Web Technologies Laboratory 05

Laboratory Continuous Assessment (LCA) [As per Rubrics]

Understanding of the Objective (5)	Performance (5)	Journal Submission and Ethics (Neatness, Handwriting, Timely submission) (5)	Orals (5)	Total (20)	Remarks	Instructor's Sign

Aim: Write server side script in PHP to perform form validation and create database application using PHP and MySQL to perform insert, update, delete and search operations.

Objectives:

1. To understand Server Side Script
2. To learn database connectivity in PHP
3. To perform insert, update, delete and search operations on database

Theory:

1. PHP Architecture
2. Steps for Database connectivity in PHP

FAQ:

1. What are Client Side Scripts and Server Side Scripts?
2. What are the advantages of Server Side Scripting?
3. Write some differences between Client Side Scripting and Server Side Scripting?
4. List some Server Side Scripting languages.
5. What is XAMPP and phpMyAdmin?
6. What are the two ways to connect to database in PHP?

Output: Screenshots of the output to be attached.

PHP Introduction

The term PHP is an acronym for **PHP: Hypertext Preprocessor**. PHP is a server-side scripting language designed specifically for web development. Websites like www.facebook.com, www.yahoo.com are also built on PHP. One of the main reasons behind this is that PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file.

The thing that differentiates PHP with client-side language like HTML is, PHP codes are executed on server whereas HTML codes are directly rendered on the browser. PHP codes are first executed on the server and then the result is returned to the browser. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP files have extension ".php"

The only information that the client or browser knows is the result returned after executing the PHP script on the server and not the actual PHP codes present in the PHP file. Also, PHP files can support other client-side scripting languages like CSS and JavaScript.

Before learning PHP, one should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

What can PHP do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is easy to learn and runs efficiently on the server side

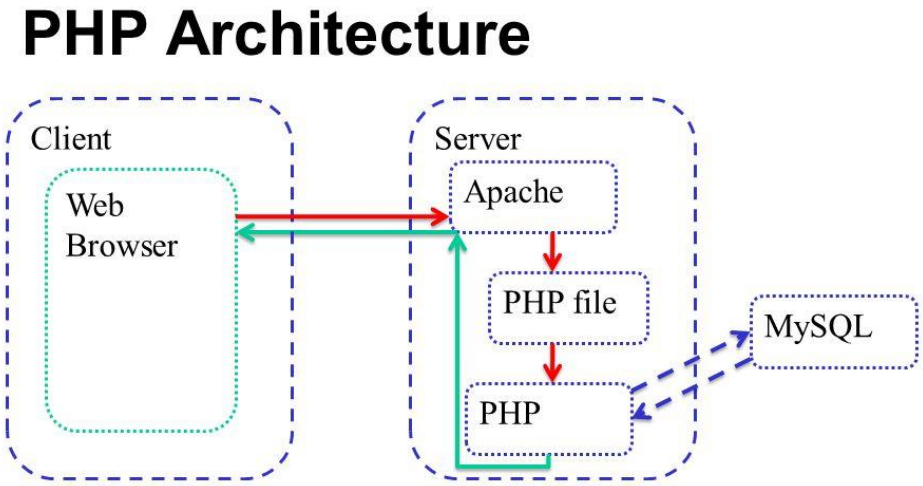
PHP Architecture: PHP is based on the Model-View-Controller is a concept involved in software development evolved in the late 1980s. It is a software architecture built on the idea that the logic of an application should be separated from its presentation. A system developed on the MVC architecture should allow a front-end developer and a back-end developer to work on the same system without interfering with each other.

- **Model**
  Model is the name given to the component that will communicate with the database to manipulate the data. It acts as a bridge between the View component and the Controller

component in the overall architecture. It doesn't matter to the Model component what happens to the data when it is passed to the View or Controller components.

- **View**
  The View requests for data from the Model component and then its final output is determined. View interacts with the user, and then transfers the user's reaction to the Controller component to respond accordingly. An example of this is a link generated by the View component, when a user clicks and an action gets triggered in the Controller.
- **Controller**
  The Controller's job is to handle data that the user inputs or submits through the forms, and then Model updates this accordingly in the database. The Controller is nothing without the user's interactions, which happens through the View component.

- A simple way to understand how MVC works is given below.
  1) A user interacts with View.
  2) The Controller handles the user input, and sends the information to the model.
  3) Then the Model receives the information and manipulates it (either saving it or updating it by communicating with the database).
  4) The View checks the state of the Model and responds accordingly (lists updated information).



PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.
A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Output:

My first PHP page
Hello World!

Note: PHP statements end with a semicolon (;)

Steps to Run a PHP program in XAMPP Server

PHP program can be run under various like WAMP, XAMPP etc.

- **WAMP Server**: This server is a web development platform which helps in creating dynamic web applications.
- **XAMPP Server**: It is a free open source cross-platform web server package.

Download XAMPP from the following link:
https://www.apachefriends.org/download.html
After downloading, just follow the following step to start XAMPP server:

Step 1

Install XAMPP

Step 2

Assume you installed XAMPP in C Drive.
Go to: **C:\xampp\htdocs**

Create your own folder; name it for example as **tutorialspoint**.

