Web Systems Fundamentals (Grundläggande webbsystem) 7,5 HP DI4004 VT19

Laboration 3

Course responsible:
Wagner Ourique de Morais

Laboration 3 – Introduction to PHP and MySQL

In this lab students will have the opportunity to:

- Learn how to develop dynamic websites
- Create scripts in PHP
- Use phpMyAdmin to data definition and manipulation in MySQL
- Create dynamic pages using PHP and MySQL
- Apply basic web site security

Introduction

In this laboratory exercise, you will learn how to develop dynamic websites using PHP and MySQL. This includes creating a web site that structures the content of its web pages using a server-side programming language, in this case PHP. The laboratory exercises were created and tested on the LAMP Server at ideweb2. Therefore, a prerequisite to get the tasks working is that you ensure that you can login to the web server (http://ideweb2.hh.se/~user_name) and to the database server (https://ideweb2.hh.se/phpmyadmin).
The web site that you will create in this laboratory exercise will be connected to a database server that can handle logins, store information about users, products and purchases.

Important

Consider the footnotes. They provide links to web sites in which you can get more references about a particular command or functionality.

Help!

If at this had any problems at this point, please use the Discussion Board in the course page in Blackboard to get help.
If you don't know how to accomplish a given task, try to find information in the course literature or online. You can also check how a webpage has solved various issues by right-clicking part of a webpage and selecting "Inspect element" or "Inspect". You will see the html code for that part of the website (in most cases).

W3Schools (www.w3schools.com) has some great, interactive learning material.

If you're still stuck, ask one of the lab assistants.

Accessing the LAMP Server at ideweb2

Make sure that you are able to connect to the web server:

http://ideweb2.hh.se/~user_name

You should be able to access the web server at:

https://ideweb2.hh.se:8443/WebInterface/login.html

You should be able to access the database server at:

https://ideweb2.hh.se/phpmyadmin

In the web server (https://ideweb2.hh.se:8443/WebInterface/login.html), the task here is to create a few directories for the files you will create during the lab.

- Create a directory called `lab3` within `~/public_html`
- Create the following directories within `~/public_html/lab3`
  - `css`
  - `img`
  - `js`
- Create the following files within `~/public_html/lab3`
  - `about.php`
  - `index.php`
  - `template.php`

At the end of this exercise, you will be able to see the result at:

http://ideweb2.hh.se/~user_name/lab3.

(remember to replace user_name with your loggin ID).

From now on, `~/public_html/lab3` will be your working directory.

Part 1: Creating and Editing PHP-files (content and layout)

In this task you will learn and use the echo statement, variables, the <<< (Heredoc) operator, and the include statement

index.php – step 1

The index.php file is the home page of your web site. Please consider the following code:

```
<?php
$content = <<<END
<h1>Welcome to this website</h1>
<p>
This is gonna be a webshop
</p>
END;
echo $content;
?>
```

The code contains the following:

- A PHP[1] script starts with **<?php** and ends with **?>**. The PHP has to be inside of these tags or else our webpage won't understand.
    - A PHP file normally contains HTML tags, and some PHP scripting code.
    - PHP statements end with a semicolon (;).
    - PHP keywords (e.g. if, else, while, echo, etc.), classes and functions are NOT case-sensitive.
    - However, all variable names in PHP are case-sensitive.
- **$content** – this is a variable in PHP and starts with the $ sign, followed by the name of the variable.
    - In this particular case the variable $content contains a few rows of HTML code delimitated by the <<< (heredoc) operator.
    - **<<<** – heredoc operator and is used to delimit strings.
    - Syntax: <<<. After this operator, an identifier is provided, then a newline. The string itself follows, and then the same identifier again to close the quotation[2].
    - In this particular case <<<END...END; delimit and hold html code.
- **echo $content** – the echo stamen is used to print out literal values and variables in PHP.
- In this case it will print out the contents of $content which is an HTML code!

Tasks

1. Edit the index.php file and add the PHP code above to it.
2. Using a web browser, enter http://ideweb2.hh.se/~user_name/lab3

[1] http://www.w3schools.com/php/php_syntax.asp
[2] http://php.net/manual/en/language.types.string.php#language.types.string.syntax.heredoc
http://en.wikipedia.org/wiki/Here_document#PHP

template.php – step 1

The `template.php` is the file for a page that contains all the layout code (menus, links to .css and .js files, etc) that is used on multiple pages. This page can be used later on to connect the site to a database and let clients save information about sessions.

It is very easy to create a simple main structure. As a start, you will create a simple menu or navigation that is reused in all other pages. Please consider the following code:

```
<?php
$navigation = <<<EOT
<nav>
    <a href="index.php">Home</a>
    <a href="about.php">About</a>
</nav>
EOT;
?>
```

The code contains the following:

- **$navigation** – this is also a variable in PHP and its contents is also delimited using the <<< (heredoc) operator. Note that a different name (EOT – end of text) is provided to the heredoc identifier.

Tasks

1. Edit the `template.php` file and add the PHP code above to it.

index.php – step 2

Now we want to add the menu to the index.php. However, instead of writing the same php into several .php files, we will use the include statement.

The include (or require) statement: "The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement. Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website."[3]

- *The require statement is also used to include a file into the PHP code.*
- *However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute.*
- *Use require when the file is required by the application.*
- *Use include when the file is not required and application should continue when file is not found.*

Please consider the following code:

```
<!DOCTYPE html>
<html>
<head>
<title>Webshop</title>
</head>
<body>
<?php
include('template.php');
$content = <<<END
<h1>Welcome to this website</h1>
<p>
This is gonna be a webshop
</p>
END;
echo $navigation;
echo $content;
?>
</body>
</html>
```

The code contains the following:

- `include('template.php');` - the include statement indicating to include the `template.php` file
  - From now on, we can reference to the variable $navigation, which is defined in the `template.php` file
- `echo $navigation` – is the variable defined in the template.php file. It is used to add the navigation bar to the web page.

Tasks

1. Edit the index.php file and replace its content with PHP code above.
2. Using a web browser, enter http://ideweb2.hh.se/~user_name/lab3
   a. Try to click in the hyperlinks. Some might not work.
3. Try and check:
   a. Change `echo $navigation` to `Echo $navigation`. Does it work?
   b. Change `echo $navigation` again to `echo $Navigation`. Does it work?
   c. Change `echo $navigation` again to `echo navigation`. Does it work?

[3] http://www.w3schools.com/php/php_includes.asp

about.php – step 1

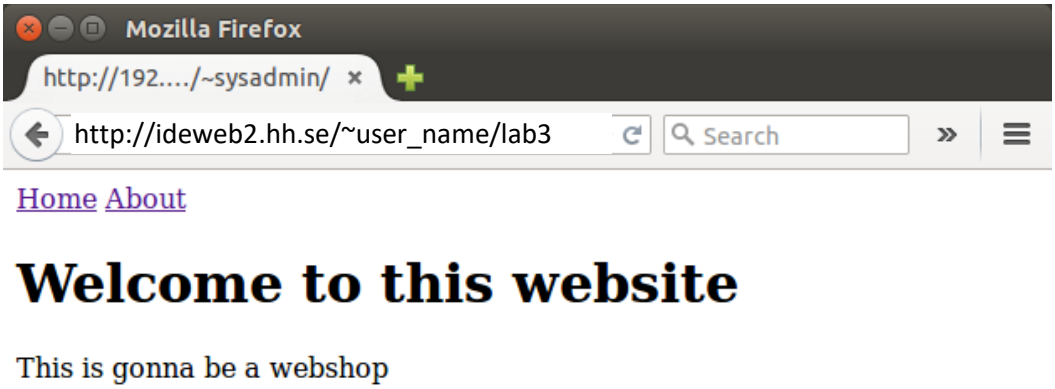
Please consider the following code:

```
<?php
include('template.php');
$content = <<<END
<h1>About</h1>
<p>
This is gonna be a webshop
</p>
END;
echo $navigation;
echo $content;
?>
```

Task

1. Edit the about.php file and add the PHP code above to it.
2. Using a web browser, enter http://ideweb2.hh.se/~user_name/lab3
    a. Navigate in your web site using the menu.

