Login and Protect

# Objectives

* Simple authentication - login and logout.
* Allow access of web pages by authenticated users only.

## 1\_1

## Preparation

1. You must have completed the previous lab “Database write” to continue work on this lab. Otherwise, copy the folder contact\ from solution of the previous lab to folder C:\wamp64\www\.
   1. We assume you have installed WAMP in C:\wamp64\.
   2. If you have installed WAMP in another folder, please change accordingly.

## 1\_2

## Create a new table login in database sis

1. Login to phpMyAdmin, select database sis.
2. Click the “Import” at the top row of menu item of the right frame.

|  |
| --- |
|  |

Figure 1: Select menu item “Import” at the top row of left frame.

1. Click the button “Choose File” and select user.sql from the given resource.
2. Click button “Go”.
3. When done successfully, a success message “Import has been successfully finished, …” (or equivalent) is displayed. There should be table user with the following four records.
   1. The actual password is shown below but it is not an actual column in the database.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **username** | **pwd\_hash** | **Active** | **usertype** | **Actual password**  **(not a database field)** | | zack | Hash value of the actual password; not shown here | y | admin | zackp | | yew | y | normal | yewp | | wong | n | admin | wongp | | tan | n | normal | tanp | |

Figure 2 Four records in table user.

## 1\_3

## User and UserDAO

1. Copy the given User.php and UserDAO.php to your web folder contact\model\.
2. Open the 2 files to understand what attributes and methods are available.

## 2\_1

## Login

1. Create login.php that has a form that prompts for username and password and has a button “Login” in your folder contact\
   1. The password field should just display asterisks instead of the actual password typed in

|  |
| --- |
|  |

Figure 3: Login.php with username and password field

1. The form posts to process\_login.php.
2. Create process\_login.php that retrieves the User object for the submitted username via UserDAO.
3. If the username is valid (i.e. a user is found) and the submitted password is correct, redirect to index.php.
   1. Use the PHP predefined function password\_verify($password, $hash) to check the submitted password against the User object’s passwordHash attribute.
4. If password is incorrect, redirect back to login.php which displays an error message “Incorrect username or password”.

|  |
| --- |
|  |

Figure 4: When login fails, login.php displays an error message.

## 3\_1

## “Remembering” the login user

1. Edit process\_login.php such that when username and password are correct, set the User object to session variable “login” before redirecting to index.php.

|  |
| --- |
| // correct username and password  // set User object to session  $\_SESSION['login'] = $user;    // redirect to index.php  header("Location: index.php");  exit(); |

Figure 5: When username and password are correct, set the User object to session.

1. Update index.php to
   1. Retrieve User object stored in session variable “login”.
   2. Display “Welcome <username>!” (heading1) at the top of the page.
   3. Replace <username> by the User object’s username.

|  |
| --- |
|  |

Figure 6: index.php displays a welcome message.

## 4\_1

## Logout

1. Update index.php to add a hyperlink “logout” which points to logout.php.

|  |
| --- |
|  |

Figure 7: index.php has a new “logout” hyperlink

1. Create logout.php that does the following:
   1. Remove the session variable login.
      1. Note: logout.php needs common.php which will start the session before you remove any session variable.
   2. Redirect to login.php.
2. Test clicking the hyperlink “logout” and it should bring you to login.php.

## 4\_2

## Protected pages

1. After a successful logout and without login, if you go to index.php by typing directly to browser’s address bar, you notice PHP error messages:
   1. “Undefined index: login in …”
   2. “Trying to get property of non-object in …”.

|  |
| --- |
|  |

Figure 8: PHP errors if you do direct to index.php without a successful login.

1. These errors are due to the code that retrieve session variable login for the welcome message.
2. We want to prevent unauthenticated users from accessing index.php. In fact, we want to protect all the create.php, edit.php and so on too.
3. To do so, create a protect.php with the following code:

|  |
| --- |
| <?php  require\_once 'common.php';  // Is the session variable “login” available  if ( !isset($\_SESSION['login'])) {  // No session variable "login" implies no login user\_error    // redirect to login page  header("Location: login.php");  exit();  } |

Figure 9: Code for protect.php

1. To protect index.php, we insert “require\_once 'protect.php';” after the line for common.php.

|  |
| --- |
| <?php  require\_once 'common.php'; require\_once 'protect.php'; // protect this page |

