_fetch_assoc($result)) {
    $firstname = $row["firstname"];
    $lastname = $row["lastname"];
    echo "User Details: $firstname $lastname";
  }
}
else {
  echo "No results found";
}
```

MySQL Database access build in Functions in PHP

Procedural Approach

mysqli_close(): This function is used to **close a MySQL database connection**.

- It takes one parameter: the **connection variable**. For example:

```
mysqli_close($conn);
```

MySQL Database access build in Functions in PHP – Object-Oriented Approach

- You can also use an **object-oriented approach** to access the MySQL database, which involves **creating** a **mysqli object ($conn)** to **create the database connection.**
- In this example, a **mysqli** object is created in variable **$conn** with the **new keyword** and **the connection parameters** are **passed** as **arguments** to the **constructor**.

```
// Create connection like mysqli_connect()
$conn = new mysqli("localhost", "root", "", "mydb");
```

- Then you can **access properties** or **call methods** **on that $conn object** to **execute queries** and **retrieve data**.

MySQL Database access build in Functions in PHP – Object-Oriented Approach

- Once the **connection is established**, you can **use** the **object's method query()** to **execute queries** and **retrieve data** **(like mysqli_query())**.
- Here is an example of how to execute a **query** and retrieve data using the **$conn** mysqli object:

```
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);
```

MySQL Database access build in Functions in PHP – Object-Oriented Approach

- The **query() method** is called on the **$conn mysqli object** to execute an SQL query and return a result set (**$result**).
- The **num_rows property** of the **$result** is then checked to determine if **any rows were returned.**
- If yes, the **fetch_assoc() method** is **called on** the **$result** to **retrieve each row** as an **associative array**.

```
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output data of each row
    while($row = $result->fetch_assoc()) {
      $firstname = $row["firstname"];
      $lastname = $row["lastname"];
      echo "User Details: $firstname $lastname";
    }
}
else {
    echo "No results found";
}
```

MySQL Database access build in Functions in PHP – Object-Oriented Approach

- Once you are **finished using the database connection**, you can close it using the **close()** method on the **$conn** mysqli object:

```
$conn->close();
```

MySQL Database access build in Functions in PHP

- A comprehensive list of MySQLi database access build in Functions in PHP can be found here both in procedural and object oriented approach.

MySQL Database access build in Functions in PHP

Some more properties and functions

->connect_errno is a property that stores **the error code** of the most recent call used to establish a connection to the database

->connect_error is a property stores **the error description** the most recent call to connect to the database

```
// Create a new mysqli object and connect to the database
$conn = new mysqli("localhost", "username", "password", "database");

// Check for errors when connecting
if ($conn->connect_errno) {
    echo "Failed to connect to MySQL: " . $conn->connect_error;
    exit();
}

// Connection successful, continue with other database operations
```

MySQL Database access build in Functions in PHP
Some more properties or functions

Here are some **common error codes** (integer values) that may be returned by **connect_errno.** Next to it is the error description that will be stored in **connect_error** property:

- **1045**: Access denied for user (wrong username or password)
- **1049**: Unknown database (database does not exist)
- **2002**: Can't connect to local MySQL server (connection refused)
- **2003**: Can't connect to MySQL server on (hostname) (connection refused)

If there are no errors during the connection process, the **connect_errno** property will be set to **0**.

MySQL Database access build in Functions in PHP
Some more properties or functions

->data_seek(row_number) is a method that is used to **move the internal pointer of a result set** to a specified row number.

->fetch_object() is a method that **returns the current row** of data **as an object**

```
// Create a new mysqli object and execute a SELECT query
$conn = new mysqli("localhost", "username", "password", "database");
$result = $conn->query("SELECT * FROM users");

// Move the internal pointer to the third row of the result set. The
// first row has index 0.
$result->data_seek(2);

// Get the current row of data as an object
$row = $result->fetch_object();

// Output the data as a JSON object
echo json_encode($data);
```

```php
<?php
// MySQL server configuration
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mydb";
// Create a connection to the MySQL server
$conn = new mysqli($servername, $username, $password, $dbname);

// Check for errors when connecting
if ($conn->connect_errno) {
    echo "Failed to connect to MySQL: " . $conn->connect_error;
    exit();
}

// Perform a query on the database
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);

// Output the results of the query
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "First Name: " . $row["firstname"] . " Last Name: " . $row["lastname"] . "<br>";
    }
}
else {
    echo "No results found";
}

// Close the database connection
$conn->close();
?>
```

First define the MySQL server configuration and create a **new mysqli** object (**$conn**) to connect to the server using the provided **servername**, **username**, **password**, and **database** name. Note that, in this example we **have not set a root password** for MySQL in XAMPP

In this code, we use the **default MySQL username ("root")** and an **empty password** ("") to **create** a **new MySQLi object** and **connect** to the **"mydb" database** on the server with name **"localhost"**. *Note that if you set a password for the root user, you will need to use that password in the **$password** variable.*

Example of PHP code connecting to a MySQL Database

Example of PHP code connecting to a MySQL Database

```php
<?php
// MySQL server configuration
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mydb";
// Create a connection to the MySQL server
$conn = new mysqli($servername, $username, $password, $dbname);

// Check for errors when connecting
if ($conn->connect_errno) {
    echo "Failed to connect to MySQL: " . $conn->connect_error;
    exit();
}

// Perform a query on the database
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);

// Output the results of the query
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "First Name: " . $row["firstname"] . " Last Name: " . $row["lastname"] . "<br>";
    }
}
else {
    echo "No results found";
}

// Close the database connection
$conn->close();
?>
```

Check if the **connection was successful**, and if so, **perform** a **query** on the database using the **SELECT SQL statement** to retrieve all data from a table named **"mytable"**.
The **$conn->query($sql)** method is used to execute the SQL statement on the database, and the result set is stored in the **$result variable**.

Example of PHP code connecting to a MySQL Database

```php
<?php
// MySQL server configuration
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mydb";
// Create a connection to the MySQL server
$conn = new mysqli($servername, $username, $password, $dbname);

// Check for errors when connecting
if ($conn->connect_errno) {
    echo "Failed to connect to MySQL: " . $conn->connect_error;
    exit();
}

// Perform a query on the database
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);

// Output the results of the query
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "First Name: " . $row["firstname"] . " Last Name: " . $row["lastname"] . "<br>";
    }
}
else {
    echo "No results found";
}

// Close the database connection
$conn->close();
?>
```

The **$result object,** which is an **instance** of **mysqli_result class,** provides a **number** of **properties** and **methods** for **accessing** and **manipulating** the **data returned** by the query.

In this code, we use the **num_rows property** to **check if the query returned any rows**, and the **fetch_assoc() method** to **retrieve the data** for **each row** as an **associative array and store it in variable $row**.

Finally, we close the database connection using the **close()** method.

Example of PHP code connecting to a MySQL Database

- To **prevent unauthorized access** to **sensitive information**, as well as to **make it easier to change** the **username** and **password**, if necessary, without modifying the code in all php scripts, the username and password for the database connection will typically be **stored in a separate configuration file** **outside** of the **web root directory**.

- For example, you can create a file called ***config.php*** in a directory **outside of the web root directory***, and define there variables for the database username and password the server and the database name.

* files stored outside of the web root directory **cannot be accessed directly** via a URL. By keeping sensitive files outside of the web root directory, you can help to ensure that they are only accessible through the appropriate channels (such as via a PHP script) and not directly by a user through a web browser. This can help to improve the security of your application.

Example of PHP code connecting to a MySQL Database

- Then, in your PHP code, you can include this configuration file and use the variables to establish a connection to the database (using **require_once('path/to/config.php’);** function)
- The **require_once()** function is **similar to** the **require** function, but it checks to see **if the file has already been included** **before attempting to include it again**. This can help prevent errors caused by attempting to include the same file multiple times.