You should then see that the following content and be able to navigate between them.



template.php – step 2

Now that you got an early version of your web site working, let's apply some style the content.

Task

1. Create a new file in the css directory called stylesheet.css. Fill stylesheet.css with the css that you want. You could use a style from CSS ZEN GARDEN[4], W3.CSS[5]. Otherwise a simple example is:

```
h1 {
      color: blue;
}
```

2. Then go back to template.php and add a reference to the external css file, i.e.¨

```
<link rel="stylesheet" href="css/stylesheet.css">
```

```
<link rel="stylesheet" href="css/stylesheet.css">
<?php
$navigation = <<<EOT
<nav>
    <a href="index.php">Home</a>
    <a href="about.php">About</a>
</nav>
EOT;
?>
```

The code contains the following:

- The <link> tag defines a link between a document and an external resource.
    - The rel="stylesheet" indicates that such a resource is a stylesheet.
    - href =" css/stylesheet.css" is the path to the stylesheet.
    - A similar approach is used to include external libraries and JavaScript files

[4] http://www.csszengarden.com/
[5] https://www.w3schools.com/w3css/default.asp

Part 2: Create a contact form and send messages by email

We are now going to create a simple contact form on the about.php page and use the mail function to send the messages to an email-address.
Please consider the following code:

```
<meta charset="utf-8">
<?php
include('template.php');
$content = <<<END
 <h1>About this website</h1>
 <p>Work in progress</p>
 <h3>Send a message</h3>
 <form action="send.php" method="post">
   <input type="text" name="name" placeholder="Name">
   <br>
   <textarea name="msg" placeholder="Message"></textarea>
   <br>
   <input type="submit" value="Send">
 </form>
END;
echo $navigation;
echo $content;
?>
```

The code contains the following:

- Form action - The page where we want to send the incoming-data we receive.
- Form method - the http-method we want to use to send our input. In short POST is more secure than using GET[6].
- input type - Is a control of what kind of data it will receive.
- input name - The index (name) our input will have when we send it. This is used when we want to retrieve the input in another page.
- input placeholder - Shows what content is planned to be in the empty field.
- input value - The text that shows up on the buttons (as submit/reset)

Forms will come back in later parts and uses so be sure to have a clear understanding of how they work.

[6] http://www.w3schools.com/tags/ref_httpmethods.asp

send.php – step 1

In the form we created on about.php we used send.php as action. This means that the form will send the input to send.php, which is a problem since that file doesn't exist yet. Let's create it!

Tasks

1. Create a file called send.php in the ~/public_html/ lab3 directory.

Since we selected POST[7][8] as our method of sending the input, it will store all the data in a variable that is automatically named $_POST. $_POST is an array that has different indexes for values. The index names are created during design time, i.e., the index names are defined when the form is created. POST creates an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

```
<form action="send.php" method="post">
  <input type="text" name="name" placeholder="Name"><br>
  <textarea name="msg" placeholder="Message"></textarea><br>
  <input type="submit" value="Send">
</form>
```

When you design a form, you define the attribute "name" for each input element within the form (see code above). Each input element corresponds to an input in the $_POST array. Later, you can retrieve the contained data using $_POST["name"] and the message from $_POST["msg"].

You can check whether a form successfully managed to send a $_POST to the page send.php by using the php function called isset(). It simply checks if a variable is set.

Task

2. Edit the send.php file and add the following code:

```
<?php
if (isset($_POST)) {
  echo 'Post-vairable was sent!';
}
?>
```

3. Using a web browser, enter http://ideweb2.hh.se/~user_name/lab3
   - Using the menu, navigate to the About web page. Try to fill the form with content and click send.

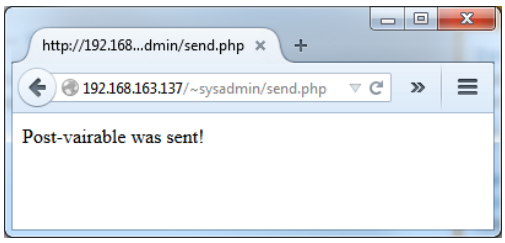


Figure 1. $_POST worked!

[7] http://www.w3schools.com/tags/ref_httpmethods.asp
[8] http://www.w3schools.com/php/php_forms.asp

If you see that message you know that your $_POST has worked. If you want to see all the content of the post you can try the function **var_dump($variablename),** which is an useful function when troubleshooting forms and finding out what kind of data is sent between pages.

Tasks

1. Edit the send.php file and use the var_dump function to check the content of $_POST.

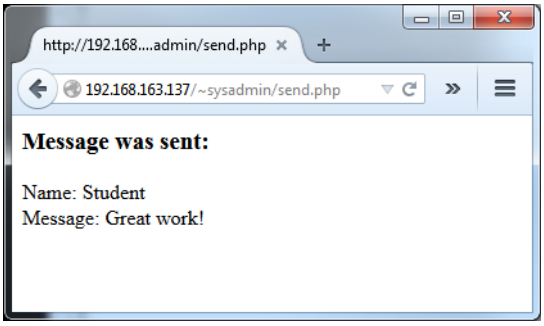
Now, let's try to expand the feedback of send.php to be more user friendly. Please consider the following code:

```
<?php
if (isset($_POST)) {
  $content = <<<END
      <h3>Message was sent:</h3>
        Name: {$_POST["name"]}
        <br>
        Message: {$_POST["msg"]}
END;
}
echo $content;
?>
```

Please note that the brackets {} is used around the $_POST[.... to include the php-variable inside of HTML.

Tasks

1. Edit the send.php file and replace its contents with the PHP code above.
2. Using a web browser, enter http://ideweb2.hh.se/~user_name/lab3
    - Using the menu, navigate to the About web page. Try to fill the form with content and click send.



You should now see a more user friendly display.

And OH! The navigation bar is missing.

Tasks

1. Add the navigation bar or menu to the send.php file. Don't remember? Scroll up!:)

send.php – step 2

To send messages as email we can use the function mail()[9].The mail() function allows you to send emails directly from a script.

- Syntax

```
mail($to,$subject,$message);
```

Normally it requires a sender that can be added into the headers of the email.

```
mail($to,$subject,$message,$headers)
```

Let's try adding the email-function with parameters in our website.

- Note that this requires that you have a SMTP-server in the server.

Please consider the following code:

```
<?php
include('template.php');
if (isset($_POST)) {
  $content = <<<END
       <h3>Message was sent:</h3>
        Name: {$_POST["name"]}
        <br>
        Message: {$_POST["msg"]}
END;
  $to      = "user18@student.hh.se"; //change this to your email address
  $subject = "Test-mail";
  $msg     = $_POST["msg"];
  $headers = "From: " . $_POST["name"];
  mail($to, $subject, $msg, $headers);
}
echo $navigation;
echo $content;
?>
```

Tasks

1. Edit the send.php file and add the PHP code above to it.
    - Switch the email in ($to) for your own email address. If you want to be able to reply to the emails, also replace $_POST["name"] with an email-address. (Again this only work if you have a SMTP-server on localhost or if you upload your files to ideweb2.hh)

[9] http://www.w3schools.com/php/func_mail_mail.asp

Part 3: MySQL with phpMyAdmin

In this part you will continue developing the web site you have created in this exercise. Once you complete this exercise, you will have a login and logout functionality in your web site.

For that, you will be using phpMyAdmin to create a database. Let's check how to connect to the phpMyAdmin webinterface.

IMPORTANT: the phpMyAdmin images used in this material might differ from the current phpMyAdmin GUI.

Tasks

1. Using a web browser, login to phpMyAdmin at ideweb2:
   - https://ideweb2.hh.se/phpmyadmin
   - Please visit Blackboard and select **School Server Environment** at the Laborations menu. The file SchoolServerEnvironment.pdf will provide you with information about how to login and use the MySQL server at ideweb2.

