Development of PHP Forum

Designed and Built by Lucas Laibly 10/12 - 10/13 of 2019

The task was assigned by Delta Defense for evaluation of my coding abilities. I had never touched PHP until this, but I did not let that hold me back. I wanted to atleast bring something to the table; I watched a series of guides, read more articles than I ever wanted to, and scanned through StackOverflow for tips. This pdf is to document my transition for inital idea to conclusion of where I am.

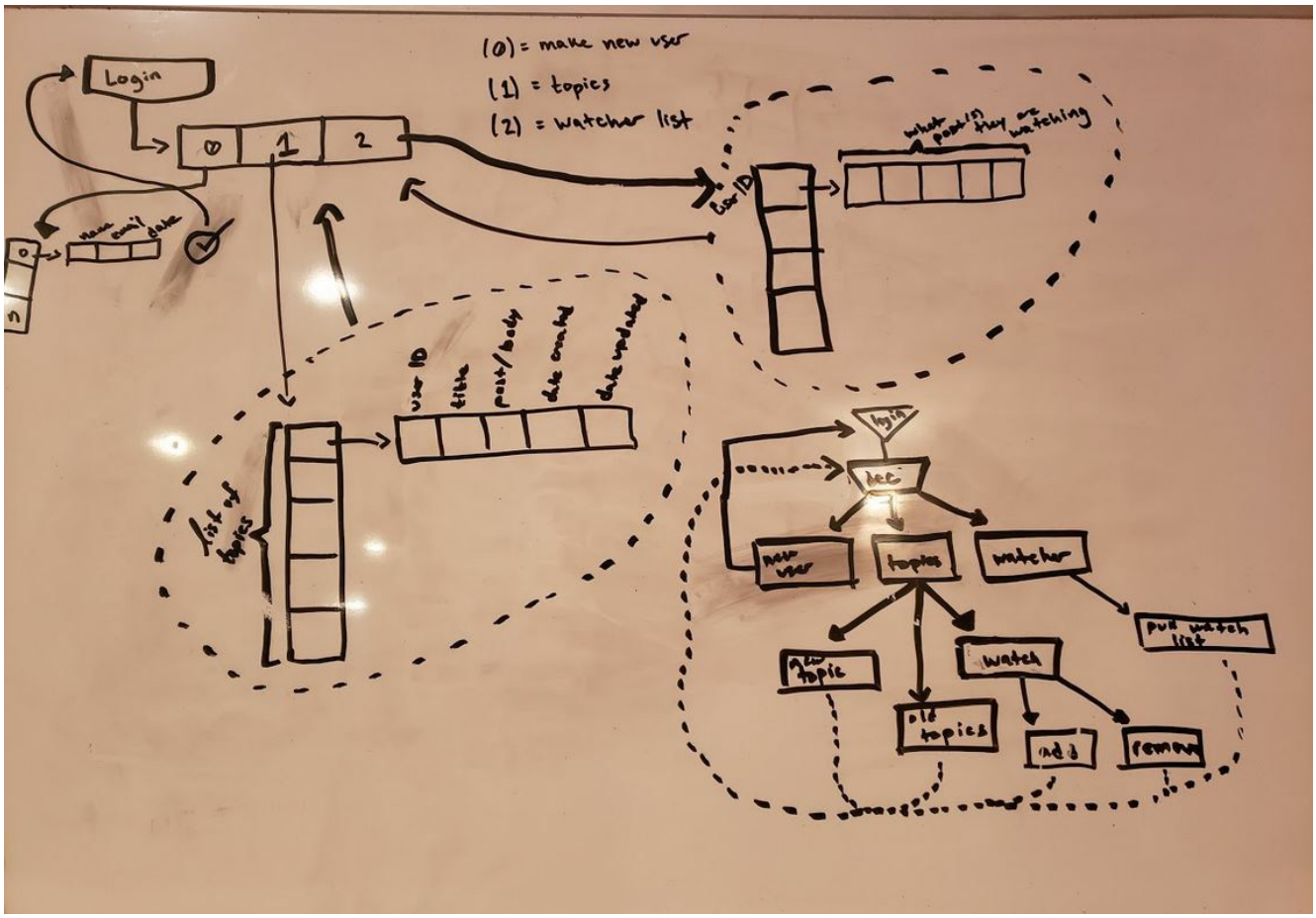
Table of Contents:

The expectations are as followed:

Develop and API that has the following objects: User, Topic, Replies, and Watchers

1. User - username, email, date created
2. Topic - Title, post body, user id, created date, updated date
3. Replies - Post body, user id, topic id, created date, updated date
4. Watcher - User id, topic id (allow users to "watch" topics)

Image A



The base idea that I had given the requirements.

I would like to discuss the absolute disaster that is Image A.
It begins with requesting the user to login and force all users to enter a username and valid email. The User class had a funciton within it that gathered the date and attached it to the other two items. I decided to put them into an array because that was easier for me to visualize, and being lost in the documentation of a new language I wanted something that made solid sense. I visualized the User class holding a private 2D array, where each array within it was the array with a username, email, and date it was created.

