COSC 360

Forum Page Developer's Guide

Zikora Dozie (17514019)
Diego Sebastian Montes (78738853)

Displaying Posts

Posts are retrieved from the database by using the WHERE operator to find the posts that are assigned to a specific board. These are then stored in an array. A while loop is used to iterate through the array and pull each post individually alongside all of its information. Each time the loop iterates, a different <article> is created with the post's information. All of the posts are inside a scrollable container. Each post is created with a like button and a link to redirect the user to that specific post.

There is an SQL table that links two foreign keys, USER ID with a POST ID, for every post that user has liked. This way, the page is able to retrieve quickly whether a user has liked a post. If the user has already liked a post, the like button will be displayed red ('liked') instead of empty ('unliked'). This process occurs when populating the list of posts as well.

Whenever the user may want to sort the posts by Date (most recent) or Likes (most liked) a POST request is made which reloads the page with a POST value called sort_value which tells the page which value to sort by. Then a u_sort function is used to sort the array. There is a default value (most liked) if there exists no POST request.

Liking Posts and Comments

Each like button (comment and posts) is a <button>. Through JQuery and JavaScript there exists an onClick function for each like button. The onClick function will change the image of the liked button to change between a red and an empty heart to showcase it being liked. The onClick function will also make an ajax call which will run a .php with an SQL query. The sql query will run simultaneously changing the database informing it of which user has liked which post/comment. Therefore, if the page is refreshed the 'liked' button will remain because this information is already reflected in the database.

Each comment and post has a value attribute in its html which corresponds to its POST ID or COMMENT ID. This way, the JavaScript function (onClick) can gather which post or comment is being liked when making the ajax php call through POST. The number of likes is retrieved from the innerHTML and then changed on the SQL through a query. As mentioned before, the foreign keys of USER ID and COMMENTID/POSTID are linked in a likedBy table.

If a user has not logged in, any likes will not be reflected in the database. Moreover, the visual change in the like button is done asynchronous, without refreshing the page. The change to the database is done so as well.

Viewing Posts

To redirect a user to a post we use the GET method and have UNIQUE IDs for each post by changing the links respectively: "<a href='post.php?post=POSTID'>". Therefore each post has

its own unique page which is dynamically built. A query is made to the database requesting the post's information and then it is echoed unto the HTML.

Searching Posts

Searching posts is done similar to how they are shown in the home page. A query is made to the database with the LIKE operator. An array is built from the query and then the information is displayed similar to the home page.

Viewing Comments

Comments are also queried from the database and saved unto an array, similar to how posts are displayed. A while loop is used to iterate through the array and echo multiple <article>. A like comment is also attached to each comment. Comments are sorted by most liked by sorting the array with a usort.

Logging In and Out

To keep track of which user is currently logged in, we utilize PHP's $_SESSION global array. This array is able to transfer data between pages. Two main variables are kept: A boolean 'logged_in' and a string 'username.'

These session cookies will inform the pages whether a user has logged in and which user that is. This is useful when switching pages or refreshing by keeping the session ongoing.

If a user has logged in into their account, they have access to extra functionalities. For example, they are able to comment. The comment <form> is not echoed unto the HTML unless $_SESSION['logged_in'] is true. Furthermore, when liking, the likes will not be reflected on the database unless logged in. Similarly, a user's profile picture in the header will only showcase when logged in is true. The profile picture is queried from the database by using the variable $_SESSSION['username'].

When logging out the $_SESSION['username'] is unset and 'logged_in' is set to false.and the page is refreshed.

Dynamic Board List

The navigation side bar which allows the user to select different boards is dynamically built. Also the list of boards in the search bar and the make post page are dynamically built. A while loop is used to create multiple <a> links or <option> values. An array is used through the loop which contains the list of boards which is queried from the SQL database.

Saving and Displaying Images

The database stores images for user profile pictures and posts. However, the SQL only saves a string which is the path where to find the respective images. The images are saved on the website folder/server. Whenever a user uploads a post with an image the image is renamed with a dynamically increasing pathname. Its pathname depends on the UNIQUE ID given to the post. This way there are no duplicates. Moreover, when the database is queried for an image the pathname and path directory is returned in order to find the picture and display it.

Form Validation

Making posts or commenting is done through forms. However, these have to be validated. A user cannot make an empty comment or an empty post. Therefore, JQuery is used to make sure the required inputs are filled in. These are selected with a class identifier. After they are not empty, the form is allowed to be submitted through JavaScript. If the forms are not filled in, an alert will inform the user. The form will not be submitted.This validation is done asynchronously.

ADMIN: Deleting Comments and Posts

An administrator may delete posts and comments. When an administrator logs in there exists a php file and function which will query the database to check if it's an administrator. If the user is an admin this $isAdmin boolean will be true. This will make the php code echo extra buttons onto the page. The buttons are delete buttons to delete specific posts/comments. These have an onClick function in JQuery which will alert the user and after confirmation run an ajax call. The ajax call is a php function which will query the database to drop the respective post or comment from the database. The page is then refreshed to reflect any changes made to the post list or comment list.