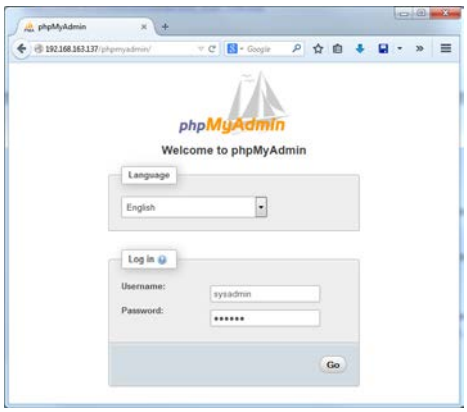


Figure 2. phpMyAdmin

2. Once the login succeeds, click at **Databases** (green arrow) on the top tab.

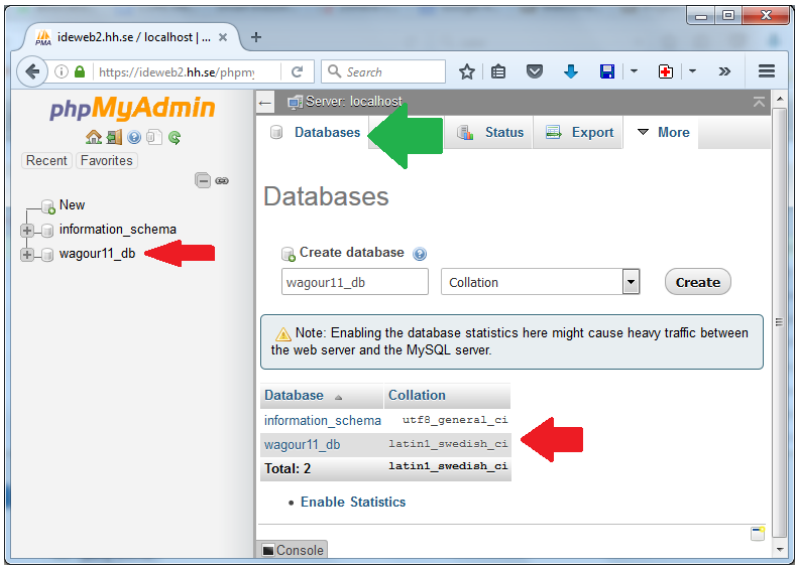


Figure 3. Visualizing the user's database.

3. You will see that there is already a database for your user (red arrows). The database name is `user_name_db`.

4. After that, click on the text database on the left side menu (indicated by the red arrow in Figure 6).
5. The objective now is to create a table called **users**, which will contain **3** columns.
    - Once you have informed the table name and the number of columns, click **Go**.

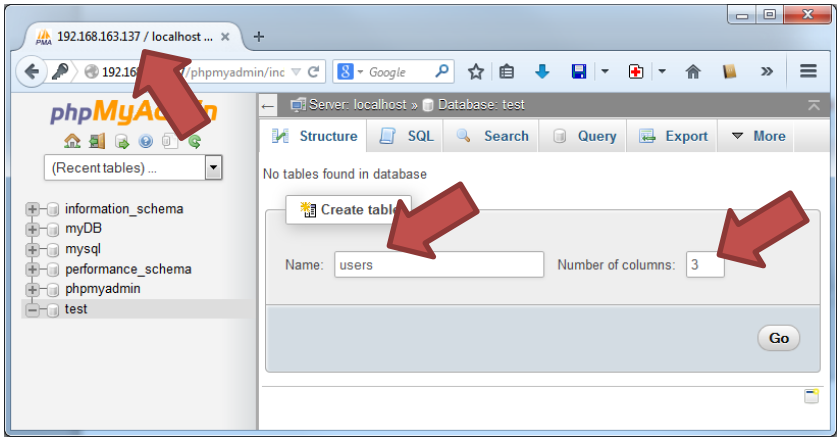


Figure 4. Creating the table users with 3 columns.

After the table is created you shall see 3 columns. We are going to fill them out and add a primary key and attributes for username and password.

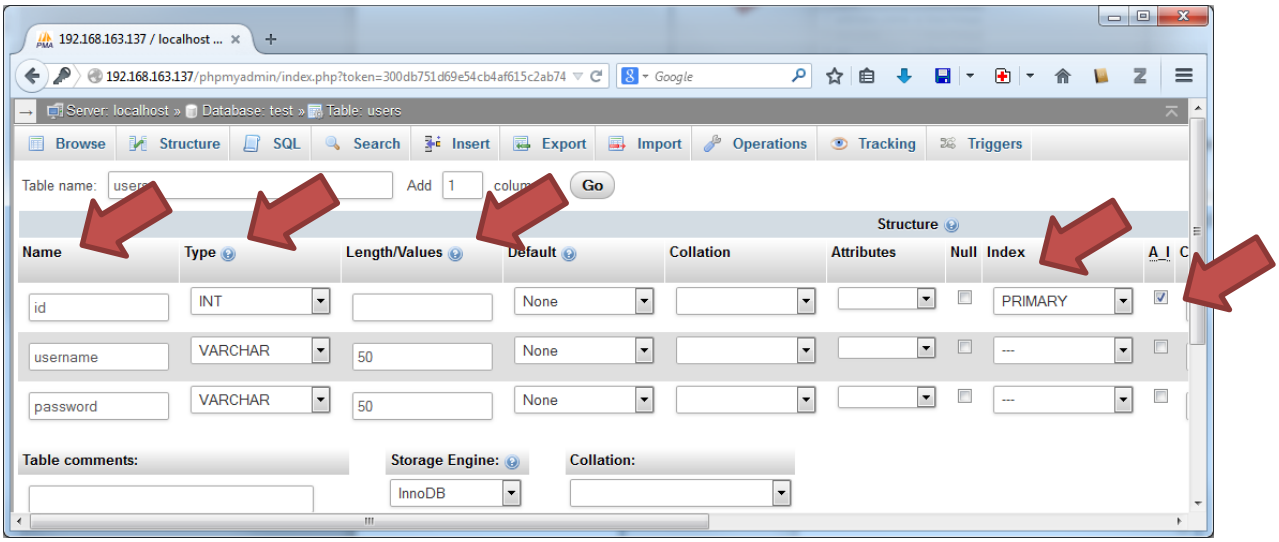


Figure 5. Columns setup

6. Setup the columns as follows
    - id
        - The id is of data type INT.
        - Set the Index to PRIMARY, since this column will be the primary key for table users.
        - Check the AI (Auto_Increment), which will attribute automatically values to this column when inserting new users
    - username
        - The data type for the username is VARCHAR
        - The maximum length is 50.

- Password
  - The data type for the username is VARCHAR
  - The maximum length is 50.

7. Scroll down on the page and click **Save**. It is located in the right side corner.

Once the operation is finished, you shall see a web interface similar to Figure 10.

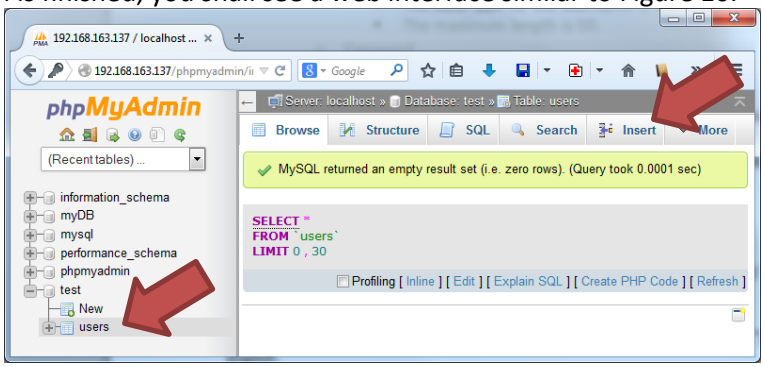


Figure 6. Table users

8. Now, the objective is to manually add a user to the table users.
   - To accomplish that, locate the table users (Use Figure 10 as a reference).
   - Click on Insert (Lägg till).

