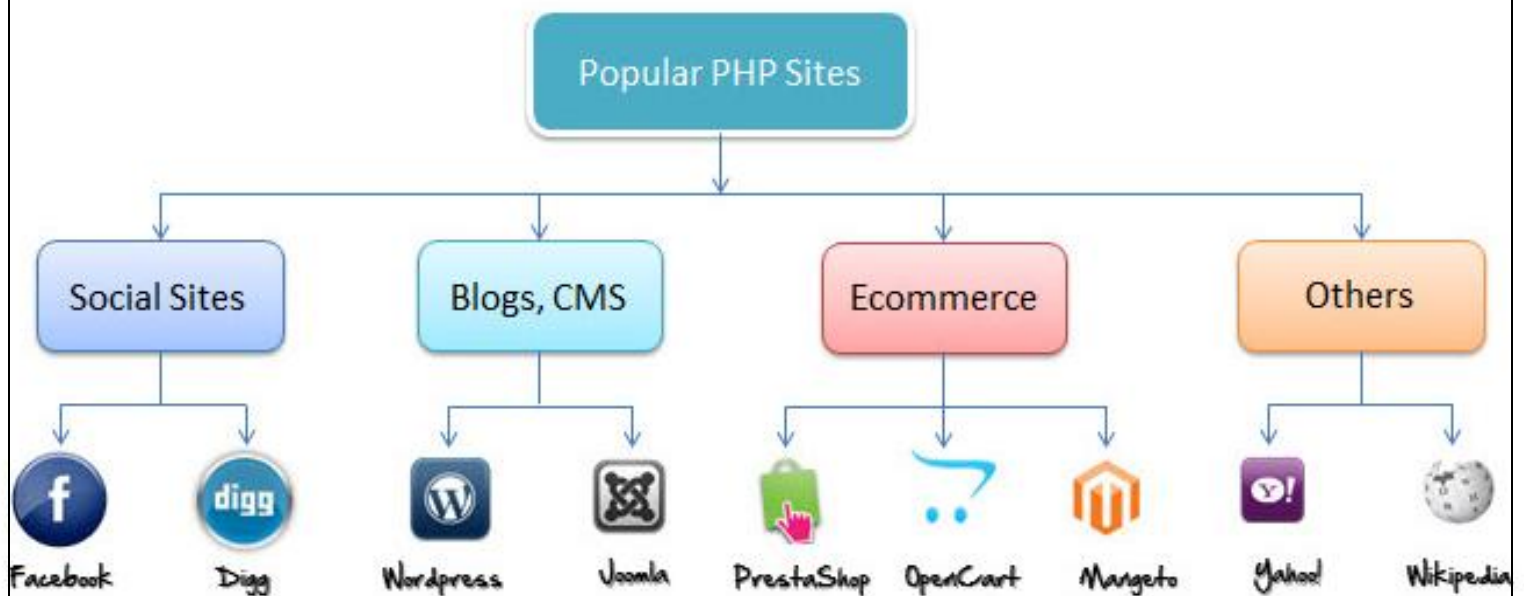


Lab Session #07

(Connectivity: PHP with MySQL)

PHP:

PHP is the most popular scripting language (i-e A scripting language is a language that interprets scripts at runtime) for web development. It is free, open source and server-side (the code is executed on the server).



