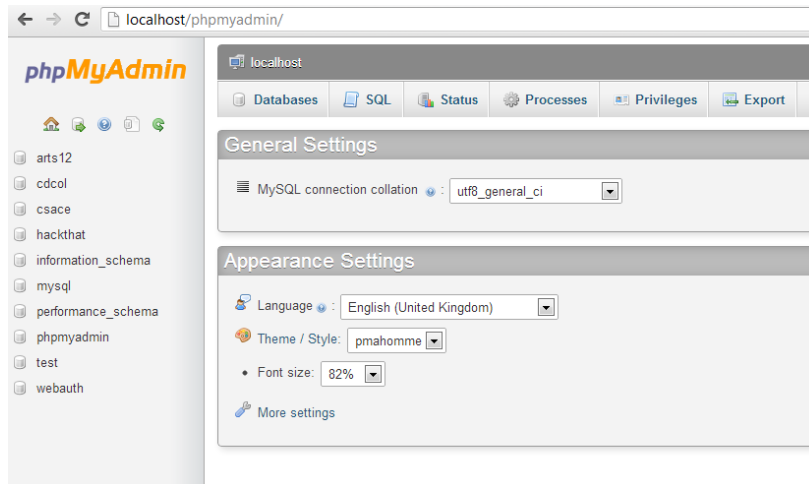


PHP and MySQL – Database Integration

Creating a MySQL user account

Your installation of XAMPP comes with MySQL which allows you to create databases. There is a useful interface to administrate your databases called phpmyadmin – you can get to it by browsing to

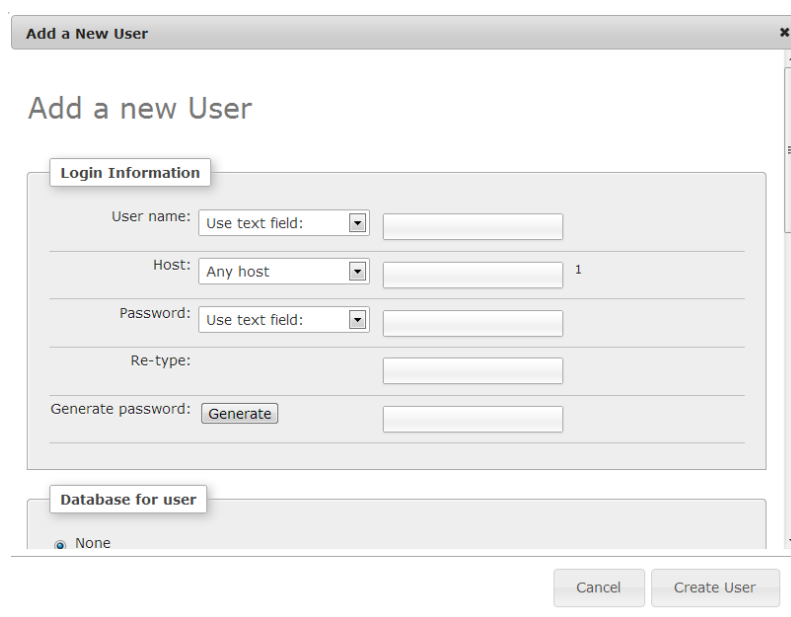
<http://localhost/phpmyadmin>



This is the phpmyadmin interface – you can see some of the databases I have already created in the list on the left.

Some of the databases are premade e.g. *information_schema* – don't delete them or phpmyadmin may go wrong

To create a user account for yourself so that you can connect to databases, click on **Privileges** at the top and then click on **Add a new user**



Choose a username.
Host should be localhost
Choose a password
Don't click on generate password!

Then scroll down and tick all of the boxes (click on check all) to make your user able to do everything.