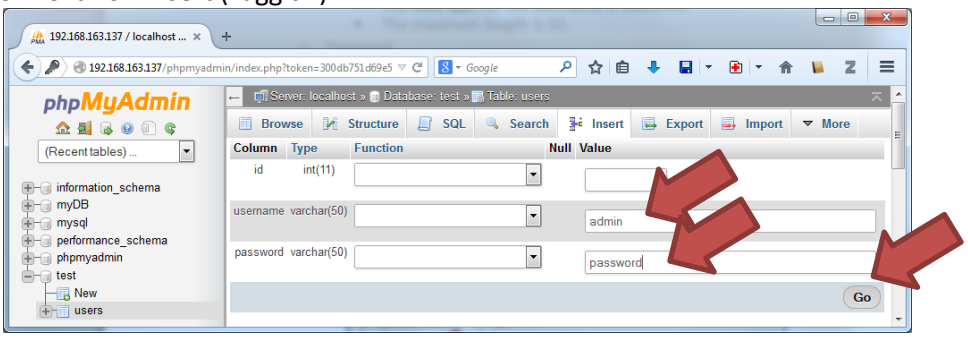


Figure 7. Adding the admin user

9. Let's add an Admin user to the table. You can select other names, but for exemplification purposes, the following values are used (which are not secure):
   - username = admin
   - password = password
   - Leave the id **empty** since it is auto_increment.
10. Click at Go (Kör) on the right corner to create the new user.

Congratulations. Now you have a good idea about how to create a database and tables as well as how to insert data into tables using phpMyAdmin. Now, lets' connect PHP and MYSQL.

Part 4 – PHP MySQL Database

To connect a website to a database management system (DBMS), the programing language needs a method to communicate with the DBMS. Such functionality is provided by DBMS drivers.
For example, to connect PHP with a MySQL DBMS, two DBMS drivers are available[10]:

- MySQLi extension (the "i" stands for improved)
- PDO (PHP Data Objects)

In this exercise, you will be using the mysqli driver.
Before we can access data in the MySQL database, you need connect to the server. For that, you need to know the IP address of the database server, the user with credentials to connect to the database, the user's password, and the database name.

```
$mysqli = new mysqli('serveraddress','username','password','databasename');
```

- The server address can be localhost or the IP address of the server.
- Remember, login information is found at **School Server Environment** at the Laborations menu in blackboard.
- The database name, in this exercise, will be the existing `user_name_db` database in the server.

Therefore, for this exercise, the connection object is created as follows:

```
$mysqli = new mysqli('localhost','user_name','user_pwd','user_name_db');
```

The best practice is to have database connections details and configuration files in a separate php file. Web pages using the database use the include statement to access the variables containing such details.

Another practice is that after the connection is made you might want to save the information inside of a session[11].
Session variables store user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. Therefore, you can use a session variable to store the database connection object, so you don't need to login every time.

[10] http://www.w3schools.com/php/php_mysql_connect.asp
[11] http://www.w3schools.com/php/php_sessions.asp

template.php – step 3

For a while, you will have the connection to the database inside of the template.php file. Please consider the following code:

```
<link rel="stylesheet" href="css/stylesheet.css">
<?php
session_name('Website');
session_start();
$mysqli     = new mysqli("localhost", "sysadmin", "dbik2015", "test");
$navigation = <<<END
<nav>
    <a href="index.php">Home</a>
    <a href="about.php">About</a>
</nav>
END;
?>
```

The code contains the following:

- session_name('Website'); - set the session name. In this case, the session name is “Website”.
- session_start(); - A session is started with the session_start()[12] function.

```
<link rel="stylesheet" href="css/stylesheet.css">
<?php
session_name('Website');
session_start();
$host       = "localhost";
$user       = "user_name"; // e.g. evanil18
$pwd        = "user_pwd"; // e.g takeAbath@06h30
$db         = "user_name_db"; // e.g evanil18_db
$mysqli     = new mysqli($host, $user, $pwd, $db);
$navigation = <<<END
 <nav>
      <a href="index.php">Home</a>
      <a href="about.php">About</a>
  </nav>
END;
?>
```

The code contains the following:

- $host, $user, $pwd, $db – variables containing the values for the server address, user, password and database used in the connection string.
    - $mysqli = new mysqli($host, $user, $pwd, $db);

[12] http://www.w3schools.com/php/php_sessions.asp

login.php – step 1

Now, the objective is to create a login page so your web site can have registered users.
The page login.php will contain a form with two inputs (username and password), as well as a submit button. Please consider the following code:

```
<?php
include('template.php');
$content = <<<END
<form action="login.php" method="post">
<input type="text" name="username" placeholder="username">
<input type="password" name="password" placeholder="password">
<input type="submit" value="Login">
</form>
END;
echo $navigation;
echo $content;
?>
```

Tasks

1. Create a new file at ~/public_html/lab3 and name it login.php.
2. A add the PHP code above to it.
    - Don't forget to include template.php because that is where the database connection is located. To access login.php for now you just have to change the URL.
3. Using a web browser, try to load http://ideweb2.hh.se/~user_name/lab3login.php

If you remember how forms worked then you might have noticed that the form action is pointing to itself. This is because we want to send the input to the same page. We will need to add the code that retrieves the input from the post and then compares the username and password with the database.

We are going to start with a conditional statement (if) to check whether the fields in form were filled out correctly or not. If so, you might want to check whether the informed username and password exist in the database.

Any requests made to a database are called queries. To make a query you use the $mysqli->query(SQL) function. However since your request in SQL can be pretty long, it is a good practice to create a variable that contains your query, lets name the variable $query.

Please consider the following code:

```
<?php
include('template.php');
if (isset($_POST['username']) and isset($_POST['password'])) {
  $query = <<<END
   SELECT username, password, id FROM users
    WHERE username = '{$_POST['username']}'
    AND password = '{$_POST['password']}'
END;
}
$content = <<<END
<form action="login.php" method="post">
<input type="text" name="username" placeholder="username">
<input type="password" name="password" placeholder="password">
<input type="submit" value="Login">
</form>
END;
echo $navigation;
echo $content;
?>
```

The code contains the following:

1. $query - variable containing the SQL statement, which is delimitated using the heredoc operator.
    - Remember that in a heredoc you have to mark the PHP code with brackets {}.
    - In SQL a string is marked by using single quotations 'string'.
    - So {$_POST['username']} is for HTML to understand that it's a PHP variable, and then enclosed by ' ' to make SQL understand that is a string.
        - The resulting SQL commend can be something similar to:

```
SELECT username, password, id
FROM users
WHERE username = 'admin'   AND password = 'password'
```

Now we have a variable called $query that contains SQL statements to retrieve a result set from table users and will contain the username, password and id.
The returned result set shall have the same username and password as in the $_POST values are the same as in the database.
However, you might not want only to execute a SQL statement. You might want to do something else in case the user exists or not in the database.

For this reason, you might want to store the result of the execution of the SQL statement in some variable, for posterior processing. You can achieve that by doing:

```
$result = $mysqli->query($query)
```

Now, the result of the SQL statement is stored at the $result variable. To check whether the SQL statement was successfully executed, you might want to use the function num_rows() checks if there are more than zero rows returned[13].

If the execution of the SQL statement returns multiple rows, han zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. Another

[13] http://www.w3schools.com/php/php_mysql_select.asp

approach is to use the function fetch_object(). We want to fetch data from our $result, and this is done by adding $result->fetch_object() to a new variable.
This is done to support less typing and also easier to read code. So later inside of the page we can simply use the new variable to fetch data.