```php
<?php
// Include the configuration file
require_once('path/to/config.php');

// Create a connection to the MySQL server
$conn = new mysqli($servername, $username, $password, $dbname);

// Check for errors when connecting
if ($conn->connect_errno) {
    echo "Failed to connect to MySQL: " . $conn->connect_error;
    exit();
}

// Perform a query on the database
$sql = "SELECT * FROM mytable";
$result = $conn->query($sql);

// Output the results of the query
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "First Name: " . $row["firstname"] . " Last Name: " . $row["lastname"] . "<br>";
    }
}
else {
    echo "No results found";
}

// Close the database connection
$conn->close();
?>
```

The path/to/config.php should be outside of the web root directory

```php
<?php
// MySQL server configuration
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mydb";
?>
```

config.php

Example of PHP code connecting to a MySQL Database

Installing MySQL on your Local Machine

To **use the functions** for **database access** in PHP and MySQL, it is important to **have a MySQL database installed** on your local machine or web server.

Installing MySQL on your computer or server **can be done by** downloading the MySQL installer from the official website or by **using a package management system** like **XAMPP**, which **bundles MySQL** with **other web development tools**.

Installing MySQL → XAMPP

- XAMPP is a free and open-source software package that **provides an easy way** to **install** and **run** a **complete web development environment** on your local machine. It **includes** several components that are commonly used in web development, including:
  - Apache web server
  - MySQL database server
  - PHP programming language