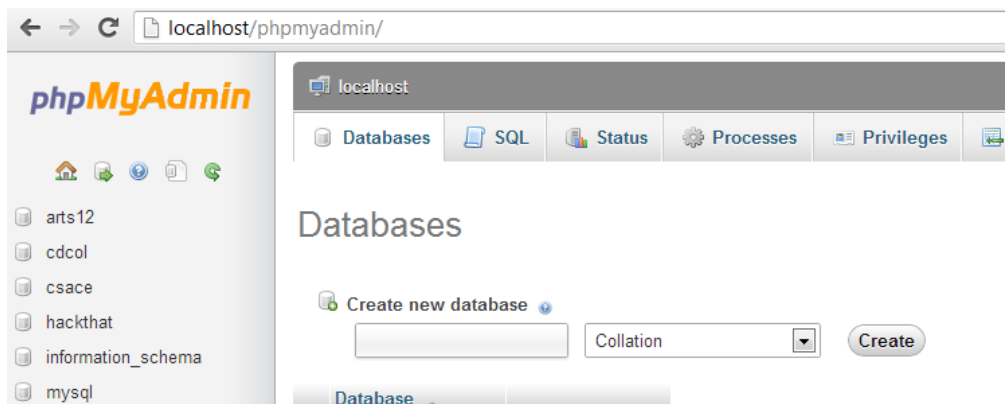
Then click the "Create User" button



Your installation of phpmyadmin might look slightly different to mine, but I'm sure you can figure out how it works!

Creating a database

Now that you have a user account, you need to create a database. Click on the Databases tab at the top:



Type the name of your database in the box and click “Create”. You can ignore the Collation drop down. I called mine shoutbox because it is going to be for the project we will do later.

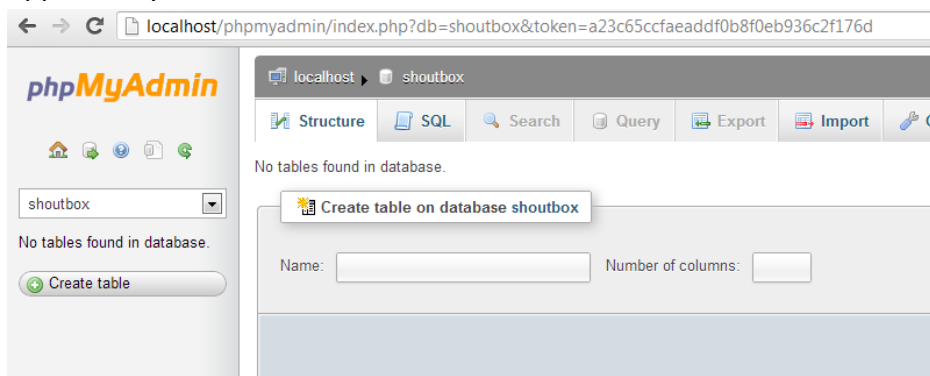


Don't use any spaces in your database name use underscores or camel/Pascal case to distinguish between words.

You should see the name of your database appear in the list on the left. You can click on its name to see it (there is nothing inside it yet).

Creating a table

When you have clicked on your database in the left hand side list you will be given an opportunity to create a table.



Type the name of the table and the number of columns inside the boxes. I am going to call my table **shouts** and I want 3 columns – again, this is for our mini project that is coming up.

You will be asked to fill in the details for your table. *Column* is the field name, *Type* is the data type of this field, *Length/values* is how long a space to reserve. You can set which field is the primary key by selecting an option for it in the *Index* drop down.

Here is how I am going to fill in this form for my shouts table:

Create Table			
Table name: shouts			
Structure			
Column	shout_id	shout_text	shout_date
Type	INT	VARCHAR	TIMESTAMP
Length/Values1	4	255	
Default2	None	None	CURRENT_TIMESTAMP
Collation			
Attributes			
Null	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index	PRIMARY	---	---
AUTO_INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments			

Cancel

As you have probably figured out by now, I'm working up to showing you how to create a basic project – a **shoutbox**! This is a bit like a very basic Facebook wall, with a box where a user can write comments and the previous comments also displayed.

localhost / localhost / sho x Shoutbox x

localhost/shoutbox/index.php

Message:

Welcome to my basic shoutbox
 The latest messages appear on top
 This is my shoutbox
 Hello post readers!

Connecting PHP with MySQL

PHP is really good for making applications which are linked to a database. You will have learnt about SQL and databases in your F453 module so now it is time to apply those concepts.



There are two sets of functions in PHP which allow you to connect to the database – **mysql** and **mysqli**. They are NOT interchangeable which means you must pick one and stick to it. I recommend picking **mysqli** as this is the new version – mysql is out of date, or to use the proper term it is **deprecated**.

