PHP assignment

Hi, welcome to our ‘code’ restaurant! Today we will begin with a small starter that should wake your brain enough to be ready for the main course. This will demand from you a bit more work, more focus and more imagination, but when you finish you will feel good, trust me. We will finish with a delicious dessert.

Have fun and bon appétit :)

(Please read the instructions below carefully. It's very important to completely understand the goal of each assignment. If you have any questions, feel free to email me.)

Starter - Algorithmic pleasure

A: The sum of all natural numbers below **10** that are multiples of **3 or 5** are **23** (3 + 5 + 6 + 9)

Write a php script that will find the sum of all the multiples of **3 or 5** below **1000**. The script should run from command line and put the result on screen. We will judge this task based on simplicity, efficiency and cleverness of the code.

Extra: Create a second algorithm to find the sum of all the multiples of 3 **and** 4 below 1000.

B: Make a function that will calculate the power of a number x to index y, but you cannot use multiplication! to make it a bit simpler please take into account only natural numbers.

C: Calculate and print 10 numbers for fibonacci series. Use recursion.

Extra: Create second algorithm for fibonacci but without recursion.

Task Two - Architectural struggle

The attachment named **example.php** is a simple php script. It's job is to serve one json-encoded string in response to http request. The script is also using **examples.csv** for fetching data. Your job is to refactor this script. Remove all mistakes and bad practices, create some structure of folders and files that will contain classes, design architecture as if this simple script was a small part of a much bigger project. In other words: pimp this script up! This is more about architecture of your code and less about function itself. We will judge on structure, hierarchy of objects and overall design. Take into consideration how clean your code is and how easy it is to read and understand. Avoid tight coupling between any layer you introduce to this code and remember that this code should be easy for further extension by somebody else. You have total freedom in how to achieve this. There are, however, some rules as described below:

- You **may not** use any external library or framework **except from the ones used to Unit Test the solution**
- After refactoring, this code **must** do the same thing.
- You **have to** use MVC pattern (or similar)
- You **could** use any other pattern if you feel like it
- You **should** conform to programming principles
- You **have to** use OOP
- You **have to** do it in RESTful style
- You **should** introduce some framework-like feature (routing, dispatching, autoloading)
- You **have to** use at least php 5.3 (namespace is mandatory)
- You **can** use psr-0 autoloading system
- You **can** put comments where you describe why you do certain things or just explaining some more high level decision
- You **can** change storage system for contact data.

- **All above rules are very important!**
    - Have to, may not- this is mandatory
    - could, should - this is strongly recommended
    - can - it's up to you

Desert - The icing on the cake

Using the code created in **Task Two** add full CRUD feature to it. You may change e data storage system if you want. In the end the system should allow to add a new entry and/or remove update existing ones. There is no need to create any form of UI.

Final notes:

- Remember that this is web based application, http statuses are important
- Code should have some basic exception control and data validation
- You can use REST architecture (it will be a plus ;)
- Design of this code is more important that it's function. Focus on good, clean architecture!
- Follow programming principles! If you don't know what that is search SOLID in object oriented design.

Support

If you have any questions, please contact me via email.