  - phpMyAdmin database management tool
  - Perl programming language
  - FileZilla FTP server
  - Mercury Mail Transport System
  - Tomcat web application server

X ("cross-platform"), **A**pache, **M**ariaDB, **P**HP, **P**erl

MySQL replaced MariaDB in 2015

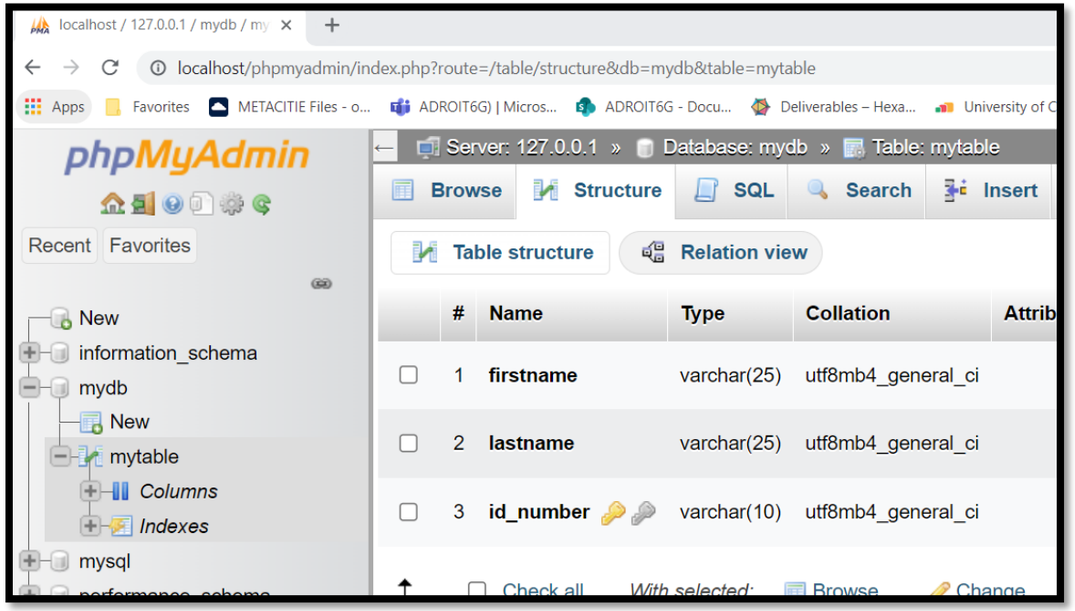
Installing MySQL → XAMPP

- **Installing XAMPP** is a **simple** and **easy way** to get started with MySQL and PHP development, which is actually our goal for this lecture:
- Here are the steps to install and use MySQL with XAMPP:
  - **Download** and install XAMPP from the official Apache Friends website: **https://www.apachefriends.org/index.html**
  - During the installation process, select the components you want to install, including MySQL and PHP.
  - Once XAMPP is installed, start the Apache and MySQL services from the XAMPP Control Panel.

Creating a MySQL Database → XAMPP

- Open a web browser and go to **http://localhost/phpmyadmin/**. This will open the phpMyAdmin interface, which you can use to manage your MySQL databases.

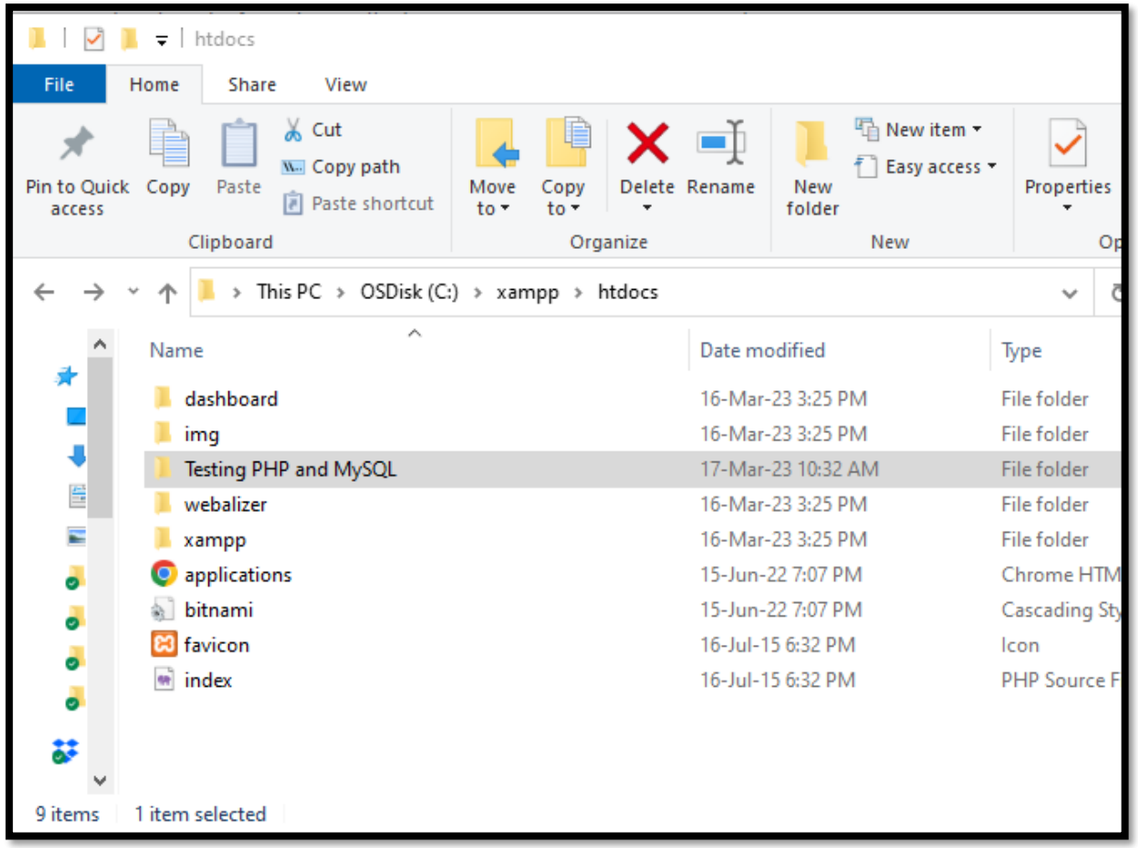
- To **create a new database**, click on the "Databases" tab, enter a name for your database (i.e., "**mydb**"), and click "Create".



- You can then **create tables** (i.e., "**mytable**") and add data to your database using the phpMyAdmin interface, or write PHP code to interact with the database programmatically.

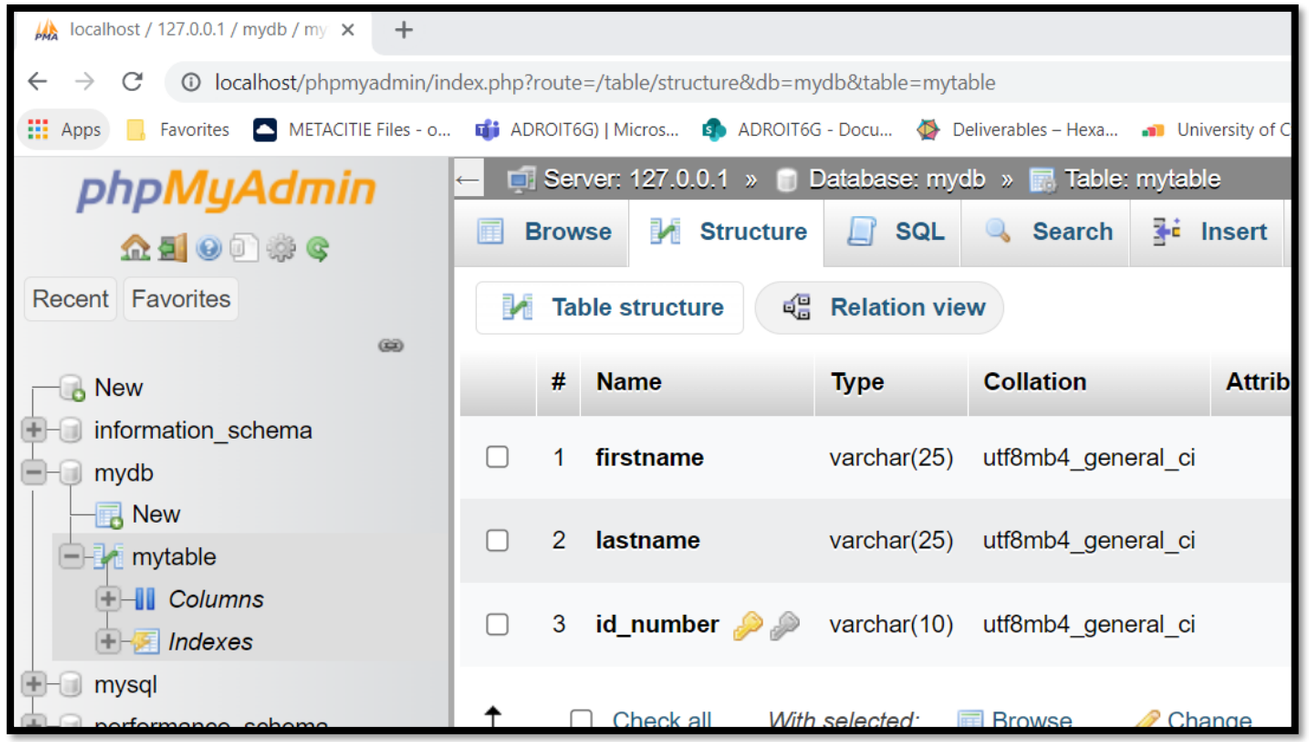
Where to put your website files ? → XAMPP

- In XAMPP, the **default root directory** for your website files is **htdocs**. You can access this folder by navigating to the XAMPP installation directory in your PC and locating the **htdocs** folder.
- For example, **on Windows**, the **default installation path** for XAMPP is **C:\xampp**.
