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

Lab 2 – PHP and Database

Aim

The aim of this lab is to learn how to handle database through different tools.

Tips:

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

2. The M-Dev-Store online videos are good references while our labs have different focus. If you want to be an expert, you are recommended take both labs and on-line videos.

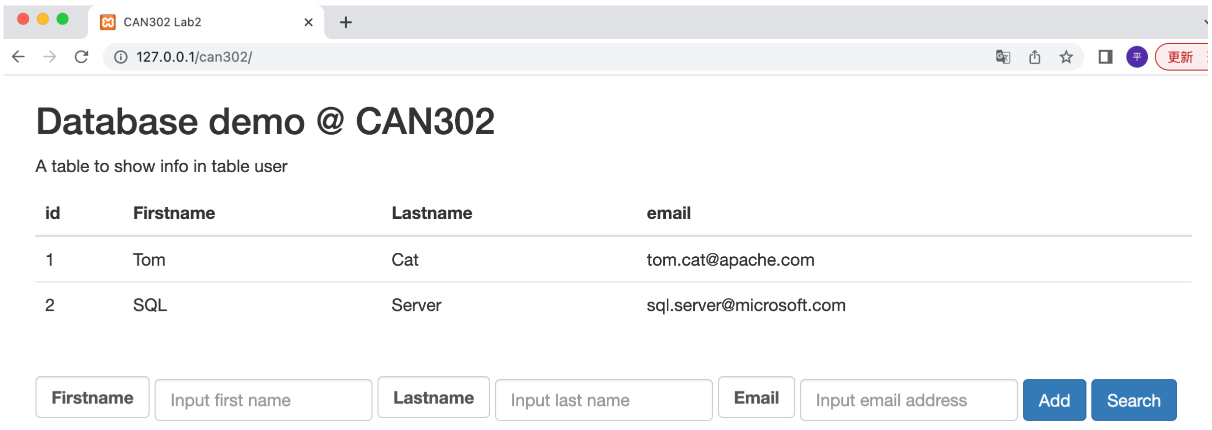
3. The forums @ LMO are available for questions and discussions.

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!

Database and MariaDB:

1. A collection of related pieces of data, whose purpose is to solve the data management needs of an institution is called a Database. Database Management Systems (DBMS), on the other hand, are very complex software that save the data on the secondary storage devices and which are used to manipulate databases. SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database. Though there are many different DBMS like Oracle, MariaDB, MySQL, SQL Server …, the SQL is same for all of them. However, the tool used to operate different DBMS is different from each other. “phpMyAdmin” is a web tool that can operate MariaDB. We can use it to create database, table and **C**reate (insert), **R**ead (query), **U**pdate, **D**elete data or CRUD.

   Let’s make the 1st demo as following:



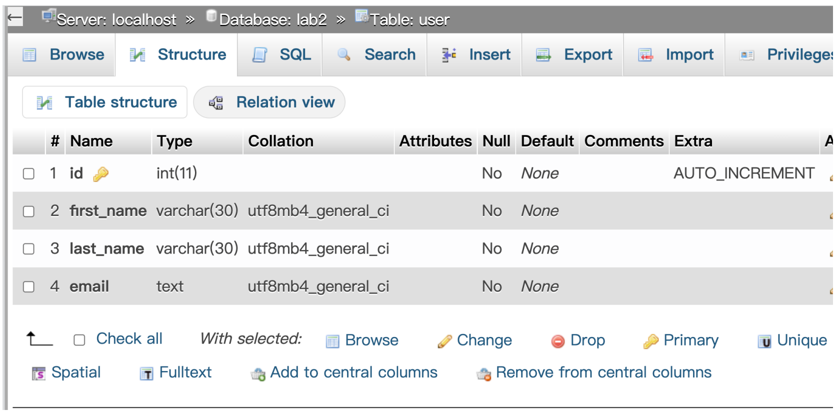
Database demo @ CAN302

A table to show info in table user

id	Firstname	Lastname	email
1	Tom	Cat	tom.cat@apache.com
2	SQL	Server	sql.server@microsoft.com

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.

