Division of communication and network
Xi'an Jiaotong-Liverpool University

Lab 1 – Introduction to PHP and XAMPP

Aim

The aim of this lab is to learn the very basics of PHP and XAMPP. At the end of the lab, you should be able to use PHP create a webpage, and display data stored in database.

Resources

1. **XAMPP** is an easy to install Apache server distribution containing MariaDB, PHP, and Perl. We can make our web and database server ready by simply click "next"!
   You may download XAMPP installation file from LMO or it's official website.
2. **Brackets 2.0.1** is the recommended IDE for CAN302. **IDE** (Integrated development environment) will help us to have a better development experience. There are many other choices, such as VS code, Sublime Text, WebStorm …… you can choose your favourite one.
   You may download Brackets installation file from LMO or it's official website.
3. **M-dev-store.zip** is project skeleton with Bootstrap3.37 and many empty folders.
   You may download the zip file from LMO.

Tips:

1. If you are not sure why you are doing something, ask a TA. This is what they are here for.

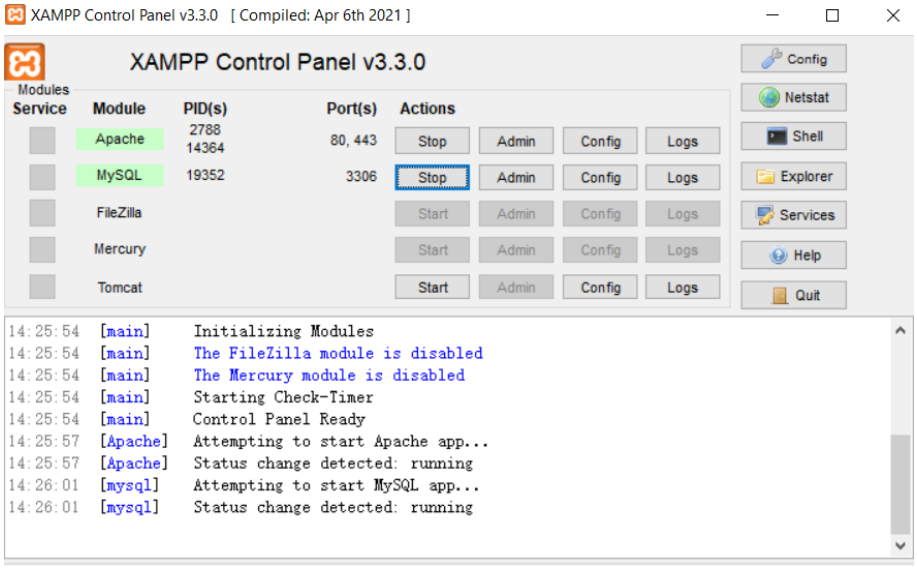
2. Our labs have different focus than the the M-Dev-Store online videos. While if you want to be an expert, you are recommended take both labs and on-line videos.