- So, you should place your website files in the **C:\xampp\htdocs folder or a subfolder within it**.



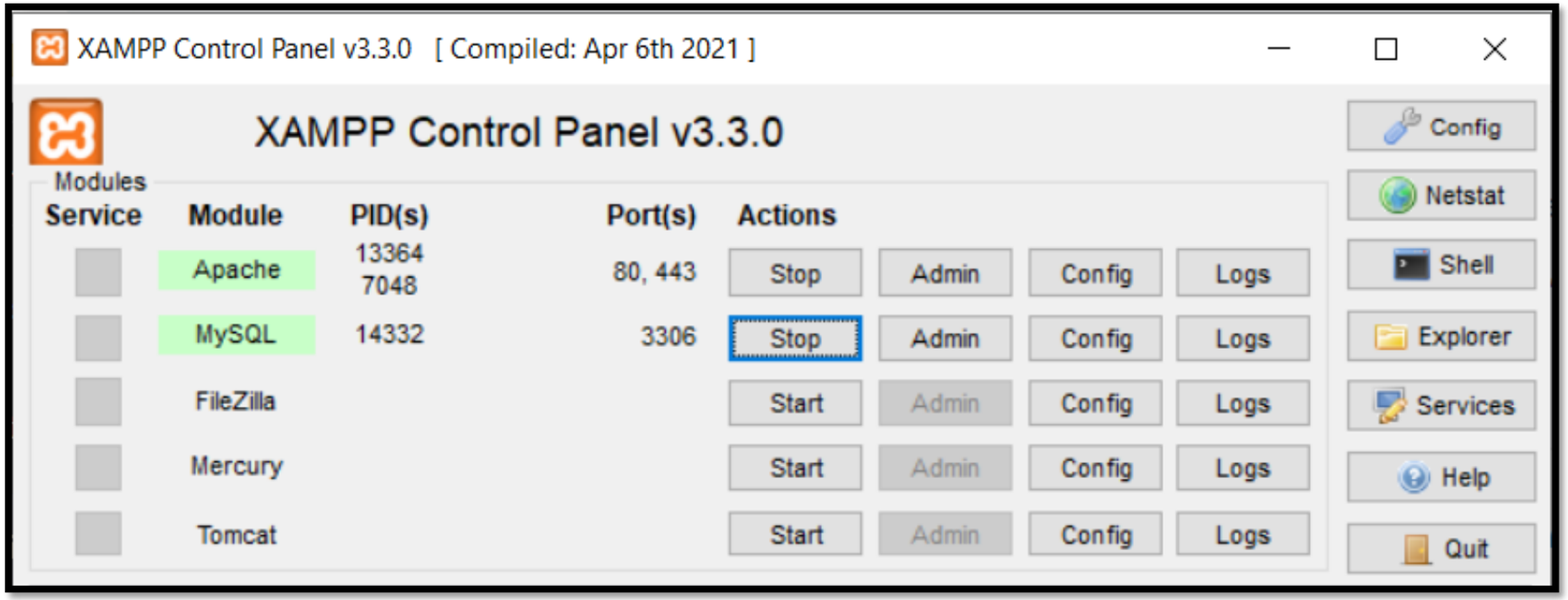
A Complete Example (HTML, JavaScript, PHP, MySQL)

- In this example we will create **two forms**:
  - The **first form** will allow **the user** to **create an account** by including its **First Name**, **Last Name** and **ID Number** and store this info in the "**mydb**" Database in "**mytable**" Table that we created using MySQL in XAMPP.
  - The **second form** will allow **the administrator** to search in the "**mydb**" Database in "**mytable**" Table, for the **First Name** and **Last Name** of **a user** by using its **ID Number**.



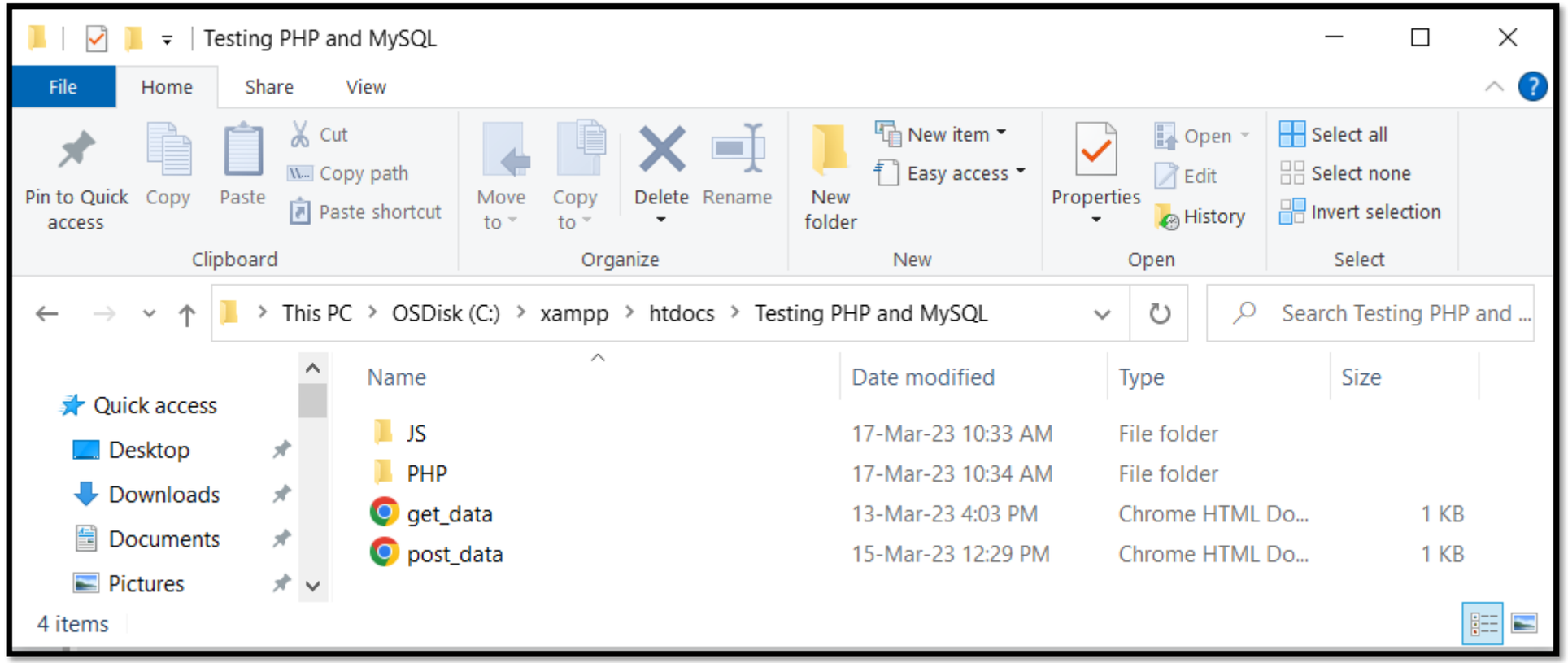
A Complete Example (HTML, JavaScript, PHP, MySQL)

- Before continuing, make sure that your **XAMPP Apache** and **MySQL services are running**. You can check this by opening the XAMPP Control Panel and looking at the "Status" column for Apache and MySQL. If they are running, they will have a green background color.



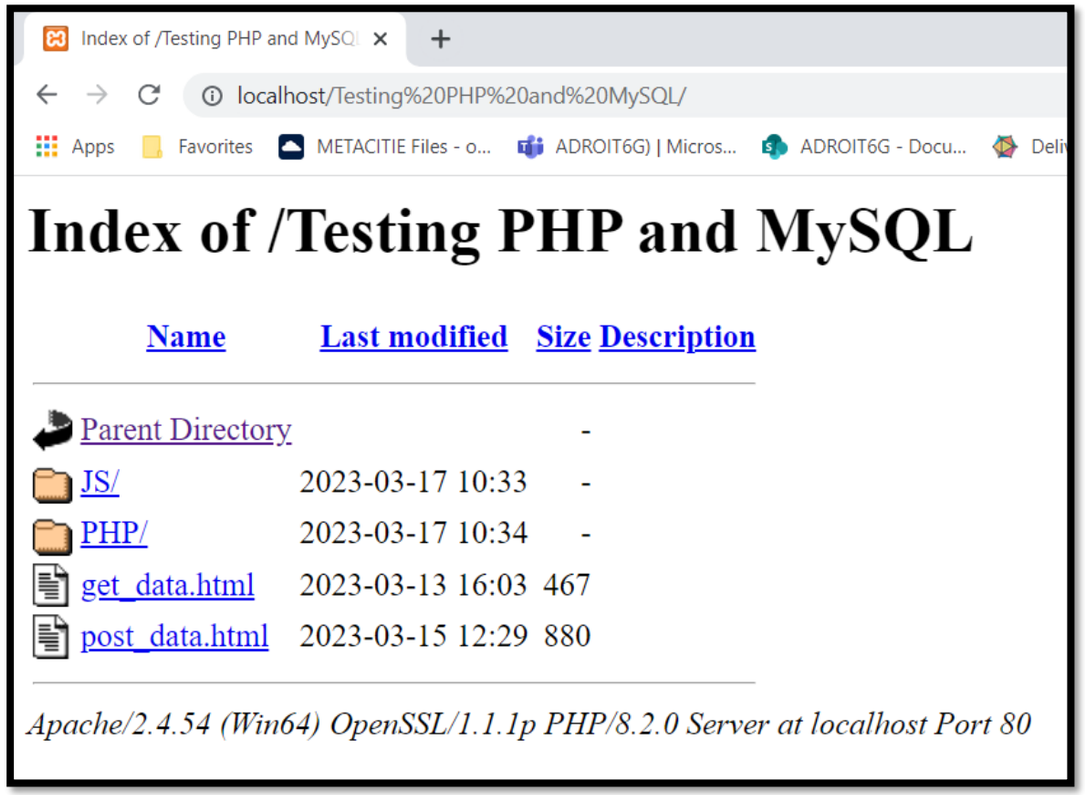
A Complete Example (HTML, JavaScript, PHP, MySQL)

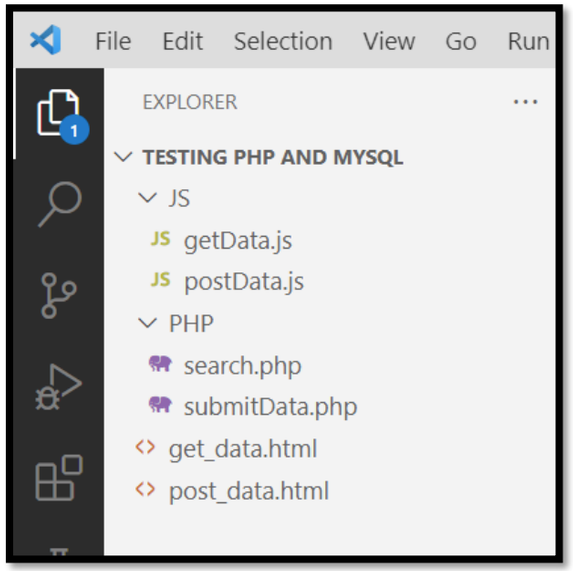
- The website files are placed in a subfolder with title "**Testing PHP and MySQL**" in the **C:\xampp\htdocs** folder.



A Complete Example (HTML, JavaScript, PHP, MySQL)

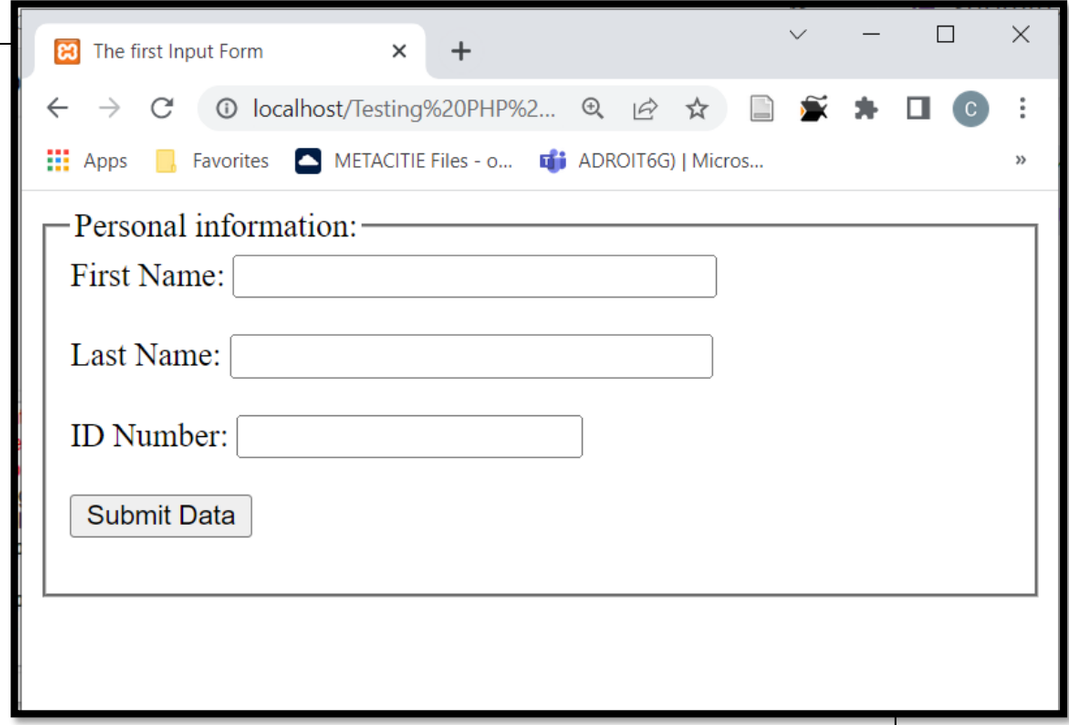
- To start using the website files open your web browser and navigate to **http://localhost/Testing PHP and MySQL** folder path.
- We will start by creating the HTML, JavaScript, and PHP files connecting to MySQL Database for **posting the user's data** and **storing** these in the Database.





post_data.html