Step 3

Now create your first php program in XAMPP and name it as **"add.php"**:

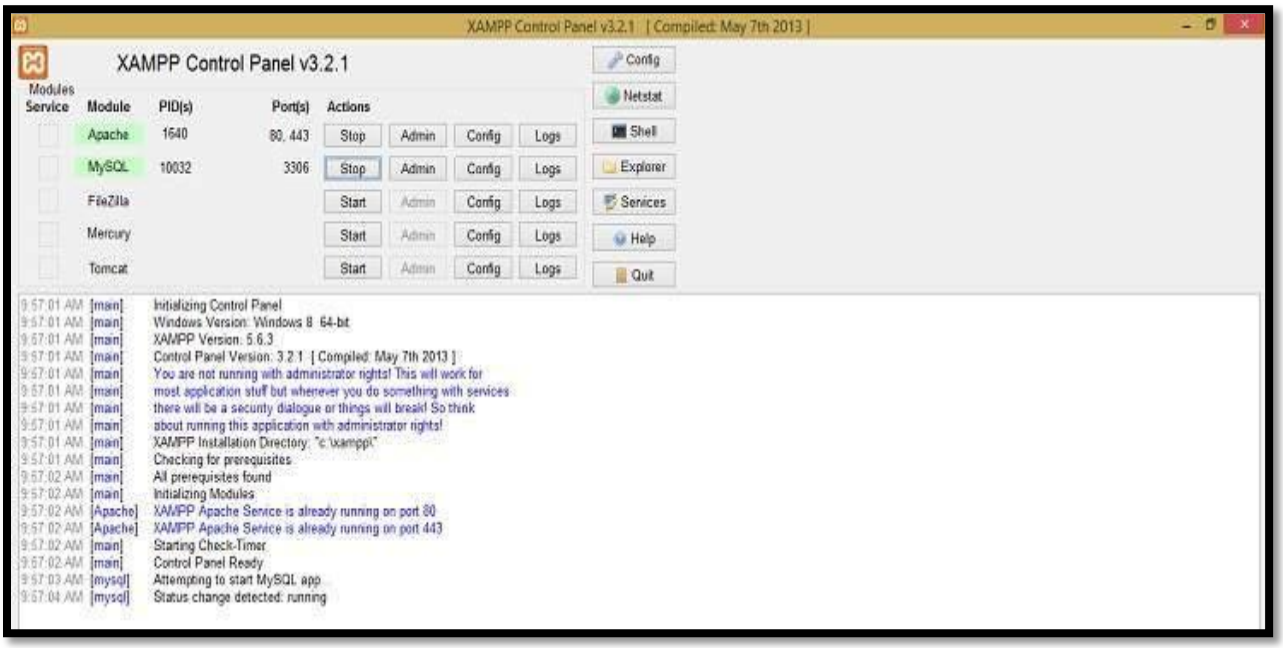
```
<html>
<head><title>Addition php</title></head>
<body>

<?php
        # operator
        print "<h2>php program to add two numbers...</h2><br />";
              $val1 = 20;
              $val2 = 20;
              $sum = $val2 + $val2;   /* Assignment operator */
              echo "Result(SUM): $sum";
?>

</body>
</html>
```

Step 4

Now double click on "**XAAMP CONTROL PANEL**" on desktop and START "**Apache**" (icon also appears on the bottom)



Step 5

Type **localhost** on your browser and press enter:

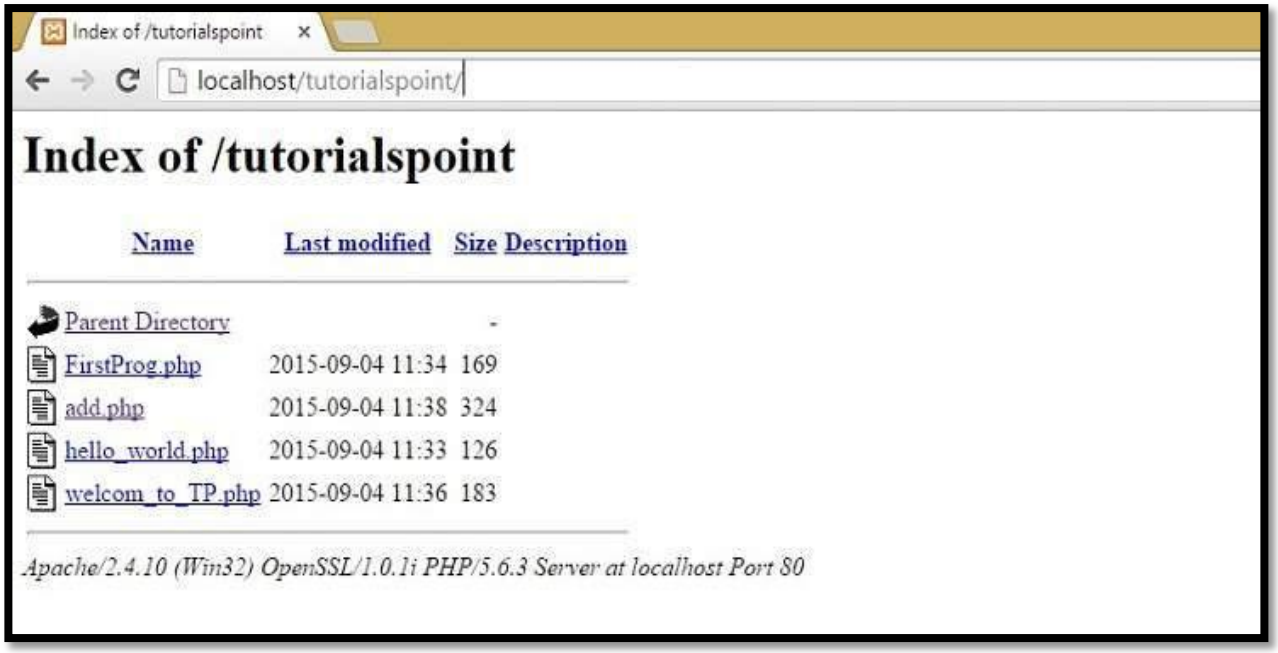
It will show the following:

Step 6

Now type the following in browser:

http://localhost/tutorialspoint/

Below screenshot shows php files created under folder "tutorialspoint"



Step 7

Click on "**add.php**" and it will show the following:

PHP Form Handling

The PHP superglobals $_GET and $_POST are used to collect form-data.
The example below displays a simple HTML form with two input fields and a submit button:
Example

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?> <br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

The output could be something like this:

Welcome John
Your email address is john.doe@example.com

The same result could also be achieved using the HTTP GET method:
Example

```
<html>
<body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?> <br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET vs. POST

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. GET may be used for sending non-sensitive data.

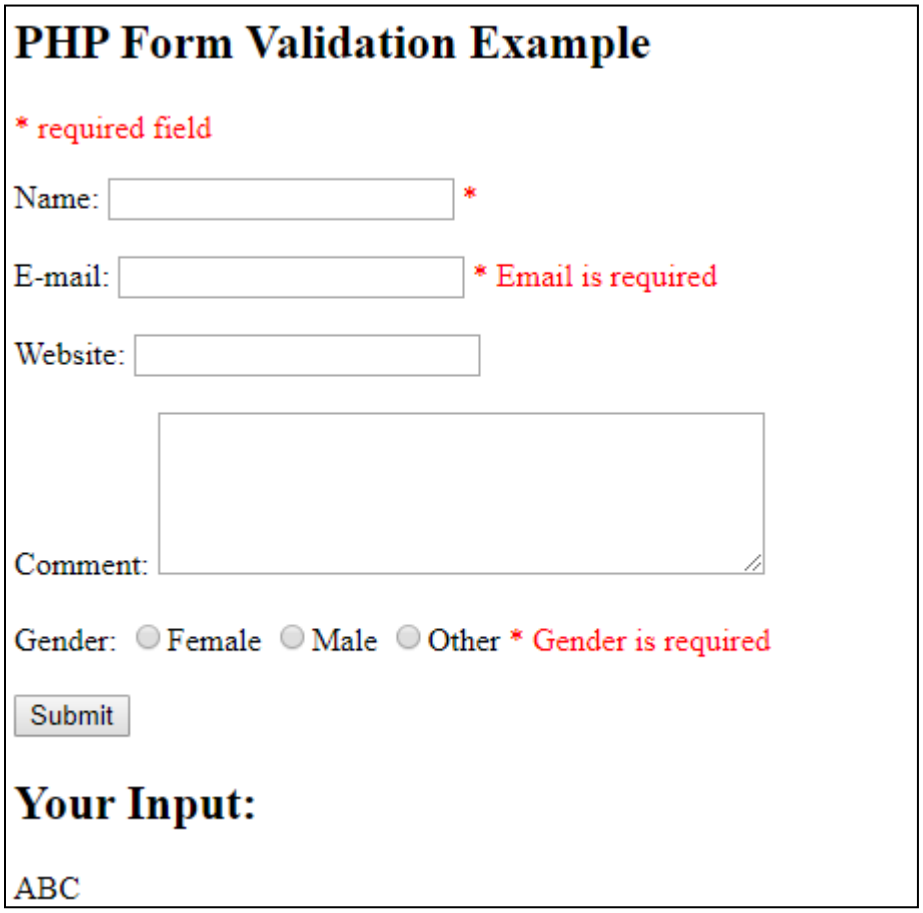
Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. However, because the variables are not displayed in the URL, it is not possible to bookmark the page. Developers prefer POST for sending form data.

PHP Form Validation

The HTML form which we will be using; contains various input fields: required and optional text fields, radio buttons, and a submit button:



The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element
The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The **$_SERVER["PHP_SELF"]** is a super global variable that returns the filename of the currently executing script. So, the **$_SERVER["PHP_SELF"]** sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The **htmlspecialchars()** function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

In the following code, some new variables are added: $nameErr, $emailErr, $genderErr, and $websiteErr. These error variables will hold error messages for the required fields. We have also added an if else statement for each $_POST variable. This checks if the $_POST variable is empty (with the PHP empty() function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the test_input() function:

PHP - Display the Error Messages

Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields).

PHP Forms - Validate Name, E-mail and URL

The next step is to validate the input data, that is "Does the Name field contain only letters and whitespace?", and "Does the E-mail field contain a valid e-mail address syntax?", and if filled out, "Does the Website field contain a valid URL?".

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters and whitespace. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name))
{
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's **filter_var()** function.

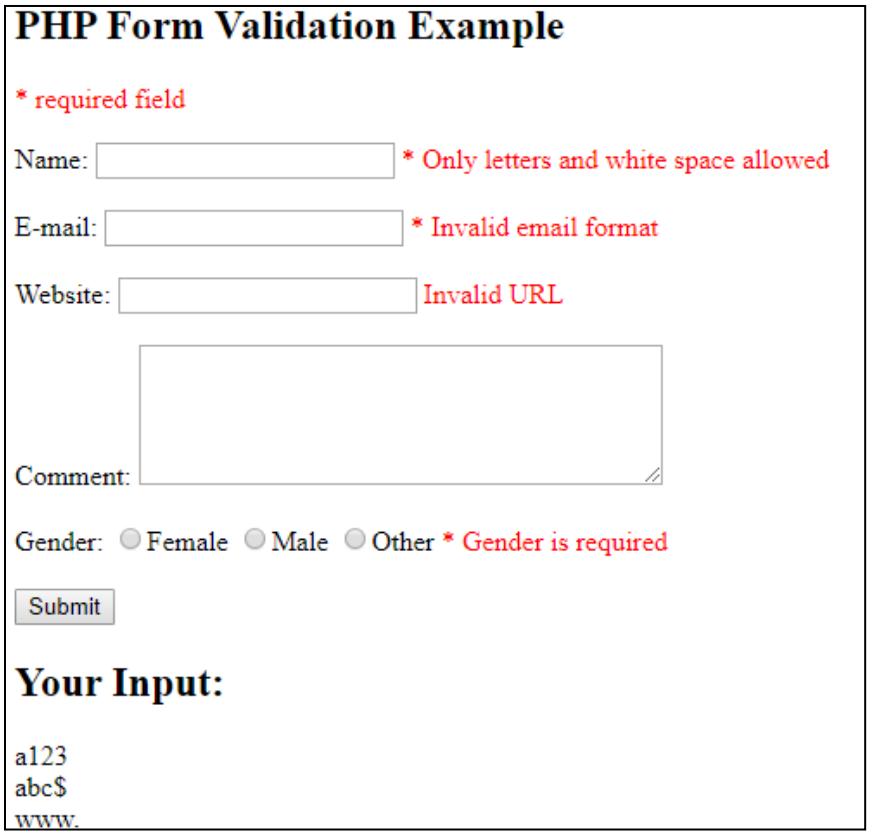
In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL))
{
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website))
{
  $websiteErr = "Invalid URL";
 }
```