3. The forums are available for questions and discussions.

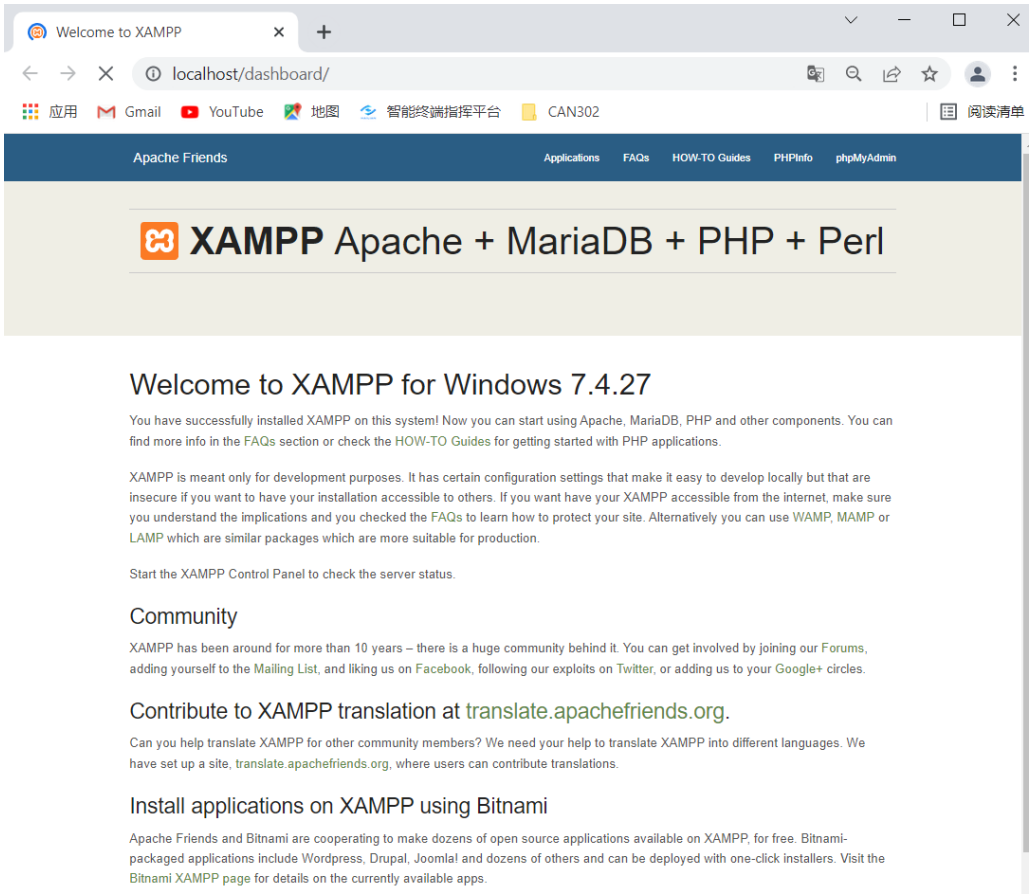
4. These labs are expected take more than the 2 allocated hours. You should complete them in your own time before the next lab. Practice makes perfect!

Install XAMPP:

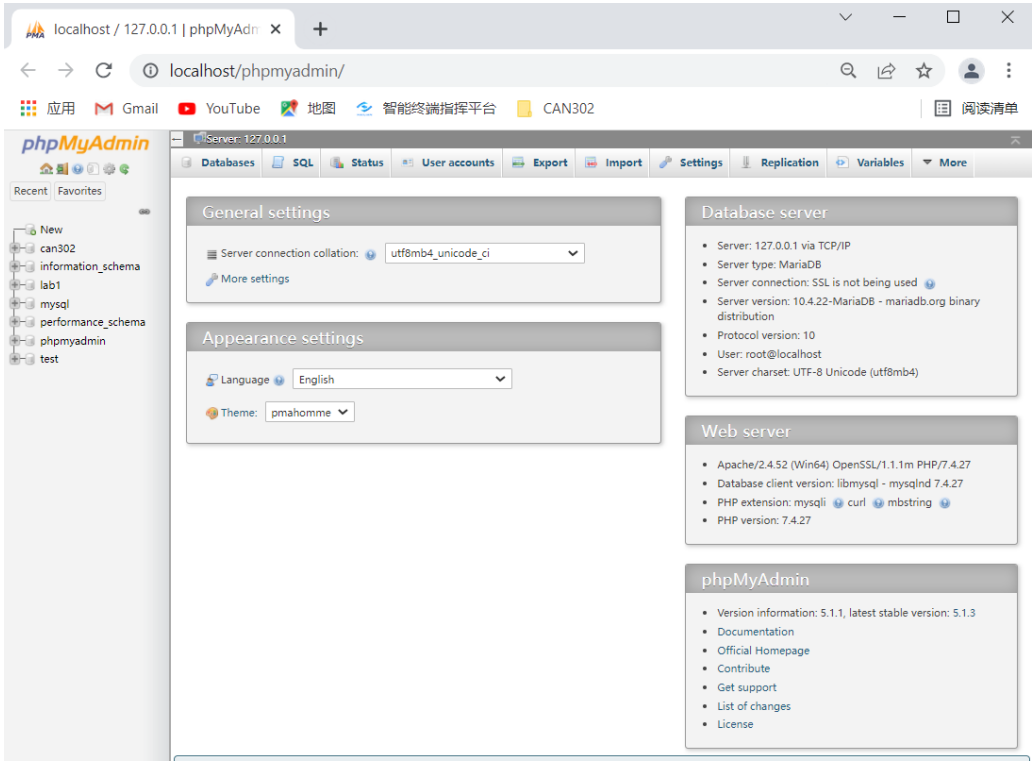
1. Links for installation files (both Windows and Mac) can be found in LMO. Download them and install them in your computer.
   Open the XAMPP control panel, start both Apache Server and MySQL server. It will look like:



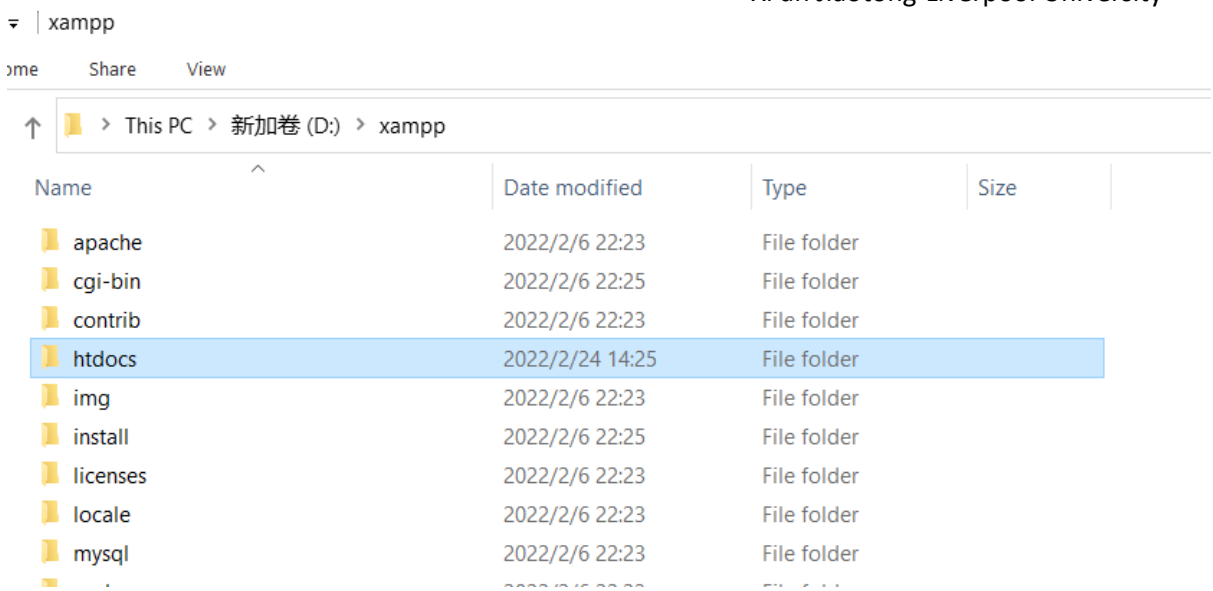
If everything works fine, by click admin button of Apache server will like this:



And by click admin button of MySQL server will like this:



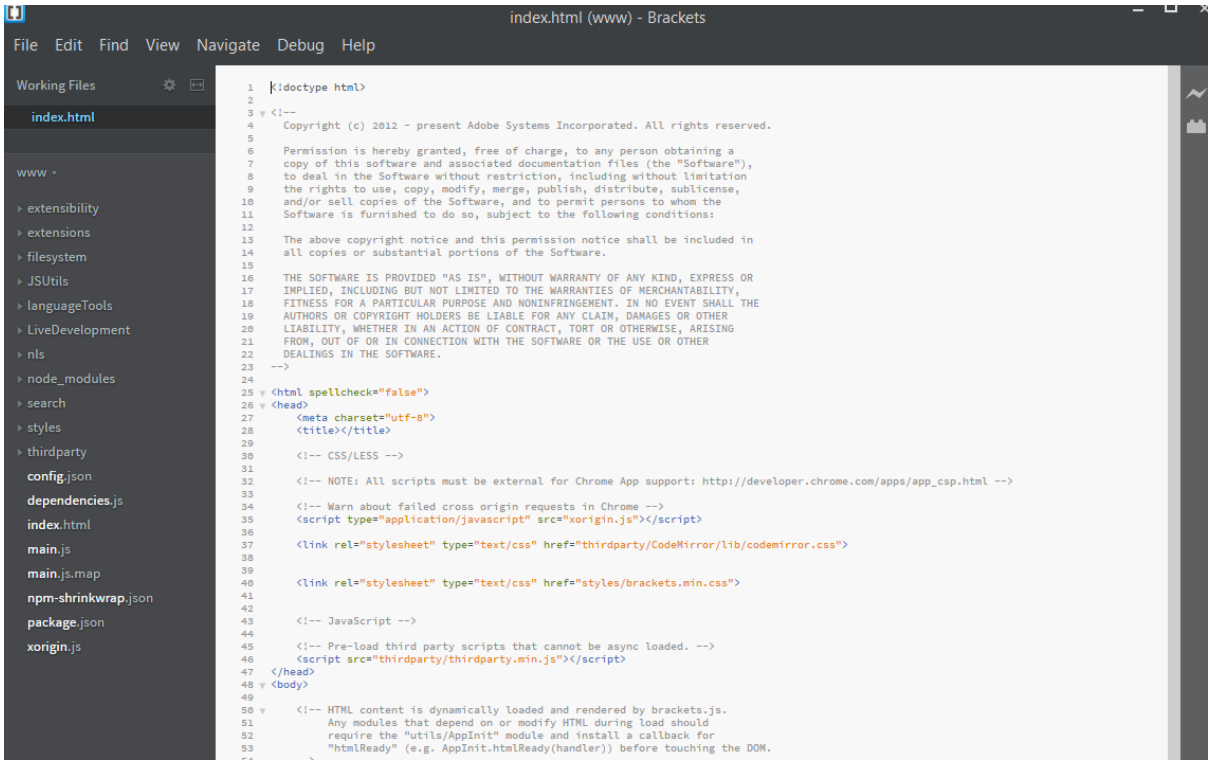
The host files are under folder "htdocs" within XAMPP application. You can find them as:



You may create a shortcut to "htdocs" for easy access.

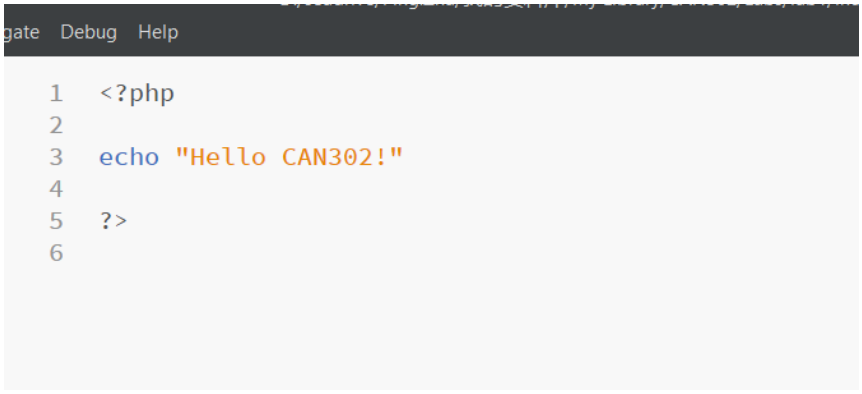
Install Brackets (optional):

2. Many IDEs can be used for php programming. Here is an example for Brackets. After installation, it looks like:



Hello CAN302:

3. Unzip M-dev-store.zip under htdocs folder and may rename it to can302. Open the "index.php" and write the code as:



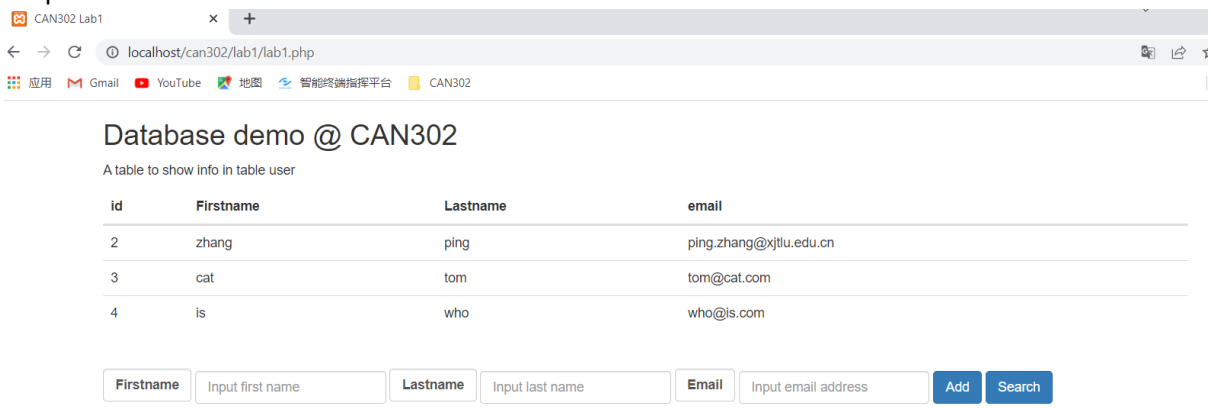
"<?php" tells server php code start from here and "?>" is the end of php code.

Then, use web-browser to view http://localhost/can302/, you should able to see:

Congratulation for your Hello world by PHP

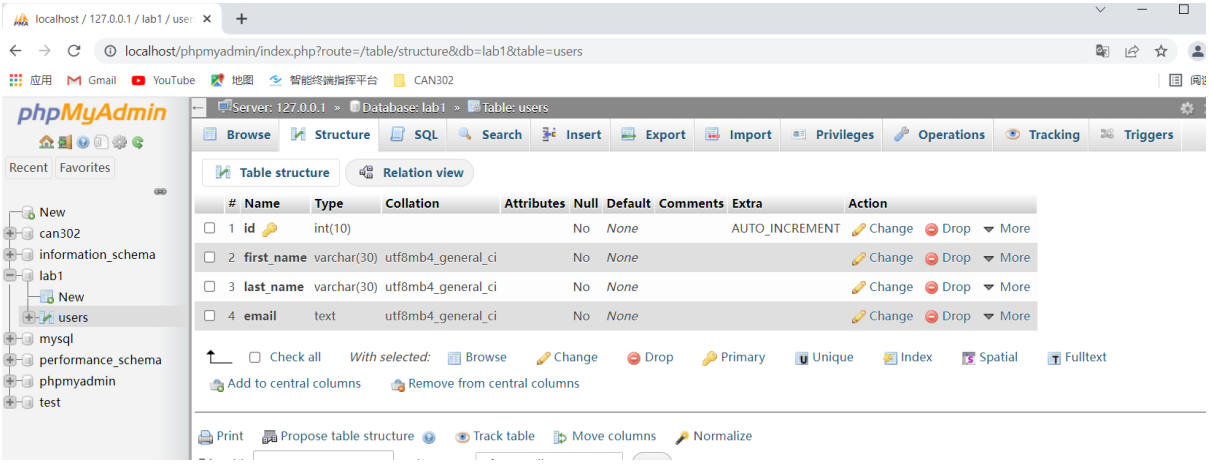
First demo with database:

4. Let's create a folder named "lab1" under "can302" and new a file named "lab1.php". The final output should look like:



In above, we get data from database and show them as a table in the webpage. There are two buttons: by "add", we can add new data into database; by "search", we can filter the result in the table by conditions.