Login.php – step 2

Please consider the following code:

```
<?php
include('template.php');
if (isset($_POST['username']) and isset($_POST['password'])) {
  $query  = <<<END
   SELECT username, password, id FROM users
    WHERE username = '{$_POST['username']}'
    AND password = '{$_POST['password']}'
END;
  $result = $mysqli->query($query);
  if ($result->num_rows > 0) {
    $row                  = $result->fetch_object();
    $_SESSION["username"] = $row->username;
    $_SESSION["userId"]   = $row->id;
    header("Location:index.php");
  } else {
    echo "Wrong username or password. Try again";
  }
}
$content = <<<END
<form action="login.php" method="post">
<input type="text" name="username" placeholder="username">
<input type="password" name="password" placeholder="password">
<input type="submit" value="Login">
</form>
END;
echo $navigation;
echo $content;
?>
```

Now analyze the code above until you think that it makes perfect sense. Also notice how we saved the username and userid inside of two new sessions.
Session variables are set with the PHP global variable: $_SESSION[14].
Using sessions, the web site can know which users are currently logged in.
Sessions can be used to change websites depending on the user, for instance a logged in user may have access to edit or updating tools. If we create a session when we log in and we can use this session from other pages. The session will last until:

2. You finish it using the commands:
    - session_unset(); - removes all session variables
    - session_destroy(); - destroys the session
    - or, you close the browser.

In the last part we use header("location") to send the successful logins to the index.php.
The header() function sends a raw HTTP header to a client[15].

Tasks

1. Edit the login.php file and add the PHP code on page 21 to it.
2. Using a web browser, test the login page

[14] http://www.w3schools.com/php/php_sessions.asp
[15] http://www.w3schools.com/php/func_http_header.asp

template.php – step 4

In our template.php we want to change the displayed menu depending on if you are logged in or not. Let's add a segment of code to the navigation variable. Again we can use the isset to see if a user is logged in. It's also time to see how sessions are powerful and can be used across the whole website.

This time we are going to append the new contents to navigation by using **.=**
Notice how the </nav> tag is removed from the first $navigation segment, this is because we don't want our heredoc to end before we have checked if a user is logged in.
Please consider the following code:

```
<link rel="stylesheet" property="stylesheet" type="text/css"
href="css/stylesheet.css">

<?php
session_name('Website');
session_start();
$host = "localhost";
$user = "sysadmin";
$pwd = "dbik2015";
$db = "test";
$mysqli = new mysqli($host, $user, $pwd, $db);
$navigation = <<<END
    <nav>
     <a href="index.php">Home</a>
     <a href="about.php">About</a>
    END;

    if (isset($_SESSION['userId']))
    {
     $navigation .= <<<END
     <a href="logout.php">Logout</a>
     Logged in as {$_SESSION['username']}
    END;
    }
    $navigation .= '</nav>';
    ?>
```

Tasks

1. Edit the template.php file and add the PHP code above to it.
2. Using a web browser, enter test again the login page in your web site.

logout.php – step 1

Now we have to create a site that can handle logout requests.
As you already know, session variables are set with the PHP global variable: $_SESSION, and once a session[16] is created, it will last until:

3. You finish it using the commands:
    - session_unset(); - removes all session variables
    - session_destroy(); - destroys the session
    - or, you close the browser.

Alot of times you want to make sure the data is lost so before destroying the session it might be a good idea to write an empty array over the current data.
Please consider the following code:

```
<?php
include('template.php');
$_SESSION = array();
session_destroy();
header("Location:index.php");
?>
```

Tasks

1. Create a file called logout.php at ~/public_html/lab3 directory and add the PHP code above to the file.
2. Using a web browser, enter navigate in your web site. Try logging in and out, also after logging out try going back to see if you really are logged out.
    a. You might need to enter as address
        http://ideweb2.hh.se/~user_name/lab3/login.php

Only one problem left to handle, a login button or menu item!

[16] http://www.w3schools.com/php/php_sessions.asp

template.php – step 5

The template now lets you logout if you are logged in. Now the objective is to add a button so that we can login without changing the URL. Since the template.php code has already a conditional statement to check whether a user is logged in or not, you can add an alternative statement is care the user is not logged in.

Tasks

3. Edit the template.php file and add the login button or option on your own!
   HINT: See how the logout button or option is added.
4. Before continuing the lab, **READ** about two common security vulnerabilities and how to protect your website against them.

- SQL Injection at W3Schools[17]
  - SQL injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input.
  - Injected SQL commands can alter SQL statement and compromise the security of a web application.
- Cross-site scripting (XSS) at W3Schools[18] and Sitepoint[19]
  - Cross-site scripting is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

[17] http://www.w3schools.com/sql/sql_injection.asp
[18] http://www.w3schools.com/php/php_form_validation.asp
[19] http://www.sitepoint.com/php-security-cross-site-scripting-attacks-xss/

register.php – step 1

Now that users can login/out we want to offer the possibility to new users to register.
To do this we have to learn how to insert data into our database. This is done with INSERT INTO statement[20]. The syntax is:

- The first form:
  INSERT INTO table_name
  VALUES (value1,value2,value3,...);

- The second form:
  INSERT INTO table_name (column1,column2,column3,...)
  VALUES (value1,value2,value3,...);

Please consider the following code:

```
<?php
include('template.php');
if (isset($_POST['username']) and isset($_POST['password'])) {
  $query = <<<END
  INSERT INTO users(username,password,email,fname,lname)
    VALUES('{$_POST['username']}','{$_POST['password']}','
    {$_POST['email']}','{$_POST['fname']}','{$_POST['lname']}')
END;
  if ($mysqli->query($query) !== TRUE) {
    die("Could not query database" . $mysqli->errno . " : " . $mysqli-
>error);
    header('Location:index.php');
  }
}
$content = <<<END
<form method="post" action="register.php">
<input type = "text" name="username" placeholder="username" ><br>
<input type="password" name="password" placeholder="password"><br>
<input type="text" name="email" placeholder="email"><br>
<input type="text" name="fname" placeholder="first name"><br>
<input type="text" name="lname" placeholder="last name"><br>
<input type="submit" value="Register">
<input type="Reset" value="reset">
</form>
END;
echo $navigation;
echo $content;
?>
```

Considering the code above:

- Remember that in heredoc you have to use single quotations around the $_POST so that SQL knows it's a string.
- The SQL statement might not correspond to the user table that you created in Part 3.
- Notice that the execution of query($query) is checked in a conditional statement, and if unsuccessful, the die() function is executed.
  - The die()[21] function is used to print a message and exits the current script. This function is an alias of the exit() function.

[20] http://www.w3schools.com/sql/sql_insert.asp
[21] http://www.w3schools.com/php/func_misc_die.asp

- This is normally used on pages where a connection to the database or the execution of a particular statement is vital for the page to work.
  - A register page containing problems, such as database-related issues, is useless so you kill the script execution if a problem happens.
- If a problem related to database happens, the mysqli->errorno returns the corresponding error code and the mysqli->error returns a message to the actual error.

Tasks

1. Create a file called register.php at the `~/public_html/lab3` directory and add the PHP code above to the file.
2. Using a web browser, test the register page

If you try to register a user after coding the page above you will get an error message that looks like:

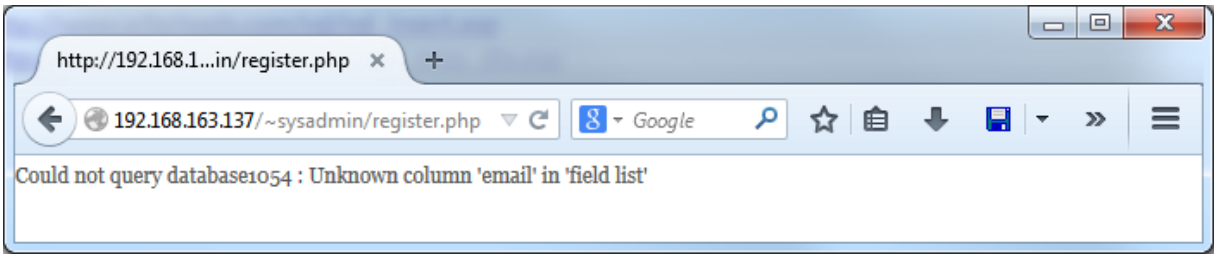


Figure 8. Problem with the user registration web page.