```html
<!DOCTYPE html>
<html>

<head>
    <title>The first Input Form</title>
    <script src="JS/postData.js" defer></script>
</head>

<body>
    <form id="form1">
        <fieldset>
            <legend>Personal information:</legend>
            <label for="firstname">First Name: </label>
            <input type="text" name="firstname" size=30 required><br><br>
            <label for="lastname">Last Name: </label>
            <input type="text" name="lastname" size=30 required><br><br>
            <label for="id_number">ID Number: </label>
            <input type="text" name="id_number" required><br><br>
            <button type="button" id="submitButton">Submit Data</button> <br><br>
        </fieldset>
    </form>
    <!-- We will use this p HTML Element to print the status to the user after submission -->
    <p id="status" style="color:blue; font-weight: bold;"></p>
</body>

</html>
```



JS/postData.js

```
// Here we add a 'click' event listener on the button. After the 'Submit Data' button
// is clicked, the postValues method is triggered.
var button = document.getElementById("submitButton");
button.addEventListener('click', postValues);

// We declare the httpPostRequest here so as to be visible in all our code
var httpPostRequest;

// In this example we will use AJAX to "POST" the values of the form to the server.
// The new FormData(form) creates a new FormData object that represents the data
// submitted in the HTML form. It provides a way to construct a set of key=value pairs
// that represent form data, that can be sent using an XMLHttpRequest (XHR).
function postValues() {
    var form = document.getElementById("form1");
    var formData = new FormData(form);

    httpPostRequest = new XMLHttpRequest();
    httpPostRequest.open('POST', 'PHP/submitData.php', true);
    httpPostRequest.onreadystatechange = handleResponse;
    httpPostRequest.send(formData);
}
```

Example 1

```
// Here we define what will happen when the response from the Server is received
function handleResponse() {
    // Process the server response here.
    if (httpPostRequest.readyState === XMLHttpRequest.DONE && httpPostRequest.status === 200) {
        // Perfect!! Everything is good, the response was received. Request Successfully Received
        var responseMessage = httpPostRequest.responseText;
        document.getElementById("status").innerHTML = responseMessage;
    }
    else {
        // There was a problem with the request.
        // For example, the response may have a 404 (Not Found)
        // or 500 (Internal Server Error) response code.
    }
}
```

JS/postData.js

PHP/submitData.php

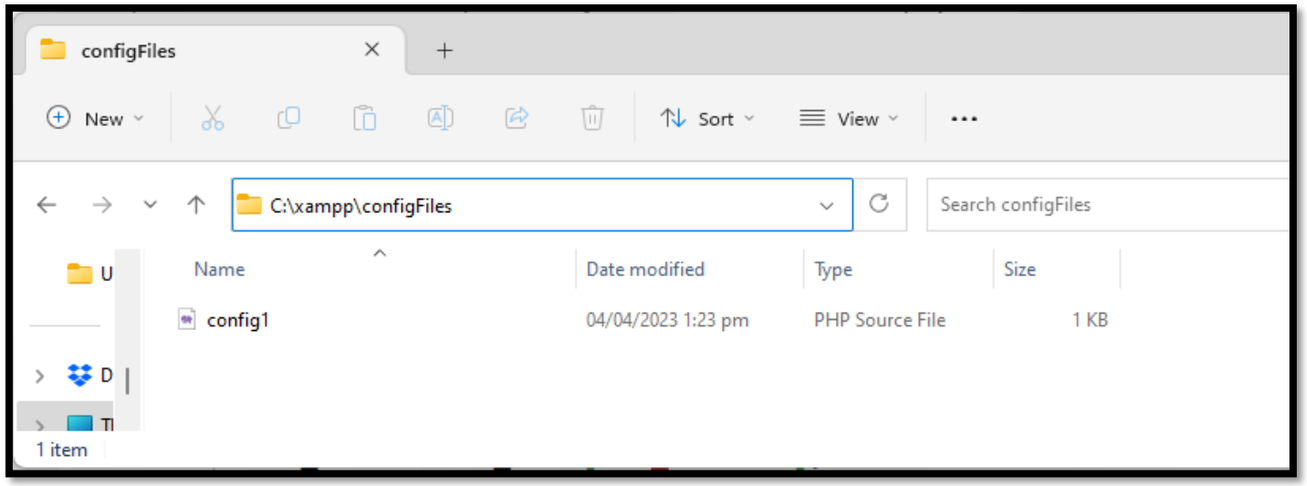
```php
<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Get form data
    $firstname = $_POST['firstname'];
    $lastname = $_POST['lastname'];
    $id_number = $_POST['id_number'];

