

Homework1_DOS

Hassan Saed Hassan Abu Thiab

11714642

Dependencies and App requirements:

- Requests package
- Python3
- Restful & flask are all in the requirements.txt file.
 - o Pip install -r requirements.txt
 - o Use pip3 for Python
- Sqlite3

Code Design:

The Code is designed using Python. It utilizes the Socket package in order to get the Servers running on their current machine IP, **Requests** in Order to communicate between the server's **components** and the **Client** as well. **Restful** happens in front-**end** and **backend servers** only, while the **Client** has simple instructions to allow inputs from the terminal to trigger functions in **ClientServer**. A Simple Database formed by **SQLAlchemy** and **SQLITE3** for the Catalog Server while the **Orderlog** Server uses **TXT** as it's form of permanent data.

Client Code: Asks for inputs from User then calls the ClientServer

The application asks the User to enter the **ClientServer** IP in terminal. This is done so the Client can communicate with the server through the calls and save the effort of writing the Link again and again!

```
#Link for Client Server
BASE = input("Enter the IP Server for Client Front Server (ONLY enter the server private IP EX:192.168.1.123\n")
```

Base → ClientServer

- If the User enters 1 → Search by topic so the link should be the Catalog Port
- If the user enters 2 → Search by ID at Catalog port
- If the user enters 3 → Purchase by ID at Orderlog port
- If 4 then it exists
- We use the request import here to perform the requests towards ClientServer. The response is then changed into Json/text and printed onto the terminal

ClientServer: Redirects requests from the Client application towards the Catalog Server and Orderlog Server

- Operations Search Info and Purchase all function the same as in, they will redirect the requests from the client and call the Backend servers based on the operation required.

Catalog Server: Performs Info and Search and returns the response.

1) The Catalog server performs three operations. Being Search, Info and Update for the purchase operation from the Orderlog.

2) The Catalog Server uses an **SQLITE3** DB that's connected to **SQLAlchemy**, make sure the **the data.db is placed within the same folder as the Python file**

3) We define the Model for the Database Table as follows:

The Database format isn't serializable into Json in it's current state so we use marshall with library to help us perform that.

1- **Search Operation:** We perform Query on the DB then return all the lines that match the attribute. We take an argument that's Topicreq and use it as the parameter for the query since that's the topic we wanna match

2- **Info Operation:** Performs the same thing as Search except this time we are looping and stopping on the very first result. We send the ID that's requested from the client and query based on it.

3- **Update Operation:** This Operation is called by orderlog. It queries based on ID then takes the Stock attribute and -1s it. If the stock is empty we replenish it by 5

The element in the table is returned to Orderlog then after committing the changes in Catalog DB

Orderlog.py:

We have the Link to Catalog as an input in order to have the application work on different devices. **So please when you run Orderlog remember to enter the Catalog Server twice if it asks for it two times until it says Running on 192..**

Purchase Operation:

We use put instead of GET since we are updating. Then in order to store the element we received from the Catalog server we create/open a TXT file where we append the Data into it in the form of an orderlog.

The Client receives the Item name+ it being purchased successfully

How the Code works:

When the User enters the servers IPs for **ClientServer** and the other IPs for the other servers, the **Client.py** will ask the user to specify an Operation through a number.

Once the User specifies an Operation they will be asked to enter a value fitting of that Operation, For example **Topic** → Must enter **distributed systems**.

This will trigger a request towards **ClientServer** which will direct it to the **backends** server. If it was a Purchase it will call the **Orderlog** Server using RESTFUL, otherwise it will contact the **Catalog** server.

