Assignment 2

Applikationer för internet, ID1354

Max Körlinge, korlinge@kth.se

November 12, 2017

Contents

1 Introduction

This assignment concerns working on the PHP server to implement user authentication, users being able to write comments on recipes, and users being able to delete those comments. Optional tasks were being able to register new users on the site, and storing all recipe data in XML format. I chose to do all tasks (optional and mandatory). I completed the assignment alone.

2 Literature Study

To complete the tasks I first studied the course lecture notes on the PHP language, PHP for the web server, and XML.

3 Method

3.1 Task 1

To implement authentication I created a new login page with a form that submits to a PHP script. The script uses the PDO library to make a connection to the mySQL database, which I setup, to check wheither there is a matching username - password entry in the database using a SQL query. I use the `$_SESSION` variable to store the username and implemented a check in the header included in every page to see if the user is logged in or not. To clarify for the user that there is loading going, on a redirection page was implemented, telling the user what is happening.

3.2 Task 2

The commenting functionality was implemented by including a form at the bottom of the page if the user is logged in (by using the function developed for task 1). The form submits using a PHP script that uses the `$_POST` and `$_SESSION` variables to find the comment submitted and the username submitting it, and uses a PDO connection to the

mySQL database to store it. While developing this it was also necessery to rework the recipe sites to load comments from the database, instead of hardcoding them as before.

3.3 Task 3

I used the method described in the lecture notes on using hidden input fields in forms to implement being able to delete a comment. The hidden value field is given the `recipe_id` stored in the mySQL database, and then the form submits a script which simply querys the database to delete this entry from the `comments` table. While generating the list of comments to the page from the database, the program checks wheither the processed comment was written by the currently logged in user, and then places the delete form next to that comment. No delete button is shown for comments that are not written by the currently logged in user.

3.4 Optional Task 1

To register new users I added a new page for registrations which is very similar to the login page, to keep the design coherent. It uses a similar method to the login page: submitting a form to a PHP script, and the PHP script extracting the new username and password, checking the database for wheither this username already exists (if so returning to the registration page), and if it does not, adding the username and password to the database. The user is then able to log in with the new account.

3.5 Optional Task 2

To store the recipes in XML I used the provided mycookbook template to convert my recipes stored in HTML to properly matching XML tags for this template. I chose to put all recipes in the same XML file, to preserve an order which also could be used to find a matching recipe ID for the database. For example, the first recipe in the XML file has `recipe_id 1` in the database. I used SimpleXML to parse the XML, and rewrote the recipe sites to call a PHP function `RecipeSite($name)` where you can input the name of a recipe in the database, and it will be output the respective XML contents correctly to the site.

4 Result

The git repository can be found at https://github.com/fongie/TastyRecipes/tree/assignment2

4.1 Task 1

A login screen is provided (Fig. 4.1) and a PHP script checks the mySQL database for a matching username and password combination, as seen in Fig 4.2. The query will always yield 1 or 0 since usernames are unique in the database, so if it gives a result higher than zero then the user is authenticated. To give user feedback on system status, it is clearly stated that the system is logging you in, until the query is finished. Also, when you are logged in, it clearly says so at the top right of the screen. This is persistent as long as the session is active, since the logged-in script is present in the header of each page.