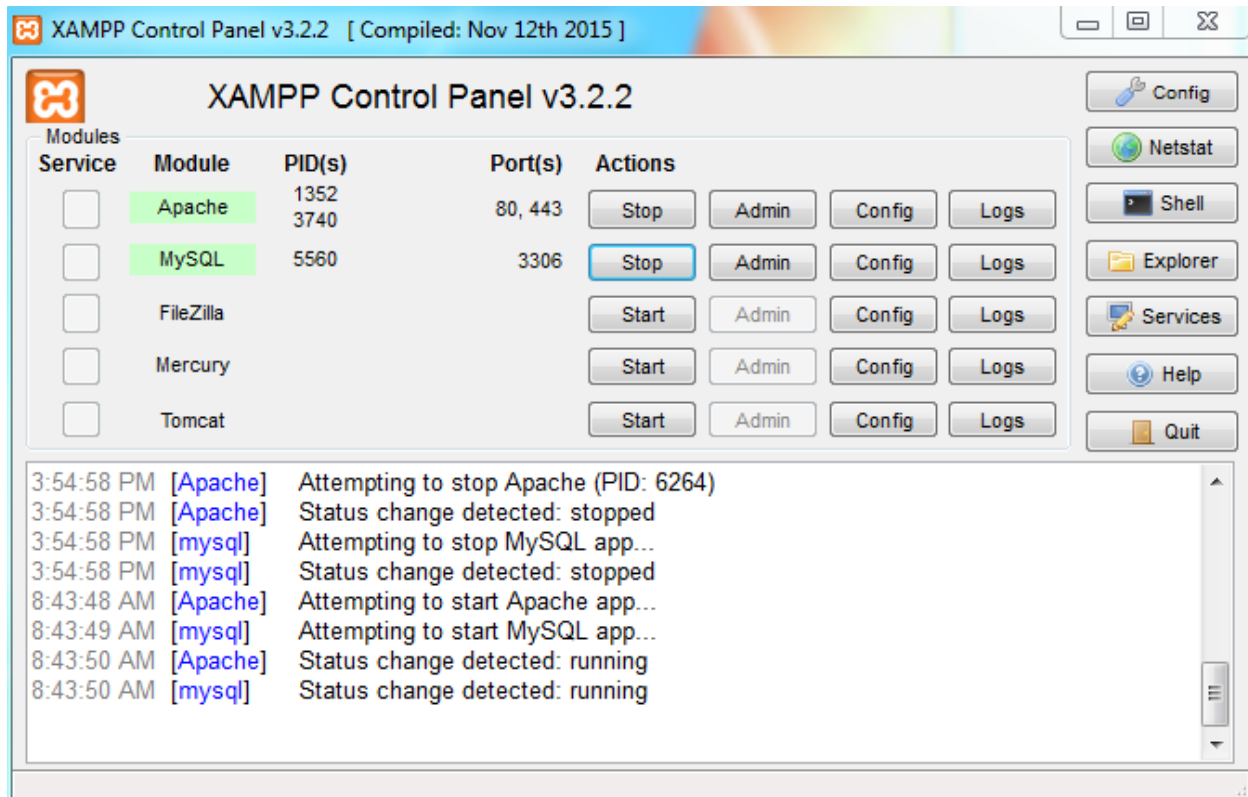
MySQL:

MySQL is a Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). It is also free and open source. The combination of PHP and MySQL gives unmet options to create just about any kind of website - from small contact form to large corporate portal.

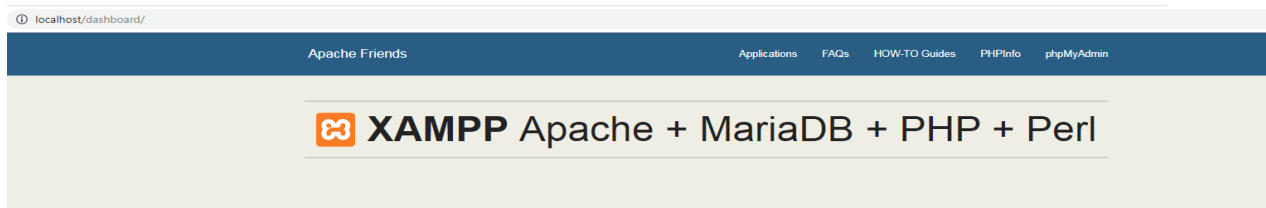
What is required?

1. Text Editor (i-e Notepad++, Brackets, Dreamweaver etc.)
2. Wamp or Xampp Server
3. Web browser (Firefox, Chrome etc.)

Xampp_Control Panel:



Apache _Admin Console:



Welcome to XAMPP for Windows 5.6.31

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, adding yourself to the Mailing List, and liking us on Facebook, following our exploits on Twitter, or adding us to your Google+ circles.

