REST interface that utilises JSON-RPC, XML-RPC and GraphQL.

Description

A company has contacted you to develop a REST interface that utilises JSON-RPC, XML-RPC and GraphQL as many of the people in the company are using different systems to interact with the existing customers. To tie all the technologies together, they wish for Flask to be used as a REST interface that will be provided to the customers.

The company has mentioned that they do not have a complete Python implementation and have asked that some components be developed in the existing JSON/XML RPC server examples.

Specifics

- Python Flask REST service should be used to generate the main web endpoint for the service that returns JSON marked up data. This endpoint should have different defined URLs:
    - /getStockIDs – Returns a list of stock IDs
    - /getStockNames Returns a list of item names
    - /getAllData – Returns all the data from the stock file
    - /getQty – Returns the qty of each produce along with the name
    - /getPrice – Returns the price for each item

    Each of these URLS is responsible for reading the sample product information from the **products.txt** file, convert the required content to JSON and return it to the caller.

- The client has mentioned to you that they would like an additional JSON RPC interface developed using existing technologies which allows a client to call a function on the JSON RPC server when the /getAllDataJSONRPC function is called in the main Flask application. JSON content will be returned from the JSON RPC server and returned to the caller.

- To keep track of all the different calls that are made to the server, a log file should be created called “calls.log”. Whenever a request comes into the Flask server, a new record should be printed into the file, containing the time/date and the name of the function that was called. This should be completed with a Python 3 logging framework.

- GraphQL provides the ability to retrieve data in a structured format. Create a custom URL that will take in up to three parameters from the user via the /converToGraph URL and return the parameter marked up as a GraphQL query that can be manually executed by the user.

They key to this assignment is creating valid JSON that will be sent to the server and valid JSON that will be retrieved from the server. Use an online JSON validation tool when testing your application JSON. E.g., https://jsonlint.com/

Tips

- To aid the development process, use Postman to emulate POST/GET requests.

References / Further reading

- https://docs.python.org/3/library/xmlrpc.client.html
- https://github.com/hprose/hprose-nodejs