Advanced Web Tutorial 10

Editor – Brackets / Visual Studio Code

Goals

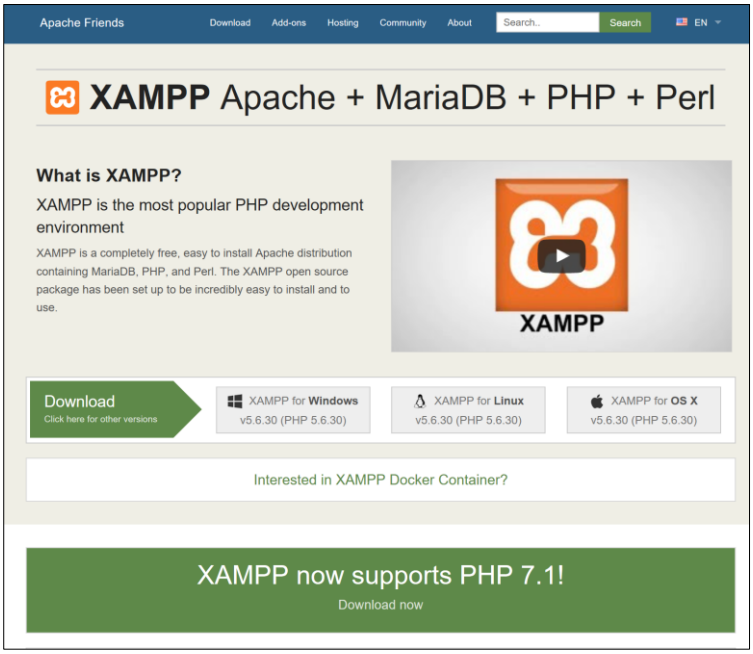
Creating a blog with PHP and MySql.

- Set up and configuration of Xampp
- Learning Data flow using Create/Read/Update and Delete

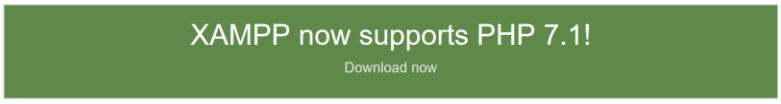
Things to note: Each week Xampp will need to be installed. Xampp is Windows software, similar software is available for Mac, called Mamp.

Installing and configuring Xampp

Go to the website: https://www.apachefriends.org/index.html



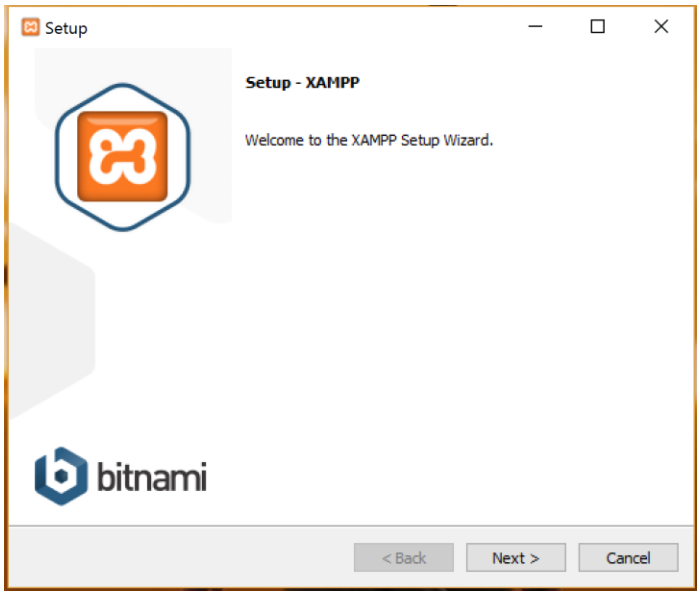
Download the Xampp with PHP 7.1, click on the following link



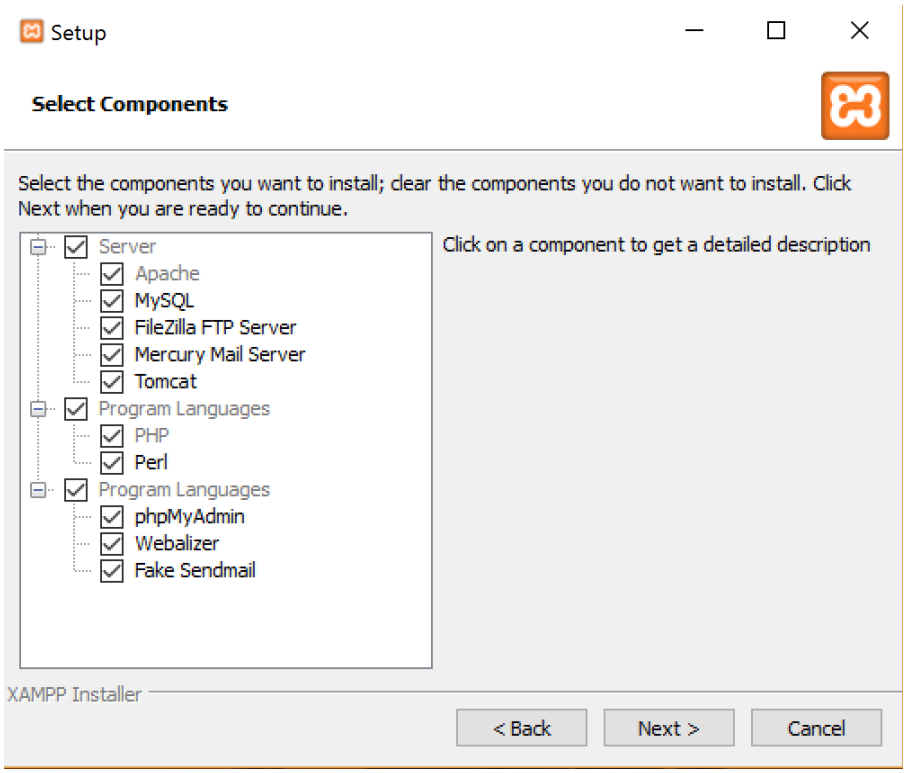
Then click on

Download to the desktop, Once there, double click on the file and following the prompts

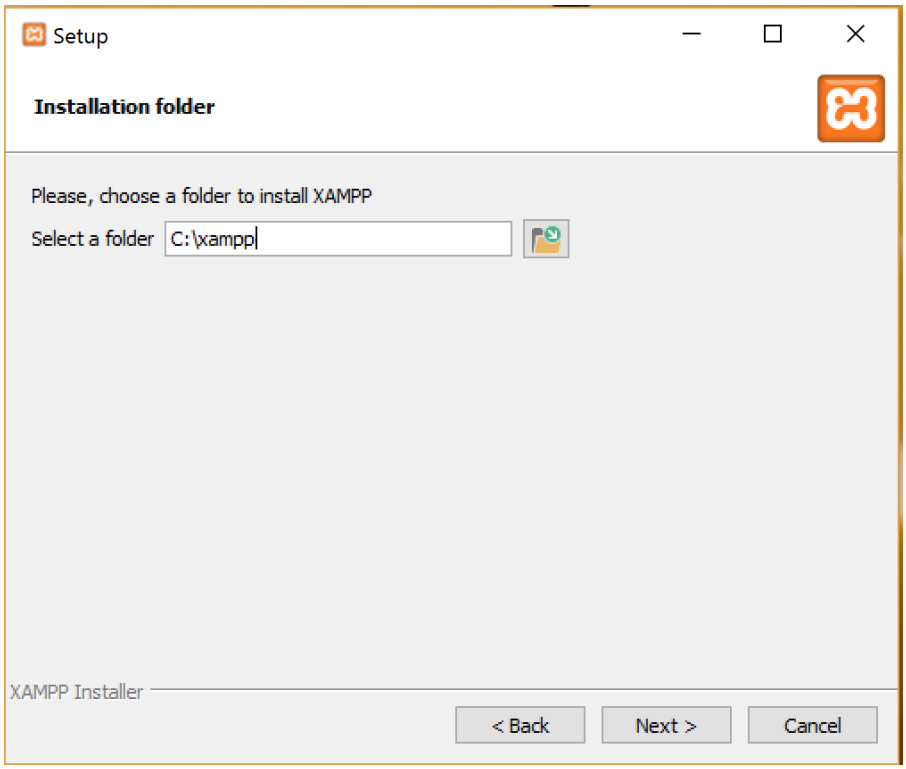
Click Next



Click Next

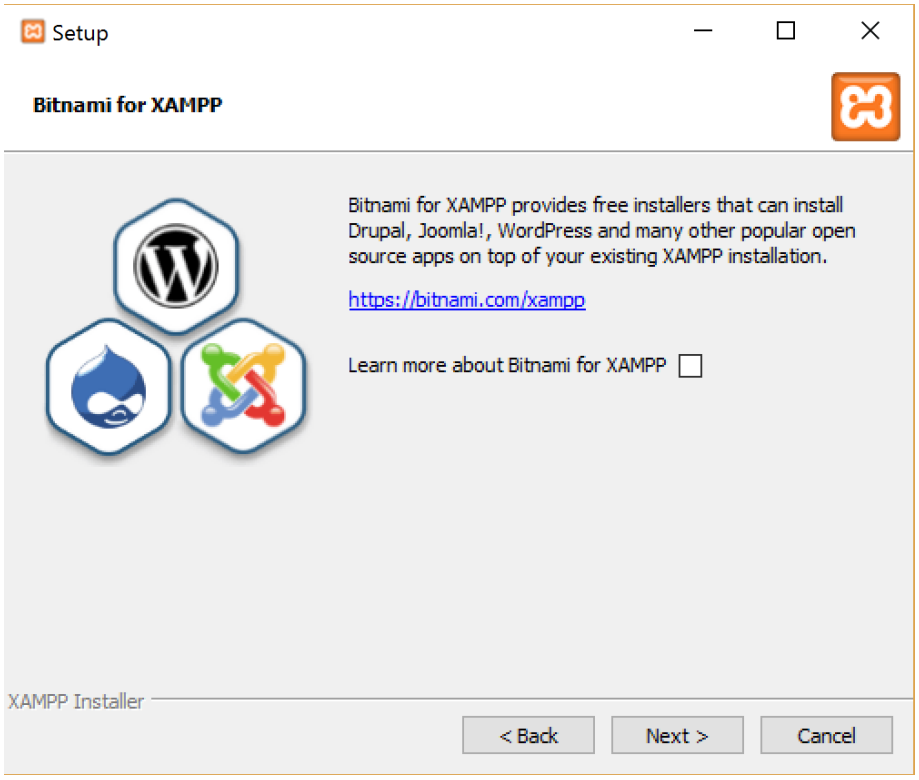


Click Next

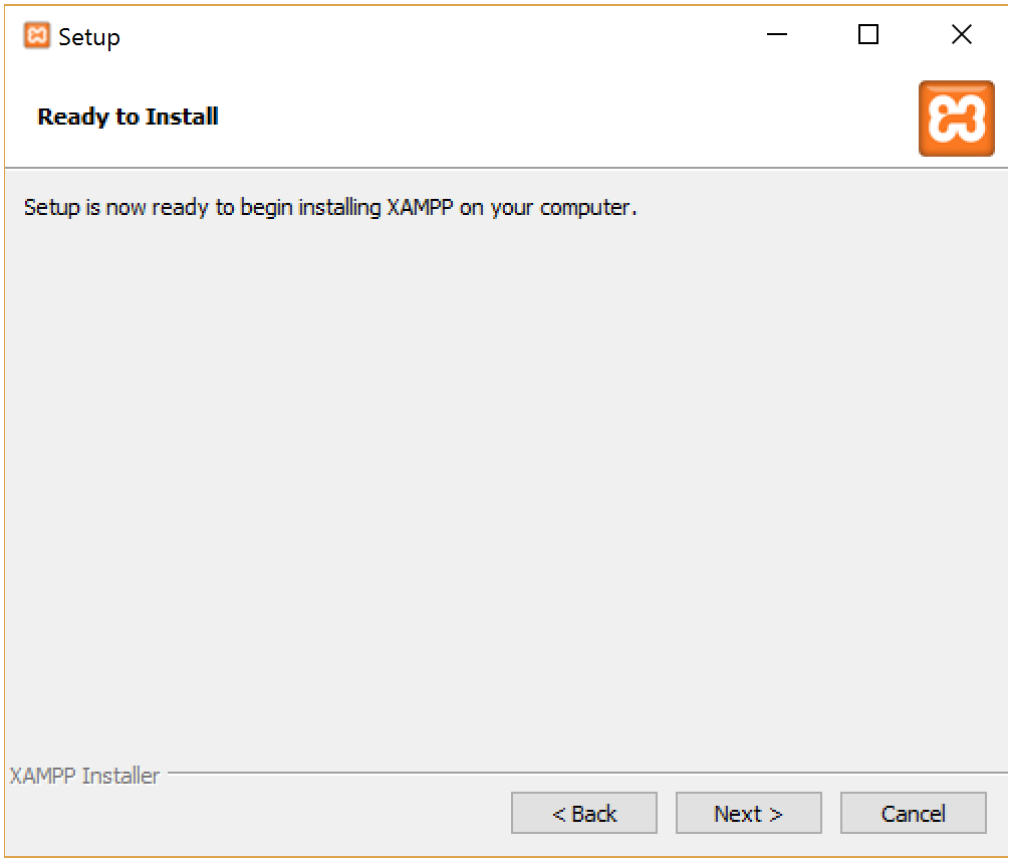


NB: This is the installation folder, all of your web pages will belong in c:\xampp\htdocs