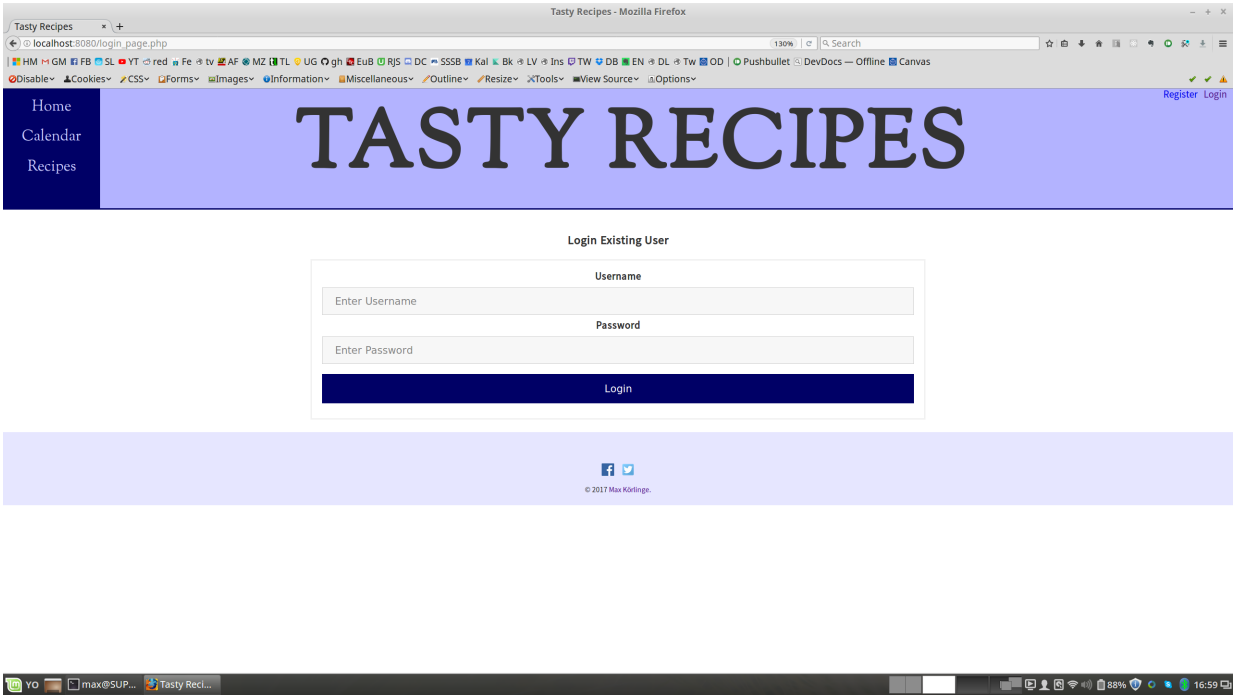

Figure 4.1: The login page.

4.2 Task 2

Comments are fetched from the database and rendered into each recipe page. If the user is logged in, there is also a form at the bottom of the page where the user can enter a comment. The comment is stored in the database together with the user ID who posted it, and the page reloads to show the new comment. The page is shown in Figure 4.3.

The design of the form is in line with the rest of the website, and when a comment is being posted the user is shown a message, so that they know what is going on, to be in line with the heuristics for user design.

```php
# what was just posted by form
$username = $_POST["uname"];
$password = $_POST["psw"];

# mysql connection
$pdo = new pdo("mysql:host=localhost;dbname=tasty_recipes;charset=utf8mb4", "tasty_user", "tasty");

# returns above 0 if found a matching username/pass combination
$query = "SELECT COUNT(*) FROM user_accounts WHERE username='$username' AND password='$password'";

$result = $pdo->query($query);
if ($result->fetchColumn() > 0) {
    #login

    echo "Logging in as $username...";
```