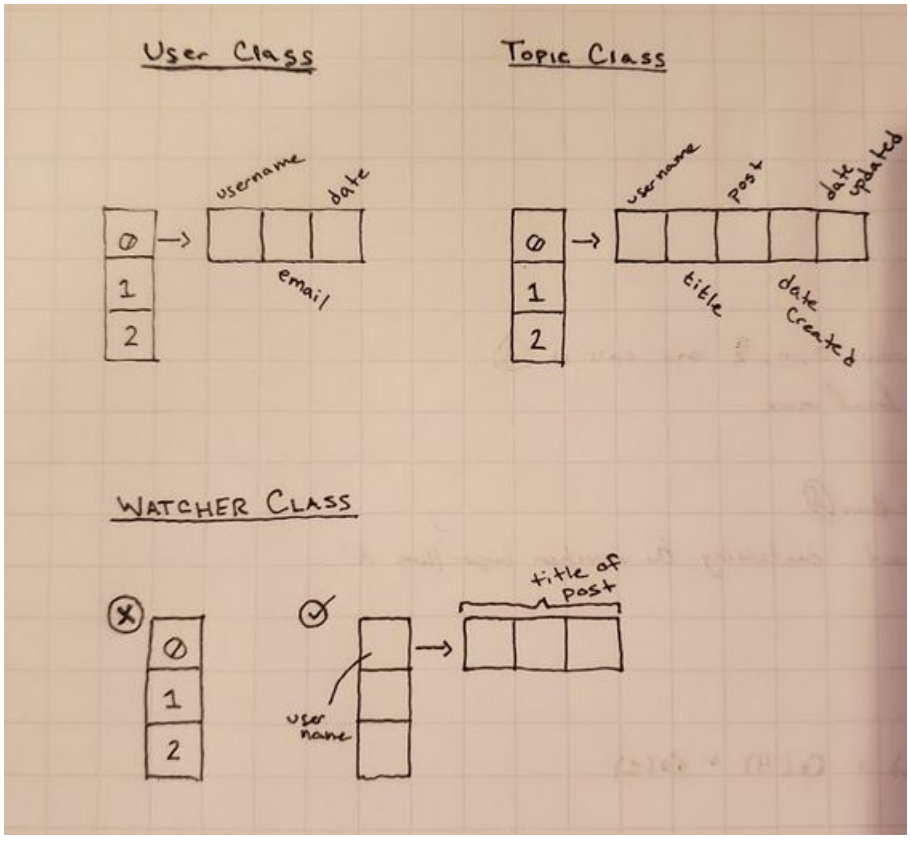
userInput	userInput	generate
username	email	date created

This is actually where the issues began, but I will touch on that in the Reflection section. The idea was to have some overall array that directed to each unique user.
Moving forward, the user is greeted with a page where there are two buttons: (1) new topic and (2) topics.

I imagined there being an obvious choice : any given user can make a new topic, or see the topics that already exist. Upon selecting the choice they are directed to the correct output. For new topic, a webpage loads that asks for their username, title of their post, and then an expandable text box for them to write in. Once you submit your topic, you are greeted with a "You have submitted". On the other hand, if you select see topics, you are directed to a webpage that *should* iterate over the entire topic array and print them for the user to read through.

Before moving forward I want to discuss how the topic array works. I made a Topic class that hold all the functions you can do to the topic object. This includes adding items, which would be placed into an array and then pushed into another. Making a monstrous 2D array. Each row hold the username, the title of their post, the post itself, the date it was created, and the date it was updated. Using a foreach loop, I can easily check the keys of each array and find specific ones more easily.

Image B



With a base idea set, and code coming together painfully, I realized I needed to rethink how I was going to approach adding a Watcher class (not fully implemented yet). Instead of using numbers, I set the keys to be each of the user's name ID and then each subarray was just the title of the post. PHP's dynamic arrays allow us to easily move items in and out of it.

Image C

```php
//add a user based on what they input
public function addUser($user, $email)
{
    $this->dateCreated();
    $date = $this->dateCreate;

    //put into unique 1D array
    $tempUserArray = array(0 => $user, 1 => $email, 2 => $date);

    //push into $userArray => 2D array with username and email unique
    array_push( &array: $this->userArray, $tempUserArray);
}

public function dateCreated()
{
    $date = getdate();
    $day = $date['mday'];
    $month = $date['mon'];
    $year =  $date['year'];

    $totalDate = $day . "/" . $month . "/" . $year;
    $this->dateCreate = $totalDate;
}
```

In short each user is passed through these two methods, and looking at it now...I made it a lot harder than it needs to be. Specifically that I do not need a function for getting the date, I can do that in the addUser() function. Regardless, I wanted to share this snippet because I wanted to show that each subarray within the User array has each of its components placed in a specific place (this is important for later).