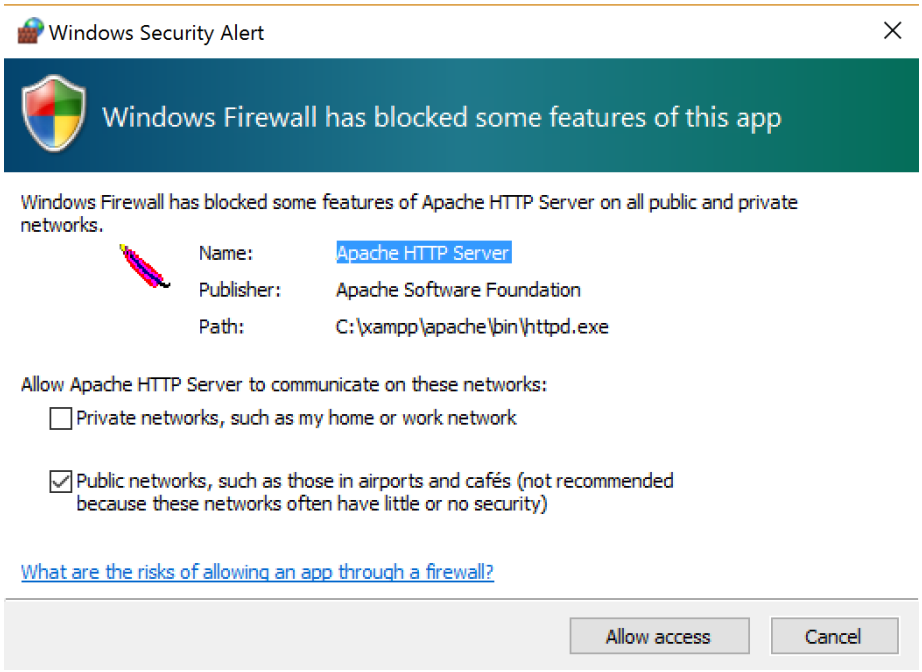
Untick Bitnami and click Next



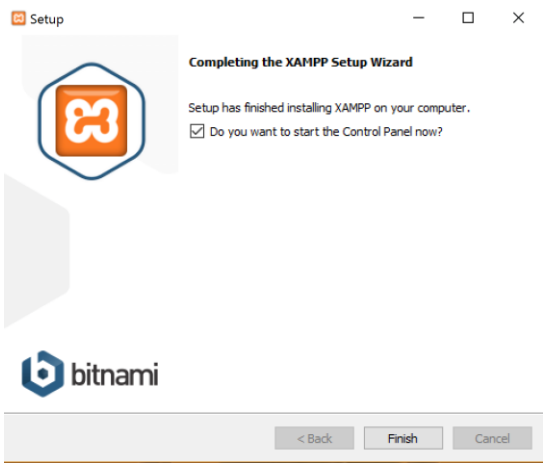
Click next to install



Allow any pop ups such as Apache or MySql

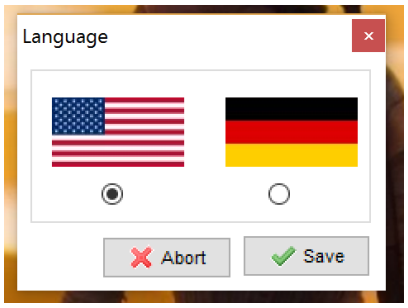


Then click finish



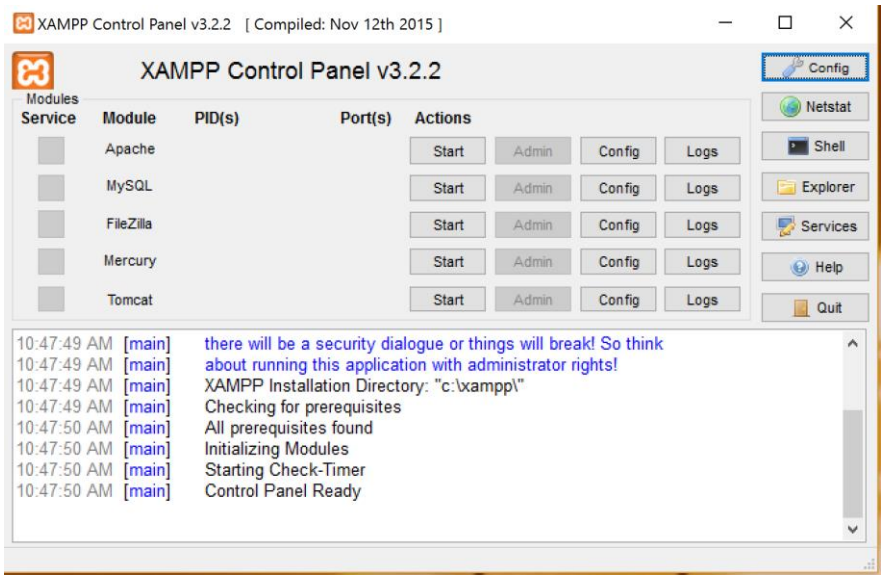
Now we configure Xampp.

Assuming you left the “Do you want to start the Control Panel” ticked, you should see the following on your screen.



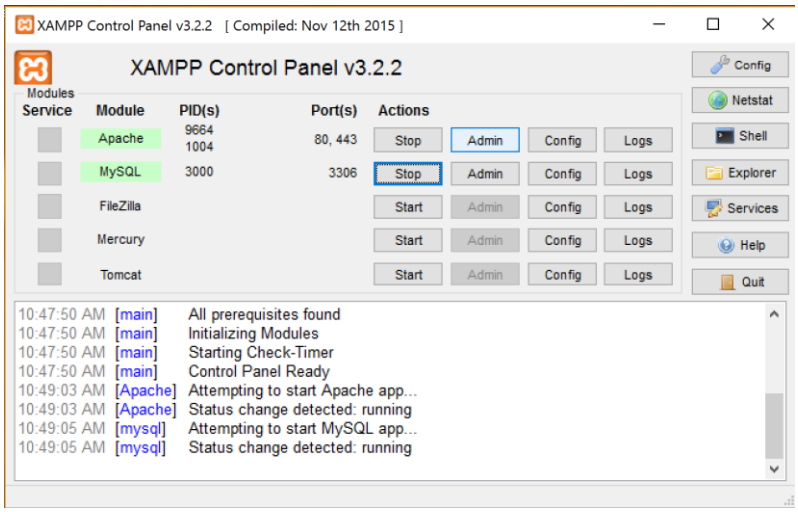
Click on Save

The following should appear:



From here we need to click Start on Apache and MySql.

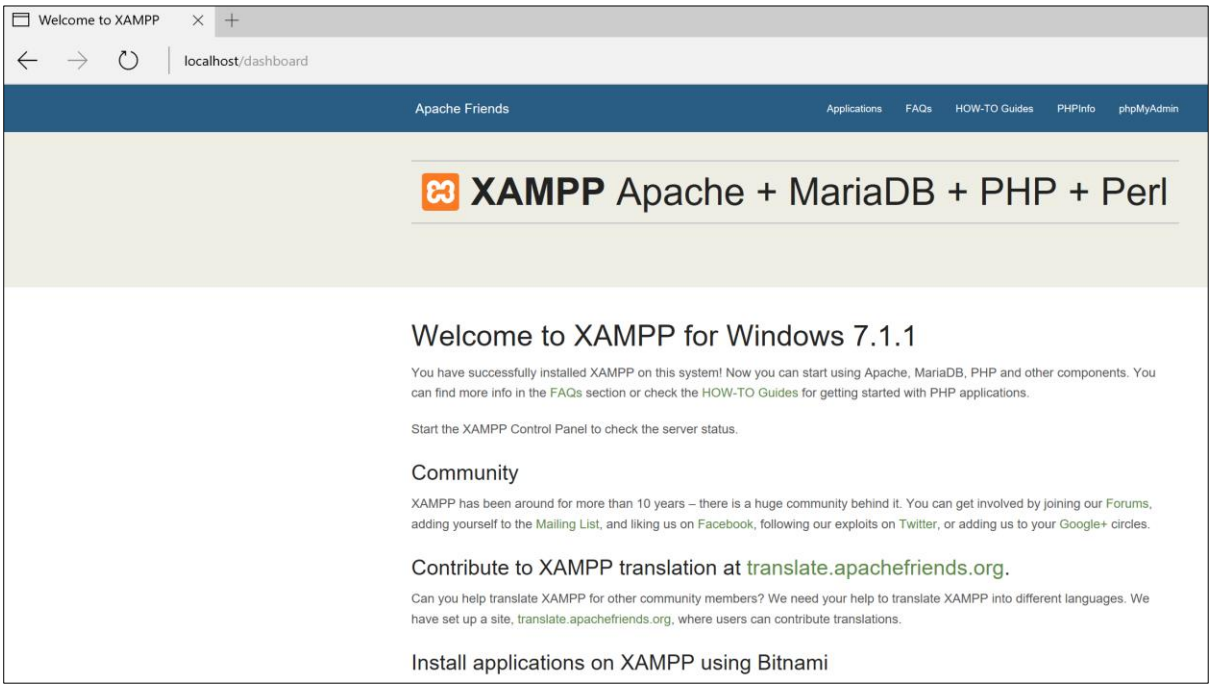
You should see the following.



We now have the Apache webserver up and running with MySql.

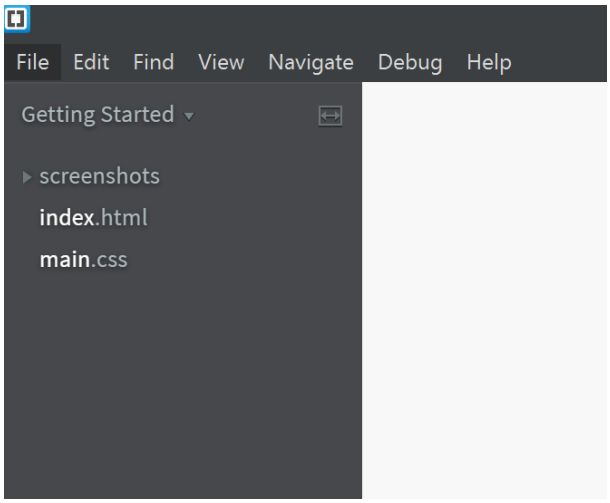
To check that it is working load up a browser and type localhost into the address bar.

You should see the following:

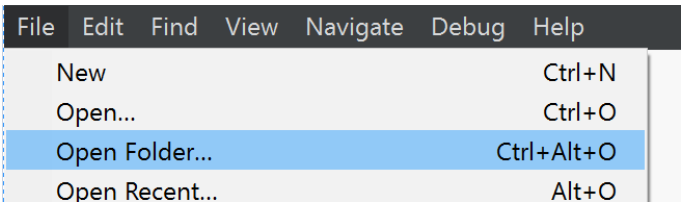


Now, this is all configuration information from Xampp, what we should do is clean out the folder htdocs, to put our own files. To do this, we will use brackets to clean out the folder.