Figure 4.2: Part of the PHP script executing on submitting login form

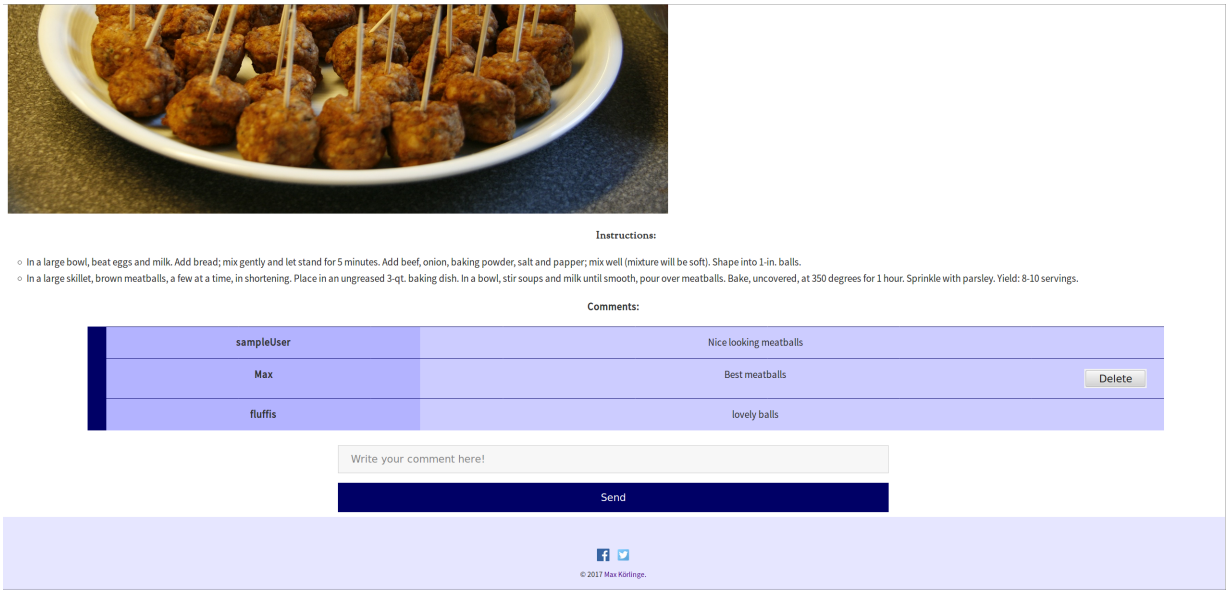

Figure 4.3: Comments on the website, showing the form to enter new comment and the option to delete your own comments.

4.3 Task 3

As shown in Figure 4.3 there is an option to delete one's own comments. As the `RecipeSite` function renders all comments from the database, it adds the delete button to comments that have been posted by the same user that is currently logged in (Figure 4.4).

```
63
 1      # While there are rows left in the query, fetch them as array, associated by the name of the column in the database
 2      $row = $res->fetch(PDO::FETCH_ASSOC);
 3      while ($row) {
 4          $username = $row["username"];
 5          $comment = $row["comment"];
 6          $commentID = $row["comment_id"];
 7          echo "
 8              <li>
 9                  <div class='comment'>
10                      <div class='username-div'>
11                          <p><strong>$username</strong></p>
12                      </div>
13                      <div class='comment-div'>
14                          <p>$comment</p>
15                      ";
16          #add delete button on comments written by the currently logged in user
17          if ($username === $_SESSION["uname"]) {
18              echo '<div class="delete-div">
19                      <form action="/actions/delete_comment.php" method="post">
20                          <input type="hidden" value="'.$commentID.'" name="commentID">
21                          <input type="submit" value="Delete">
22                      </form>
23                  </div>';
24          }
25
26          echo "</div>
27                  </div>
28              </li>
29              ";
30          $row = $res->fetch(PDO::FETCH_ASSOC);
31      }
32      echo "</ul>";
```

Figure 4.4: Code that adds the delete button.

4.4 Optional Task 1

New users are registered on a registration page very similar to the login page shown in Figure 4.1, only substituting the headline and the input text telling the users to write their own new username and password. If the username does not already exist, it is added to the database by a PHP script using a SQL query shown in Figure 4.5. While the request is being processed, the user is shown a message, to clearly state the status of the system.

4.5 Optional Task 2

Recipes are stored in a single XML file, one after another. It uses the mycookbook template provided in the assignment. The XML is loaded using SimpleXML parser, where you then easily can pick out the values to the keys you want, and render them into the page. This is done in a function called `RecipeSite($name)` which is called in each recipe site. You can see the part where the XML is parsed to PHP variables in Figure 4.6.

```php
<?php

session_start();

# what was just posted by form
$username = $_POST["uname"];
$password = $_POST["psw"];

# mysql connection
$pdo = new pdo("mysql:host=localhost;dbname=tasty_recipes;charset=utf8mb4", "tasty_user", "tasty");

# if username exists, do not execute
$query = "SELECT COUNT(*) FROM user_accounts WHERE username='$username'";
$result = $pdo->query($query);
require_once $_SERVER['DOCUMENT_ROOT'].'/actions/redirects.php';
if ($result->fetchColumn() > 0) {
    echo 'Username already exists!! Please try again. Redirecting in 3 seconds..';
    sleep(5);
    redirect_to('/registration_page.php');
} else {

    $insertAccount = "INSERT INTO user_accounts(username, password) VALUES ('$username', '$password');";
    $pdo->query($insertAccount);

    echo 'Registering your account, please wait... You will be redirected to the login page when finished';
    redirect_to('/login_page.php');
}

?>
```

