Assignment 2 - Arrays and Files

Due Nov 3, 2023 by 11:59p.m. **Points** 100 **Submitting** a file upload **File Types** txt
Available Oct 19, 2023 at 8a.m. - Nov 6, 2023 at 11:59p.m.

This assignment was locked Nov 6, 2023 at 11:59p.m..

Please note that all assignments must be submitted both to CSUNIX and also to Canvas. Except for the filename difference noted below, your files in both submissions must be identical or your work will be graded 0.

Please follow the submission instructions carefully to prevent a grade of 0 for incomplete submission.

Do not edit any of the starter code that has been provided. Your server side solution must interact with the provided code correctly.

For this course programs like ChatGPT or Co-pilot will be treated exactly the same as code copied from an internet source. Any evidence of their use will be treated as a level 2 academic integrity violation.

Purpose

The purpose of this assignment is to move us from basic function writing into more sophisticated web development techniques that include processing files. In this assignment you will need to perform file I/O, manage data in arrays, and use encoding to respond to AJAX queries from the provided starter code. Specifically we'll be working with modular programming, sanitize/validate inputs, XSS checking, arrays, reading a file, writing a file, and accepting a file upload.

Starter Code

Please download the starter code for assignment 2 **(https://mycanvas.mohawkcollege.ca/courses/92989/files/17505477?wrap=1)** **(https://mycanvas.mohawkcollege.ca/courses/92989/files/17505477/download?download_frd=1)** . As indicated previously, we aren't assessing your abilities in producing client side code this term, and we want you to engage fully with the server side code. As such you are forbidden from altering the front end code. If we download your solutions and put it with the starter code file, it must work without alteration. To help you understand whether this is just some academic nonsense or if it's a real world restriction that could occur: during team programming one of the first steps is to agree on an interface which allows the teams to work in parallel without frequent interactions. Neither team is permitted to alter the interface once it is created. If a change is required, it must come from the project manager/team lead, which in this setting is your professor. This is a pattern that occurs in many kinds of programming and allows for teams to build software in parallel much quicker than an individual could generate the code. It also allows for the creation of standard tests at the time the specification is written so that we can validate that the created code meets the requirements. If we are going to strive to help you improve your skills as you move through the

program with the goal of a professional level of practice, it feels reasonable to set up a restriction such as this.

Getting Started

Please review the **documentation standards (https://mycanvas.mohawkcollege.ca/courses/92989/pages/documentation-standards)** for the homework in this course. All PHP files must follow this standard.

Download the starter code and extract the files to your /xampp/htdocs/10260/a2 folder (or the equivalent on your development machine).

Create a new file called **me.php** that outputs your name and student number and place it in the same folder. Remember to apply the documentation standards even to this file.

Question 1

Use the provided client-side HTML code "a2-q1.html" and "a2-q1.js" to complete this exercise.

The code defines two buttons, "Pokémon" and "Movies", and a dropdown list with the following options: "Show Results (Ascending)", "Show Results (Descending)". The button will provide a GET parameter named "choice" and a value either of "pokemon" or "movies" (note that the GET parameter will not use the accented character, just plain ASCII, not a typo). The drop down list will provide a parameter named "sort" with the value "a" or "d" for ascending or descending respectively.

Your task is to create the server-side PHP script that processes the button click and dropdown selection combination. The data for the response will come from a corresponding data file, sorts it based on the chosen option, and sends the results back to the client-side. All inputs, regardless of source, should have validation or sanitization applied as appropriate.

Name your solution "`data_processor.php`" and don't forget your documentation.