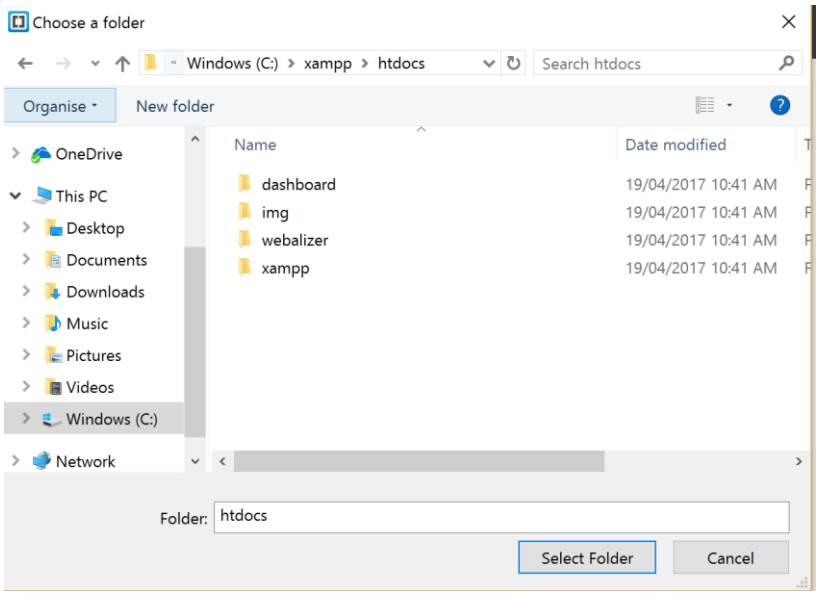
Load up brackets



From here, go to open folder

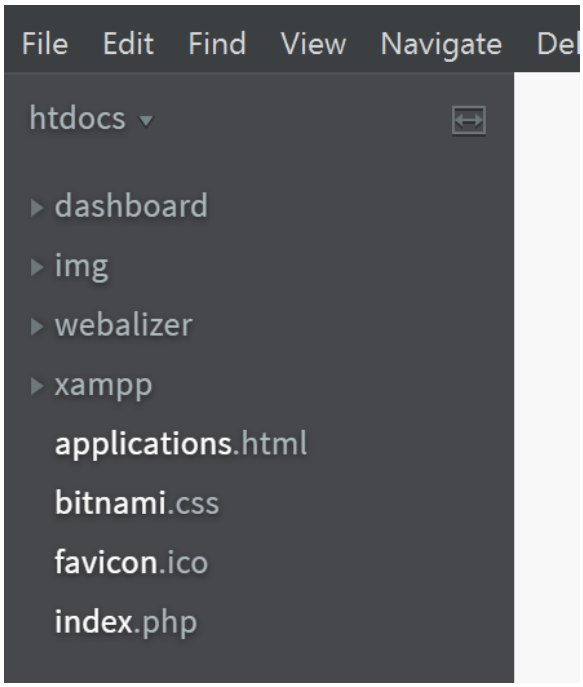


Navigate to c:\xampp\htdocs

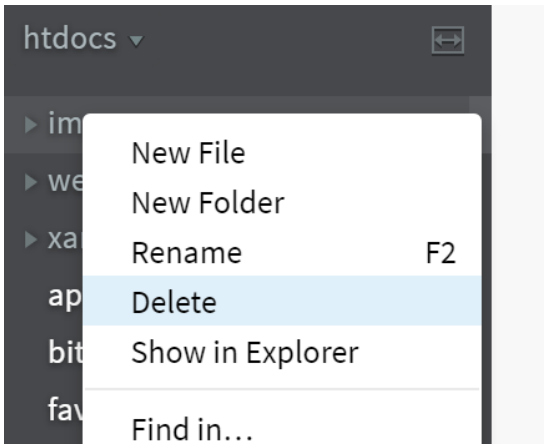


Click Select

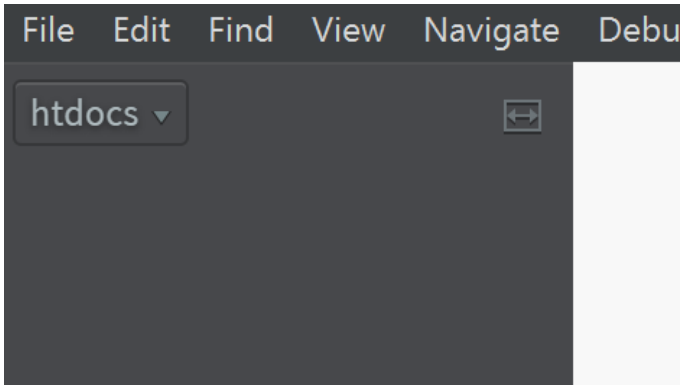
This should give you the following set of files in the navigation pane of brackets:



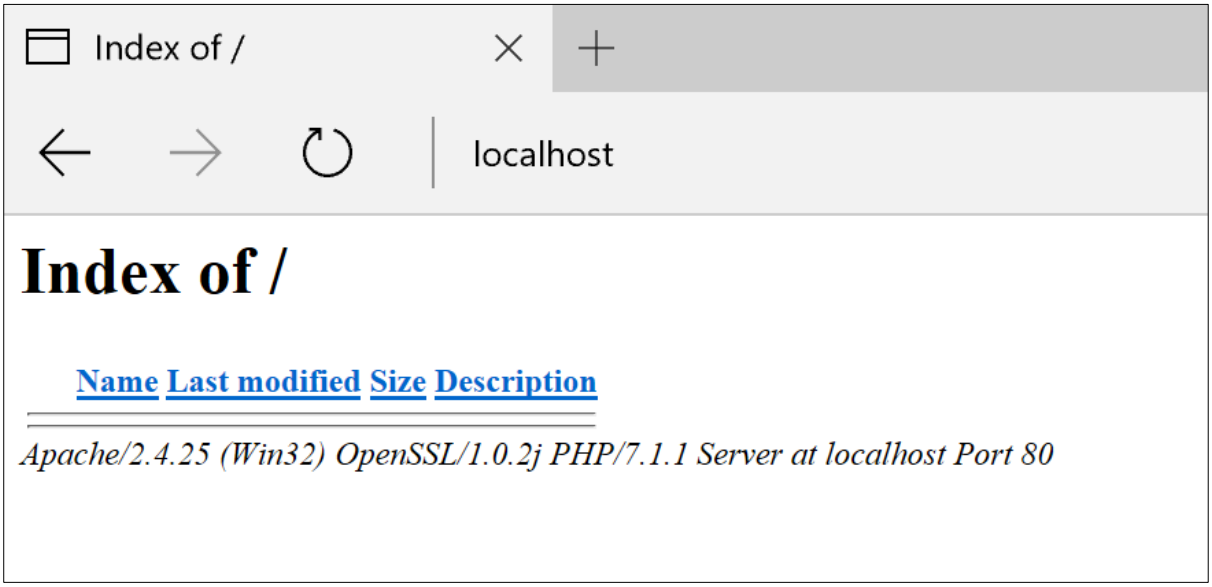
Highlight and right click on the navigation elements and delete everything in the folder



Once you have emptied the folder it should look like



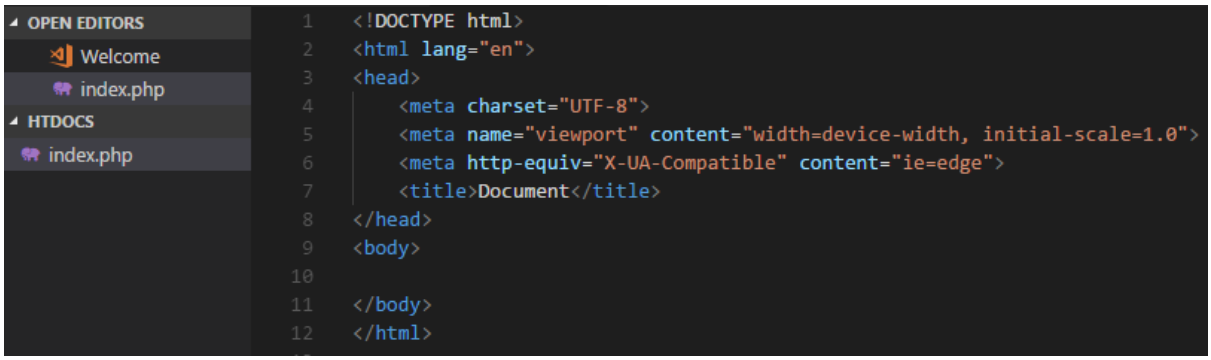
Now, we check to ensure that the system is clean, go to your browser and navigate to http://localhost/ You should see the following



Now we have a clean server to start creating with.

Creating your php Website

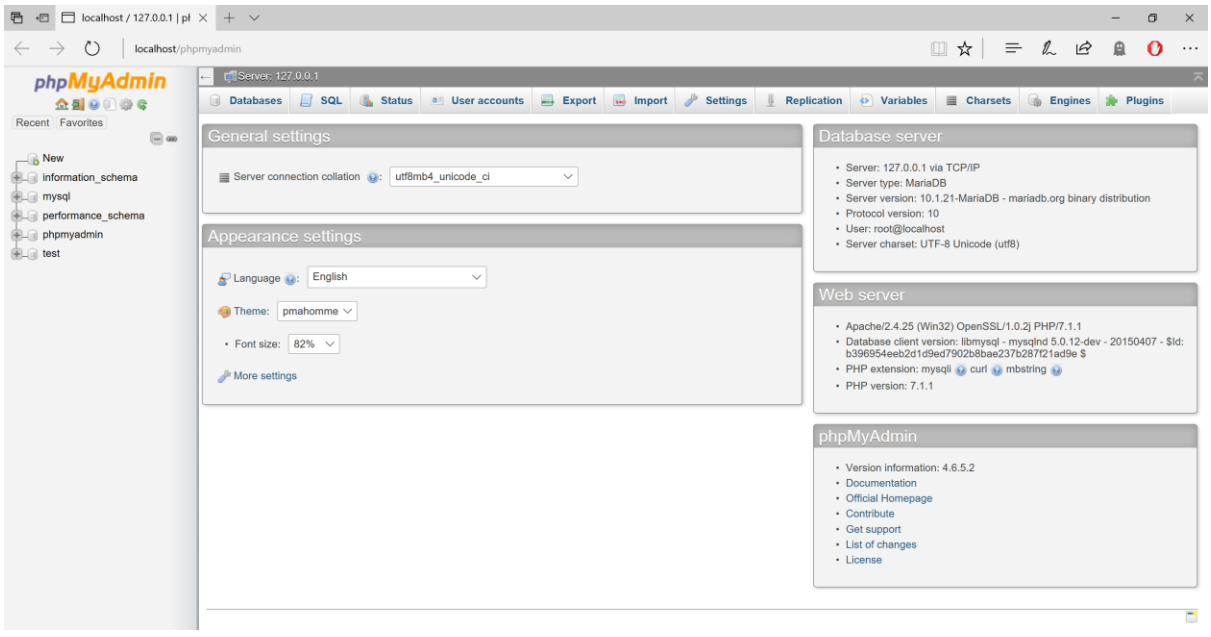
From here, right click and create a new file called index.php and put in the following content:



In this lesson we will be communicating with a database through our php files. There are 4 aspects to database communication; Create, Read, Update and Delete otherwise referred to as CRUD. But before we get into the code, let's look at the structure of a database and how we can access and manipulate the database service.

There are a couple of ways in which we can do this, one is via command prompt and the other is through a web interface. In our case, to make life as easy as possible, we will be using the web interface; this interface is called phpmyadmin.

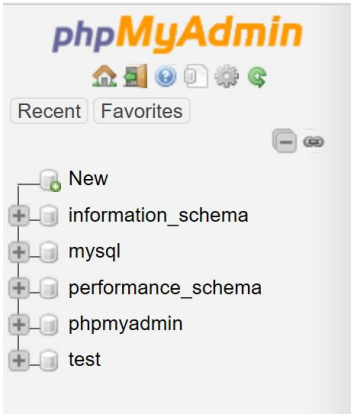
To get access to the web interface, open a browser and go to the address localhost/phpmyadmin and you should see the following:



This is the normal screen in which you can manipulate all of the databases you require for most web projects.