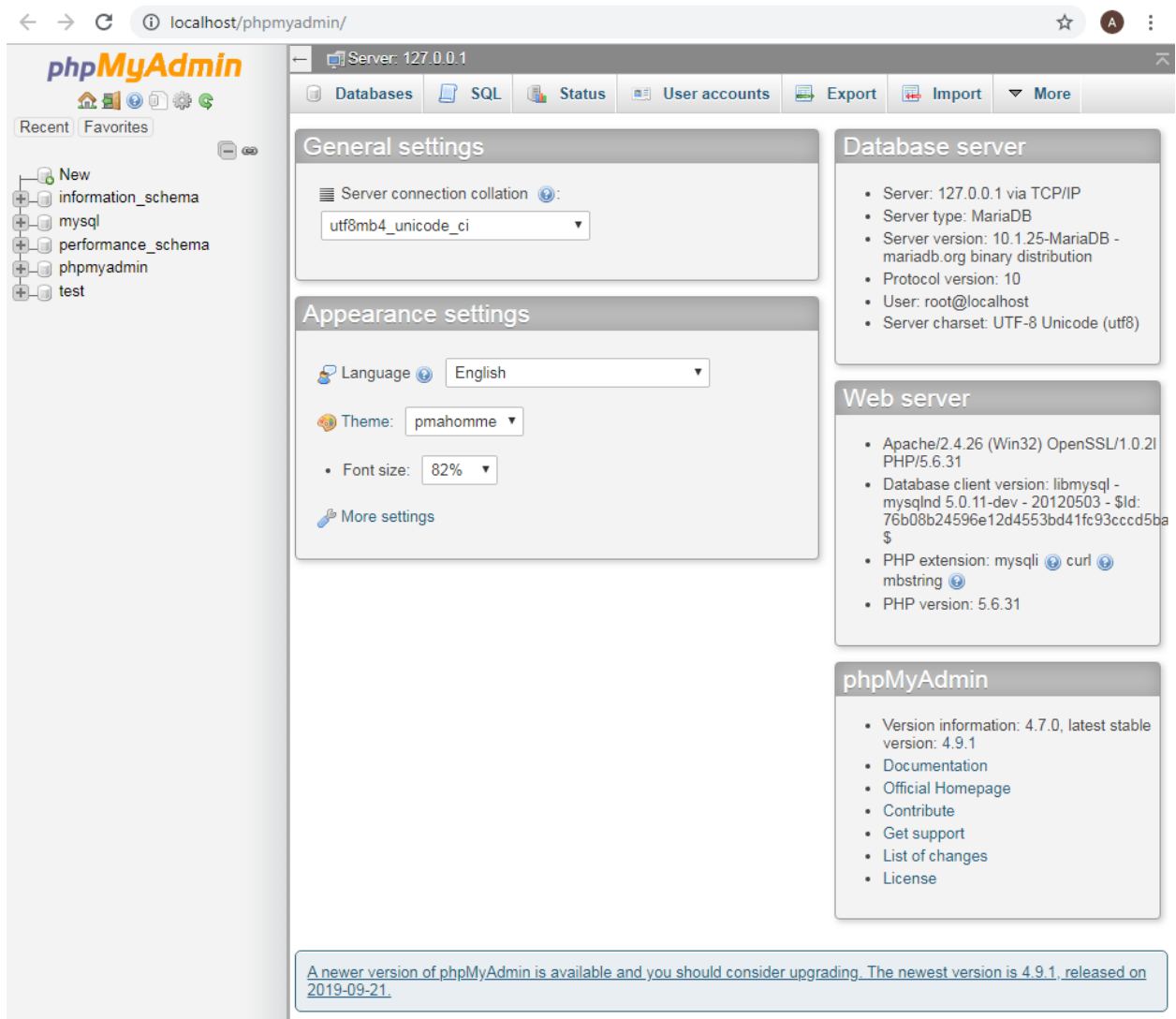
Contribute to XAMPP translation at translate.apachefriends.org.

Can you help translate XAMPP for other community members? We need your help to translate XAMPP into different languages. We have set up a site, translate.apachefriends.org, where users can contribute translations.

Install applications on XAMPP using Bitnami

Apache Friends and Bitnami are cooperating to make dozens of open source applications available on XAMPP, for free. Bitnami-packaged applications include Wordpress, Drupal, Joomla! and dozens of others and can be deployed with one-click installers. Visit the Bitnami XAMPP page for details on the currently available apps.

MySQL-phpMyAdmin Panel:



Creating Users:

Follow these Steps:

1. Go to User Accounts in phpMyAdmin.
2. Click Add user account.
3. Set username, hostname, password and privileges for this user.
4. Click Go.

phpMyAdmin

Server: 127.0.0.1

Databases SQL Status User accounts Export Import Settings Replication Variables

Add user account

Login Information

User name: Use text field: ammara

Host name: Local localhost

Password: Use text field: Strength: Good

Re-type:

Authentication Plugin: Native MySQL authentication

Generate password: Generate

Database for user account

☐ Create database with same name and grant all privileges.

☐ Grant all privileges on wildcard name (username_%).

