<u>Checkpoint 2:</u> Build a HTTP server

Write a simple HTTP server, which can handle HTTP GET requests sent from a browser. Upon receiving the requests, the server will first authenticate the user, by checking whether a correct password is contained in the request parameter. Then, the server will lock/unlock the door according to the requests, and send a response back to the client's browser. A python example of HTTP server is provided along with the assignment. You are encouraged to append on the example, but you can also choose to use the language you are familiar with.

The python example (**HTTPServerWithLibs.py, available in week2's lab practice folder**) uses http.server and urllib. The example includes all basic operations required for implementing this checkpoint. To know more about these libraries, please refer to:
https://docs.python.org/3/library/http.server.html
https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse

<u>Introduction</u>

HTTP server is a process that runs on a machine to "listen" and "handle" the requests sent to this process.

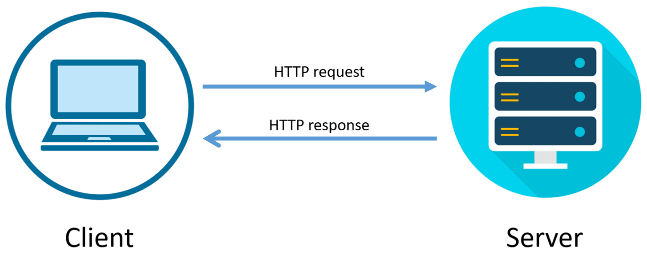


Image source: https://bytesofgigabytes.com

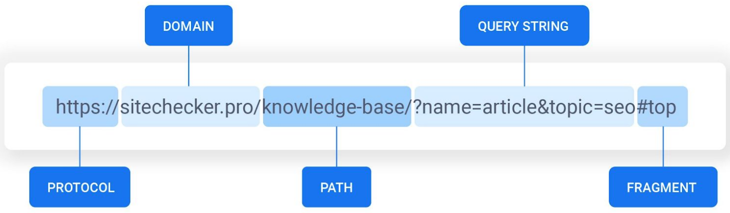
1. **Listen** : A web server is built over a socket. This socket is bound to port 80 by default (port number can change, depending upon the availability). The server actively listens at the port to accept the *requests* sent by the web browser/client.
2. **Handle**: once a request has been received, the server reads the header of the request, processes it and sends back the *response*.

The two major methods used in HTTP are GET and POST. The GET method sends parameters through the requested URL. For more information, please refer to:
https://www.w3schools.com/tags/ref_httpmethods.asp

To read more about HTTP, please refer to:
https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

URL, Path, Parameters: When a URL (Uniform Resource Locator) is sent from the browser, A request is created and sent to the server. In the Apache HTTP server, the URL is parsed, and the corresponding file under the folder of your web server is sent to the client as a response. For example, if you create a folder test under your root directory (by default, C:\xampp\htdocs) and create a file index.html inside that folder, the URL to access that file is: http://localhost/test/index.html.

In our HTTP server, no file will be served, and the URL will be used to denote actions (i.e., path in the graph below) and parameters (Query String). In the given example, we use the Python built-in http.server package, which uses the variable *path* to refer to both PATH and QUERY STRING. Hence, in the given code example, we use the variable *URI* to refer to PATH.


DOMAIN
QUERY STRING
https://sitechecker.pro/knowledge-base/?name=article&topic=seo#top
PROTOCOL
PATH
FRAGMENT

To read more about URL, please refer to:

1. https://en.wikipedia.org/wiki/Query_string
2. https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

Assumptions:

1. You will write a HTTP server program and visit that server using your browser as a client.
2. A password will be hardcoded in your server program.
3. A file on the server stores the status code of the door, where 0 for unlocked and 1 for locked. Nothing else apart from the status code of the door is stored in the file. Everytime the status is changed, the file is updated. Example: if the door is locked, the status file has 1 in it, else it will have a 0 in it.
4. A file on the server stores the log of the operation, where each line represents a request. Examples:

```
01-18-2021 21:08:30 request:lock result:200 client_ip:192.168.1.21
01-18-2021 21:08:30 request:lock result:401 client_ip:192.168.1.21
```

Tasks:

1. Accept GET requests to the given URIs (or say PATH): “lock”, “unlock”, and “viewLog”. All accepted requests will be appended to the log file. For other URIs, return a 404 Error.
2. A password will be sent as a URL parameter in the GET requests (e.g., URL: http://192.168.1.2/lock?pwd=xxx).

3. Compare the received password with a correct password hardcoded in your program. If the right password is not provided, return a “401” error.
4. For authorized users:
    a. Lock and unlock: upon receiving a lock or unlock request, the server modifies the status of the lock file according, regardless of the file’s current status. The server then returns 200 OK in the response status line and the content of the lock status file to be displayed on the client’s browser.
    b. viewLog: the server returns 200 OK in the response status line and the content of the log file to be displayed on the client’s browser.

Turn in:

1. Source code
2. A report that contains three basic sections:
    a. Explain the design and functioning of your program.
    b. Present your test cases and their results (screenshots).
    c. Briefly explain your understanding of the HTTP server.

Rubric:

- Server can process the specified requests correctly
- Log client’s requests as per the guidelines
- User authentication as per the guidelines
- Exception handling complaint with the status code of the HTTP protocol
- Comments and code format

Notes:

1) By default, the browser will request for a favicon file (favicon.ico) from any server. You will see the request when you run the server in the example code.
2) When running your own server, please make sure the port is not occupied by other programs (i.e., the given code example).