This is because the script is trying to insert values into columns that do not exist in table users.
If you look back at the user table, the columns email, fname and lname do not exist.

Back to phpMyAdmin

The objective now is to use phpMyAdmin and add the missing columns to table users.

Tasks

1. Login again at phpMyAdmin, since the session might have expired.
2. Select the database test on the left side list of databases
3. Select the table users and click at the SQL tab (Figure 13).

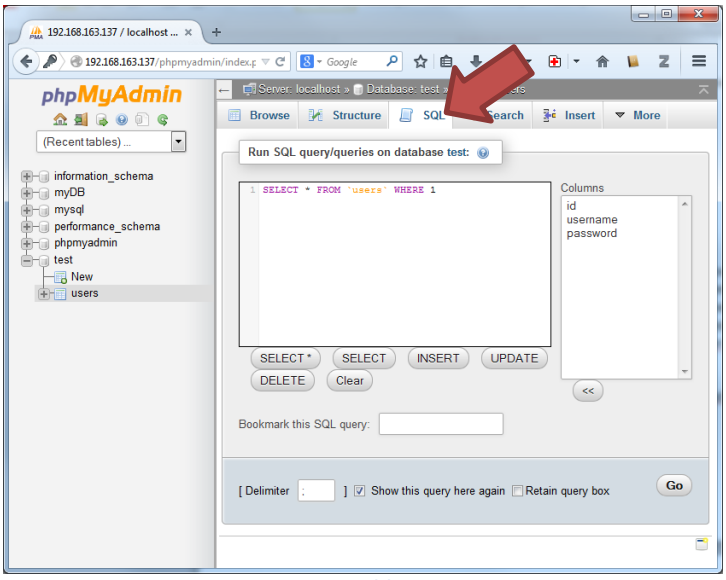


Figure 9. Table users.

In the “Run SQL query/queries on database test” text box, you can input SQL statements, such as for adding columns to a given table.
The ALTER TABLE[22] statement is used to add, delete, or modify columns in an existing table. Syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

However, before adding columns to a table, you have to decide about the datatype of the new columns as well as some values constraints, such as the maximum size for strings.
In this particular case, email ,fname and lname can be of VARCHAR datatype and the size limit can be 30 characters.
Once you added the new columns, the registration page will work and new users can be added to the system.

Tasks

1. Add the 3 new columns to table user using SQL statements in phpMyAdmin.
   HINT: Figure 14. To run the commands press the button Go on the botton right.

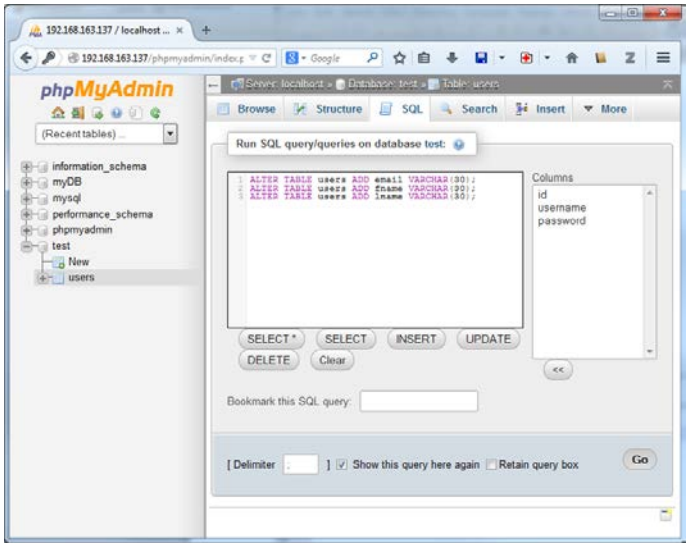


Figure 10. Adding columns to table user using SQL statements.

2. Using the registration page, add a new user.

[22] http://www.w3schools.com/sql/sql_alter.asp

More phpMyAdmin

The objective now is add to the database new relations or tables so your web site website can do more than user's management. Remember, you are creating a web shop.
Because you are going to sell produces, you need to create an extra table in the database, the table products.
For example, a product can have an identifier (id), a name (name), a price (price), a description (description) and a date/time indicating when the product was added to the system (created_at).

Tasks

1. Create a new table called products in the database test and the table must have 5 columns. The table structure is presented in Figure 15.
   Different datatypes are used to each column. To read more about them visit Tutorialspoint on MySQL[23].

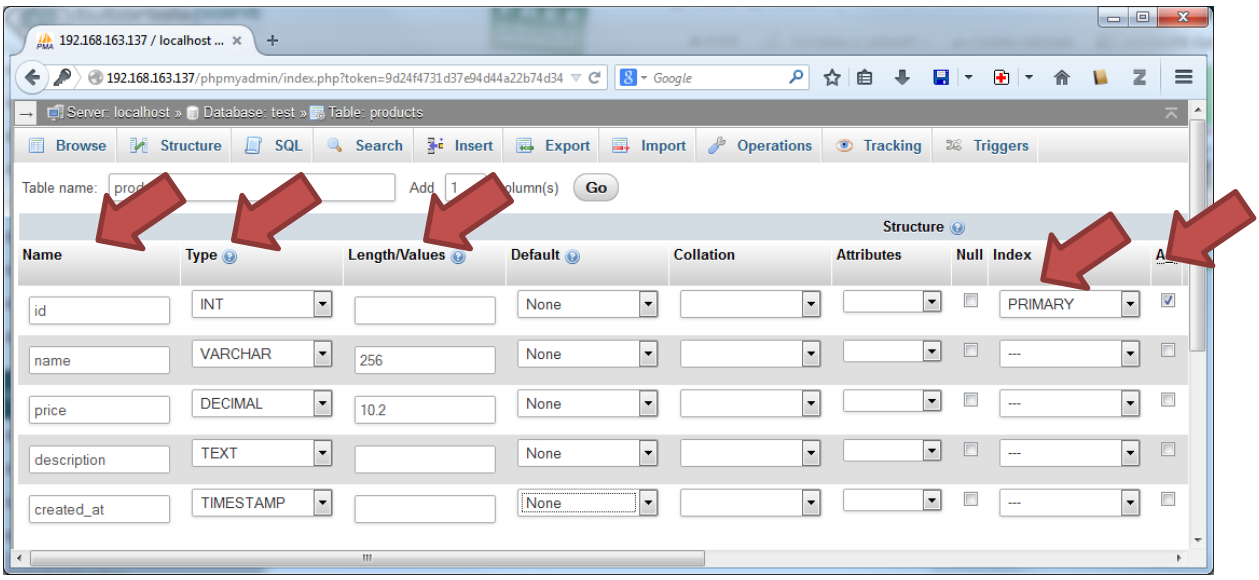


Figure 11. Creating table products.

[23] http://www.tutorialspoint.com/mysql/mysql-data-types.htm

CRUD

Create, Read, Update and Delete are the four ground principle functions for handling data in a database. You have used SELECT(read) and INSERT(create) so far.
A normal usage of PHP is to have separate files for functions, this is so that it's easier to build and have control over your website.

add_product.php

Let's now a new php page to add products to the database. Similarly to the php script for the registration page, you will be using the INSERT INTO SQL statement to add a new product to table products.
Please consider the following code:

```
<?php
include_once('template.php');
if (isset($_POST['name'])) {
  $query = <<<END
   INSERT INTO products(name,price,description)
    VALUES('{$_POST['name']}','{$_POST['price']}','{$_POST['description']
}')
END;
  $mysqli->query($query);
  echo 'Product added to the database!';
}
$content = <<<END
<h1>Add product</h1>
<form method="post" action="add_product.php">
<input type="text" name="name" placeholder="Productname"><br>
<input type="text" name="price" placeholder="Price"><br>
<textarea name="description" placeholder="Description"></textarea><br>
<input type="submit" value="Add product">
<input type="reset" value="reset">
</form>
END;
echo $navigation;
echo $content;
?>
```

Tasks

1. Create a new file called add_product.php at ~/public_html/lab3 directory add the PHP code above to the file.
2. Add menu options for the register.php and add_product.php to the template.php.
3. Once you finish, you the web browser to see how your web site looks like.
4. Also, think about if all features should be available for the users. Perhaps your web site should only offer the option add_product.php to an admin who is logged in!!!