Figure 4.5: The code that adds the new user to the database.

```php
function RecipeSite($name) {

    # Load XML and mysql connection
    $xml = simplexml_load_file($_SERVER['DOCUMENT_ROOT'].'/recipes/xml/recipes.xml');
    //print_r($xml);
    $pdo = new pdo("mysql:host=localhost;dbname=tasty_recipes;charset=utf8mb4", "tasty_user", "tasty");

    # Find current recipe ID. This ID MUST be the same position as the recipes appear in the xml recipes.xml
    $query = 'SELECT id FROM recipes WHERE name="'. $name .'"';
    $res = $pdo->query($query);
    $recipeID = $res->fetchColumn() - 1;

    # Pick out the XML tags we want and put them into variables ready to use
    $title = $xml->recipe[$recipeID]->title;
    $image = $xml->recipe[$recipeID]->imagepath;
    $ingredients = $xml->recipe[$recipeID]->ingredient->li;
    $instructions = $xml->recipe[$recipeID]->recipetext->li;

```

Figure 4.6: The RecipeSite function which uses XML parsing to find recipe data.

5 Discussion

I completed all tasks for this assignment, as shown in the Result chapter above. A requirement throughout the tasks was to keep the five basic heuristics for user interface design fulfilled. In seminar one I found that some of them were hard to apply to those tasks. One of those was the one stating that the user should be shown the status of the system, but this time it has been important to have that in mind since there is more

going on server-side. I have solved this design problem by showing the user that they are logged in in the header of the page, and by echoing a message to the user telling them that a request is being processed every time a connection to the database is being made and some loading time could occur, such as when the user logs in, registers, and posts or deletes a comment.

When authenticating the user I decided to use the `$_SESSION` variable to make the login persist across pages and over the whole browser session. I am aware that you could also accomplish this by using browser cookies, and then you could also have the login persist even after closing down and reopening the browser. I think both ways have merits and accomplish the given task, and my decision to use this method was circumstantial. It would not be difficult to change to using cookies if required at some point.

When dealing with comments on the recipe pages I moved big parts of the recipe pages to PHP functions that dynamically render the data respective to each recipe. I still kept the two pages for the recipes as files in the project, and an improvement to the project would be to render each page according to how many recipes there are in the database, eliminating the need for one PHP file for each recipe. I decided not to do this because of time constraints and because it was not mentioned in the requirements. I did however fulfill the requirements about posting and deleting comments for logged-in users.

I decided early on to use a mySQL database because I wanted to learn how to do it, and also because I know that it is a task in a future assignment. I am also currently studying a course on databases. Because of implementing a database, I think some tasks were easily implemented once that structure was in place and I knew how to interact with it using PHP. For example, deleting comments was as easy as a simple `if` clause in the code, together with a one-line SQL query. Also, registering new users was just a short query to insert the data after checking for uniqueness.

Working with XML was not so difficult since it is so similar to HTML, and the SimpleXML parser made it possible to retrieve values through keys corresponding to the XML tags. I used the mycookbook template, but in the end I could remove a majority of the tags and stick with a few ones that seemed appropriate, such as imageURL, recipeText and ingredients. Since no recipe data was then present in the HTML or PHP code, that task was fulfilled as specified.

6 Comments About the Course

I spent about 30 hours on this assignment, finding it to be quite a large task despite having fiddled around a little in the web world before. It was interesting to finally find out how server side PHP code works.

I found that working with the code from the first assignment as a starting point lead to very unstructured code at the end of the second, with mixed HTML and PHP code everywhere, despite trying to follow the lecture notes on architecture. I look forward to

clean this mess up in the next assignment - since I believe this assignment did not have requirements on code structure.