I will be using the procedural style in the following examples, OO is available – see php.net

Connecting to the database

The easiest way to do this is to create a file and save it as **connection.php**. Your connection file should have the following information inside it:

```
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'my_db');
?>
```

Obviously, fill in your own MySQL user, password and database name!

Then, whenever you need to connect to the database in a program, use the line:

```
<?php require_once("connection.php"); ?>
```

This will grab the contents of the connection.php file and insert it into the page so that you can use the `$link` variable as the connection you've already set up.

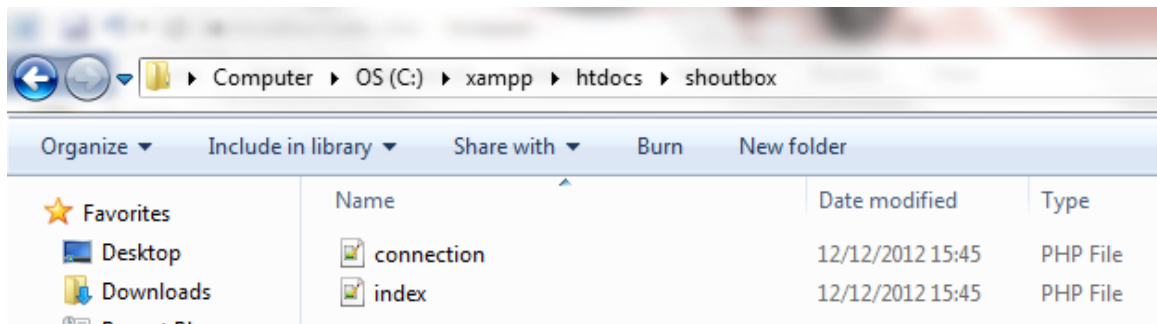


Why do I need a separate file? Can't I just put the connection line on the top of each page? Well yes, you can. But imagine if you change your server details – which will definitely happen if you are testing your site on XAMPP and uploading to an online webserver later on. You'll have to change the details in every single page. If you put them in a connection file and require that file wherever a connection is needed, you only need to change the details once in the connection file instead of everywhere. Genius!

Mini Project 2 – Shoutbox

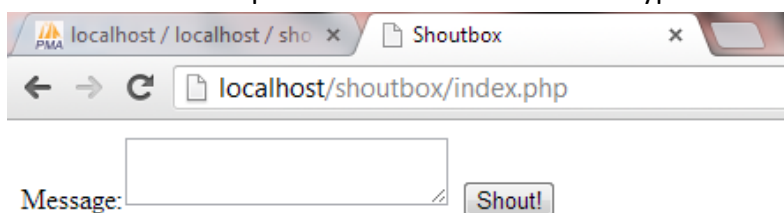
We've actually already started this project, so if you have not already done so please start from the "PHP and MySQL – Database Integration" section on page 21.

Create a subfolder called **shoutbox** inside htdocs, and inside there create a file called **index.php**. Put the connection file you just created inside this folder as well.



Displaying the input form

We need a basic input form where the user can type in their shout, like this:



To make a larger box like the one above, you could use a `<textarea>`, the code looks like this:

```
<textarea name="shout"></textarea>
```

If you can't remember how to make the form, have a look on page 14 for some tips. You can style the form to your liking using CSS if you want to make it look prettier.

Processing the data

When the user types their shout in the box and clicks the button, the code should send this data to a processing page called **send.php**

- This means that the form action inside index.php should be `<form action="send.php">`
- Create and save a file called **send.php** inside the shoutbox folder in htdocs

Putting data in to the database

Assuming you have already included your connection file, this is the basic format for a query