The next step is to show how to prevent the form from emptying all the input fields when the user submits the form.

PHP - Keep the Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.

Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also allows dashes in
the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
```

```
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]
);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?
></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?> value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?> value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="other") echo "checked";?> value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
```

```
 <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

PHP Form Validation Example

* required field

Name: ABC12 * Only letters and white space allowed

E-mail: ABC123@ * Invalid email format

Website:

Comment:

Gender: Female Male Other * Gender is required

Submit

Your Input:

ABC12
ABC123@

PHP MySQL Database

With PHP, you can connect to and manipulate databases. MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web and runs on a server
- MySQL is very fast, reliable, and easy to use
- MySQL compiles on a number of platforms, is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

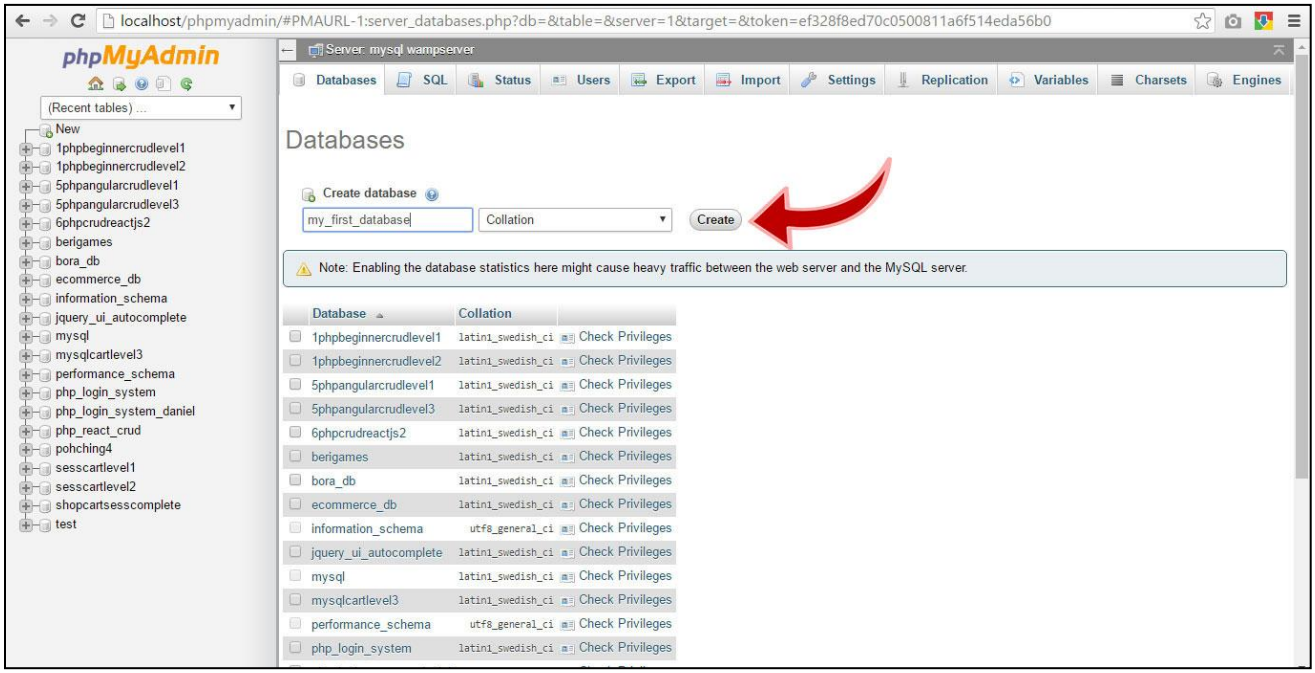
Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Steps to manage MYSQL with PHPMYADMIN

phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. In the following example, we will see how easy we can handle MySQL with phpMyAdmin.