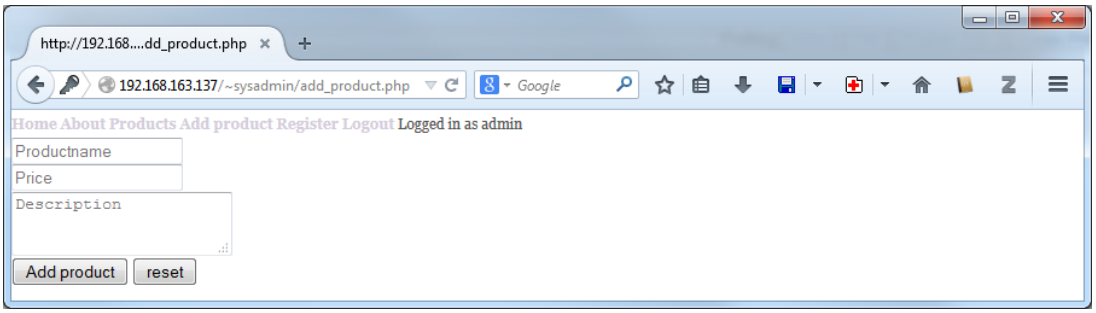


Figure 12. This is how my page looks now. Try using the add product function a few times so that your database holds a few items before proceeding to the next task.

When adding products you should note that there is nothing that prevents you from attacking the web site with SQL-injection techniques. Try adding some SQL code and see how your database treats it. REMEMBER: depending on the SQL statement, you might have to recreate a table!!!!
Later on in this exercise, you will protect your web site against this type of attack.

products.php – step 1

You use the SELECT statement to retrieving data from the database, and later one some script code to show such data.
Now, you are going to create a web page to list all products stored in the database. Newly added products must be in the top of this list.
This can be done with the SQL keyword ORDER BY, and for this particular case, you will ORDER BY the date/time column of the product table.
To list all stored products, you will need a loop control structure, such as FOR and WHILE.
Please consider the following code:

```
<?php
require_once('template.php');
$content = '<h1>Products</h1>';
$query   = <<<END
SELECT * FROM products
ORDER BY created_at DESC
END;
$res     = $mysqli->query($query);
if ($res->num_rows > 0) {
  while ($row = $res->fetch_object()) {
    $content .= <<<END
       {$row->name}|
        {$row->price}
        <hr>
END;
  }
}
echo $navigation;
echo $content;
?>
```

Tasks

- Add a new file called products.php to ~/public_html/lab3.
- Add the PHP code above to the file.
- Use a web browser to test the page.
- Try changing DESC to ASC, what happens and why?

products.php – step 2

Now, let's add some php code to the product.php file. The idea is create a hyperlink for each product in the list, so when the user a link called "Description", a web page will be presented with the details of that particular product.
For that, you will use the HTTP method GET.
READ about POST and GET at http://www.w3schools.com/tags/ref_httpmethods.asp.

GET requests data from a specified resource and sends data via the URL, using this syntax:

URL?variablename=value

The GET methods will be used to the product "id" when the user clicks on the description link.
Please consider the following code:

```
<?php
require_once('template.php');
$content = '<h1>Products</h1>';
$query   = <<<END
SELECT * FROM products
ORDER BY created_at DESC
END;
$res     = $mysqli->query($query);
if ($res->num_rows > 0) {
  while ($row = $res->fetch_object()) {
    $content .= <<<END
       {$row->name}|
        {$row->price}
        <br>
        <a href="product_details.php?id={$row->id}">Description</a><br>
        <hr>
END;
  }
}
echo $navigation;
echo $content;
?>
```

Tasks

- Edit the products.php file and modify its code according to the PHP code above.
- Use a web browser to test the page.
- Click at the "Description" hyperlink.
    - If you select the first product, the URL in the browser should be similar to: http://ideweb2.hh.se/~user_name/lab3/products_details.php?id=3

product_details.php – step 1

The task now is to create a page that receives the produc "id" using the GET method and displays the details about the select product.
Using GET method to retrieve values is very similar to retrieving values from the POST method.
"Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special."[24]

Note that only the "id" is sent, as you can see with ?id=1. Sometimes you might want to send multiple values. In that case, you separate the variables with an & sign. So ?id=1&price=10 would be two values and both could be retrieved with
$_GET['id'] and $_GET['price']

Please consider the following code:

```
<?php
include_once('template.php');
$content = '<h1>Product Details</h1>';
if (isset($_GET['id'])) {
  $query = <<<END
   SELECT * FROM products
    WHERE id = '{$_GET['id']}'
END;
  $res = $mysqli-
>query($query) or die("Could not query database" . $mysqli-
>errno . " : " . $mysqli->error);
  if ($res->num_rows > 0) {
    $row     = $res->fetch_object();
    $content = <<<END
   Product id: {$row->id}<br>
    Title: {$row->name}<br>
    Price: {$row->price}<br>
    Description: {$row->description}<br>
    Date: {$row->created_at}
END;
  }
}
echo $navigation;
echo $content;
?>
```

Task

1. Create file called product_details.php at ~/public_html/lab3 and add the PHP code above to the file.
2. Test the web site by selecting products.

HINT: Do not forget to update your template.php so you can access every page in the web site.

[24] http://www.w3schools.com/php/php_forms.asp

products.php – step 3

The next step is to add to the products page other options than description, for example, the options of removing and modifying products.
Please consider the following code:

```
<?php
require_once('template.php');
$content = '<h1>Products</h1>';
$query   = <<<END
SELECT * FROM products
ORDER BY created_at DESC
END;
$res     = $mysqli->query($query);
if ($res->num_rows > 0) {
  while ($row = $res->fetch_object()) {
    $content .= <<<END
      {$row->name}|
        {$row->price}<br>
        <a href="product_details.php?id={$row->id}">Description</a>|
        <a href="delete_product.php?id={$row-
>id}" onclick="return confirm('Are you sure?')">
        Remove product</a>|
        <a href="edit_product.php?id={$row->id}">Edit product</a><br>
        <hr>
END;
  }
}
echo $navigation;
echo $content;
?>
```

Tasks

1. Edit the products.php file according to the PHP code above.
2. Identify the main differences.
3. Note also the javascript code on the delete hyperlink, which is used to misclicking accidents.
4. Test your web site.

delete_product.php – step 1

To remove a product from the database, you will use DELETE SQL statement. However, it is important to think about security aspects first. If you use the GET method to remove products, people could remove products deliberately using …/delete_product.php?id=...
To prevent this, you can use again the isset() function to check if the is logged in.
Please consider the following code:

```
<?php
include_once('template.php');
if (isset($_GET['id']) and isset($_SESSION['userId'])) {
  $query = <<<END
   DELETE FROM products
    WHERE id = '{$_GET['id']}'
END;
  $mysqli->query($query);
  header('Location:products.php');
}
echo $navigation;
?>
```

Tasks

1. Create a file called delete_product.php at ~/public_html/lab3 and add the PHP code above to the file.
2. Test the new functionality.

products.php – step 4

Another method to prevent accidental deletes is to remove the option “Remove product” if the user is not logged in.