We will run through the interface, to increase your ability to navigate and understand the way it works and how it will work for your code.

- The following pane, contains a list of all currently existing databases within the MySql system.

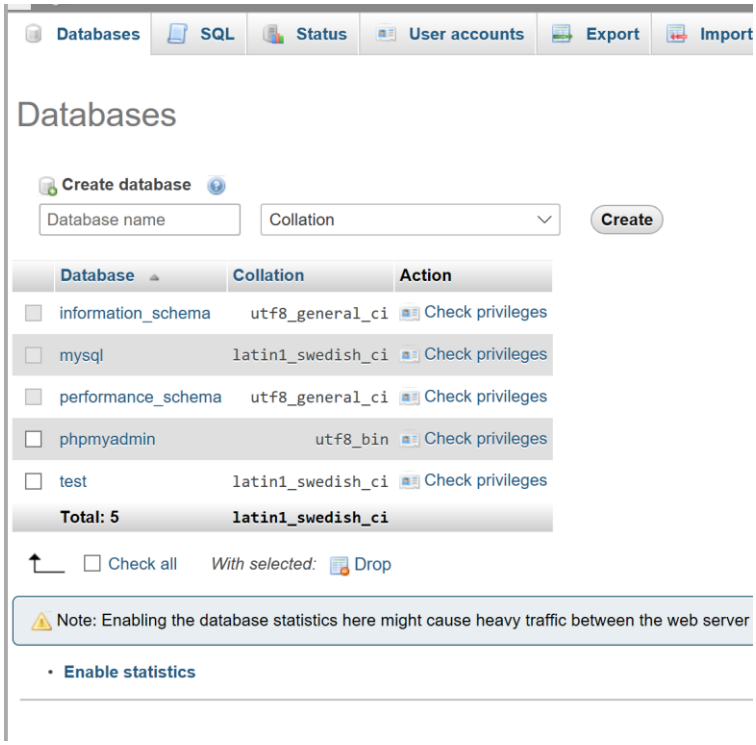


-
- Each of the databases can be viewed in more by expanding via the plus sign. Also note, that there is a home (house) icon under the title, this returns you to this page.
- This part of the interface:

-
- Allows you to access specific elements of MySQL. There are only a few of them that we will require for this lesson.
    - Database – This is where you can create and delete databases
    - User Accounts – This is where you will create/delete user accounts that allow access to the database. This is a security element that is very important, if your server has multiple databases and each database is access via one user account, if there is a security breach then only one database can be compromised. So always create a specific user for a database.
    - Export – This is where you can make a backup of the database. When saving your work/submitting your assignment. You will need to export your database to a .sql file.
    - Import – This is where you bring a .sql file back into the MySQL system. So, if you have worked on a database, exported the work and saved it to USB, if the computer removes all the server details (All university computers are cleaned, so your files will be removed each week) it means that when you return to the computer, you will be able to import all the database information (Tables and data). Thereby removing the potential requirement to recreate and re-enter.

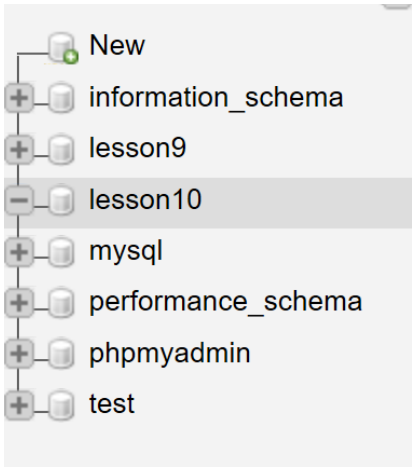
Now that we have a basic overview of the interface, let's create the database structure, account, and tables within the database

Click on the database link, you should see the following:



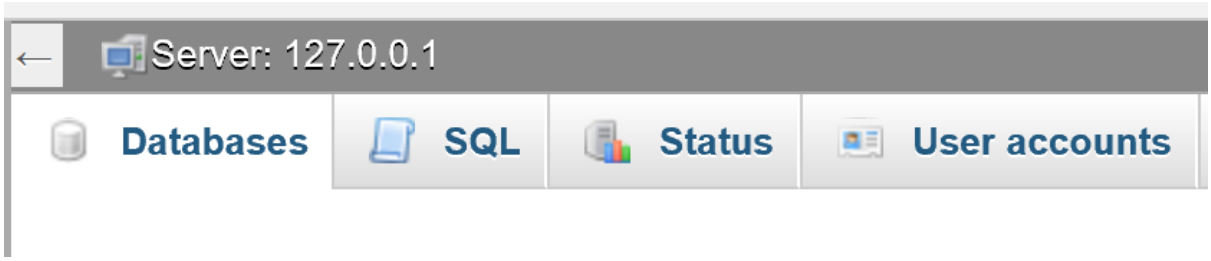
From here enter in the database name and click create. NB When it comes to your assignments, try using your student number as your database name. In this case, we will name the database lesson10.

You should have a structure looking like this once done. Once the database was created, you might be automatically taken into the database, click on the home icon and then database link to see the following.

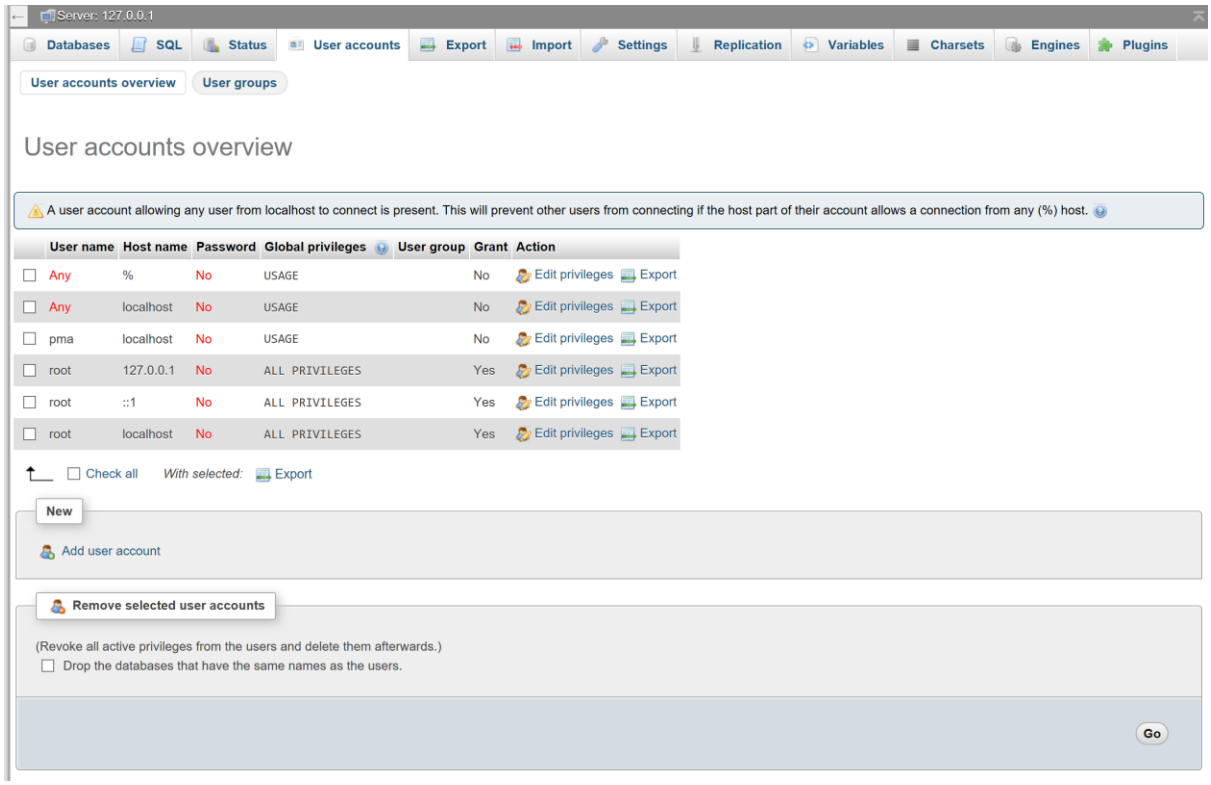


As you can see, the database is listed in the left navigation bar as well as in the list of databases.

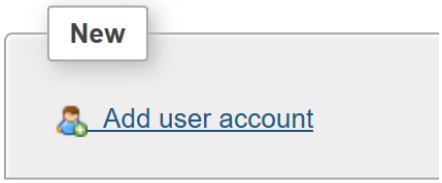
Next click on the user accounts link.



You should see the following:



This is a list of all current user accounts within the MySQL system. To ensure that we work securely with our database, we need to create a new account. As such, click on Add User Account:

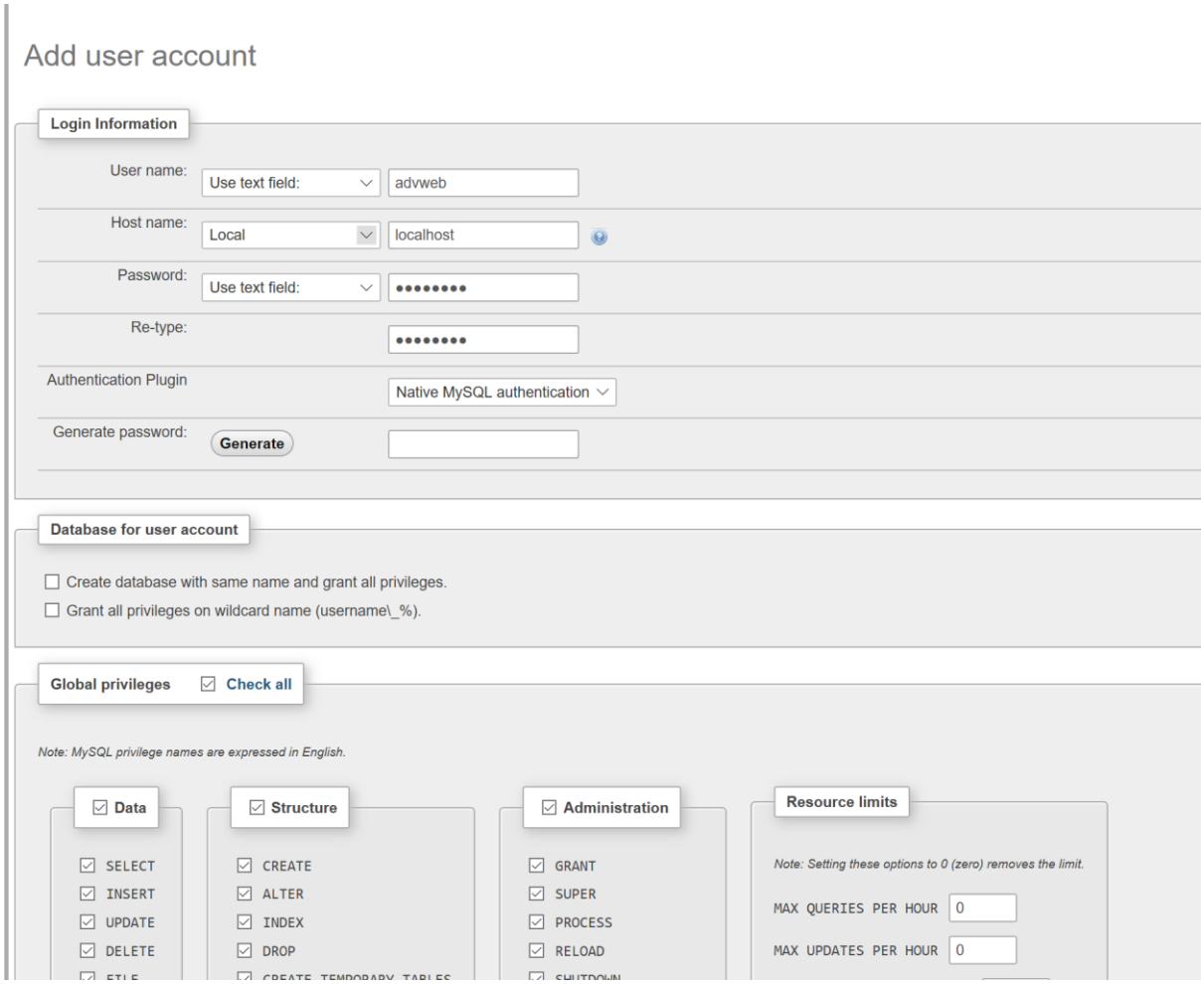


Here you will be prompted for details to enter. User the following:

- Username: advweb
- Host: localhost
- Password: password

Also, ensure that you click the check all part. Once the details a re filled out scroll down and click go. This will create the user and apply it's permissions.