First of all, we need to create a database and then a table to hold such data. We can use "phpmyadmin" for that purpose as:



In above, a database named lab1 was created and a table named users was created with four columns: id, first_name, last_name and email.

HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li> and many others.

Bootstrap has many pre-defined classes (styles and interaction). In the <head> tag, we include the necessary files of bootstrap. we can make the front-end job much easier. Some key code is as:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>CAN302 Lab1</title>
    <link rel="stylesheet" href="/can302/styles/bootstrap-337.min.css">
    <link rel="stylesheet" href="/can302/font-awsome/css/font-awesome.min.css">
    <link rel="stylesheet" href="/can302/styles/style.css">
    <script src="/can302/js/jquery-331.min.js"></script>
    <script src="/can302/js/bootstrap-337.min.js"></script>
</head>
```

In html, "form" is an important method for webpage to interact with server. We can use the form tag to create a form as:

```
<div class="container">
<form class="form-inline" role="form" action="" method="post" >
    <label class="form-control" for="first"> Firstname </label>
    <input type="text" class="form-control" id="first" placeholder="Input first name" name="first">
    <label class="form-control" for="last"> Lastname </label>
    <input type="text" class="form-control" id="last" placeholder="Input last name" name="last">
    <label class="form-control" for="email"> Email </label>
    <input type="text" class="form-control" id="email" placeholder="Input email address" name="email">
    <button type="submit" class="btn btn-primary" id="add" name="add" value="add"> Add </button>
    <button type="submit" class="btn btn-primary" id="search" name="search" value="search"> Search </button>
</form>
</div>
```

There are "name" for each element in the form, which is the parameter for receiving the posted data.

We can receive the post data by different webpage or by the same webpage. Here, we use the same webpage to receive the data. So, the value of "action" is "".

To receive the data, we may have a safe function to avoid "null" error. Which is as:

```
function mypost($str) {
    $val = !empty($_POST[$str]) ? $_POST[$str] : '';
    return $val;
}
```

We need to link to our database for operation. In php, we can use "mysqli" to operate database, the code is as:

```
//connect to database
$con = mysqli_connect("localhost", "root", "", "lab1");
if (mysqli_connect_errno($con)) {
    die("Connect to MySQL failed: " . mysqli_connect_error());
}
```

In which, localhost means the MySQL server address, root and "" are the username and password for the database and "lab1" is the name of database.

Then we can have the following logic:

```
$last = mypost('last');
$first = mypost('first');
$email = mypost('email');

if (isset($_POST['add'])) {
    $sql = "INSERT INTO `users` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, '".$first."', '".$last."',
    '".$email."')";
    $query = mysqli_query($con,$sql);
}

if (isset($_POST['search'])) {
    $sql = "select * from users where first_name LIKE '%".$first."%' and last_name LIKE '%".$last."%' and email LIKE
    '%".$email."%'";
}
else {
    $sql = "select * from users";
}
$query = mysqli_query($con,$sql);
```

We receive all parameters first. If user click "add" button, then we insert the received data into table "users". If user click "search" button, then we make the sql statement with certain conditions, otherwise we will get all data from table "users".

Then we need to show the data in the webpage. The code is as:

```
<div class="container">
    <h2> Database demo @ CAN302 </h2>
    <p>  A table to show info in table user</p>
    <table class="table">
        <thead>
            <tr>
                <th>id</th>
                <th>Firstname</th>
                <th>Lastname</th>
                <th>email</th>
            </tr>
        </thead>
        <tbody>
            <?php
            while($row = mysqli_fetch_array($query)){
                echo "<tr>";
                echo "<td>".$row['id']."</td>";
                echo "<td>".$row['last_name']."</td>";
                echo "<td>".$row['first_name']."</td>";
                echo "<td>".$row['email']."</td>";
                echo "</tr>";
            }
            mysqli_close($con);
            ?>
        </tbody>
    </table>
</div>
```

Create the table head with tag <thead> and using "while" loop to output data from the search result.