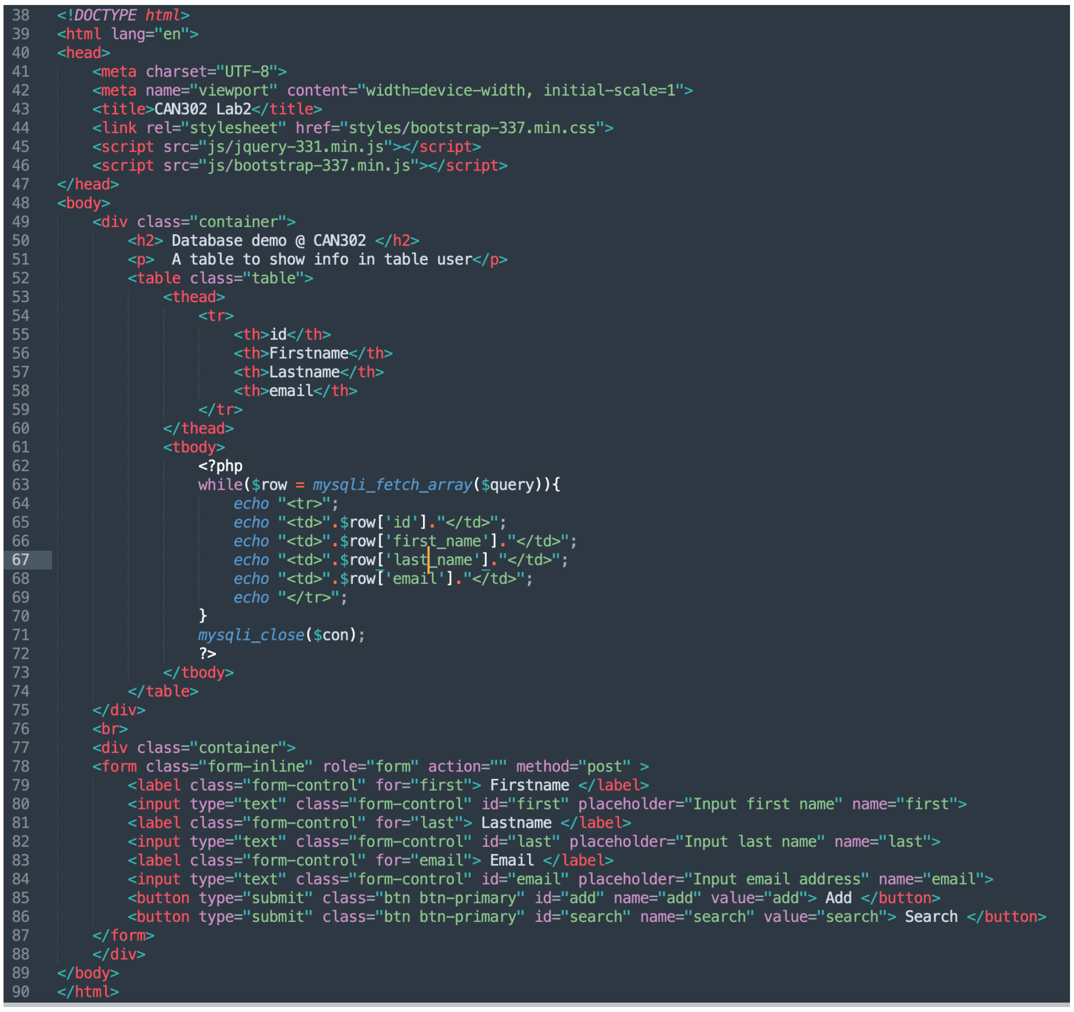
First of all, we need to create a database and then a table to hold such data. We can use "phpMyAdmin" for that purpose. The data structure is as:



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	id	int(11)			No	*None*		AUTO_INCREMENT
☐	2	first_name	varchar(30)	utf8mb4_general_ci		No	*None*		
☐	3	last_name	varchar(30)	utf8mb4_general_ci		No	*None*		
☐	4	email	text	utf8mb4_general_ci		No	*None*		

In above, a database named "lab2" was created and a table named "user" was created with four columns: id, first_name, last_name and email.

To show the content on the webpage as the demo, the html part code is as:

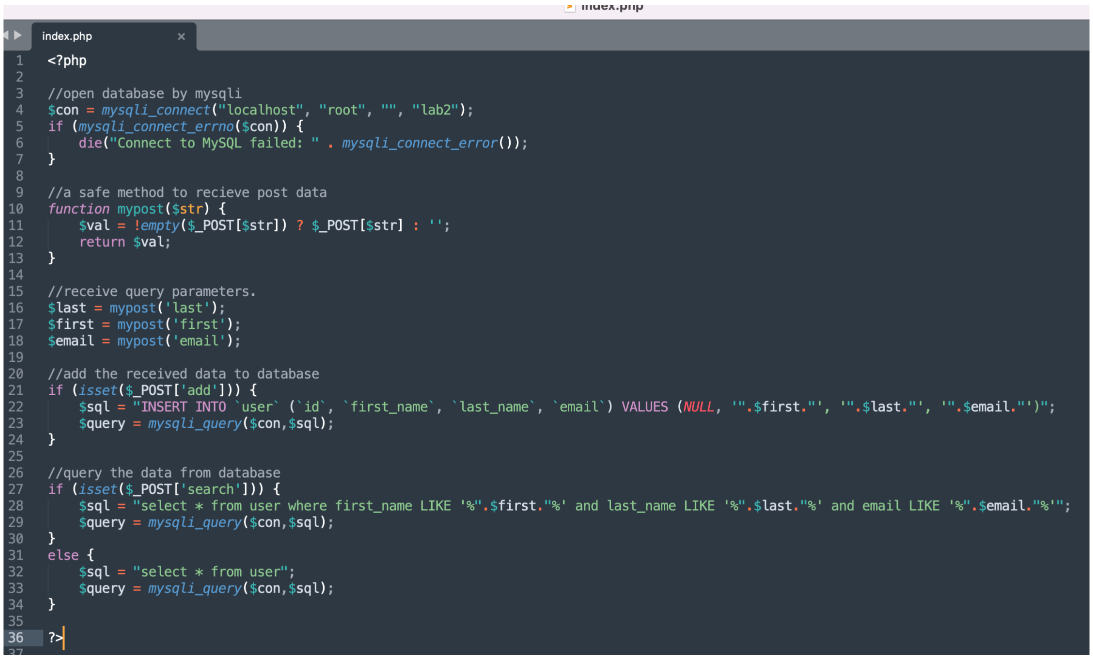


```
38  <!DOCTYPE html>
39  <html lang="en">
40  <head>
41      <meta charset="UTF-8">
42      <meta name="viewport" content="width=device-width, initial-scale=1">
43      <title>CAN302 Lab2</title>
44      <link rel="stylesheet" href="styles/bootstrap-337.min.css">
45      <script src="js/jquery-331.min.js"></script>
46      <script src="js/bootstrap-337.min.js"></script>
47  </head>
48  <body>
49      <div class="container">
50          <h2> Database demo @ CAN302 </h2>
51          <p>  A table to show info in table user</p>
52          <table class="table">
53              <thead>
54                  <tr>
55                      <th>id</th>
56                      <th>Firstname</th>
57                      <th>Lastname</th>
58                      <th>email</th>
59                  </tr>
60              </thead>
61              <tbody>
62                  <?php
63                  while($row = mysqli_fetch_array($query)){
64                      echo "<tr>";
65                      echo "<td>".$row['id']."</td>";
66                      echo "<td>".$row['first_name']."</td>";
67                      echo "<td>".$row['last_name']."</td>";
68                      echo "<td>".$row['email']."</td>";
69                      echo "</tr>";
70                  }
71                  mysqli_close($con);
72                  ?>
73              </tbody>
74          </table>
75      </div>
76      <br>
77      <div class="container">
78      <form class="form-inline" role="form" action="" method="post" >
79          <label class="form-control" for="first"> Firstname </label>
80          <input type="text" class="form-control" id="first" placeholder="Input first name" name="first">
81          <label class="form-control" for="last"> Lastname </label>
82          <input type="text" class="form-control" id="last" placeholder="Input last name" name="last">
83          <label class="form-control" for="email"> Email </label>
84          <input type="text" class="form-control" id="email" placeholder="Input email address" name="email">
85          <button type="submit" class="btn btn-primary" id="add" name="add" value="add"> Add </button>
86          <button type="submit" class="btn btn-primary" id="search" name="search" value="search"> Search </button>
87      </form>
88      </div>
89  </body>
90  </html>
```

From line62 to line72 are PHP code, the dynamic data are retrieved from database and output to the html by using "while" loop.

To interact with server, there are mainly two methods: GET and POST. GET method is used to appends form data to the URL in name or value pair. If you use GET, the length of URL will remain limited. It helps users to submit the bookmark the result. GET is better for the data which does not require any security or having images or word documents. POST is a method that is supported by HTTP and depicts that a web server accepts the data included in the body of the message. POST is often used by World Wide Web to send user generated data to the web server or when you upload file. To use POST method, we need to use the <form> tag from line77 to line88. Please notice the bootstrap elements have been used. In the <form> tag (line78), the action will determine the target webpage. In this demo, we keep it blank. That means the form will post to the same page.