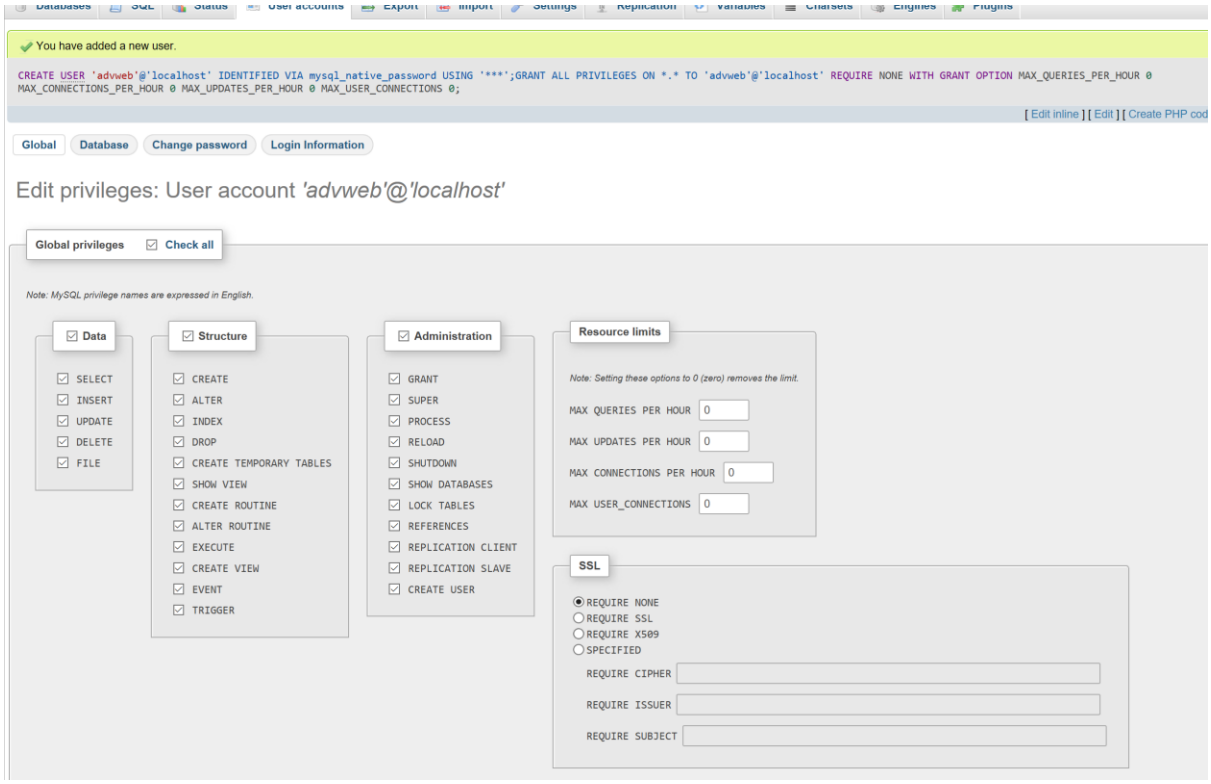
It should look like the following:



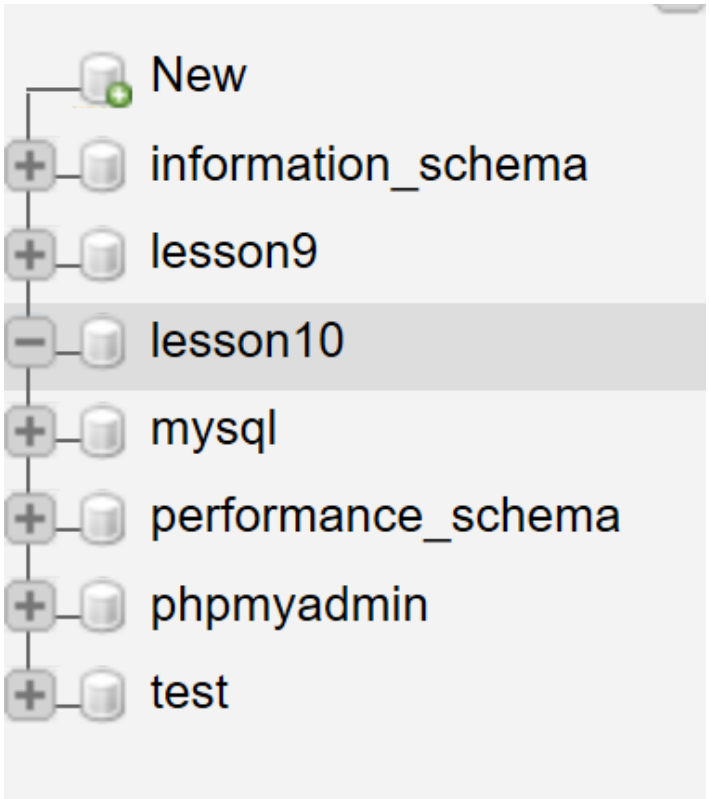
Once you have clicked go, you should be redirected to a page that indicates a successful user addition. If there is an error message that appears regards c:\xampp\mysql\lib\plugins and the message states dir not found. You will have to create the folder structure. Open file explorer and navigate to c:\xampp. Once there, go inside the mysql folder and create a new folder called lib. Once this has been done, go inside the newly created lib folder and create the folder called plugins.

After the creation of these folders, go back to the web browser and re-create the user. It should work successfully this time.

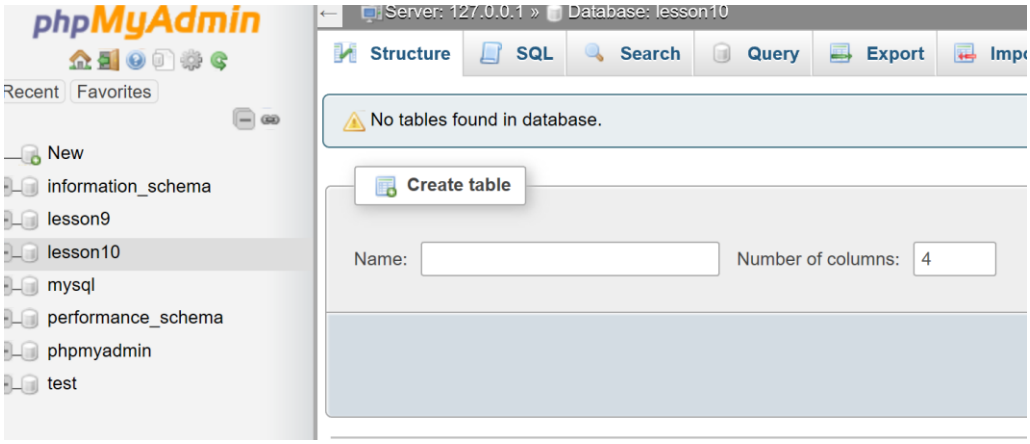
You should be directed to the following page:



Now that the user and database have been created, we need to add some tables to the database, to do this, we need to access the database. On the left hand navigation panel, click on the database called lesson9.



This will bring you to the following screen:

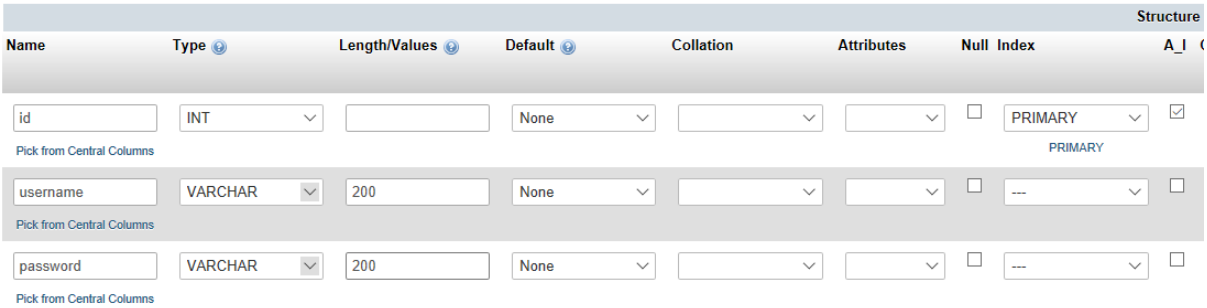


Now that we are inside the database we need to create a table to hold information.

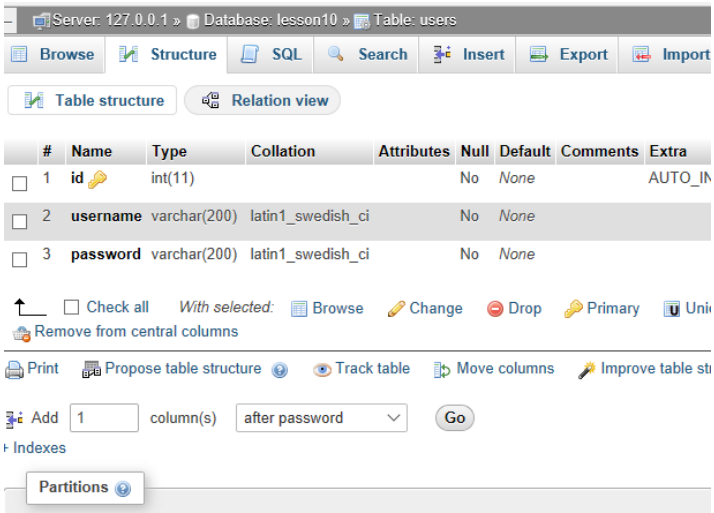
We will create a users table that contains an id, username and password.

Field Name	Field Attributes
Id	Int, primary key, Auto Incrementing
Username	Varchar 200
password	Varchar 200

Fill out the table like so



Then click save you should see the following:



Now we will create the pages.

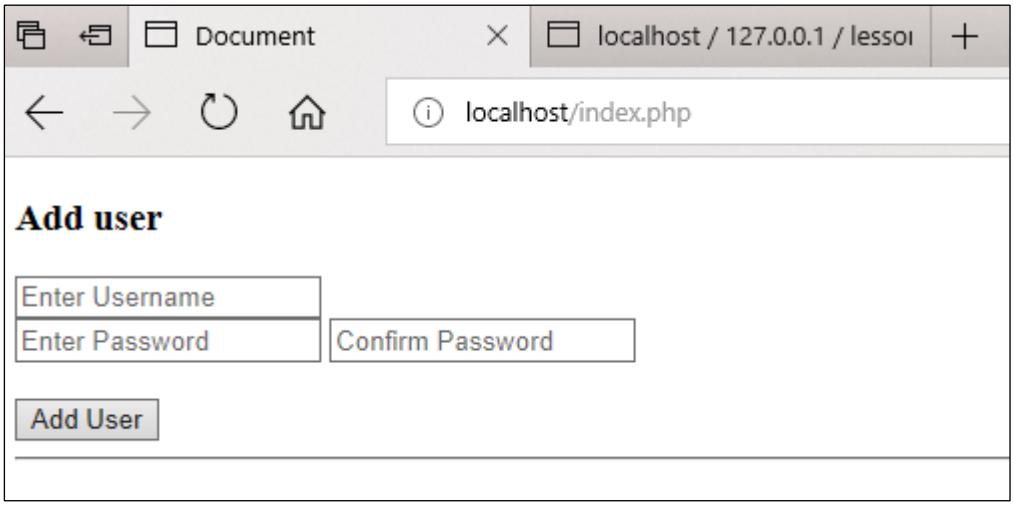
PHP pages

First page is index.php, we will put a simple form on this page to collect information from the end user.