Global privileges ☐ Check all

Note: MySQL privilege names are expressed in English.

Data	Structure	Administration	Resource limits
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> GRANT	Note: Setting these options to 0 (zero) removes the limit. MAX QUERIES PER HOUR: 0 MAX UPDATES PER HOUR: 0 MAX CONNECTIONS PER HOUR: 0 MAX USER CONNECTIONS: 0
<input checked="" type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> SUPER	
<input checked="" type="checkbox"/> UPDATE	<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> PROCESS	
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP	<input checked="" type="checkbox"/> RELOAD	
<input checked="" type="checkbox"/> FILE	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> SHUTDOWN	
	<input checked="" type="checkbox"/> SHOW VIEW	<input checked="" type="checkbox"/> SHOW DATABASES	
	<input checked="" type="checkbox"/> CREATE ROUTINE	<input checked="" type="checkbox"/> LOCK TABLES	

PHP Connection to MySQL:

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5
6 // Create connection
7 $conn = mysqli_connect($servername, $username, $password);
8
9 // Check connection
10 if (!$conn) {
11     die("Connection failed: " . mysqli_connect_error());
12 }
13 echo "Connected Successfully";
14 ?>
```

Save this file in C:\xampp\htdocs folder.

To run the script type in the browser localhost/filename.php

Creating Database:

Two ways

1. Directly on phpMyAdmin
2. Front end (Text Editor)

1. Direct on phpMyAdmin:

Follow these steps:

1. Go to Databases in phpMyAdmin.
2. Fill the Database name field then Click Create.

The screenshot shows the phpMyAdmin interface for a MySQL server at 127.0.0.1. The 'Databases' tab is selected. On the left, a sidebar shows a tree of databases: New, example, information_schema, mysql, new, performance_schema, phpmyadmin, and test. The main area displays the 'Databases' section with a 'Create database' form. The form has a text input for the database name containing 'Lab_Example' and a dropdown for 'Collation' set to 'latin1_swedish_ci'. A 'Create' button is to the right. Below the form is a table listing existing databases:

Database	Collation	Action
<input type="checkbox"/> example	latin1_swedish_ci	Check privileges
<input type="checkbox"/> information_schema	utf8_general_ci	Check privileges
<input type="checkbox"/> mysql	latin1_swedish_ci	Check privileges
<input type="checkbox"/> new	latin1_swedish_ci	Check privileges
<input type="checkbox"/> performance_schema	utf8_general_ci	Check privileges
<input type="checkbox"/> phpmyadmin	utf8_bin	Check privileges
<input type="checkbox"/> test	latin1_swedish_ci	Check privileges
Total: 7		latin1_swedish_ci

At the bottom, there is a 'Check all' checkbox and a 'With selected: Drop' button.

2. Front End (Text Editor)

```
1  <?php
2  $servername = "localhost";
3  $username = "root";
4  $password = "";
5
6  // Create connection
7  $conn = new mysqli($servername, $username, $password);
8  // Check connection
9  if ($conn->connect_error) {
10     die("Connection failed: " . $conn->connect_error);
11 }
12
13 // Create database
14 $sql = "CREATE DATABASE new";
15 if ($conn->query($sql) == TRUE) {
16     echo "Database created successfully";
17 } else {
18     echo "Error creating database: " . $conn->error;
19 }
20
21 $conn->close();
22 ?>
```

Creating Tables:

Two ways

1. Directly on phpMyAdmin
2. Front end (Text Editor)

1. Direct on phpMyAdmin:

Follow these steps:

1. Select the database in which you want to create table.
2. Fill out the table name and quantity of fields then click Go.
3. Give every field a proper name ,data type, size and Constraint (if any).
4. Click Go.

2.Front end (Text Editor)

```
1 <?php
2
3 $servername = "localhost";
4 $username = "root";
5 $password = "";
6 $dbname = "new";
7
8 // Create connection
9 $conn = new mysqli($servername, $username, $password, $dbname);
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to create table
16 $sql = "CREATE TABLE MyGuests (
17     id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
18     firstname VARCHAR(30) NOT NULL,
19     lastname VARCHAR(30) NOT NULL,
20     email VARCHAR(50),
21     reg_date TIMESTAMP
22 )";
23
24 if ($conn->query($sql) == TRUE) {
25     echo "Table MyGuests created successfully";
26 } else {
27     echo "Error creating table: " . $conn->error;
28 }
29
30 $conn->close();
31 ?>
```

Inserting Data into Table:

```
1 <?php
2
3 $servername = "localhost";
4 $username = "root";
5 $password = "";
6 $dbname = "new";
7
8 // Create connection
9 $conn = new mysqli($servername, $username, $password, $dbname);
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to create table
16 $sql = "INSERT INTO myguests(`id`, `firstname`, `lastname`, `email`) VALUES ('1','john','dalton','john@gmail.com')";
17
18 if ($conn->query($sql) == TRUE) {
19     echo "data inserted successfully";
20 } else {
21     echo "Error inserting in table: " . $conn->error;
22 }
23
24 $conn->close();
25 ?>
```


Inserting multiple records:

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "new";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9 // Check connection
10 if ($conn->connect_error) {
11     die("Connection failed: " . $conn->connect_error);
12 }
13
14 $sql = "INSERT INTO MyGuests (firstname, lastname, email)
15 VALUES ('John', 'Doe', 'john@example.com')";
16 $sql .= "INSERT INTO MyGuests (firstname, lastname, email)
17 VALUES ('Mary', 'Moe', 'mary@example.com')";
18 $sql .= "INSERT INTO MyGuests (firstname, lastname, email)
19 VALUES ('Julie', 'Dooley', 'julie@example.com')";
20
21 if ($conn->multi_query($sql) == TRUE) {
22     echo "New records created successfully";
23 } else {
24     echo "Error: " . $sql . "<br>" . $conn->error;
25 }
26
27 $conn->close();
28 ?>
```

Prepared Statements and Bound Parameters:

- A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.
- Prepared statements basically work like this:
 - ✓ Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
 - ✓ The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
 - ✓ Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Advantages:

Compared to executing SQL statements directly, prepared statements have three main advantages:

- Prepared statements reduces parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

```
1 <?php
2 $servername="localhost";
3 $username="root";
4 $password="";
5 $dbname="new";
6 //create connection
7 $conn=new mysqli($servername,$username,$password,$dbname);
8 //check connection
9 if($conn->connect_error)
10 {
11     die("connection failed:".$conn->connect_error);
12 }
13 //prepare and bind
14 $stmt=$conn->prepare("INSERT INTO myguests(firstname,lastname,email) values(?,?,?)");
15 $stmt->bind_param("sss",$firstname,$lastname,$email);
16
17 $firstname="ammara";
18 $lastname="yaseen";
19 $email="ammara@gmail.com";
20 $stmt->execute();
21
22 $firstname="ayesha";
23 $lastname="farhan";
24 $email="ayesha@gmail.com";
25 $stmt->execute();
26
27 echo "new record added";
28 $stmt->close();
29 $conn->close();
30 ?>
```

Selecting Data:

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "new";
6 // Create connection
7 $conn = new mysqli($servername, $username, $password, $dbname);
8 // Check connection
9 if ($conn->connect_error) {
10     die("Connection failed: " . $conn->connect_error);
11 }
12 $sql = "SELECT id, firstname, lastname FROM MyGuests";
13 $result = $conn->query($sql);
14 if ($result->num_rows > 0) {
15     // output data of each row
16     while($row = $result->fetch_assoc()) {
17         echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
18     }
19 } else {
20     echo "0 results";
21 }
22 $conn->close();
```

Deleting Data:

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "new";
6 // Create connection
7 $conn = new mysqli($servername, $username, $password, $dbname);
8 // Check connection
9 if ($conn->connect_error) {
10     die("Connection failed: " . $conn->connect_error);
11 }
12 // sql to delete a record
13 $sql = "DELETE FROM MyGuests WHERE id=3";
14 if ($conn->query($sql) == TRUE) {
15     echo "Record deleted successfully";
16 } else {
17     echo "Error deleting record: " . $conn->error;
18 }
19 $conn->close();
20 ?>
```

Updating Data:

```
1  <?php
2  $servername = "localhost";
3  $username = "root";
4  $password = "";
5  $dbname = "new";
6  // Create connection
7  $conn = new mysqli($servername, $username, $password, $dbname);
8  // Check connection
9  if ($conn->connect_error) {
10     die("Connection failed: " . $conn->connect_error);
11 }
12 $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";
13 if ($conn->query($sql) === TRUE) {
14     echo "Record updated successfully";
15 } else {
16     echo "Error updating record: " . $conn->error;
17 }
18 $conn->close();
19 ?>
```