    // Database connection parameters
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "mydb";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
```

Example 1

```
    // Prepare query
    $sql = "INSERT INTO mytable (firstname, lastname, id_number)
            VALUES ('$firstname', '$lastname', '$id_number')";

    // Execute query and check for errors
    if ($conn->query($sql) === TRUE) {
        echo "Form data for user with ID $id_number submitted successfully!";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    // Close connection
    $conn->close();
}
?>
```

PHP/submitData.php

Example 1 - Alternative



We created a file called ***config1.php*** in **C:\xampp\configFiles** directory **outside of the web root directory**, and define there the variables for the database username and password the server and the database name.

```php
<?php
  // Database connection parameters
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "mydb";
?>
```

C:\xampp\configFiles\config1.php

Example 1 - Alternative

PHP/submitData.php

```
<?php

require_once('C:\xampp\configFiles\config1.php');

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Get form data
    $firstname = $_POST['firstname'];
    $lastname = $_POST['lastname'];
    $id_number = $_POST['id_number’];


    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);
```

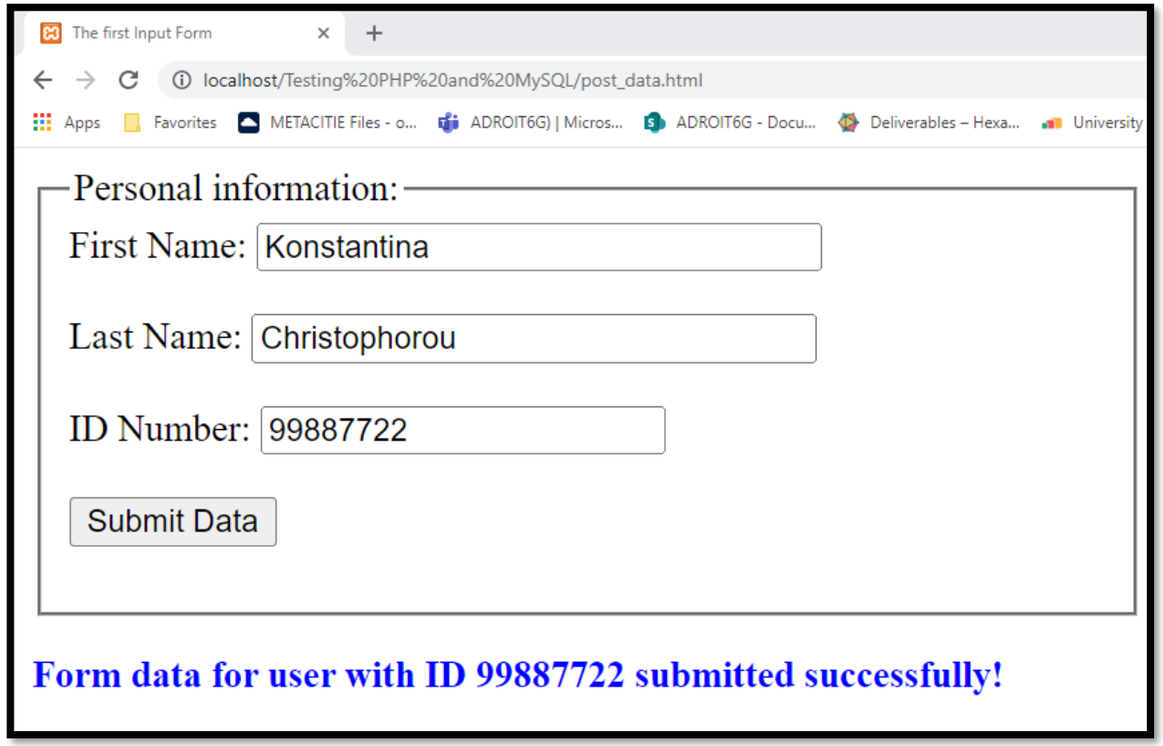
Here using **require_once(…)** function we include the configuration file **config1.php** and use the connection variables to establish a connection to the database.

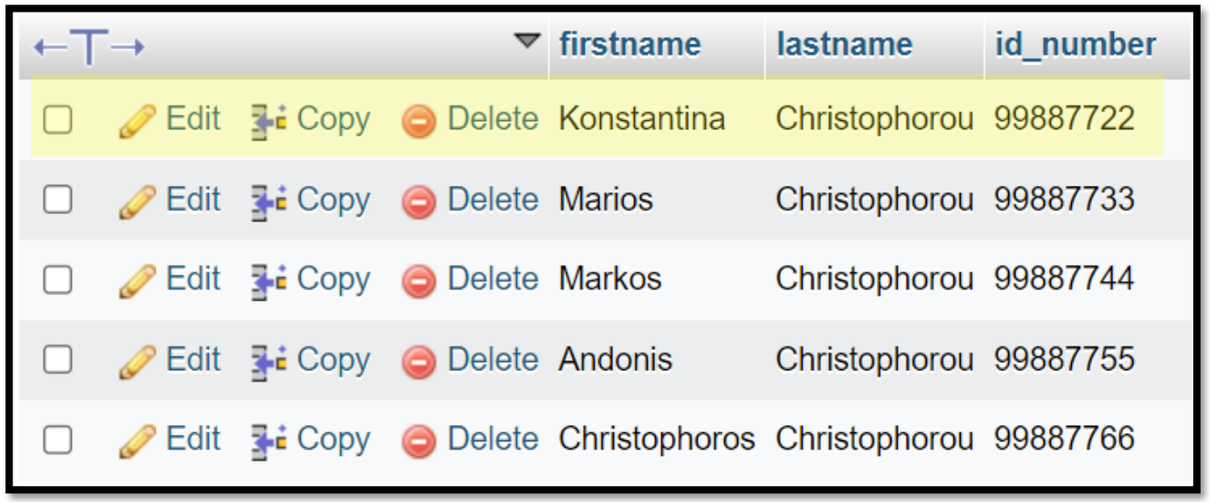
Example 1