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h3>Add user</h3>
    <form method="post" action="pages/resolveRegister.php">
    <input type="text" name="username" placeholder="Enter Username"><br>
    <input type="password" name="pwd" placeholder="Enter Password">
    <input type="password" name="pwd2" placeholder="Confirm Password"><br><br>
    <input type="submit" value="Add User"><br>
    <hr>
    </form>
</body>
</html>
```

This will produce the following output (Remember all pages are saved in the htdocs folder, index.php should be your first page with subsequent pages being stored in subfolders. Also, to view, the web address is localhost.)



Next we need to link the page to the database, this will be done by creating a functions.php page that will store the php functions that are going to be created. In this manner, all functions can be easily located. The first function we need to create is a connection function.

Functions.php

index.php functions.php

```php
<?php

function dbLink()
{
    $db_user = "advweb";
    $db_pass = "password";
    $db_host = "localhost";
    $db = "lesson10";
    try{
        $db = new PDO("mysql:host=$db_host; dbname=$db", $db_user, $db_pass);
    }catch(Exception $e){
        echo 'Unable to access database';
        exit;
    }
    error_reporting(0);
    return $db;
}

?>
```

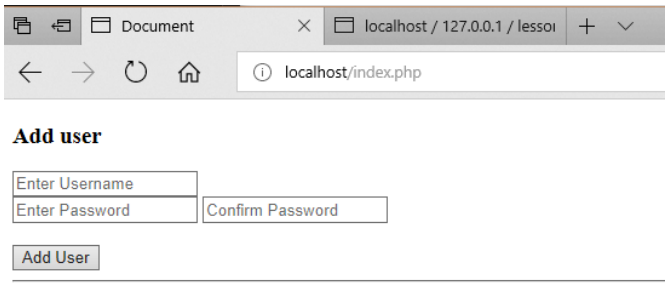
Next, we need to link the function to the index page.

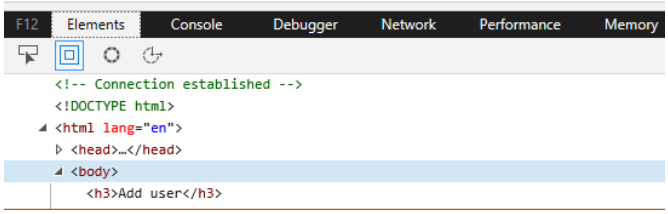
Index.php

index.php

```php
<?php
    session_start();
    include_once('functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection Established -->';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h3>Add user</h3>
    <form method="post" action="pages/resolveRegister.php">
    <input type="text" name="username" placeholder="Enter Username"><br>
    <input type="password" name="pwd" placeholder="Enter Password">
    <input type="password" name="pwd2" placeholder="Confirm Password"><br><br>
    <input type="submit" value="Add User"><br>
    <hr>
    </form>
</body>
</html>
```

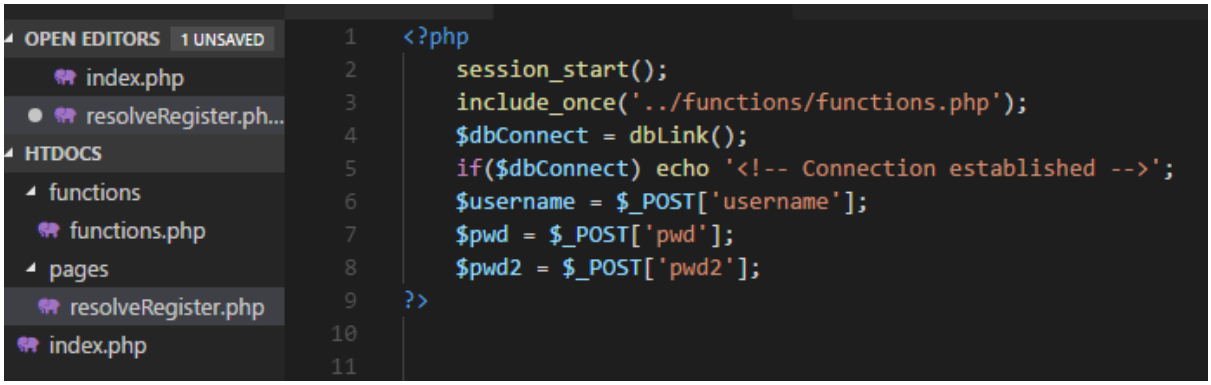
Once done, load the page in localhost and check the page via the debugger, we should be able to see the html comment.





Now that we have established connection, we need to insert data into the database, this is done by collecting all of the information gathered in the form and then sending it to the resolveRegister page.

Create a new folder called pages. Inside that folder, create a new php file called resolveRegister.php with the following code:



This piece of code is designed to capture the “posted” data and place it into variables. To check if we got the content, we need to create a showMem() function and implement that.

Functions.php

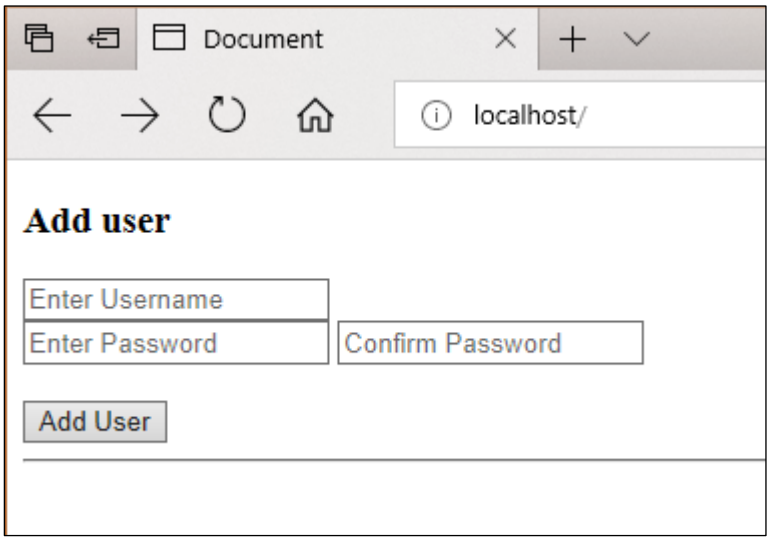
```php
function showMem(){
    echo '<h3>Post Memory</h3><pre>';
    print_r($_POST);
    echo '</pre>';
    echo '<h3>Get Memory</h3><pre>';
    print_r($_GET);
    echo '</pre>';
    echo '<h3>Session Memory</h3><pre>';
    print_r($_SESSION);
    echo '</pre>';
}
```

Implement it on the resolveRegister page

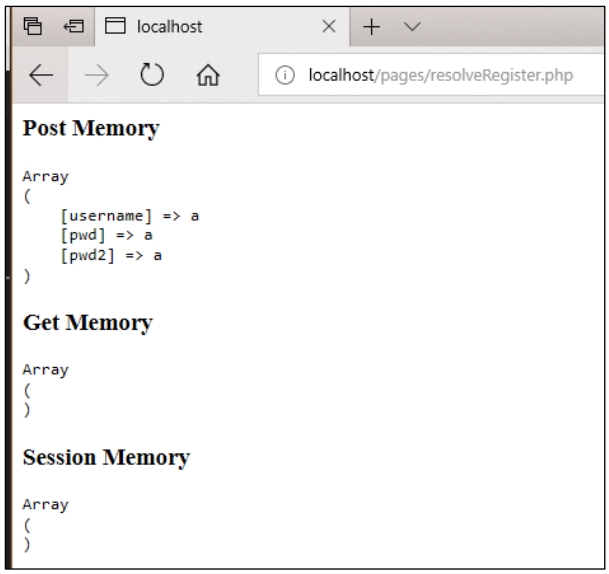
```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    showMem();

?>
```

And now test it.

Index page



Resolve register page



As you can see, the data has transferred from index to resolveRegister.

Now we need to insert the details into the database. This is done by creating an insert function and implementing it on the resolve register page.

Functions.php

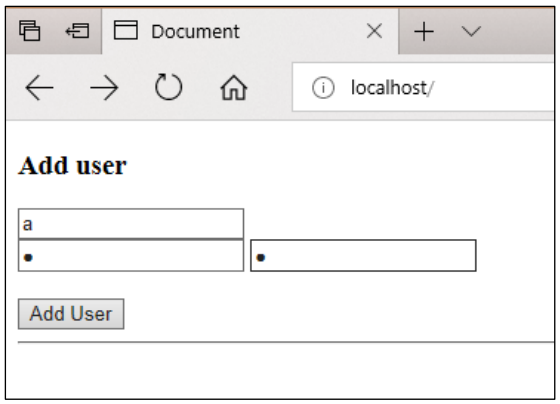
```php
function insertUser($dbConnect, $item1, $item2)
{
    $q = "INSERT into users (id,username,password) VALUES(NULL,:un,:pw);";
    $query = $dbConnect->prepare($q);
    $query->bindParam(":un",$item1);
    $query->bindParam(":pw",$item2);
    $result = $query->execute();
    return $result;
}
```

resolveRegister.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    showMem();
    $res = insertUser($dbConnect, $username, $pwd);
?>
```

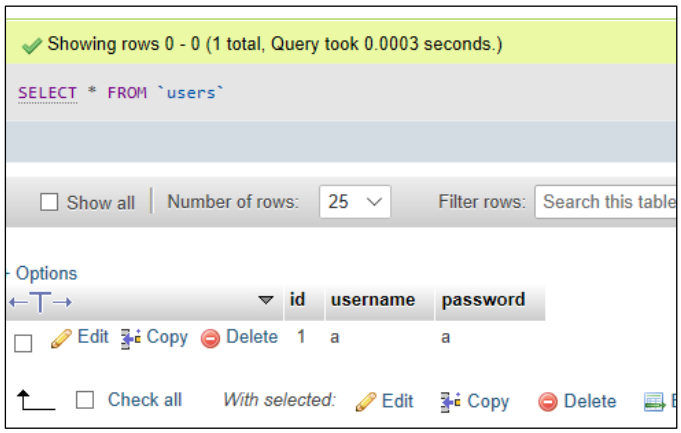
Output

Index



Resolveregister

Phpmyadmin



Now that we know data is going into the database, we will tidy up the code a bit. We will hide the showMem function in resolveRegister and create a redirect to make the page automatically return to the index page. We will also store the username and password in session memory, in case we need to look at it sometime.

resolveRegister.php

index.php × resolveRegister.php × functions.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    //showMem();
    $res = insertUser($dbConnect, $username, $pwd);
    if($res){
        $_SESSION['username'] = $username;
        $_SESSION['password'] = $pwd;
        redirectPage('../index.php',$statusCode = 303);
    }
?>
```

functions.php

```php
function redirectPage($url, $statusCode = 303)
{
   header('Location: ' . $url, true, $statusCode);
   die();
}
```

Now when testing, the index page will be the primary one seen, but the database will grow whenever data is entered.

Phpmyadmin after adding a few more users.

Options

	id	username	password
Edit Copy Delete	1	a	a
Edit Copy Delete	2	b	b
Edit Copy Delete	3	c	c

Even though this is working well, there are some issues with it. What happens if the user doesn't enter anything? What happens if the passwords don't match? These are solved by using if statements on the code.

First, we will check if the fields are empty and if they are, we will return the end user to the index page but, supply an error message.

Secondly, we will validate the passwords. Same result as previous, if they don't match, we will inform the end user.

To make it a little quicker, the messages will be added to the index code in one go.

Resolveregister.php

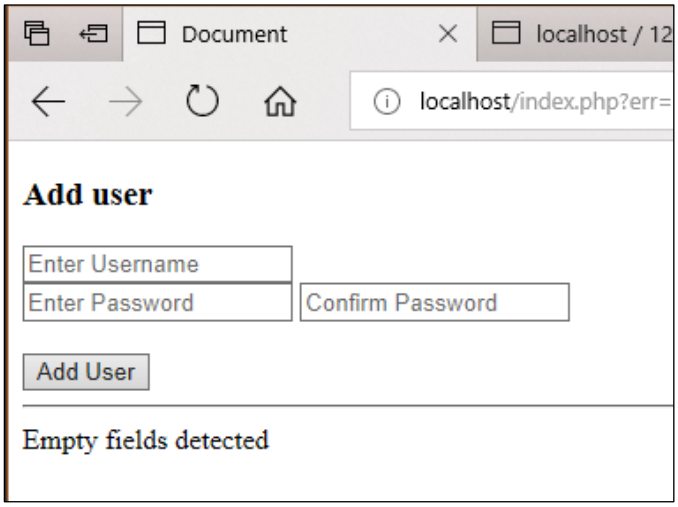
```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    //showMem();
    if($username == NULL || $pwd == NULL || $pwd2 == NULL){
        redirectPage('../index.php?err=1',$statusCode = 303);
    }
    $res = insertUser($dbConnect, $username, $pwd);
    if($res){
        $_SESSION['username'] = $username;
        $_SESSION['password'] = $pwd;
        redirectPage('../index.php',$statusCode = 303);
    }
?>
```

Index.php

index.php | resolveRegister.php | functions.php