Receive and process the GET or POST requests are basic functions of Web servers. The PHP code for this lab is as:



```
1   <?php
2
3   //open database by mysqli
4   $con = mysqli_connect("localhost", "root", "", "lab2");
5   if (mysqli_connect_errno($con)) {
6       die("Connect to MySQL failed: " . mysqli_connect_error());
7   }
8
9   //a safe method to recieve post data
10  function mypost($str) {
11      $val = !empty($_POST[$str]) ? $_POST[$str] : '';
12      return $val;
13  }
14
15  //receive query parameters.
16  $last = mypost('last');
17  $first = mypost('first');
18  $email = mypost('email');
19
20  //add the received data to database
21  if (isset($_POST['add'])) {
22      $sql = "INSERT INTO `user` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, '".$first."', '".$last."', '".$email."')";
23      $query = mysqli_query($con,$sql);
24  }
25
26  //query the data from database
27  if (isset($_POST['search'])) {
28      $sql = "select * from user where first_name LIKE '%".$first."%' and last_name LIKE '%".$last."%' and email LIKE '%".$email."%'";
29      $query = mysqli_query($con,$sql);
30  }
31  else {
32      $sql = "select * from user";
33      $query = mysqli_query($con,$sql);
34  }
35
36  ?>
```

"mysqli" is a build-in tool to operate MySQL type database, the code from line4 to line7 is to open a new connection to database. In which, localhost means the MySQL server address, root and "" are the username and password for the database and "lab2" is the name of database.

To receive the posted data, we may have a safe function to avoid "null" error. Which is from line10 to line13.

The rest logics are:

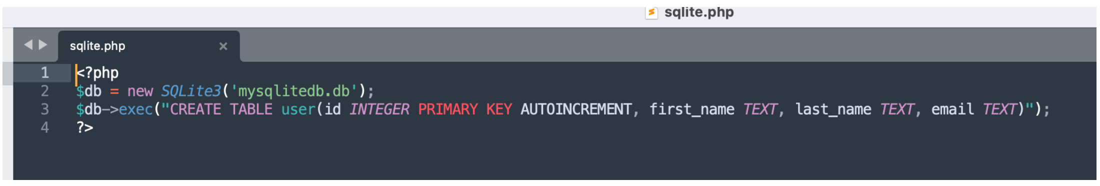
Receive all post parameters first. If user click "add" button, then insert the received data into table "user". If user click "search" button, then add the conditions to SQL statement, otherwise retrieve all data from table "user". The retrieved data will be outputted to web page by the code from line62 to line72.

Be careful, the code here is only a demo purpose. It is very weak. For example, the SQL injection can cause the database error easily.

Database and sqlite3:

2. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. The demo here will do the same function with the sqlite3 type database or DBMS.

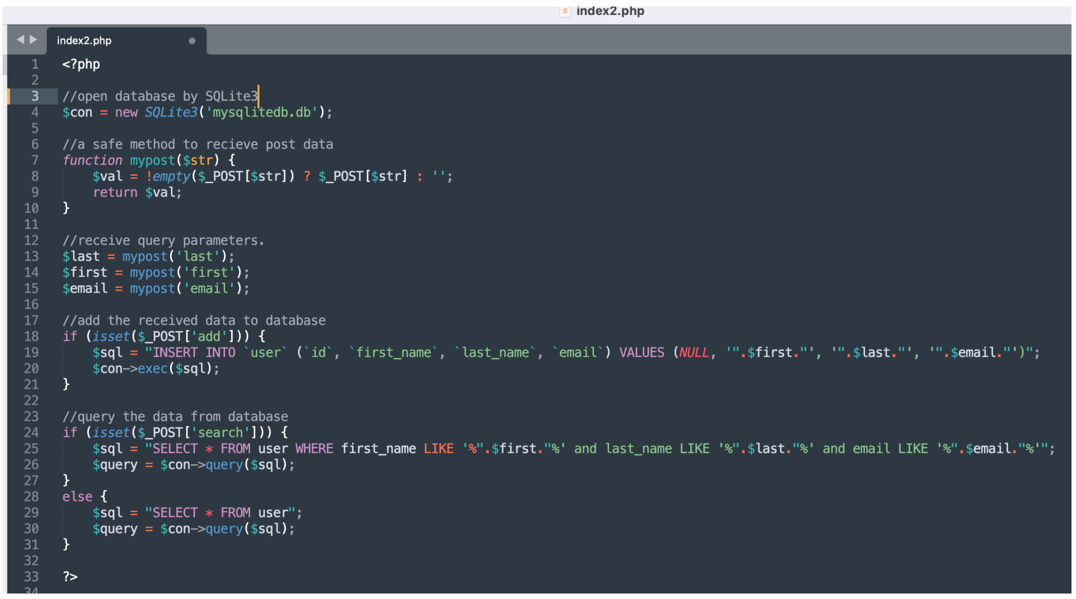
   We need to prepare the SQLite database same as the MariaDB one. The following PHP code can make it:



```php
<?php
$db = new SQLite3('mysqlitedb.db');
$db->exec("CREATE TABLE user(id INTEGER PRIMARY KEY AUTOINCREMENT, first_name TEXT, last_name TEXT, email TEXT)");
?>
```