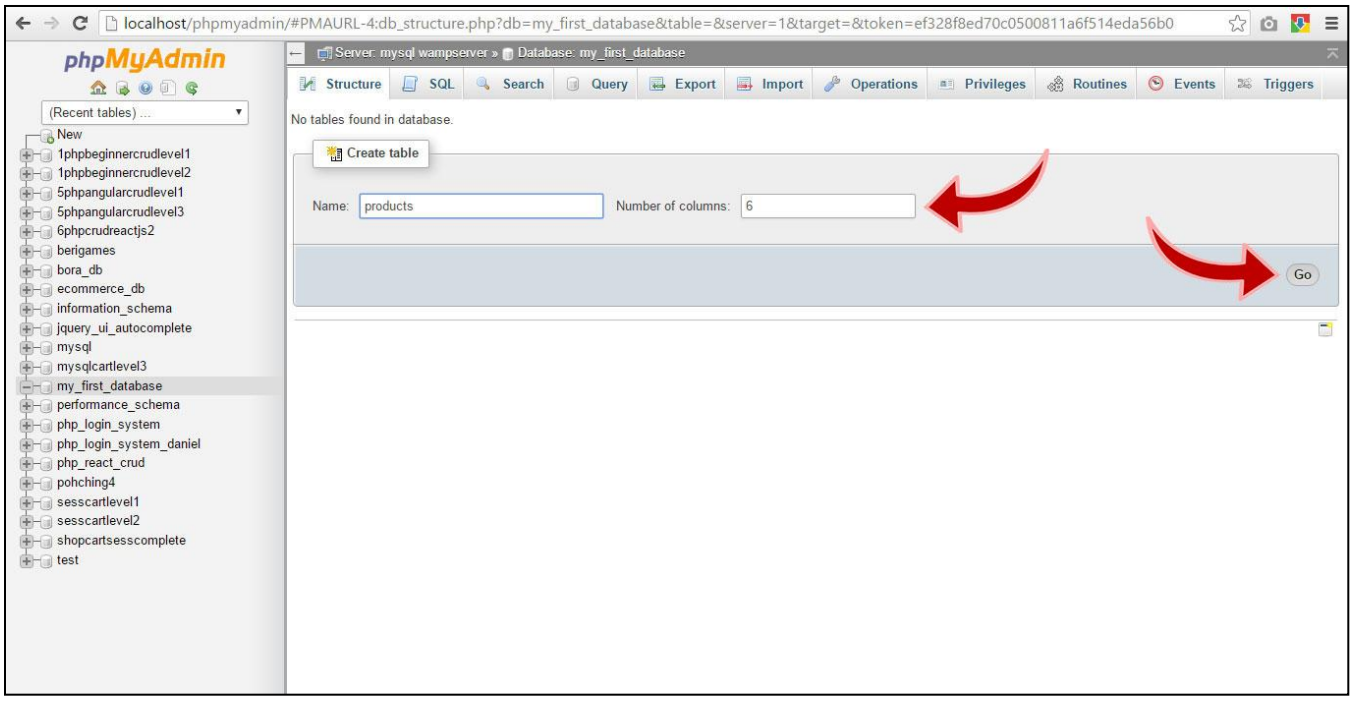
Create a Database

1. Go to **http://localhost/phpmyadmin/**
2. Click the "**New**" link on the upper left corner (under recent tables)
3. Fill out the "**Database Name**" field with "**my_first_database**".
4. Click the "**Create**" button

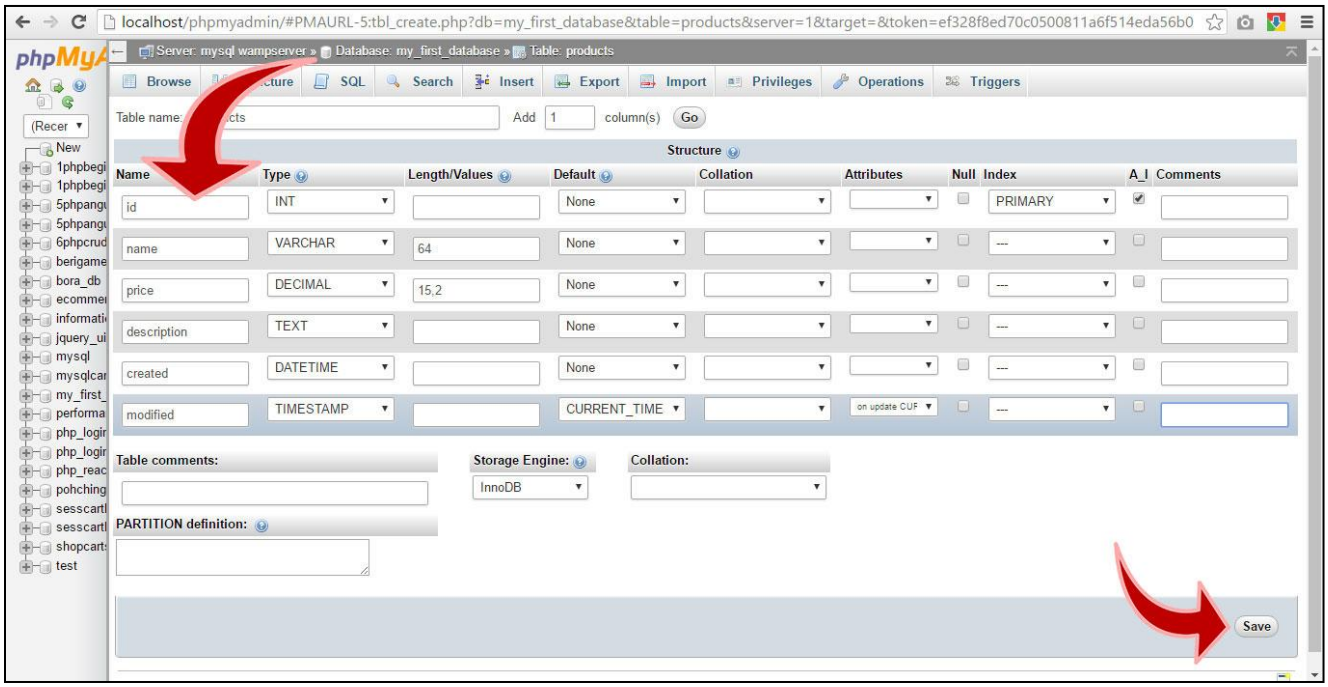


Create a Table

1. Click "**my_first_database**" on the left side of the screen
2. On the "Create Table" section, fill out the *Name* with "**products**" and *Number of Columns* with "**6**"
3. Click "**Go**" button

