CSIS 311: MODULE 3 ASSIGNMENT 3B

DR. ANDREW CHEN

Module

Module 3: PHP Syntax 2: Functions, Arrays, Loops, Sessions.

Directions

This will be an adaptation of your earlier conversion program. Use the same units.

Description

This is intended to be a fancier version of your earlier conversion program.

This may seem overwhelmingly complicated, but if you just do each stage at a time, then as you do them, you should hopefully realize that, done right, the next stage isn't really a huge amount of work relative to the previous stage. It just seems like it when you read through it at first glance. I've also done my best to make sure that you'll have enough time to do this, if you work on it steadily, pacing each of the stages out.

Objectives

Writing a larger PHP program

Use persistent arrays in PHP

You will want to do this in stages:

0.1. **Stage 1.** Earlier, you had 5 files:

(1) the program's HTML file for converting from A to B, so here the user inputs the quantity for the A unit

(2) the program's PHP file for converting from A to B, so this receives that form submission, and outputs the quantity for the B unit (amongst other things)

(3) the program's HTML file for converting from B to A, so here the user inputs the quantity for the B unit

(4) the program's PHP file for converting from B to A, so this receives that form submission, and outputs the quantity for the A unit (amongst other things)

(5) index.html (which will link to each of the above mentioned HTML forms such that the user will know which link to will bring the user to where)

Now, you'll combine them as follows:

- the index.html (you may, if you so choose, have an index.php file instead) will have two forms in it, each of which submits to the two different php files listed below
- the A to B PHP file will have, as it's output, not only the result of the conversion, but also a form, where the initial value is the B value, that can be submitted to the B to A PHP file.
- the B to A PHP file will have, as it's output, not only the result of the conversion, but also a form, where the initial value is the A value, that can be submitted to the A to B PHP file.

In summary, the user might use these as follows:

(1) User starts at index.html (or index.php) and fills out (for example) the A to B conversion form and submits it.

(2) Since the A to B conversion form was submitted, it was submitted to the A to B PHP script, which runs, and outputs the following:

- The result of the conversion from A to B
- A form for converting from B to A, where the B initial value is the result of that conversion

(3) The user could then either edit that B value and change it to something else, or leave it as it is.

(4) The user could then submit the form.

(5) The form would be submitted to the B to A PHP script, which runs, and outputs the following:

- The result of converting from B to A
- A form for converting from A to B, where the A initial value is the result of that conversion from B to A

(6) Again, the user could either edit that A value and change it to something else, or leave it as it is.

(7) The user could then submit the form.

(8) This would then go back to the step that begins "Since the A to B conversion form was submitted".

Make sure you get this stage working correctly before you proceed on to the next stage. Reach out for help as needed along the way.

0.2. **Stage 2.** As you look at your code from the previous stage, you may notice a lot of the same code in the two different PHP files. Break that code out into different functions, put them in one PHP file that you use the *include* function to include into both of those, which should hopefully drastically reduce the amount of code in either of them.

Update your index.html (or index.php) file to now, (instead) be an index.php file. At the end of it, be sure that it has buttons that allow me to view the code of each of your respective .php files.

Make sure you get this stage working correctly before you proceed on to the next stage. Reach out for help as needed along the way.

0.3. **Stage 3.** As you look at your code from the previous stage, you'll see that you are really doing persistence already, since the value that you are converting would always persist throughout each of the presses of the convert button, as it goes back and forth between the two different PHP scripts.

Now, you're going to add a persistent array and some other persistent data. I recommend that, similar to the persistent examples that you've seen so far in the class, that you have a *passData* function that writes it all out as hidden input types, and that you get it all back by using the *extract* function.

The other persistent data will be the size of the array.

The array itself will have the entire history of the conversions, with associated timestamps. An example PHP file that constructs a multidimensional persistent PHP array, and puts timestamps in it, will be available separately.

To debug this, use the *print_r* function and to make the output readable, use the open *pre* tag before it and the close *pre* tag after it.

Make sure you get this stage working correctly before you proceed on to the next stage. Reach out for help as needed along the way.

0.4. **Stage 4.** As you look at your code from the previous page, you'll see that you've got a conversion history going. The next thing that we're going to do, is to do two things.

First, we're going to make the conversion history pretty, so it isn't just the output of *print_r*.

Second, we're going to allow a user to click on any of the units in the conversion history, to redo the conversion. This may seem pointless, but it has the advantage of getting the converted value back into the text field as the initial value, which may make it easier for them to edit.

Make sure you get this stage working correctly before you proceed on to the next stage. Reach out for help as needed along the way.

0.5. **Stage 5.** Provide a nice interface for the user to delete entries from the conversion history.

Grading

Grades will be as follows:

- Not completing any stage: F
- Completing only Stage 1: D (60/100)
- Completing stages 1-2: C (70/100)
- Completing stages 1-3: B (80/100)
- Completing stages 1-4: A (90/100)
- Completing stages 1-5: A+ (100/100)

You're welcome to work together with peers, but not to share code (you should never be copying from a classmate's file and pasting into your own or anything like that). If you do work with your peers, cite please mention which ones both in your comments and in your plain text file that you upload to D2L Brightspace.

Submission Instructions

Be sure that your assignment is put in its own subdirectory of *public_html/private* on your account on *puff.mnstate.edu*

Be sure that within *public_html/private* your *index.html* file has a link to the assignment subdirectory that is named accordingly, and is part of a table that also indicates how complete that assignment is (for example, "in progress" or "complete") as well as roughly how much time you spent working on the assignment (for example, "1 hour" or "10 hours").

Look for the appropriate Assignment Folder on D2L Brightspace and upload to there a file that says the following:

- how long you took to complete the assignment
- a summary of what you learned

Submit your assignment before the due date.

Due Date

You should look on D2L Brightspace for when the folder that corresponds to this assignment is set as due. This assignment should be submitted before then.

Resources

- information given in class
- previous assignments
- given example code
- the persistent multidimensional PHP array example mentioned above (not yet available as of the time of this writing)