// Create the query

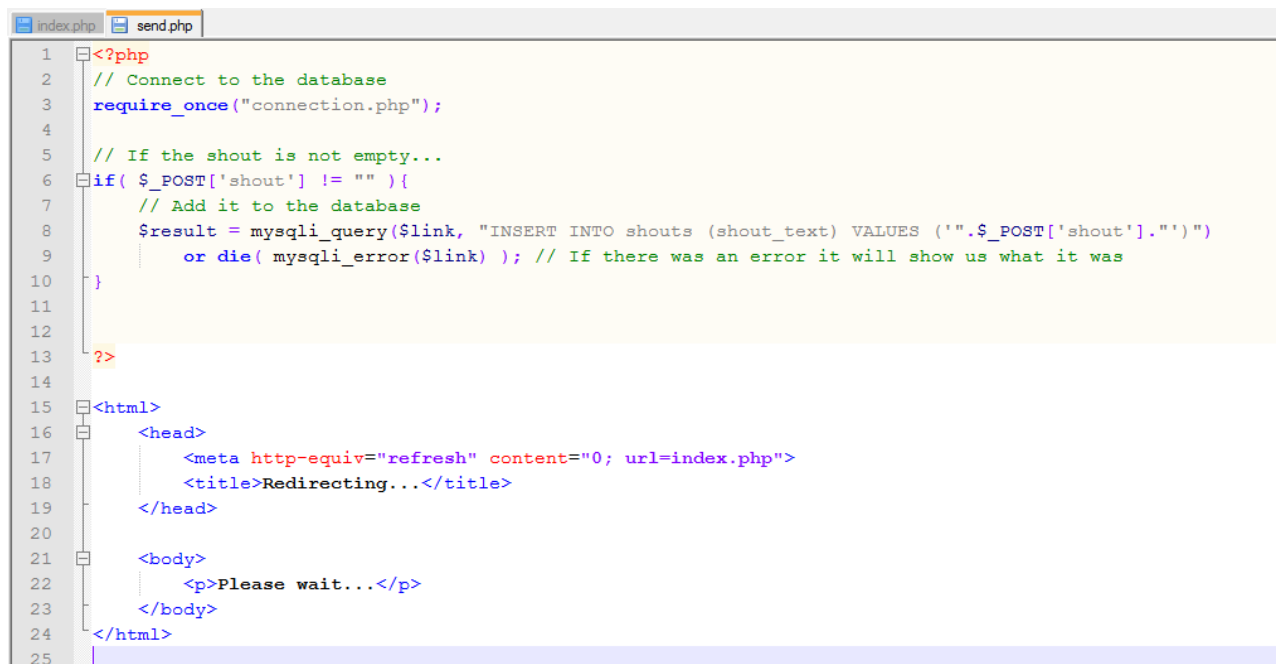
```
$query = "INSERT INTO tablename(field1, field2, field3) VALUES ('value1', 'value2', 'value3')";
```

// Execute the query

```
$result = mysqli_query($link, $query);
```

You need to put in your own table name, field names and values. The values should be in the same order as the field names, i.e. value1 will go into field1 etc.

This is the code I have inside my **send.php** page



```
1 <?php
2 // Connect to the database
3 require_once("connection.php");
4
5 // If the shout is not empty...
6 if( $_POST['shout'] != "" ){
7     // Add it to the database
8     $result = mysqli_query($link, "INSERT INTO shouts (shout_text) VALUES ('".$_POST['shout']."')");
9     or die( mysqli_error($link) ); // If there was an error it will show us what it was
10 }
11
12
13 ?>
14
15 <html>
16 <head>
17     <meta http-equiv="refresh" content="0; url=index.php">
18     <title>Redirecting...</title>
19 </head>
20
21 <body>
22     <p>Please wait...</p>
23 </body>
24 </html>
25
```

Lines 1-13 are PHP code to insert the shout into the database. Lines 15-24 are HTML code which provides a meta refresh – on line 17 the code specifies to wait 0 seconds and then go to index.php. This means that once the page is done with inserting the data into the database, it redirects quickly to the first page.

If you like the cartoon on the front page of this booklet, you may be worried about **validation** and **SQL injection** – these are improvements you can make to the shoutbox, see page 30 for details.

What's going on here?