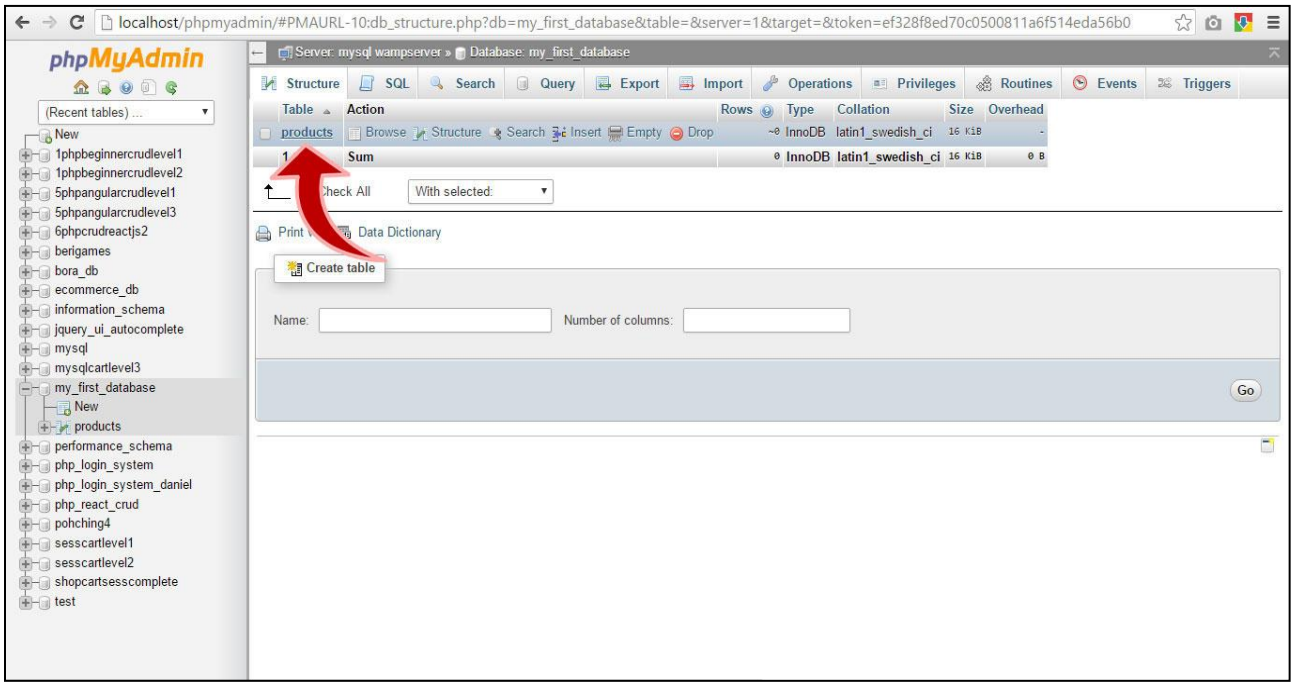


1. Fill out the fields with id, name, etc.
2. Mimic everything in the following image
3. Click the "**Save**" button