```
    // Prepare query
    $sql = "INSERT INTO mytable (firstname, lastname, id_number)
            VALUES ('$firstname', '$lastname', '$id_number')";

    // Execute query and check for errors
    if ($conn->query($sql) === TRUE) {
        echo "Form data for user with ID $id_number submitted successfully!";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    // Close connection
    $conn->close();
}
?>
```

PHP/submitData.php

A Complete Example – Creating a User Account

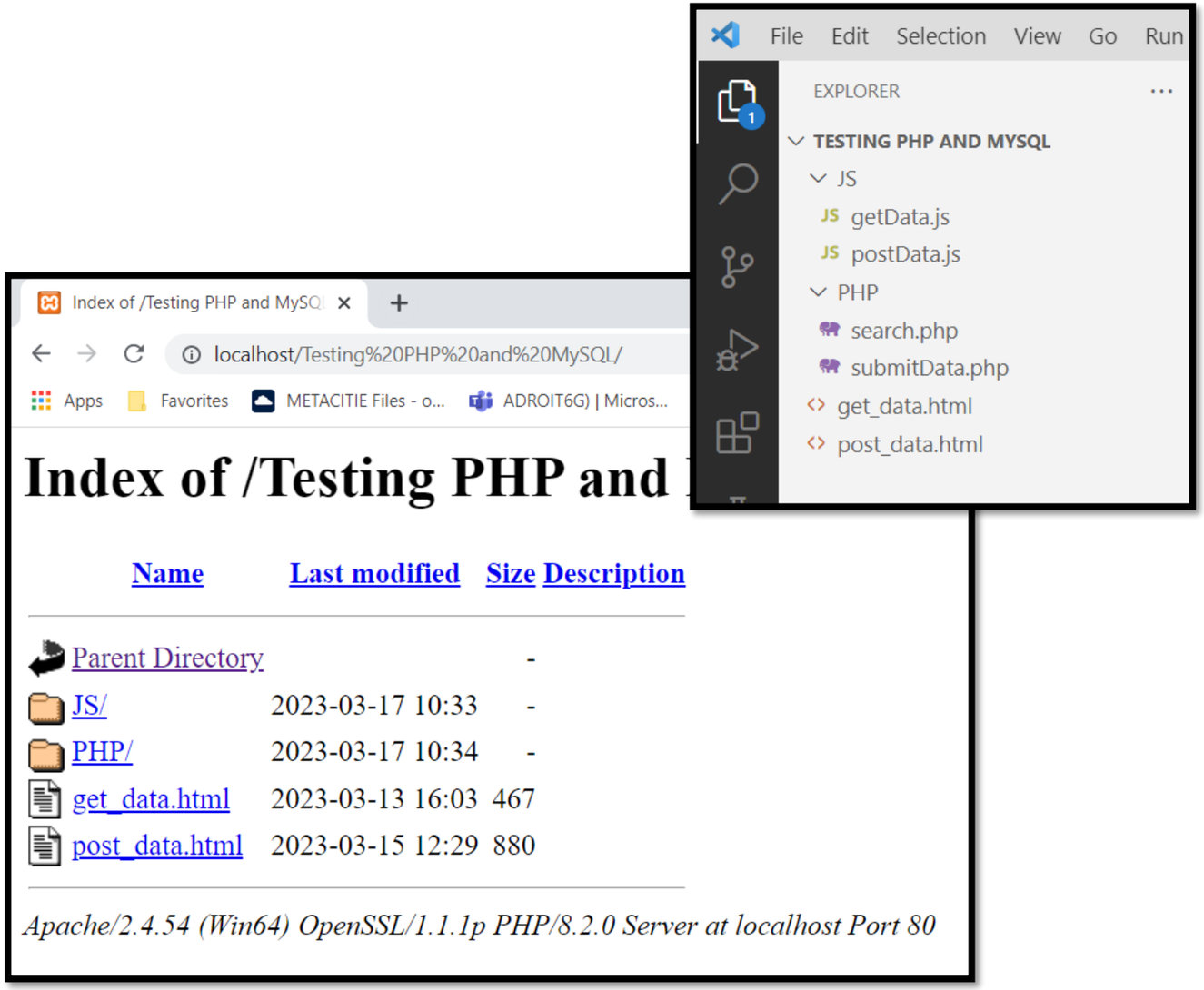




				firstname	lastname	id_number
Edit	Copy	Delete		Konstantina	Christophorou	99887722
Edit	Copy	Delete		Marios	Christophorou	99887733
Edit	Copy	Delete		Markos	Christophorou	99887744
Edit	Copy	Delete		Andonis	Christophorou	99887755
Edit	Copy	Delete		Christophoros	Christophorou	99887766

A Complete Example (HTML, JavaScript, PHP, MySQL)

- Now we will create the HTML, JavaScript, and PHP files connecting to MySQL Database for **getting** the user's data from the Database based on his/hers ID Number

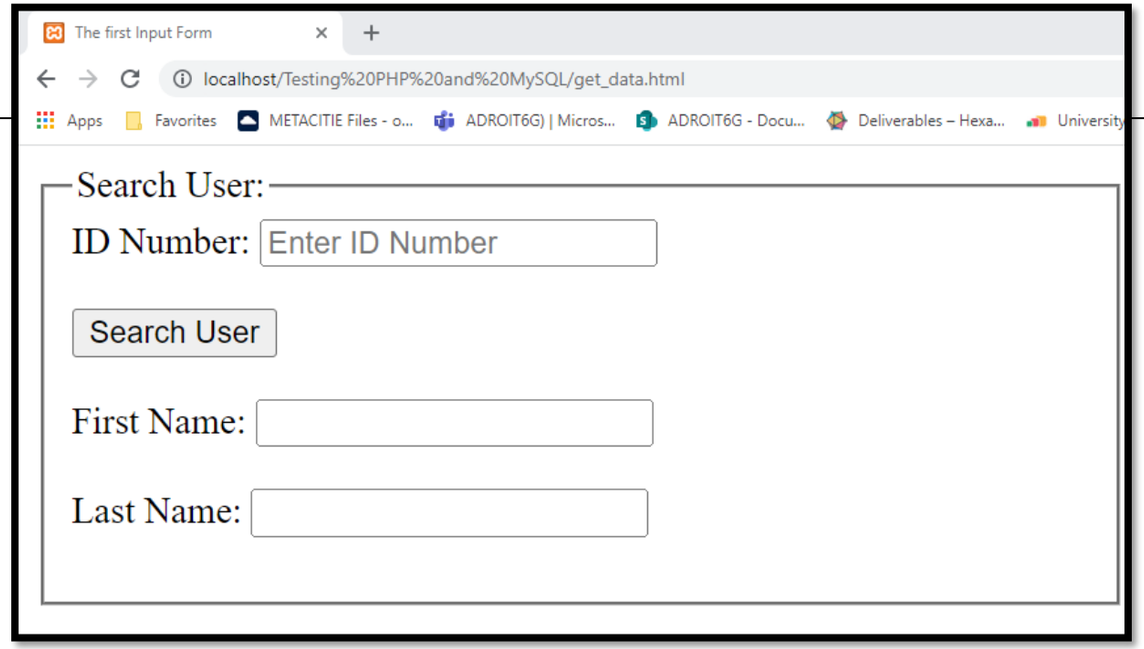


A Complete Example – Searching a User Account

←T→			firstname	lastname	id_number
Edit	Copy	Delete	Konstantina	Christophorou	99887722
Edit	Copy	Delete	Marios	Christophorou	99887733
Edit	Copy	Delete	Markos	Christophorou	99887744
Edit	Copy	Delete	Andonis	Christophorou	99887755
Edit	Copy	Delete	Christophoros	Christophorou	99887766

mydb with data included in mytable

get_data.html