Search/Info:

It will trigger Info/Search in the Catalog server, which will query the DB and returns the elements to the ClientServer which are then returned to Client and displayed in the form of JSON

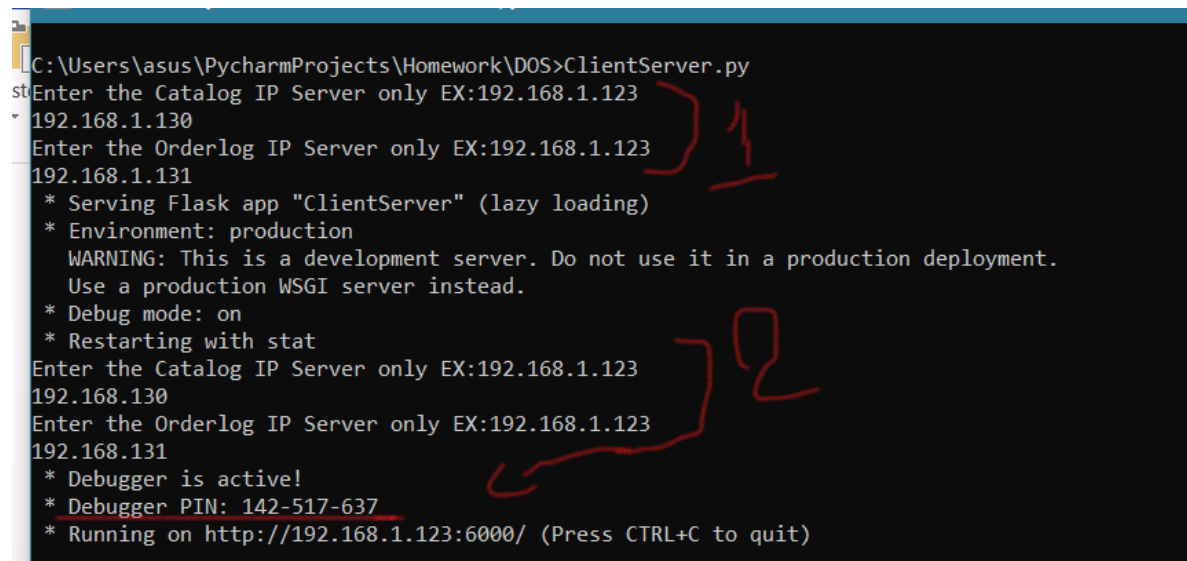
Purchase:

The orderlog will trigger an Update request in the Catalog Server in order to update the DB there... The Orderlog server then receives the transaction and saves it in the TXT. Then the Orderlog returns "Successful+ID" to the ClientServer which the ClientServer returns to the Client and displays it in the form of TXT

How to run the Entire thing:

First you need three VM and one host. Run each component on VM and Client on host or however you feel is best, but so long you use 4 machines overall.

- On each machine except the client.py run the **Command hostname -I for Linux and ipconfig for Windows**
 - o This way you will get the IP of each server printed out as you will need to enter it as an Input later
- Run Catalog Server on machine 1
- Run Orderlog Server on machine 2
 - o **Enter the IP server for the Catalog in the TERMINAL. IT MIGHT ASK FOR IT TWICE SO WAIT UNTIL YOU SEE DEBUGGER IS ACTIVE**
- Run ClientServer on Machine 3
 - o Enter the **IP server** For **Catalog** and **Orderlog** once it asks for them. **IT MIGHT ASK FOR IT TWICE SO WAIT UNTIL YOU SEE DEBUGGER IS ACTIVE**



```
C:\Users\asus\PycharmProjects\Homework\DOS>ClientServer.py
Enter the Catalog IP Server only EX:192.168.1.123
192.168.1.130
Enter the Orderlog IP Server only EX:192.168.1.123
192.168.1.131
* Serving Flask app "ClientServer" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
Enter the Catalog IP Server only EX:192.168.1.123
192.168.130
Enter the Orderlog IP Server only EX:192.168.1.123
192.168.131
* Debugger is active!
* Debugger PIN: 142-517-637
* Running on http://192.168.1.123:6000/ (Press CTRL+C to quit)
```

- Run Client on Machine4/Host:
 - o Enter the IP server for **ClientServer**
 - o Interact with the Terminal Interface as displayed by entering 1 for topic etc..
 - o Fill the topic correctly and hit enter.
 - o You can view the Orderlog by opening the TXT and view the DB changes by performing INFO again

Please contact me in the case of any issues running the Code

Issues:

When sending inputs through terminal to the Servers (Entering IP in Orderlog and ClientServer) the Terminal will ask twice before actually running the application. I couldn't find a way around it so I would like to know a solution. But Please enter it twice until you see the word Debugger Is Running on your terminal for Both servers