Tasks

1. Edit the file products.php to only show the “Remove product” and “Edit product” options when the user is logged in.
2. Also change the template.php file so the registration page (register.php) is only visible to users logged in.
3. Test your modifications

edit_product.php – step 1

Now, the next step is to create a page which will allow users to update products.
Please consider the following code:

```
<?php
include_once('template.php');
$content = 'Edit Product';
if (isset($_GET['id'])) {
  $mysqli->query($query);
}
$query = <<<END
   SELECT * FROM products
    WHERE id = '{$_GET['id']}'
END;
$res   = $mysqli->query($query);
if ($res->num_rows > 0) {
  $row     = $res->fetch_object();
  $content = <<<END
   <form method="post" action="edit_product.php?id={$row->id}">
    <input type="text" name="name" value="{$row->name}"><br>
    <input type="text" name="price" value="{$row->price}"><br>
    <textarea name="description">{$row->description}</textarea><br>
    <input type="submit" value="save">
    </form>
END;
}
echo $navigation;
echo $content;
?>
```

Task

1. Create a file edit_product.php and add the PHP code above to the file.
    a. The edit_product.php file represents the web page used to update product's details. It first selects all the information about the selected product and then add a form with corresponding input HTML elements.
    b. The form is set POST as method and the action to itself (edit_products.php), but keeping the ID ?id={$row->id} .
2. If you test the page, you will notice that nothing happens. This is because have no code in the page that handles the incoming post values.

edit_product.php – step 2

To update a row in a table in the database, you will use the UPDATE SQL statement[25], which has the following syntax:

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

In this example, you will update a particular product in the products table. To do so, you need to identify the product that modifications should apply. The product identifier as well as the new values for the product are containing in the variable $_POST array.
Now, the next step is to create a page which will allow users to update products. Please consider the following code:

```
<?php
include_once('template.php');
$content = 'Edit Product';
if (isset($_GET['id'])) {
  if (isset($_POST['name'])) {
    $query = <<<END
       UPDATE products
        SET name = '{$_POST['name']}',
        price = '{$_POST['price']}',
        description = '{$_POST['description']}'
        WHERE id = '{$_GET['id']}'
END;
    $mysqli->query($query);
  }
  $query = <<<END
   SELECT * FROM products
    WHERE id = '{$_GET['id']}'
END;
  $res   = $mysqli->query($query);
  if ($res->num_rows > 0) {
    $row     = $res->fetch_object();
    $content = <<<END
   <form method="post" action="edit_product.php?id={$row->id}">
    <input type="text" name="name" value="{$row->name}"><br>
    <input type="text" name="price" value="{$row->price}"><br>
    <textarea name="description">{$row->description}</textarea><br>
    <input type="submit" value="save">
    </form>
END;
  }
}
echo $navigation;
echo $content;
?>
```

Task

1. Edit the edit_product.php file according to PHP code above.
   a. Note that the update query is written in the code, and consequently executed, before the select statement used to create the form. This is because the web page must display the most recent data from the database. So always do the update first.
2. Test the new functionality.

[25] http://www.w3schools.com/sql/sql_update.asp

Part 5 - Adding security to the website

The observant student might have noted that, stating from CRUD session, the use of the *include*() function was changes to *include_once(),* for example, from include('template.php'); to *include_once*('template.php');
This is not about security but about redundancy. Include will try to include the page, and if it does not succeed it will keep going like nothing happened. The reason to use include_once instead is because this prevents the same page from being loaded twice! Think about if template.php includes products.php, which then includes another page, and that page includes products.php again. Now it may load the same page a lot of time and cost resources and maybe if really unlucky even crash. include_once will only load a page once, if it recurs again it will be skipped.

The next thing to note, and change in the web site, is that when you have a critical page that needs to be loaded, you should use the *require* function instead of *include*. Let's say our template.php is critical to the products.php page since it keeps the connection with the database.
It should use require_once instead of include. Require will not load the website if it can't reach the resources that you try to include (see code below).

```
<?php
require_once('template.php');
$content = '<h1>Products</h1>';
$query   = <<<END
SELECT * FROM products
ORDER BY created_at DESC
END;
$res     = $mysqli->query($query);
if ($res->num_rows > 0) {
  while ($row = $res->fetch_object()) {
    $content .= <<<END
       {$row->name}|
        {$row->price}
        <hr>
END;
  }
}
echo $navigation;
echo $content;
?>
```

Note also that along the exercises, users had the opportunity to insert, delete and modify data in the database freely. No validation was performed, for example, while inserting products.
To secure your web site against some forms of attacks, such as XSS attacks and SQL injections, there are a few security mechanism and practices that you can to develop more secure websites.
If you haven't read about XSS attacks and SQL injections as instructed on page 25, now is the time for it, so you can explain what they are and how they work.

Let's start hardening our website!

To protect against SQL injection we have to validate that the data input is actually what we want it to be.
Every time you allow a user to input data to your website that is going to be sent to the database make sure to use:

$mysqli->real_escape_string(variable);

"The mysqli_real_escape_string() function escapes any special characters in a string for use in an SQL statement."[26]

Task

1. Apply the $mysqli->real_escape_string to the Login page.

```
<?php
include('template.php');
if (isset($_POST['username']) and isset($_POST['password'])) {
  $name   = $mysqli->real_escape_string($_POST['username']);
  $pwd    = $mysqli->real_escape_string($_POST['password']);
  $query  = <<<END
   SELECT username, password, id FROM users
    WHERE username = '{$name}'
    AND password = '{$pwd}'
END;
  $result = $mysqli->query($query);
  if ($result->num_rows > 0) {
    $row                  = $result->fetch_object();
    $_SESSION["username"] = $row->username;
    $_SESSION["userId"]   = $row->id;
    header("Location:index.php");
  } else {
    echo "Wrong username or password. Try again";
  }
}
$content = <<<END
<form action="login.php" method="post">
<input type="text" name="username" placeholder="username">
<input type="password" name="password" placeholder="password">
<input type="submit" value="Login">
</form>
END;
echo $navigation;
echo $content;
?>
```

Note that two variables are created, $name and $pwd respectively, and the corresponding content from the variable $_POST array is used as parameters to the $mysqli->real_escape_string, which results are assigned accordingly to the $name and $pwd variables.

Task

1. Revise the web pages in the web site and check if they are susceptible to SQL injection. If so, protect these pages. Some pages that you have to protect are are register.php and add_product.php, among others.

[26] http://www.w3schools.com/php/func_mysqli_real_escape_string.asp

Lastly you have to protect your web site XSS attacks. This is done by the use of htmlspecialchars() function[27]. What this function does is that it takes the strings and converts all special characters into HTML code. This way you can't send in lines of codes as values in the forms.

Task

1. Go back and have a look at send.php and about.php files. As it is, the code allows users to whatever pleases them. To try an example of XSS attack, enter "<script>alert('xss')</script>" in the name or message input of the web page.
    a. A popup message is harmless, but more damaging things can be done.

By using the htmlspecialchars() functions, you can prevent that from happening. The same concept as in sql protection applied. First you capture the input, pass it as argument to the htmlspecialchars() functions, the use its return value.

```
<?php
include('template.php');
if (isset($_POST)) {
  $name    = htmlspecialchars($_POST["name"]);
  $msg     = htmlspecialchars($_POST["msg"]);
  $content = <<<END
       <h3>Message was sent:</h3>
        Name: {$name}
        <br>
        Message: {$msg}
END;
  $to      = "youremail@student.hh.se";
  $subject = "Test-mail";
  $msg     = $_POST["msg"];
  $headers = "From: " . $_POST["name"];
  mail($to, $subject, $msg, $headers);
}
echo $navigation;
echo $content;
?>
```

Task

1. Revise the web pages in the web site and check if they are susceptible to this sort of attack. Just remember, anytime a user can input data, that place needs to be defended!

Final Task

1. Using a browser, visit the web site. Using the mouse, right click anywhere inside the web page and select to view the source code.
2. Copy the web page source code, and using http://validator.w3.org/, check whether the source code is valid or not. Try home and products pages.
3. Perhaps the source code is not a valid Web document in HTML. How can you fix it?

Congratulations, you have now completed the exercises for Laboration 3.

[27] http://www.w3schools.com/php/func_string_htmlspecialchars.asp