```php
<?php
    session_start();
    include_once('functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection Established -->';
    error_reporting(0);
    $errCode = $_GET['err'];
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h3>Add user</h3>
    <form method="post" action="pages/resolveRegister.php">
    <input type="text" name="username" placeholder="Enter Username"><br>
    <input type="password" name="pwd" placeholder="Enter Password">
    <input type="password" name="pwd2" placeholder="Confirm Password"><br><br>
    <input type="submit" value="Add User"><br>
    <hr>
    </form>
    <?php
        switch($errCode){
            case 1: echo 'Empty fields detected';
                break;
            case 2: echo 'Passwords do not match';
                break;
            case 3: echo 'Account added!';
                break;
        }
    ?>
</body>
</html>
```

Output

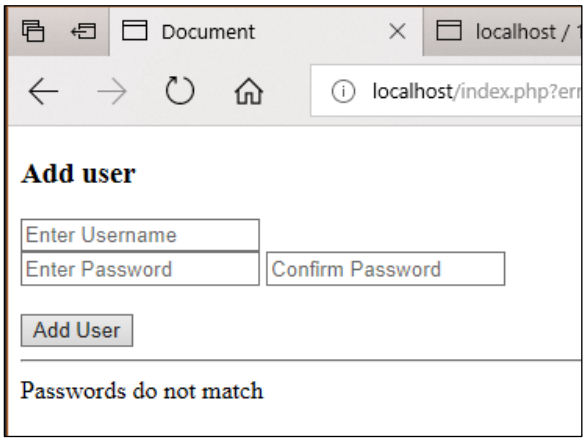


Adding notification to a successful add here.

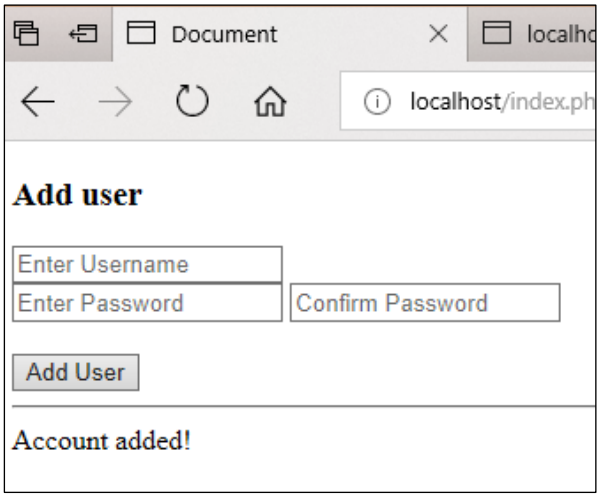
Resolveregister.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    //showMem();
    if($username == NULL || $pwd == NULL || $pwd2 == NULL){
        redirectPage('../index.php?err=1',$statusCode = 303);
    }
    if($pwd != $pwd2){
        redirectPage('../index.php?err=2',$statusCode = 303);
    }
    $res = insertUser($dbConnect, $username, $pwd);
    if($res){
        $_SESSION['username'] = $username;
        $_SESSION['password'] = $pwd;
        redirectPage('../index.php?err=3',$statusCode = 303);
    }
?>
```

Output error with passwords



Output Successful adding of account



Now that we have added and ran some validation, we will create a function that will list the accounts created. This function will run on the index page.

Functions.php

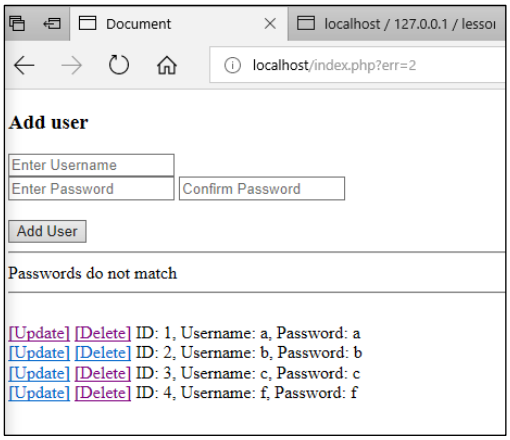
```php
function listAccounts($dbConnect){
    $sql = 'SELECT * FROM users';
    foreach ($dbConnect->query($sql) as $row)
    {
        echo '<br><a href="pages/updateForm.php?id='.$row['id'].'">[Update]</a>
        <a href="pages/delete.php?id='.$row['id'].'">[Delete]</a> ID: '
       .$row['id'].',  Username: '.$row['username'].',  Password: '.$row['password'];
    }
}
```

Index.php

index.php × resolveRegister.php functions.php

```php
<?php
    session_start();
    include_once('functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection Established -->';
    error_reporting(0);
    $errCode = $_GET['err'];
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<h3>Add user</h3>
    <form method="post" action="pages/resolveRegister.php">
    <input type="text" name="username" placeholder="Enter Username"><br>
    <input type="password" name="pwd" placeholder="Enter Password">
    <input type="password" name="pwd2" placeholder="Confirm Password"><br><br>
    <input type="submit" value="Add User"><br>
    <hr>
    </form>
    <?php
        switch($errCode){
            case 1: echo 'Empty fields detected';
                break;
            case 2: echo 'Passwords do not match';
                break;
            case 3: echo 'Account added!';
                break;
        }
    ?>
    <hr>
    <?php
        listAccounts($dbConnect);
    ?>
</body>
</html>
```

Output



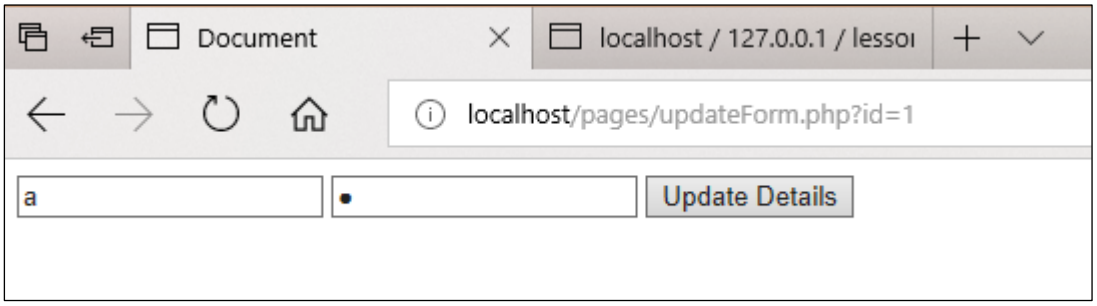
As you can see, the update and delete links are in place now, this is to save typing them up later. As you can see by examining the listAccounts functions, each link to update and delete have the items id number, this number will be used to locate the exact item in the database.

Now we modify the contents. Create a page called updateForm like this:

index.php functions.php updateForm.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<?php
    $id = $_GET['id'];
    $sql = 'SELECT * FROM users';
    foreach ($dbConnect->query($sql) as $row)
    {
        if($id == $row['id']){
            $username = $row['username'];
            $password = $row['password'];
        }
    }

?>
<form action="resolveUpdate.php" method="post">
        <input type="text" name="username" value="<?php echo $username; ?>">
        <input type="password" name="pwd" value="<?php echo $password; ?>">
        <input type="hidden" name="id" value="<?php echo $id; ?>">
        <input type="submit" value="Update Details">
    </form>
</body>
</html>
```

Like the index page, this form is designed to capture information from the end user, the primary difference is that we are pre-filling the fields. If you save and run it, the output will be:



The form is designed to go to the resolveUpdate page, this needs to be created, like this:

resolveUpdate.php

index.php | functions.php | updateForm.php | resolveUpdate.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $username = $_POST['username'];
    $pwd = $_POST['pwd'];
    $id = $_POST['id'];
    updateDetails($dbConnect,$id,$username,$pwd);
    redirectPage('../index.php',$statusCode = 303);
?>

```

It requires a function called updateDetails, this is stored in the functions.php file.

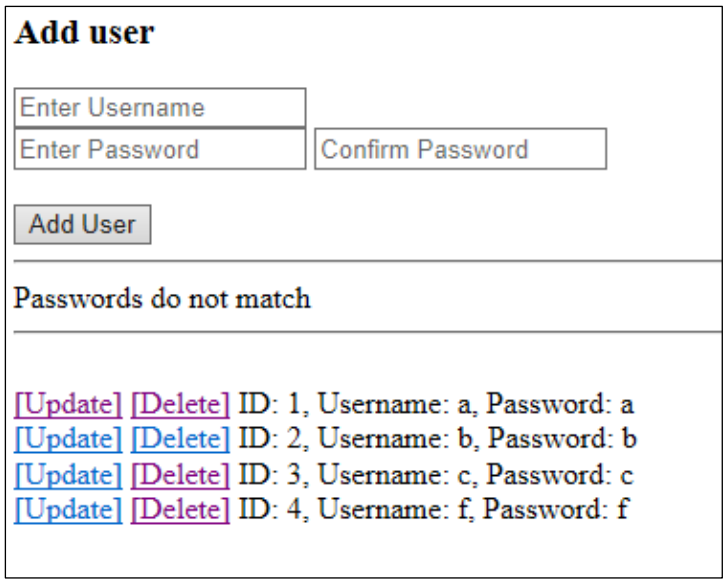
Functions.php

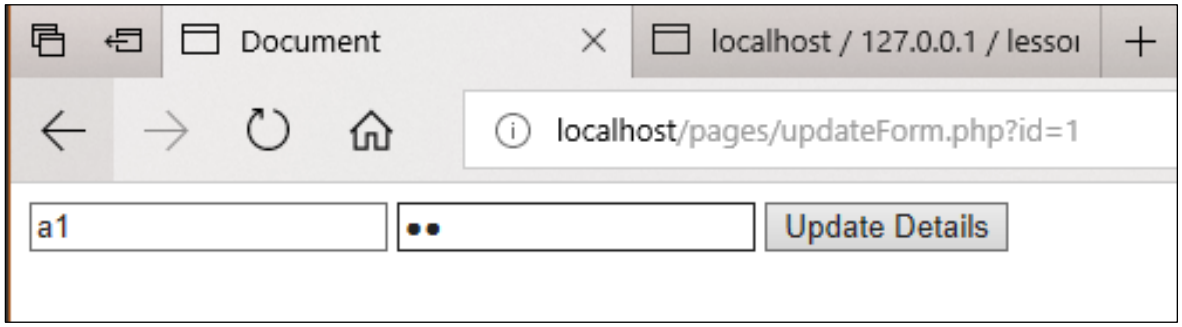
```php
function updateDetails($dbConnect,$id,$username,$pwd){
    $q = $dbConnect -> prepare("UPDATE users set username = :un WHERE id = :postid");
    $q->bindValue(':un',$username);
    $q->bindValue(':postid',$id);
    $q->execute();
    $q = $dbConnect -> prepare("UPDATE users set password = :pw WHERE id = :postid");
    $q->bindValue(':pw',$pwd);
    $q->bindValue(':postid',$id);
    $q->execute();
}
```

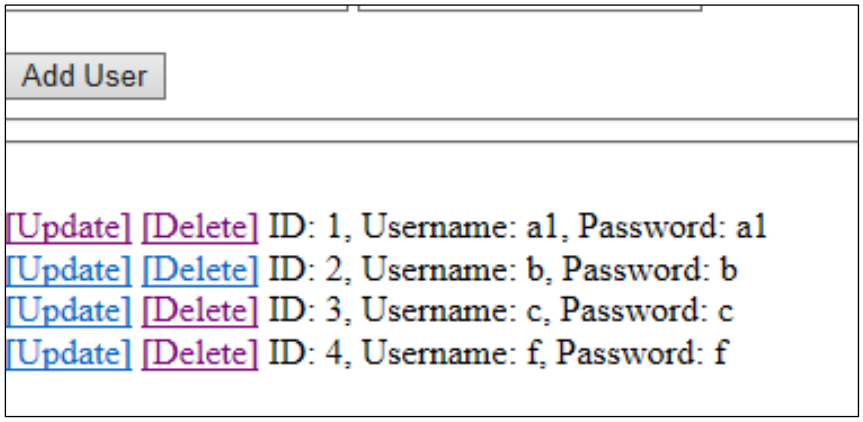
This function is designed to update the username and the password one after the other, in this case all of the updates occur in one go.

