Computer Networks 1

Lab 2c

Socket Programming: Chat Application

Student Name: **Nguyễn Quý Hải**

Student No.: **2052974**

I. Objectives

- Practice with Socket programming in Java.
- Build a simple chat application using client-server model.
- Multithreaded application.

II. Content

1. Socket programming in Python

Exercise 1: Create a program that connects to a web server and downloads the homepage of this website to local computer.

Solution:

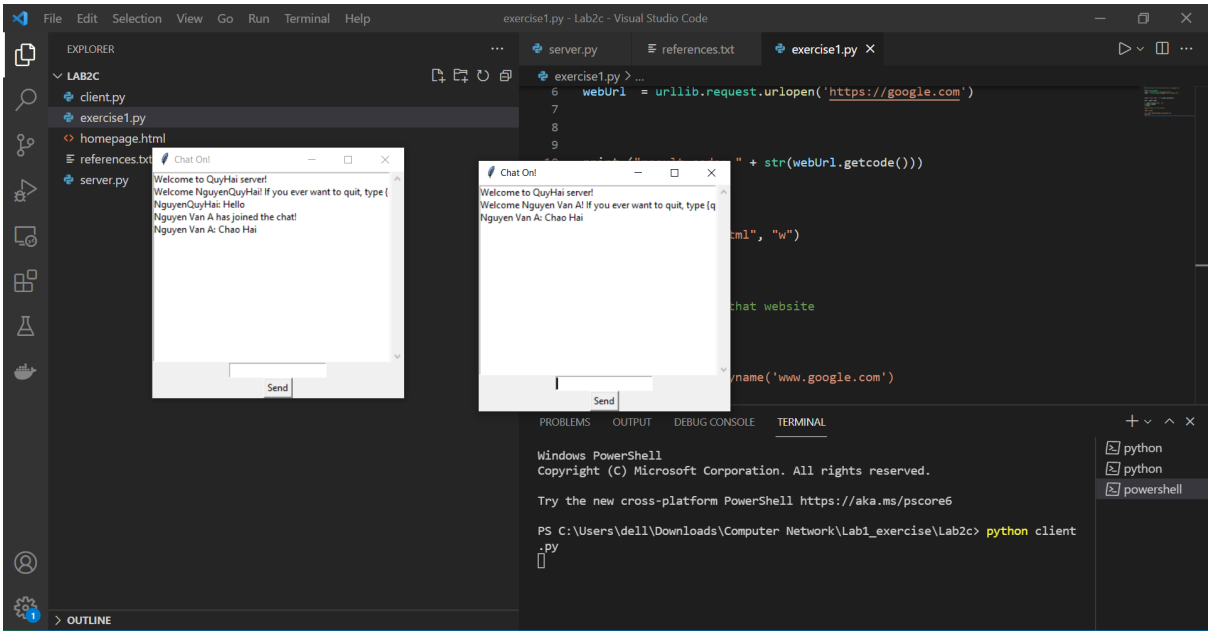
- Connect to that homepage
- Read the content
- Write it to local file.

```
import urllib.request
# open a connection to a URL using urllib
webUrl  = urllib.request.urlopen('https://google.com')



print ("result code: " + str(webUrl.getcode()))

data = webUrl.read()

f = open("homepage.html", "w")
f.write(str(data))
f.close()
```

In this example, I use `urllib` as library to open url and read data. After reading, the contents will be written to “homepage.html” in local file. To see the result, just type python exercise1.py in terminal and run, then check the homepage.html.

2. Develop a simple chat application using client-server model

Exercise 2: Design the user interface for the chat application



I use tkinter for UI design for client only. With the server side, we just need to start the server by: python server.py command in terminal. After that, we will open app for the client by command python client.py . When we have UI, the first input and hit enter is username, after that, each time we type enter, we will send the message.

```
top = tkinter.Tk()
top.title("Chat On!")

messages_frame = tkinter.Frame(top)
my_msg = tkinter.StringVar()  # For the messages to be sent.
my_msg.set("")
scrollbar = tkinter.Scrollbar(messages_frame)  # To see through
previous messages.
# this will contain the messages.
msg_list = tkinter.Listbox(messages_frame, height=15, width=50,
yscrollcommand=scrollbar.set)
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)
msg_list.pack()
messages_frame.pack()

entry_field = tkinter.Entry(top, textvariable=my_msg)
entry_field.bind("<Return>", send)
entry_field.pack()
send_button = tkinter.Button(top, text="Send", command=send)
send_button.pack()
```

In this example, I will automatically use localhost. If want to change your host for your own purpose, just configure it in the code.

3. Multithread in Python

Ex3: Using multithread programming model to make the chat application can talk to many different users concurrently.

server.py:

```
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread

def accept_incoming_connections():
    """Sets up handling for incoming clients."""
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Welcome to QuyHai server!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()

def handle_client(client):  # Takes client socket as argument.
    """Handles a single client connection."""

    name = client.recv(BUFSIZ).decode("utf8")
    welcome = 'Welcome %s! If you ever want to quit, type {quit}
to exit.' % name
    client.send(bytes(welcome, "utf8"))
    msg = "%s has joined the chat!" % name
    broadcast(bytes(msg, "utf8"))
    clients[client] = name

    while True:
        msg = client.recv(BUFSIZ)
        if msg != bytes("{quit}", "utf8"):
            broadcast(msg, name+": ")
        else:
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name,
"utf8"))
            break
```

```
def broadcast(msg, prefix=""):  # prefix is for name
identification.
    """Broadcasts a message to all the clients."""

    for sock in clients:
        sock.send(bytes(prefix, "utf8")+msg)

clients = {}
addresses = {}

HOST = 'localhost'
PORT = 33000
BUFSIZ = 1024
ADDR = (HOST, PORT)

SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(ADDR)

if __name__ == "__main__":
    SERVER.listen(5)
    print("Waiting for connection...")
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
    ACCEPT_THREAD.start()
    ACCEPT_THREAD.join()
    SERVER.close()
```

client.py

```
HOST = 'localhost'
PORT = 33000
BUFSIZ = 1024
ADDR = (HOST, PORT)

client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect(ADDR)

receive_thread = Thread(target=receive)
receive_thread.start()
tkinter.mainloop()  # for start of GUI  Interface
```

References:

https://github.com/sachans/Chat-App