1. Based on the GET parameters provided by the client-side code, identify which button was pressed and which sorting option is active (ascending or descending).
2. Open and read the corresponding text file based on the button pressed: "pokemon.txt" if the choice parameter is "pokemon" or "movies.json" if the choice parameter is "movies". The file "pokemon.txt" will have 2 pieces of information. First you will read the name of a character, then you will read a URL to a graphic for the character (2 lines total per character). The file "movies.json" will have a record that gives the movie name and release year. See the provided files for details.
3. Create 2 functions that will perform the appropriate "read file" operations depending on which file was requested. This function must return an associative array. Other functions and modularity is encouraged. You are not generating any HTML code, so MVC is not a relevant technique.
4. Read the data from the text file, and parse it if necessary, into an associative array - for the Pokémon file use the field names "name" and "image" as the index values. For the movies file use the index

values defined by the JSON code.

5. Implement the sorting (by name) based on the selected sorting option:
    - If "Show Results (Ascending)" is selected, sort the data in ascending order (A-Z).
    - If "Show Results (Descending)" is selected, sort the data in descending order (Z-A).
    - For either sorting case you may choose how to handle non-alphabetic data.
6. Create a JSON-encoded associative array containing the sorted data.
7. Send the JSON-encoded array as a response to the client-side.
8. Ensure that your script handles any potential errors or exceptions gracefully and provides informative error messages when necessary.

An online demo version is available at **https://csunix.mohawkcollege.ca/~adams/10260/a2/a2-q1 (https://csunix.mohawkcollege.ca/~adams/10260/a2/a2-q1)**

Question 2

Use the provided client-side HTML code "a2-q2.html" and "a2-q2.js", that includes an upload file form to complete this problem.

In this problem you will allow your user to upload a 3 column comma separated values (CSV) formatted file which you will process and provide the result as a JSON formatted array. This will use POST and the CSV file name will be parameter csvFile

The first row of the CSV will contain the names of the fields for that column which you will use as the associative array indices (see sample.csv for an example).

Using the parameter `sortColumn`, sort the array by the column specified (for example using sample.csv a 1 would correspond to sorting by Song Title). Sort using a natural sorting order (numbers should appear in order like 1, 2, 3, ..., 10, ..., not like 1, 10, 11, ..., 2, ...). To help you in this task, see the php.net help page for **usort() (https://www.php.net/manual/en/function.usort.php)**, especially example #2. You may also find the help page for **strnatcasecmp() (https://www.php.net/manual/en/function.strnatcasecmp.php)** helpful.

Be sure to sanitize and validate all information provided by the user, including things like the file name - see the modules for additional information on how XSS issues can creep in. Handle errors gracefully.

Name your solution "`file_processor.php`" and don't forget to document your code.

An online demo version is available at **https://csunix.mohawkcollege.ca/~adams/10260/a2/a2-q2 (https://csunix.mohawkcollege.ca/~adams/10260/a2/a2-q2)**

Submission Instructions

Programs are meant to be run, not read. Therefore in this course you must submit your work to CSUNIX so that your instructor can grade the functional performance of your work. This assignment must be uploaded to your **public_html/private/10260/a2** directory on CSUNIX (you will probably have to create this directory). If you submit this to the wrong folder it will be treated as though it was not submitted. Searching around your directory structure to find the folder is not a good use of your professor's assignment grading time. If your professor cannot find your work on CSUNIX you will be given 1 week to correct the folder name, and to email them to alert them that you have fixed the problem. After the 1 week period the grade of 0 will be permanent.

Your program must also be submitted to Canvas. First and foremost this establishes a time that you submitted your work that we can all agree on. Secondly it allows your work to be inspected for plagiarism. You will not be judged based on the simple score from TurnItIn, but on your professor's interpretation of whether the identified code is common and reasonably should appear in many student's work (i.e. **<? php** will show up a lot) or whether the code suggests copying from an unauthorized source. We are suggesting you shouldn't fully trust AI tools, and we won't either.

You will submit your files to Canvas in the following way:

For each .php file (not the .html starter code):

- Copy the file and add .txt to the end of the filename. For example: question1.php gets copied to question1.php.txt
- Upload the .txt files to Canvas

Do not archive (zip) your files, or upload files that are not requested.

