NEW YORK UNIVERSITY

Computer Networking Assignment 2

Lab 2: Web Server Lab

In this lab, you will learn the basics of socket programming for TCP connections in Python: 1) how to create a socket, 2) bind it to a specific address and port, and 3) send and receive a HTTP packet. You will also learn some basics of HTTP header format.

You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client. Make sure you include all relevant header lines, if you forget any, Gradescope will tell you. For the header 'Server' feel free to use one of your own choice/making.

Code

Below you will find the skeleton code for the Web server. You are to complete the skeleton code. The places where you need to fill in code are marked with #Fill in start and #Fill in end. Each place may require one or more lines of code.

Note: It is not recommended to deviate or remove any skeleton code. You are an engineer so you are welcome to add code outside the fill in start and fill in end areas. However, if you delete skeleton code, the TAs are limited on support we can provide if you drift too far off the intended path.

Running the Server

Put an HTML file (e.g., helloworld.html) in the same directory that the server is in. Run the server program (e.g. python solution.py). Determine the IP address of the host that is running the server (e.g., 127.0.0.1). Open a browser and provide the corresponding URL. For example: http://127.0.0.1:13331/helloworld.html

'helloworld.html' is the name of the file you placed in the server directory. Note that you should also the use of the port number after the colon. GradeScope will test your code using port 13331. In the above example, we have used the port number 13331. The browser should then display the contents of helloworld.html. If you omit ":13331", the

browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80. Next, try to get a file that is not present at the server. You should get a “404 Not Found” message.

To reiterate, your submission will be evaluated with a client testing the functionality of valid and invalid requests. You should perform testing locally to verify that your server handles valid and invalid requests as expected. A valid request should serve an html file to the client and an invalid request should serve a 404 error message.

What to Hand in

Submit the code to GradeScope using your GitHub repository. **Your submission program file name must be named webServer.py.**

Notes:

- There are clients (browsers) that will not present HTML content unless encoded HTTP headers are submitted (such as Content-Type, Server, Connection) with the message from the web server.
- HTTP status codes “200 OK” and “404 Not Found” are required to be part of the Web Server in order to receive full credit on this assignment.
- Do not send line by line, prepare your entire transmission and send as one. That means, send should be called only once for transmission.

The skeleton code can be found at
https://github.com/NYUCyberFellows-CSGY6843/assignment2-webserver

NEW YORK UNIVERSITY

FAQ

Q: I am getting the following error in gradescope:

"cp: cannot stat '/autograder/submission/webServer.py': No such file or directory"

A: If you are submitting a python solution, this python assignment submission must have the filename titled "webServer.py" (minus the quotation marks). Make sure your file meets this naming requirement.

Required Textbook Reference Reading

- Chapter 2: 2.7 Socket Programming: Creating Network Applications

Recommended Reference Reading

- Good tutorial on python sockets, https://realpython.com/python-sockets/
- Socket library docs, https://docs.python.org/3/library/socket.html

Most Common issues

1. Print statements are not commented out. If uncommented, they may cause errors in your gradescope submission. Print statements should only be used for local troubleshooting.
2. Improper encoding
    a. See https://pythontic.com/modules/socket/send
    b. You can send data as bytes in a variety of formats. You can even just read data from a file as bytes to avoid having to configure encoding later.
3. Inappropriate socket management (i.e. not opening or closing the socket appropriately)
4. Syntactical errors