Let's take a closer look at some of the PHP code in the processing page

```
5 // If the shout is not empty...
6 if( $_POST['shout'] != "" ){
```

This is some **validation** – I am checking that the shout they entered was not an empty string. I could also have done it this way:

```
if( !empty($_POST['shout']) ) { ... }
```

This checks whether the variable `$_POST['shout']` is not empty. (Don't forget we can refer to what the user typed in as `$_POST['shout']` because it is passed to this page in the `$_POST` array, and it has the index 'shout' because that was the name of our textarea on **index.php**)

```
7 // Add it to the database
8 $result = mysqli_query($link, "INSERT INTO shouts
9                                     (shout_text)
10                                    VALUES
11                                    ('".$_POST['shout']."'");
12 or die( mysqli_error($link) );
13 // If there was an error it will show us what it was
```

This part looks a bit technical, but it isn't. The first **argument** on line 8 is the database link variable that we created in the connection.php file. The second argument is the query itself.

In the above code, on line 11 we are using concatenation to put the `$_POST['shout']` variable inside the query. If the amount of quotes messes with your head, try renaming the variable (line 8) and then you can use its short name on line 9. It works exactly the same but it's a poor and inefficient way of coding and you should try to understand the concatenation method above if you can.

```
7 // Add it to the database
8 $shout = $_POST['shout'];
9 $result = mysqli_query($link, "INSERT INTO shouts (shout_text) VALUES ('$shout')");
10 or die( mysqli_error($link) ); // If there was an error it will show us what it was
```

The “or die” part means that if the query cannot be done for whatever reason, the page should stop executing and give us a mysql error message so that we can debug it.

Getting data out of the database

Now we are back on the index page, our data has been inserted into the database, but we can't yet see the shouts displaying below the message box.

Message:

Welcome to my basic shoutbox
The latest messages appear on top
This is my shoutbox
Hello post readers!

Go back and edit your **index.php** file, and after the code for the form add a PHP section

A general example

Here is a general example of how to write a SELECT query to get data from your database

```
// Write the query
$query = "SELECT fieldname FROM tablename";

// Execute the query
$result = mysqli_query($link, $query);

// If anything was found, get the results
while($data = mysqli_fetch_assoc($result)){
    print $data['fieldname'];
}
```

In this example, you specify which fields you would like from which table, when you write the query.

In this example, the results are put one by one into an array called \$data

```
$data = mysqli_fetch_assoc($result);
```

You can then reference each field by its name

e.g. if you had a field called username you could use...

```
print $data['username'];
```

...if you had a field called dateofbirth you could use

```
print $data['dateofbirth'];
```


How to display your shouts

Here is the code now in my index.php page

```
index.php send.php
1 <html>
2   <head>
3     <title>Shoutbox</title>
4   </head>
5   <body>
6
7     <form action="send.php" method="post">
8       Message:<textarea name="shout" size="50" maxlength="300"></textarea>
9       <input type="submit" value="Shout!">
10    </form>
11
12
13
14
15    <?php
16      // Connect to the database
17      require_once("connection.php");
18
19      // Get all of the shouts
20      $result = mysqli_query($link, "SELECT * FROM shouts ORDER BY shout_date DESC")
21      or die( mysqli_error($link) );
22
23      // Loop through the results, printing out each shout
24      while ($data = mysqli_fetch_assoc($result) ){
25        print $data['shout_text'];
26        print "<br>";
27      }
28
29    ?>
30
31
32   </body>
33 </html>
```

On lines 20-21 I do a query in exactly the same way as we discussed before – except that this is a SELECT query to get data from the database, and not an INSERT query to put data in the database.

Lines 24-27 are a loop which **iterates** through all of the rows of data returned by the query, printing out the shout text and a HTML newline character (
).

Improvements to the shoutbox

There are numerous ways you could improve your shoutbox code, here are some ideas for you to work on, in order of difficulty!

- Tidy up the presentation of your input form using CSS
- Make the page display the time and date of the shout as well
- Only show the top 5 most recent shouts
- Print alternate shouts in a different colour (hint: use MOD or %)
- Add another input box for the user's name, save this in the database and display it along with the shout
- Make the user's name persist in the box after they have submitted the form (hint: try a hidden field in the form?)
- Look up some PHP validation functions such as `strip_tags()` and `stripslashes()` on php.net and try to use them to validate your input
- Look up what SQL injection is and try to prevent it from being possible on your shoutbox – try `mysqli_real_escape_string()` function on PHP.net

To come in future updates of the book:

- Session variables
- Creating a basic login system