```html
<!DOCTYPE html>
<html>

<head>
    <title>The first Input Form</title>
    <script src="JS/getData.js" defer></script>
</head>

<body>
    <form id="searchForm">
        <fieldset>
            <legend>Search User:</legend>
            <label for="id_number">ID Number: </label>
            <input type="text" id="id_number" name="id_number" placeholder="Enter ID Number"><br><br>
            <button type="button" id="searchButton">Search User</button> <br><br>

            <!-- We will use the following input form elements to store the person details received -->
            <label>First Name: </label>
            <input type="text" name="firstname" id="firstname" readonly /> <br><br>
            <label>Last Name: </label>
            <input type="text" name="lastname" id="lastname" readonly /> <br><br>

        </fieldset>
    </form>
</body>

</html>
```

```
// Here we add a 'click' event listener on the button. After the 'Search User' button
// is clicked, the searchUser method is triggered.
var button = document.getElementById("searchButton");
button.addEventListener('click', searchUser);

// We declare the httpRequest here so as to be visible in all our code
var httpRequest;

// In this example we will use AJAX to "POST" the values of the form to the server.
// The new FormData(form) creates a new FormData object that represents the data
// submitted in the HTML form. It provides a way to construct a set of key=value pairs
// that represent form data, that can be sent using an XMLHttpRequest (XHR).

function searchUser() {
    var form = document.getElementById("searchForm");
    var formData = new FormData(form);

    httpRequest = new XMLHttpRequest();
    httpRequest.open('POST', 'PHP/search.php', true);
    httpRequest.onreadystatechange = handleResponse;
    httpRequest.send(formData);
}
```

JS/getData.js

Example 2

```
// Here we define what will happen when the response from the Server is received
function handleResponse() {
    // Process the server response here.
    if (httpRequest.readyState === XMLHttpRequest.DONE && httpRequest.status === 200) {
        // Perfect! Everything is good, the response was received.
        // Here we expect a JSON Associative Array string. Thus we will use JSON.parse()
        // to conver JSON string into Associative Array.
        var responseArray = JSON.parse(httpRequest.responseText);
        document.getElementById("firstname").value = responseArray["firstname"];
        document.getElementById("lastname").value = responseArray["lastname"];
    }
    else {
        // There was a problem with the request.
        // For example, the response may have a 404 (Not Found)
        // or 500 (Internal Server Error) response code.
    }
}
```

JS/getData.js

Example 2

PHP/search.php

```php
<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  // get the id_number value from the form data
  $id_number = $_POST['id_number'];

  // create a database connection
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "mydb";

  $conn = mysqli_connect($servername, $username, $password, $dbname);
```

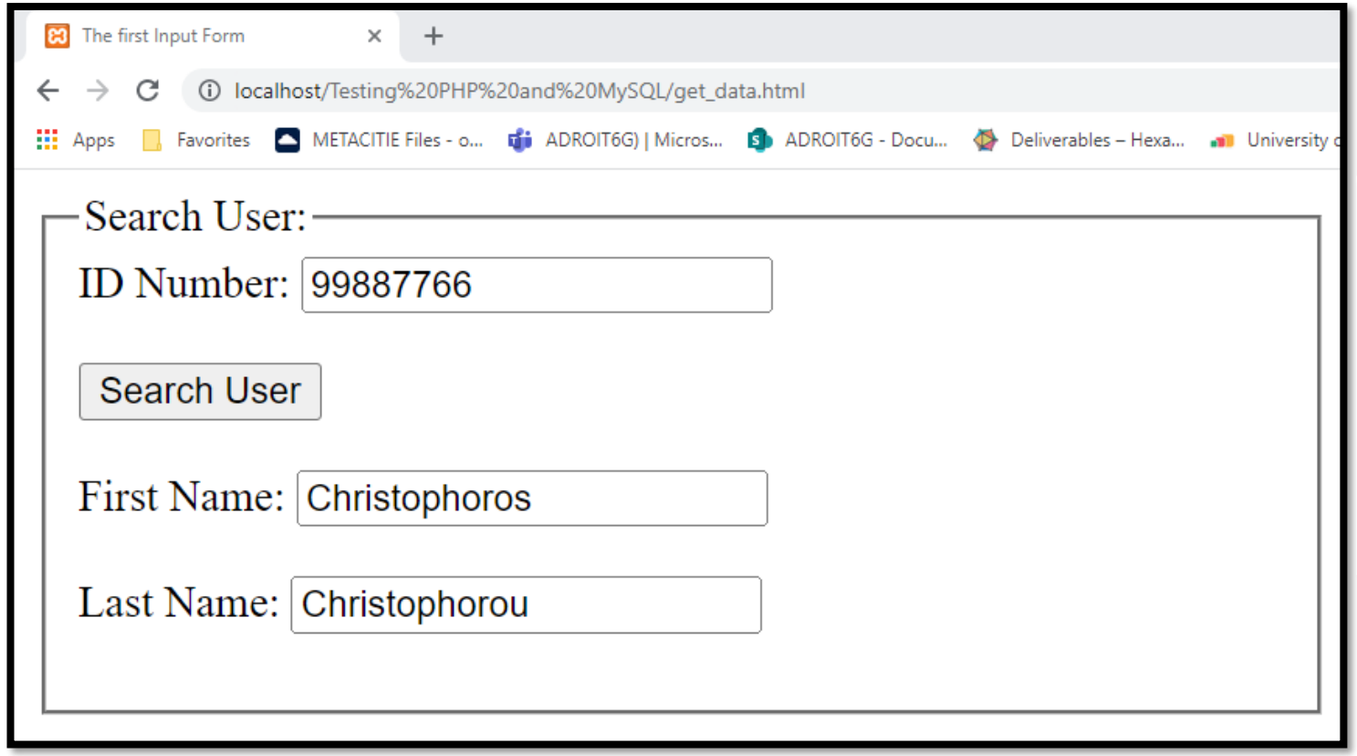
Example 2

```
// search for the user based on the id_number. The id_number is
// unique for each user, thus only one row is expected.
 $sql = "SELECT * FROM mytable WHERE id_number = '$id_number'";
 $result = $conn->query($sql);

 if ($result->num_rows !== 0) {
    // output the user details. With fetch_assoc() method the firstname and lastname will
    // be included in an associative array. Then convert this array to a JSON string and echo it
    // to the Client.
    $row = $result->fetch_assoc();
    $fullname = json_encode($row);
    echo $fullname;
 }

// close the database connection
mysqli_close($conn);
}
?>
```

Alternatively, you can use **$result->fetch_object()** that fetches a result row as an object.

PHP/search.php

A Complete Example – Searching a User Account



				firstname	lastname	id_number
Edit	Copy	Delete		Marios	Christophorou	99887733
Edit	Copy	Delete		Markos	Christophorou	99887744
Edit	Copy	Delete		Andonis	Christophorou	99887755
Edit	Copy	Delete		Christophoros	Christophorou	99887766

Ερωτήσεις?