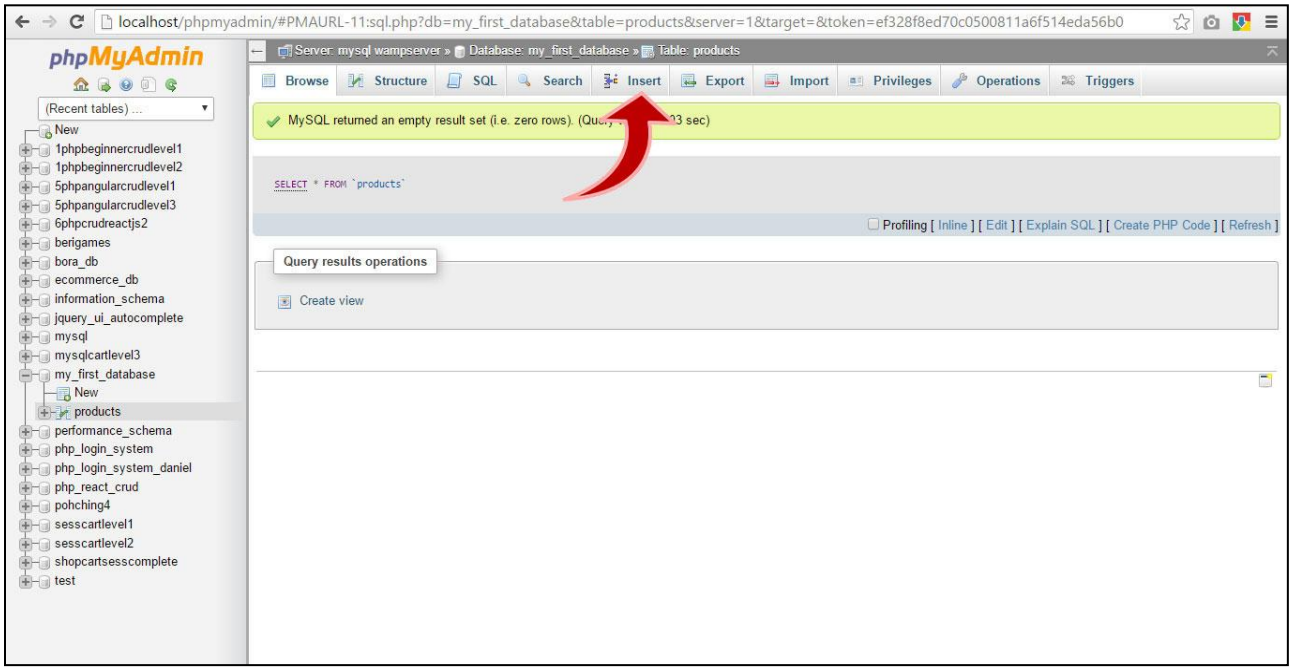


Insert Data

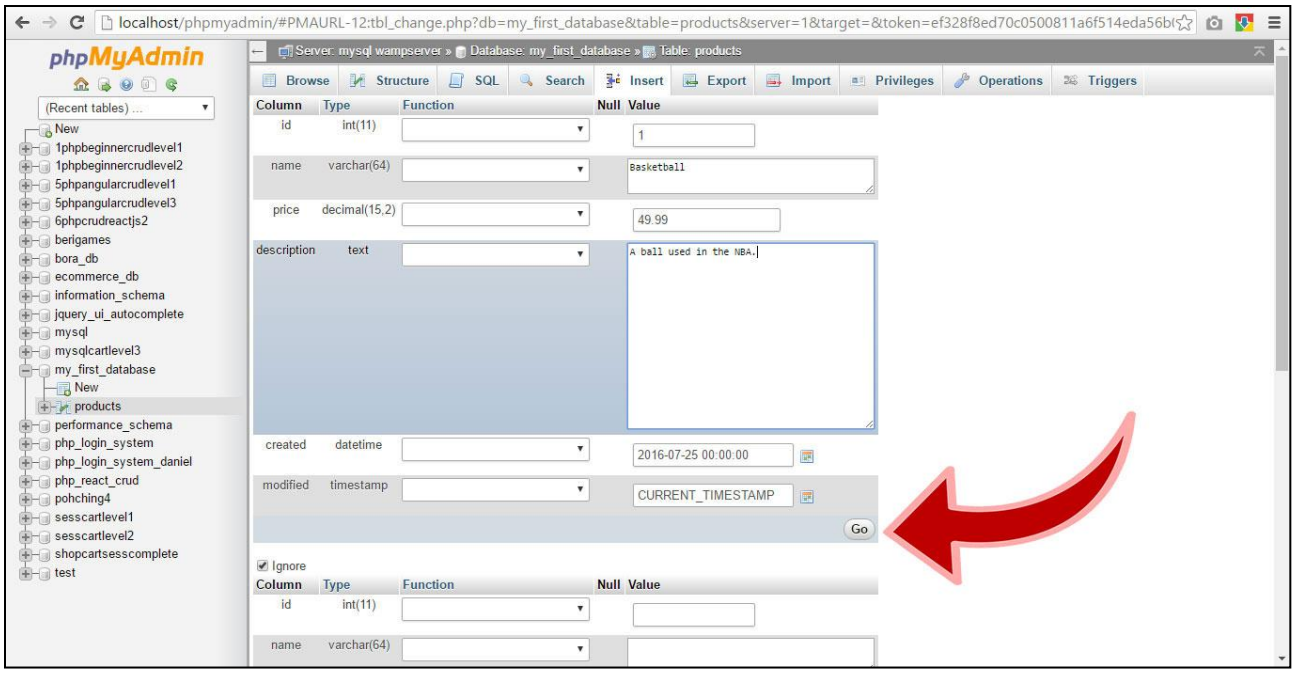
Click the "**products**" table.



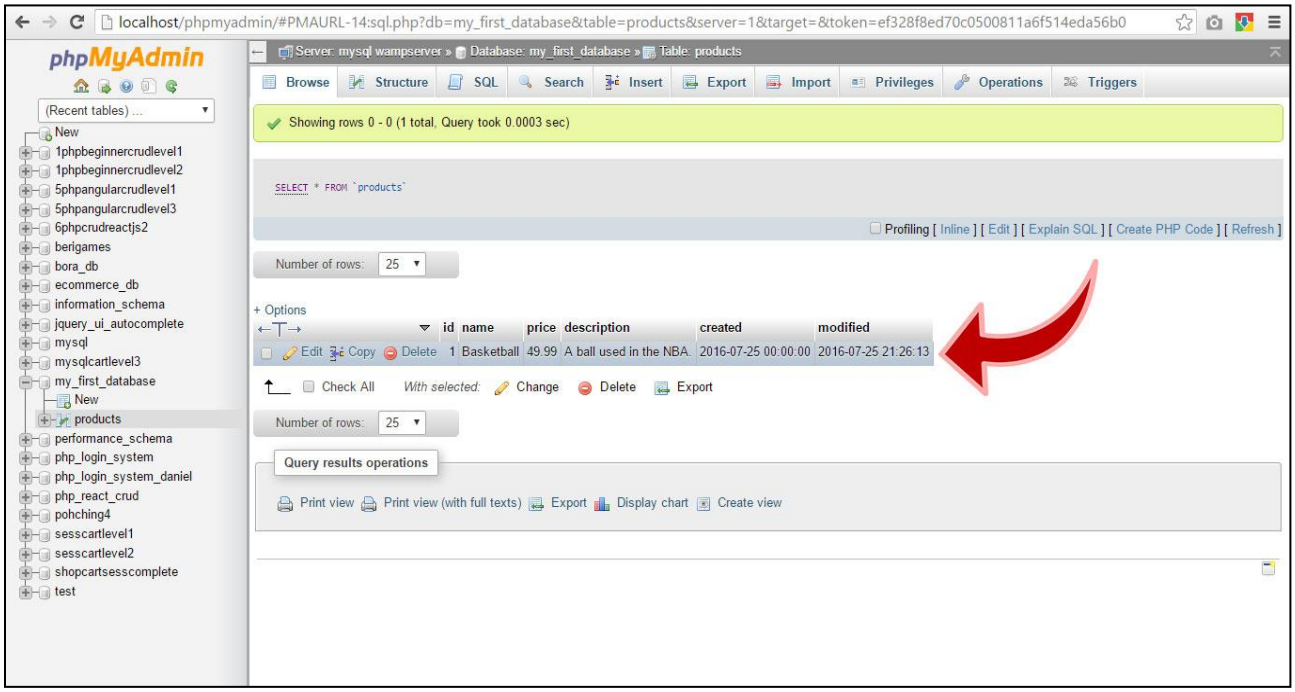
Click the "Insert" tab.