Image D

```php
public function addTopic($username, $title, $post)
{
    //call the create function for the date
    $this->dateCreated();
    $dateC = $this->dateCreate;

    //set a base line for the updated date
    $this->dateUpdated();
    $dateU = $this->dateUpdate;

    //compile the array
    $temp = array($username, $title, $post, $dateC, $dateU);

    //push the array into the super array
    $this->addArray($temp);
}
```

In the Topic class we have a similar function that generates baselines for the date's. It like wise takes in the input from the newtopic.php file. It is all passed here, and pushed brought together. Once again I am keeping the username at the front, so I can easily search through. The idea here was that the User and Topic array are almost mirrored. There is always the case that the user doesnt post or follow anything. But *in a perfect world* the user will always do something, and therefore the arrays are mirrored.

Image E

```php
else {
    // The form has not been posted
    // Show the form
    ?>
```

```
    <form id="NewTopicForm" action="newtopic.php" method="post">
        Title:<br>
        <input type="text" name="Title"><br>
        Username:<br>
        <input type="text" name="Username"><br>
        <input type="hidden" name="SubmitCheck" value="sent">
        Post:<br>
        <textarea cols="50" rows="4" name="Post"></textarea><br>
        <input type="Submit" name="NewTopicForm_Submit" value="Submit">
    </form>
    <?php
}
?>
```

Each of the forms are similar (more or less). This one is for when the user wants to make a new post (see newtopic.php). The form prompts for the Title, Username, and then the Post that they want to make and add. Now, if you see this is wrapped in an else statement. Here is where the learning curve really hit me -> I could if/else wrap whether a name from an attribute in HTML had been heard yet. See below:

Image F and G

```
// Lets test if the form has been submitted
if(isset($_POST['SubmitCheck']))
{
    $newTopic = new Topic();
```

For the longest time I was unaware of this capability. However, it is a saving grace. Regardless, the form directs where each post gets made as I listen directly for a Username, a Title, and a Post:

```
if( $_POST ['Username'] and $_POST['Title'] and $_POST['Post'])
{
    //separate variables for each one, kind of anal about this
    $username = $_POST ['Username'];
    $title = $_POST['Title'];
    $post = $_POST['Post'];
```

Though it may seem redundant to assign variables the values that are being passed in, I personally like this because there can be no ambiguity in what I am handling at any given time. From here we can setTitle() in the topic class, and addTopic() with the username, title, and post as well.

I want to put a disclaimer out there that I have yet to build in the Watcher function as well as the Replies. The Watcher class I think I have figured out and you can poke around that under the Libraries section. As for the Replies I have absolutely no idea how to implement this, please see my reflection for further discussion.

Reflection

Looking back on this there is a lot I could and can do better. Firstly, I think it would have benefited me to use an actual database framework. I *probably* chose one of the worst ways to go about this. Arrays are great in that you can easily move around within them, add to them, and certainly access elements directly (assuming you know where you are aiming). The biggest downside with this is time. A lot of the time I am blindly shooting around massive 2D arrays and hoping for a hit. With a database I can store these massive structures more easily and cleanly. I think arrays look good on the surface, but can quickly spiral out of control.

The second issue with my code is that it is one directional, and this comes from a fundamental misunderstanding of how PHP handles webpages. Later research showed my flaws. Firstly, my program will move you through the different options, and route you with what you correctly select. However, using the header(Location: file.php) function cause a ton of lag and open ended pages. I noted that I can close them with "exit;" but I still had trouble directing webpages around with header().

The third issue is that because I can not easily check if values are going where they need to be, I am not sure if I am blowing out my arrays everytime a page [re]loads. And this is troublesome because that is the whole project more or less.

Final note of issue: HTML and look of my program. This thing is ugly. There is not flavour or design put into it. I am stressed about it working so I never put any time into the actual appeal of it. Even though I was given a fairly large time frame to work on this project, I still had to learn PHP and fail my way through it.

So moving forward, I am looking to add a few things. I need to address being able to move forwards and backwards within the program. This is an actual detriment to the program. Secondly, I need to address how the directories can communicate with each other. Even with include() and every variation of it, I still am greeted with the error that the file cannot be found (and all directories have read, write, execute functionality enabled). So I resulted to giving it the full path, and this **will not work for any devices other than mine**. And finally, I really want to give it a face lift, as well as figure out how to do the Replies function. Even thinking about it now, I am still very lost in how that would work with the infrastructure I have built. I think with my 2D arrays, I would have to expand the Topics array to a 3D disgusting thing. Where the array's subarray's subarray holds the replies, if that would even be enough cause then I need to think of how deleting works. Point is, direction is very much needed with that function.

Overall, I had a ton of fun pushing myself through this. I have never done something like this before where I need to learn a new langauge and develop a functioning program on the fly. I hope this is the response you are looking for, thank you for considering me @Delta Defense. I greatly appreciate the opportunity you have given me, and the challenge as well.