Figure 10: Add protect.php to index.php

Do the same for the other web pages that you want to protect

1. Try accessing index.php directly without a valid login now and you should be brought to login.php.

## 5\_1

## Create a login user

1. Create createUser.php that contains a form that allows user to create a new login account.
2. The form has the following fields:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Form field** | **Description** | **Validation error message (if any)** | | Username |  | * Username cannot be empty or blank. * This is an existing username. | | Password | The value should be masked. | Password cannot be empty or blank. | | Confirm password | Its value should be same (case sensitive) as “Password” field. | Password must be the same as  'Confirm password'. | | Active | A single checkbox which is checked by default |  | | User Type | Radio button list to select “Normal” or “Admin”. “Normal” is selected by default. |  | | Submit | Button to submit the form |  | |

Figure 11: Form fields in createUser.php

|  |
| --- |
|  |

Figure 12: Screenshot of createUser.php

1. The form posts to process\_createUser.php. The page process\_createUser.php does server-side validation as described in the table of form fields above.
   1. You will need to use class UserDAO to check if the username exists.
2. If there is any error, send back to createUser.php to display the error messages.
3. If there is no error, page process\_createUser.php creates a new User object.
   1. The password hash is generated using function password\_hash($password, $algo).
   2. $algo is the formula to be used to generate the hash. We are going to use CRYPT\_BLOWFISH algorithm. To do so, pass predefined constant PASSWORD\_BCRYPT[[1]](#footnote-1) to the parameter $algo.

|  |
| --- |
| $passwordHash = password\_hash( $password, PASSWORD\_BCRYPT ); |

Figure 11: Code to generate password hash

1. Use class UserDAO’s create() function to insert the new login user to database.
2. Database table “user” has columns “active” and “usertype” to store values of active and usertype fields.   
   Active users have the value “y” while non-active users have the value “n” stored in the column “active”. For user types “Normal” and “Admin”, values of “normal” and “admin” respectively is stored in the column “usertype”.
3. Create a few new users and test by logging with the new user-credentials.
   1. As active and usertype fields are not used yet, check the database via PhpMyAdmin to check that the two fields are saved into the database correctly

## 6\_1

## Same header for all web pages

To have a consistent user experience, all web pages should have the same header.

1. Copy the given header.php from resource to folder contact\. Read header.php to understand its code.
   1. It displays a welcome message.
   2. It has the following list of links separated by the ‘|’ character on the second line
      1. “Home” – index.php
      2. “New contact” – create.php
      3. “New user” – createUser.php
      4. “Logout” – logout.php
2. Update index.php, create.php, edit.php and delete.php
   1. Add this line of code at the start of <body> tag.

|  |
| --- |
| …  <body>  **<?php require "header.php" ?>**  … |

Figure 12: Add code to have the same header for all web pages.

* 1. Remove duplicate links

1. Test by checking all the web pages have the same header.

|  |
| --- |
|  |

Figure 13: create.php has the same header as the other web pages.

## 6\_2

## Question

1. What is the difference between PHP keyword “require\_once” and “require”?
   1. Why do we use “require” keyword for header? What if we want to repeat the header.php at the bottom of a web page?
   2. Why do we use “require\_once” keyword for class-based PHP files?

## 7\_1

## More practice

## 7\_2

## Inactive user login always fails

1. Update login such that inactive user (User object’s attribute active is ‘n’) are sent back to login.php regardless of the password.
2. Display the same error message “Incorrect username or password

## 7\_3

## Different pages for admin versus normal users

1. Update header.php such that the “New user” hyperlink (to createUser.php) is not shown for a normal user.
   1. Admin user is indicated by User object’s attribute usertype with value ‘admin’.
   2. Normal user’s attribute usertype is ‘normal’.
2. After a successful login, admin user has “New user” link in the menu while normal user does not have that link.
3. Prevent access to unauthorized pages
   1. If a normal user types createUser.php directly into the browser’s address bar, he should be redirect to index.php.
   2. You may want to create a protect-admin.php to protect createUser.php.

# *References*

1. <http://php.net/manual/en/function.require-once.php>
2. <http://php.net/manual/en/function.include.php>
3. Password hashing:
   1. <http://whatis.techtarget.com/search/query?q=hashing>
   2. Password hashing in PHP
   3. <http://php.net/manual/en/faq.passwords.php>
   4. <http://php.net/manual/en/function.password-hash.php>
      1. Constants for hashing algorithms: <http://php.net/manual/en/password.constants.php>
   5. <http://php.net/manual/en/function.password-verify.php>

1. <http://php.net/manual/en/password.constants.php> [↑](#footnote-ref-1)