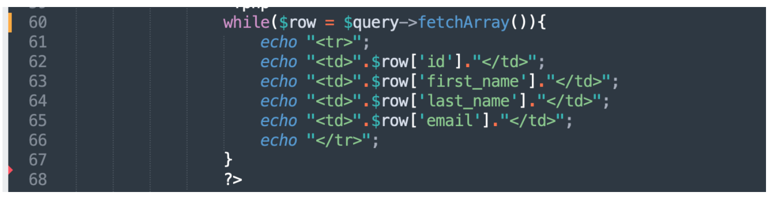
   Please note another build-in class "SQLite3" has been used to operate the sqlite3 type database. By execute this code (you can run the php in command line mode or configure your IDE that can run the php directly in your IDE), a file named "mysqlitedb.db" will be created in the same directory of the php file and a user table will be created.

   To migrate from the MariaDB to sqlite3, we need to modify the code accordingly. The changed code is as:



```php
<?php

//open database by SQLite3
$con = new SQLite3('mysqlitedb.db');

//a safe method to recieve post data
function mypost($str) {
    $val = !empty($_POST[$str]) ? $_POST[$str] : '';
    return $val;
}

//receive query parameters.
$last = mypost('last');
$first = mypost('first');
$email = mypost('email');

//add the received data to database
if (isset($_POST['add'])) {
    $sql = "INSERT INTO `user` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, '".$first."', '".$last."', '".$email."')";
    $con->exec($sql);
}

//query the data from database
if (isset($_POST['search'])) {
    $sql = "SELECT * FROM user WHERE first_name LIKE '%".$first."%' and last_name LIKE '%".$last."%' and email LIKE '%".$email."%'";
    $query = $con->query($sql);
}
else {
    $sql = "SELECT * FROM user";
    $query = $con->query($sql);
}

?>
```

   and:

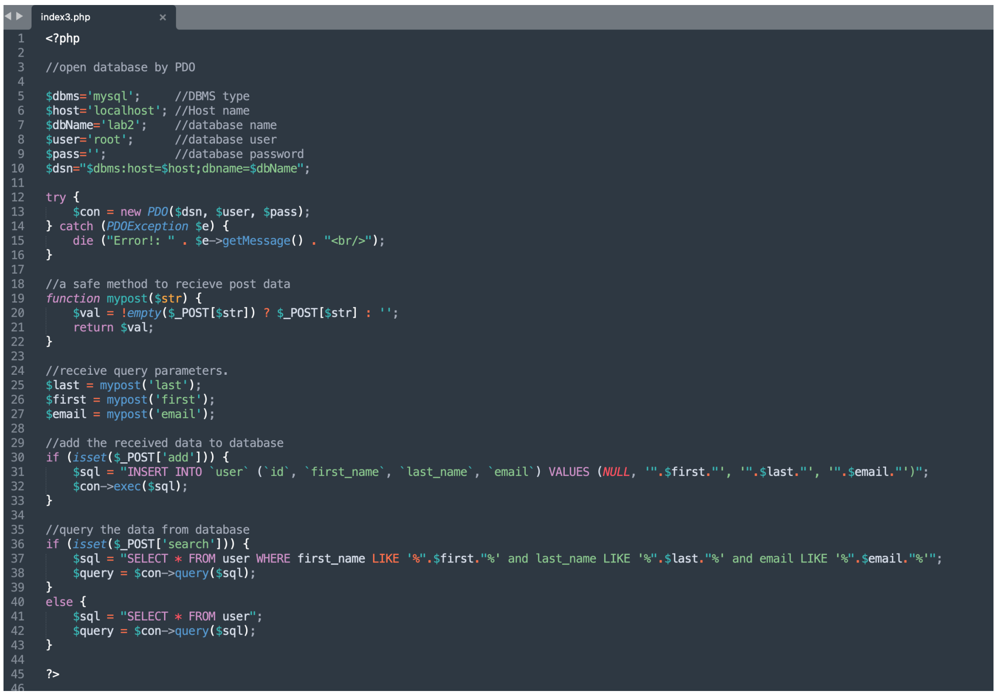


```php
while($row = $query->fetchArray()){
    echo "<tr>";
    echo "<td>".$row['id']."</td>";
    echo "<td>".$row['first_name']."</td>";
    echo "<td>".$row['last_name']."</td>";
    echo "<td>".$row['email']."</td>";
    echo "</tr>";
}
?>
```