Fill out the form; mimic the data on the following image. Click the "**Go**" button.



We now have a database, a table inside the database and a record inside the table.



Similarly, make other entries in the table.

Steps to Run PHP script with database

1. Go to XAMPP server directory
2. Go to your "C:\xampp\htdocs\" directory
3. Create read_one.php, write code inside read_one.php and save
4. On your browser window, type http://localhost/read_one.php
5. See the output

PHP script that fetches one record from the MySQL using PDO:

```
<?php
// 1. database credentials
$host = "localhost";
$db_name = "my_first_database";
$username = "root";
$password = "";

// 2. connect to database
$con = new PDO("mysql:host={$host};dbname={$db_name}", $username, $password);

// 3. prepare select query
$query = "SELECT id, name, description, price FROM products WHERE id = ? LIMIT 0,1";
$stmt = $con->prepare( $query );

// 4. sample product ID
$product_id=1;

// 5. this is the first question mark in the query
$stmt->bindParam(1, $product_id);

// 6. execute our query
$stmt->execute();

// 7. store retrieved row to the 'row' variable
$row = $stmt->fetch(PDO::FETCH_ASSOC);

// 8. show data to user
echo "<div>Name: " . $row['name'] . "</div>";
echo "<div>Description: " . $row['description'] . "</div>";
echo "<div>Price: $" . $row['price'] . "</div>";
?>
```

Output

You should see the following output.

PHP script that fetches records from the MySQL using MySQLi Procedural:

(Note: First Create Database and Table)

Database name is "Mydb" and Table name is "Data" with 3 columns as Firstname, Lastname and Age.

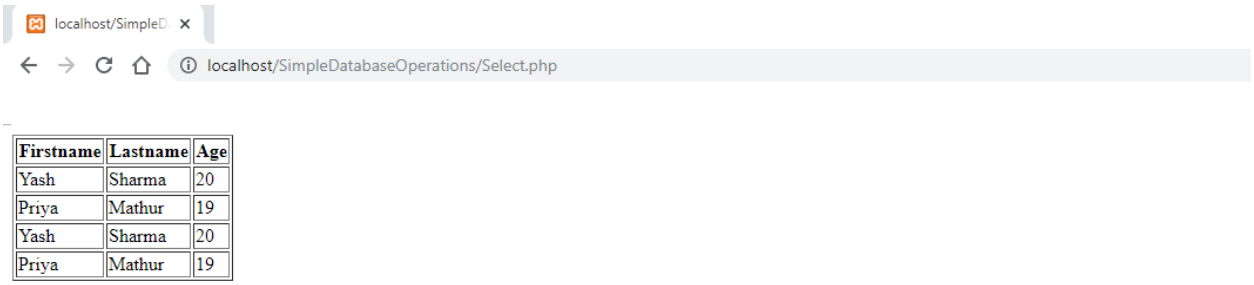
```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Mydb";

// Create connection
$link = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$link) {
   die("Connection failed: " . mysqli_connect_error());
}
$sql = "SELECT * FROM Data";
if ($res = mysqli_query($link, $sql)) {
   if (mysqli_num_rows($res) > 0) {
      echo "<table border='1'>";
      echo "<tr>";
      echo "<th>Firstname</th>";
      echo "<th>Lastname</th>";
      echo "<th>Age</th>";
      echo "</tr>";
      while ($row = mysqli_fetch_array($res)) {
         echo "<tr>";
         echo "<td>".$row['Firstname']."</td>";
         echo "<td>".$row['Lastname']."</td>";
         echo "<td>".$row['Age']."</td>";
```

```
            echo "</tr>";
        }
        echo "</table>";
       mysqli_free_result($res);
    }
    else {
        echo "No matching records are found.";
    }
}
else {
    echo "ERROR: Could not able to execute $sql. ".mysqli_error($link);
}
mysqli_close($link);
?>
```

Output

You should see the following output.



Firstname	Lastname	Age
Yash	Sharma	20
Priya	Mathur	19
Yash	Sharma	20
Priya	Mathur	19

Conclusion:

Written and successfully executed Server Side Script in PHP to generate the web pages dynamically using the database connectivity.