You should be submitting **me.php.txt**, **data_processor.php.txt**, and **file_processor.php.txt** to Canvas and uploading the HTML starter code and your 3 .php files to CSUNIX.

10260 Assignment 2 Rubric

Criteria	Ratings					Pts
me.php as required me.php is present, provides the required information, and is correctly documented.	**5 pts** **Full Marks** me.php provides the correct information and is well documented.	**3 pts** **Satisfactory** me.php provides the correct information but is not correctly documented.	**2 pts** **Poor** me.php provides incomplete or incorrect information but some effort is present.	**0 pts** **No Marks** me.php is not submitted, does not meet the technical requirements, or is not available on csunix		5 pts
question 1: non-technical merits File name is as specified, file header is present and as required, functions are documented as required. Client side starter code files and support files are unmodified.	**15 pts** **Excellent** Solution is well documented and has not modified the starter code. The solution is named as required.	**9 pts** **Good** The solution's documentation is incomplete or missing. Solution is named as required and the starter code is not modified.	**6 pts** **Satisfactory** Solution is well documented but has modified the starter code.	**3 pts** **Poor** The solution is insufficiently documented and has modified the starter code.	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	15 pts
question 1: technical merits Code is written in a manner that is consistent with course instruction and includes appropriate levels of commenting, modularity, and error	**45 pts** **Excellent** Code contains appropriate internal documentation, modularity, and error handling. Course concepts such as validation and sanitization are appropriately	**27 pts** **Good** Code contains internal documentation, modularity, and error handling but could be improved. Course concepts such as validation and sanitization are	**18 pts** **Satisfactory** Code may not contain appropriate internal documentation, modularity, and/or error handling. Course concepts such as validation and	**9 pts** **Poor** Code may not contain appropriate internal documentation, modularity, or error handling. Course concepts such as validation and sanitization	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	45 pts

Criteria	Ratings					Pts
handling. Question functionality is correctly implemented and produces correct outputs.	integrated. Functionality is correctly implemented. Starter code is not modified.	appropriately integrated. Functionality is correctly implemented. Starter code is not modified.	sanitization may not be fully integrated. Functionality may contain minor errors but is mostly correctly implemented. Starter code is not modified.	may not be appropriately integrated. Functionality contains significant deviations from the requirements. Starter code may have been modified to accommodate errors.		
question 2: non-technical merits File name is as specified, file header is present and as required, functions are documented as required. Client side starter code files and support files are unmodified.	**7.5 pts** **Excellent** Solution is well documented and has not modified the starter code. The solution is named as required.	**4.5 pts** **Good** The solution's documentation is incomplete or missing. Solution is named as required and the starter code is not modified.	**3 pts** **Satisfactory** Solution is well documented but has modified the starter code.	**1.5 pts** **Poor** The solution is insufficiently documented and has modified the starter code.	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	7.5 pts
question 2: technical merits Code is written in a manner that is consistent with course instruction and includes appropriate levels of commenting, modularity, and error	**27.5 pts** **Excellent** Code contains appropriate internal documentation, modularity, and error handling. Course concepts such as validation and sanitization are appropriately integrated.	**16.5 pts** **Good** Code contains internal documentation, modularity, and error handling but could be improved. Course concepts such as validation and sanitization are appropriately	**11 pts** **Satisfactory** Code may not contain appropriate internal documentation, modularity, and/or error handling. Course concepts such as validation and sanitization	**5.5 pts** **Poor** Code may not contain appropriate internal documentation, modularity, or error handling. Course concepts such as validation and sanitization may not be	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	27.5 pts

Criteria	Ratings					Pts
handling. Question functionality is correctly implemented and produces correct outputs.	Functionality is correctly implemented. Starter code is not modified.	integrated. Functionality is correctly implemented. Starter code is not modified.	may not be fully integrated. Functionality may contain minor errors but is mostly correctly implemented. Starter code is not modified.	appropriately integrated. Functionality contains significant deviations from the requirements. Starter code may have been modified to accommodate errors.		
						Total Points: 100