Once all this is done, save the work and test it in localhost.

Output







As you can see, the update functionality modifies the content of the database.

Finally, we will look at delete. In this instance, delete will be straightforward. We are using the id number that is listed in the link from the listAccounts function. We will send this id off to the delete page, where it will remove the line form the database.

Create the following file, delete.php

index.php | functions.php | delete.php | updateForm.php

```php
<?php
    session_start();
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if($dbConnect) echo '<!-- Connection established -->';
    $id = $_GET['id'];
    deleteDetails($dbConnect,$id);
    redirectPage('../index.php',$statusCode = 303);
?>
```

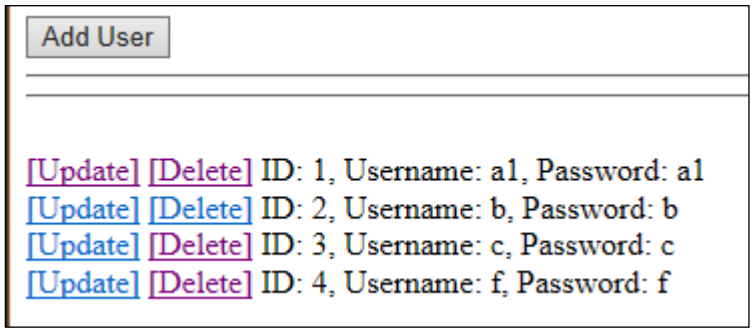
Now, we need to create the following function in the functions file.

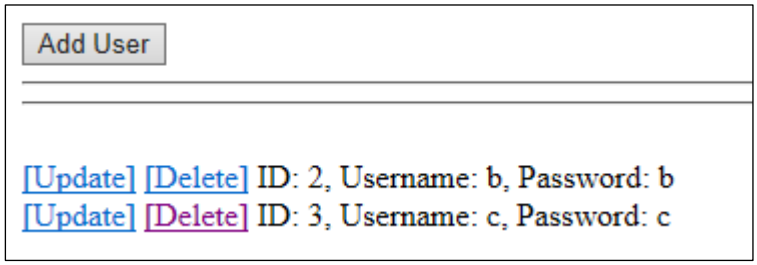
Functions.php

```php
function deleteDetails($dbConnect,$id)
{
    $sql = "DELETE FROM users WHERE id =  :id";
    $stmt = $dbConnect->prepare($sql);
    $stmt->bindParam(':id',$id);
    $stmt->execute();
}
```

Notice how it links to the users table directly, to make the function more utilitarian, we could pass in a table variable instead.

Output





In this case, I tested delete on the a1 and f accounts.

This concludes the tutorial. It has shown you Create/Read/Update and Delete functionality, each function can be manipulated to work with your own assessments. Remember to keep a backup of all files so you have access to a library of code.

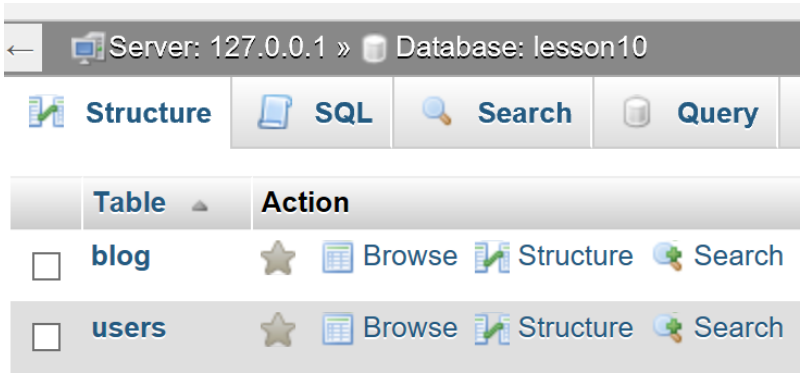
Backup

This concludes tutorial.

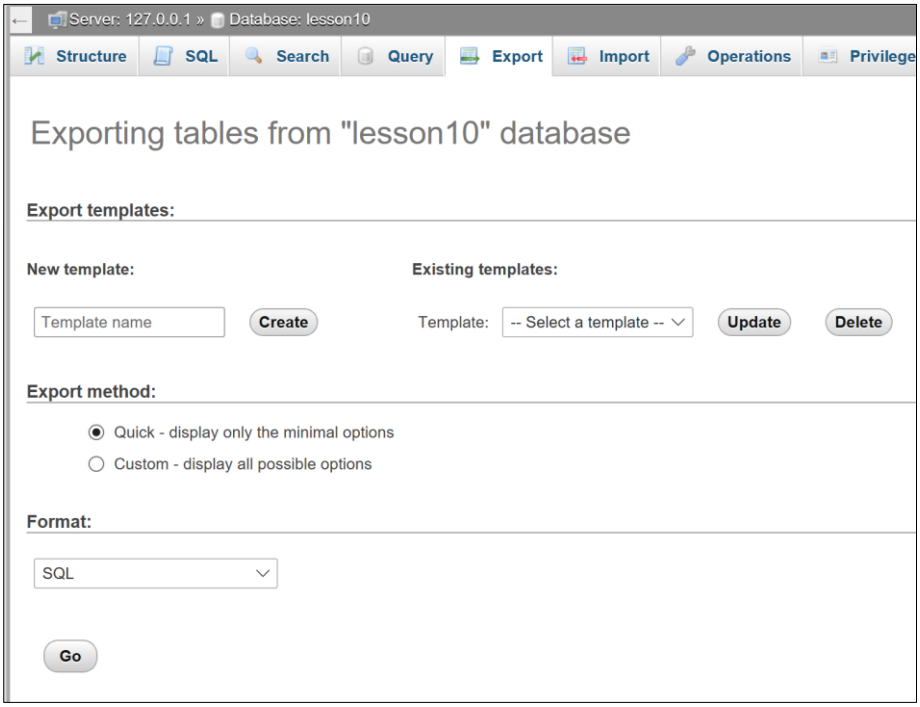
As we are now using servers, and server information, there are two areas to backing up your information.

The first area is the htdocs folder. This is all of the php/html/css/javascript files. These are all needed to view the website.

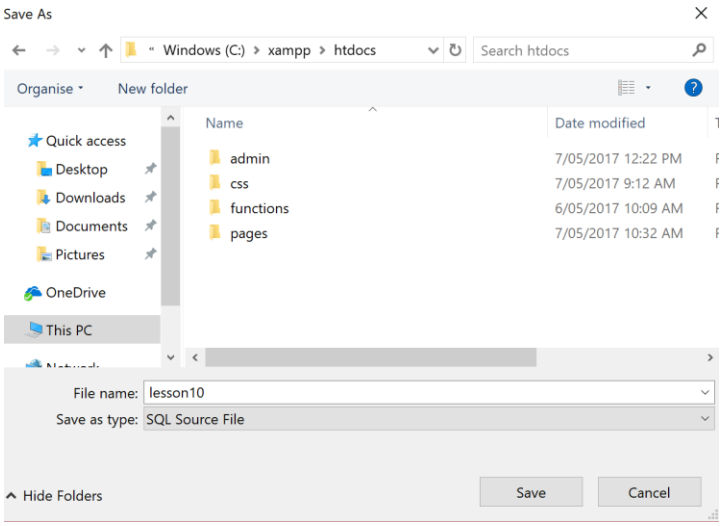
Secondly, the database itself. To back up the database, go to phpmyadmin and navigate to the root area of the database



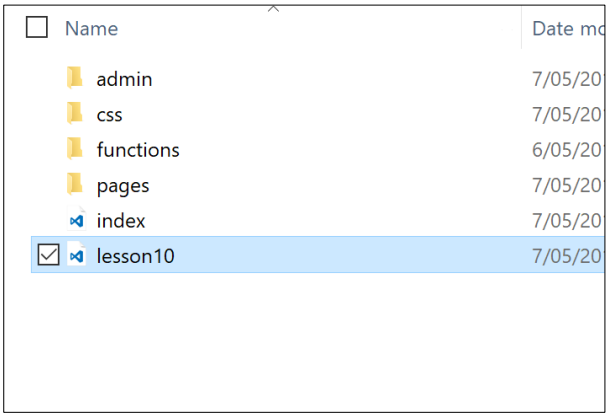
From here, click on export:



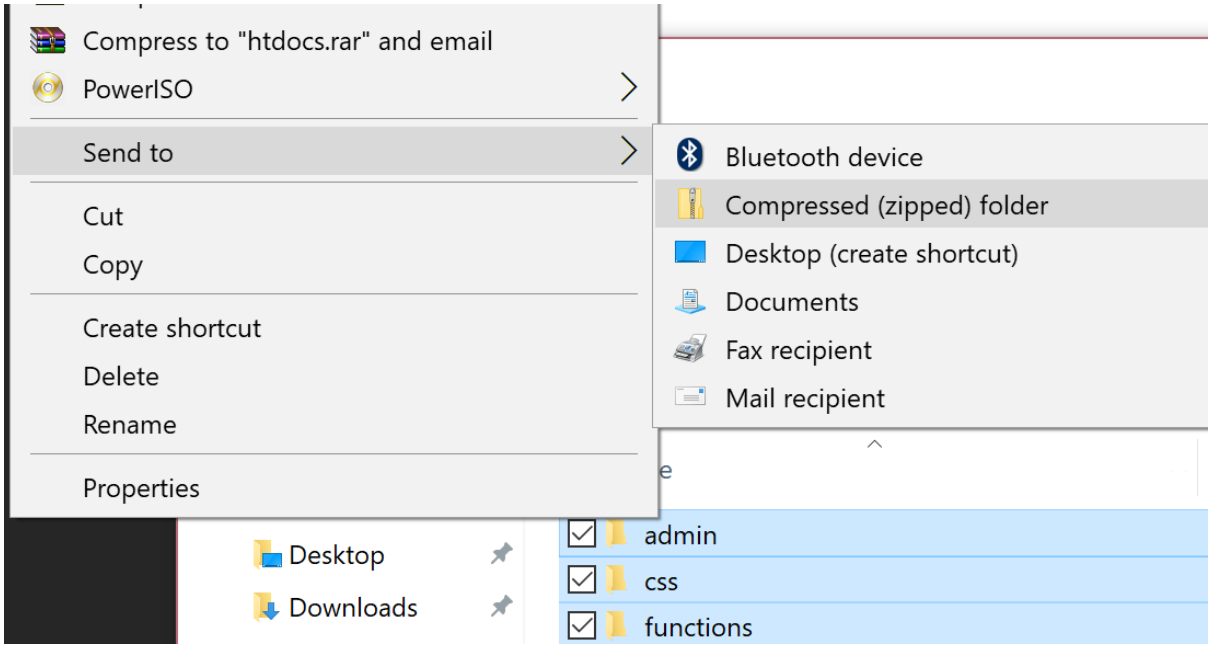
Click on go and save this .sql file into the htdocs folder



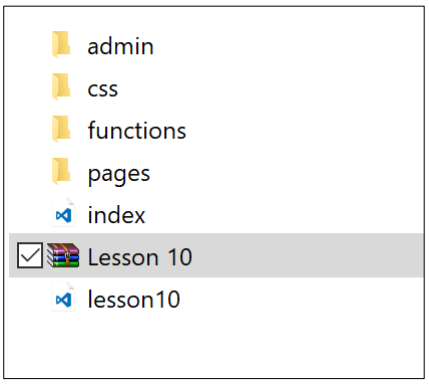
When viewing the htdocs folder you should see



Highlight everything and right click, then select send to compressed file



It will save a zip file, name this file appropriately.



If the icon is different, this is because on the machine this was created on, I use winrar as my compression tool.

Save your work (c:\xampp\htdocs) to USB, student drive, Onedrive, Dropbox or zip and email the folder to yourself.