   Now you can try to verify all functions by visiting the updated webpage. You may encounter the database read-only warning, you need to adjust the permission accordingly.

PDO:

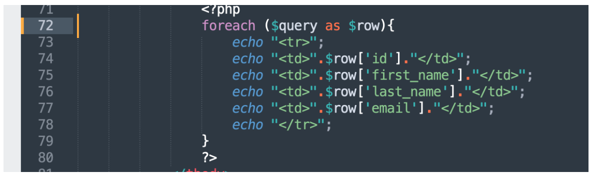
3. SQLite or MySQL? Each database has its advantages and disadvantages. There is no answer which database should be chosen if the requirements are unknown.
In the previous demos, we use two different tools to connect two different types database. Then we need to adjust the PHP code accordingly. The code is tightly coupled with the database type. Simply it limits the flexibly on database choices.
"PDO" is another choice. Thanks OO programming, PDO has implements the connections to different databases and hide the details. So, the same code can be used for different type databases.
The updated PHP code is as:



index3.php

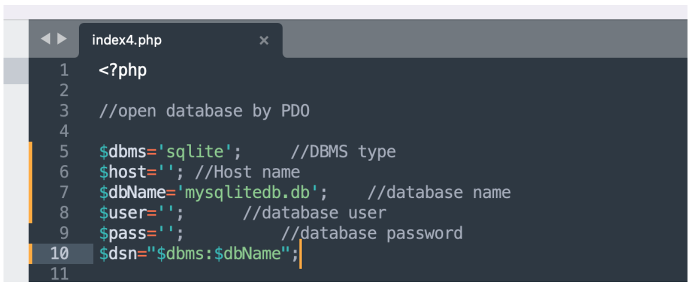
```php
<?php

//open database by PDO

$dbms='mysql';      //DBMS type
$host='localhost'; //Host name
$dbName='lab2';    //database name
$user='root';      //database user
$pass='';          //database password
$dsn="$dbms:host=$host;dbname=$dbName";

try {
    $con = new PDO($dsn, $user, $pass);
} catch (PDOException $e) {
    die ("Error!: " . $e->getMessage() . "<br/>");
}

//a safe method to recieve post data
function mypost($str) {
    $val = !empty($_POST[$str]) ? $_POST[$str] : '';
    return $val;
}

//receive query parameters.
$last = mypost('last');
$first = mypost('first');
$email = mypost('email');

//add the received data to database
if (isset($_POST['add'])) {
    $sql = "INSERT INTO `user` (`id`, `first_name`, `last_name`, `email`) VALUES (NULL, '".$first."', '".$last."', '".$email."')";
    $con->exec($sql);
}

//query the data from database
if (isset($_POST['search'])) {
    $sql = "SELECT * FROM user WHERE first_name LIKE '%".$first."%' and last_name LIKE '%".$last."%' and email LIKE '%".$email."%'";
    $query = $con->query($sql);
}
else {
    $sql = "SELECT * FROM user";
    $query = $con->query($sql);
}

?>
```

and:



```php
<?php
foreach ($query as $row){
    echo "<tr>";
    echo "<td>".$row['id']."</td>";
    echo "<td>".$row['first_name']."</td>";
    echo "<td>".$row['last_name']."</td>";
    echo "<td>".$row['email']."</td>";
    echo "</tr>";
}
?>
```

Switch to SQLite is very simple now. Only the DBMS information need to be changed as:



index4.php

```php
<?php

//open database by PDO

$dbms='sqlite';      //DBMS type
$host=''; //Host name
$dbName='mysqlitedb.db';     //database name
$user='';       //database user
$pass='';            //database password
$dsn="$dbms:$dbName";
```

Through this lab, you can gain the basic ideas how PHP interact with database. However, due to time limitation, the operation of database cannot be explained in details and many demo code are very fragile.