САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Web-программирование

Отчет

Лабораторная работа №1: Работа с сокетами

Выполнил:
Кривошапкина Айталина

Группа:
К33402

Проверил:
Говоров А. И.

Санкт-Петербург

2021 г.

Цель

Овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Ход работы

1. Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Файл server.py

```
1. import socket
2.
3. sock = socket.socket()
4. sock.bind(('', 9090))
5. sock.listen(1)
6.
7. conn, addr = sock.accept()
8.
9. print('connected:', addr)
10.
11.data = conn.recv(1024)
12.print(data.decode())
13.
14.conn.send(b'Hello, client')
15.
16.conn.close()
```

Файл client.py

```
1. import socket
2.
3. sock = socket.socket()
4. sock.connect(('', 9090))
5. sock.send(b'Hello, server')
6.
7. data = sock.recv(1024)
8. sock.close()
9.
10.print(data.decode())
```

2. Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции,

параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту. Вариант:

c. Поиск площади трапеции.

Файл server.py

```
import socket

host = socket.gethostbyname(socket.gethostname())
port = 1030

sock = socket.socket()
sock.bind((host, port))
sock.listen(1)

conn, addr = sock.accept()
print('connected:', addr)
data = conn.recv(1024).decode('utf-8')

data = data.split(',')
a = data[0]
b = data[1]
c = data[2]

S = 0.5*(int(a)+int(b))*int(c)
print(S)

conn.send(str(S).encode())
conn.close()
```

Файл client.py

```
import socket

sock = socket.socket()
sock.connect(('', 1030))

print('Поиск площади трапеции')
a = input('Введите длину первого основания: ')
b = input('Введите длину второго основания: ')
c = input('Введите длину высоты: ')

sock.send((a + ',' + b + ',' + c).encode())

data = sock.recv(1024)
sock.close()

print('Площадь трапеции равна: ', data.decode('utf-8'))
```

3. Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Задание: сделать сервер, который может:

- Принять и записать информацию о дисциплине и оценке по дисциплине.
- Отдать информацию обо всех оценах по дсициплине в виде html-страницы.

Файл server.py

```
import socket

GRADES = {}


class MyHTTPServer:
    def __init__(self, host, port):
        self.host = host
        self.port = port

    def serve_forever(self):
        serv_sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

        try:
            serv_sock.bind((self.host, self.port))
            serv_sock.listen()

            while True:
                conn, _ = serv_sock.accept()  # accept connection
                try:
                    self.serve_client(conn)
                except Exception as e:
                    print('Fail', e)
        finally:
            serv_sock.close()

    def serve_client(self, conn):
        try:
            req = self.parse_request(conn)
            resp = self.handle_request(req)
            self.send_response(conn, resp)
        except ConnectionResetError:
            conn = None

        if conn:
            conn.close()  # close connection

    def parse_request_line(self, rfile):
        line = rfile.readline(10**4)
        line = line.decode('utf-8')
        return line.split()

    def parse_request(self, conn):
        rfile = conn.makefile('rb')
        method, target, ver = self.parse_request_line(rfile)
```

```
46.
47.         request = {'data': {}, 'method': method}
48.         if '?' in target:
49.             request['method'] = 'POST'
50.             values = target.split('?')[1].split('&')
51.             for value in values:
52.                 a, b = value.split('=')
53.                 request['data'][a] = b
54.
55.         return request
56.
57.     def handle_request(self, req):
58.         if req['method'] == 'POST':
59.             return self.handle_post(req)
60.         else:
61.             return self.handle_get()
62.
63.     def handle_get(self):
64.         content_type = 'text/html; charset=utf-8'
65.         body = '<html><head><style></style></head><body>'
66.         body += '<form><label>Дисциплина</label><input
    name="discipline" /><br><label>Оценка</label><input
    name="grade"/><input type="submit"></form>'
67.         for subject in GRADES:
68.             body += f'<div><span>{subject}:
    {GRADES[subject]}</span></div>'
69.         body += '</div></body></html>'
70.         body = body.encode('utf-8')
71.         headers = [('Content-Type', content_type),
72.                    ('Content-Length', len(body))]
73.         return Response(200, 'OK', headers, body)
74.
75.     def handle_post(self, request):
76.         discipline = request['data']['discipline']
77.         grade = request['data']['grade']
78.
79.         if discipline not in GRADES:
80.             GRADES[discipline] = []
81.
82.         GRADES[discipline].append(grade)
83.
84.         return self.handle_get()
85.
86.     def send_response(self, conn, resp):
87.         rfile = conn.makefile('wb')
88.         status_line = f'HTTP/1.1 {resp.status} {resp.reason}\r\n'
89.         rfile.write(status_line.encode('utf-8'))
90.
91.         if resp.headers:
92.             for (key, value) in resp.headers:
93.                 header_line = f'{key}: {value}\r\n'
94.                 rfile.write(header_line.encode('utf-8'))
95.
96.         rfile.write(b'\r\n')
97.
98.         if resp.body:
99.             rfile.write(resp.body)
100.
101.            rfile.flush()
```

```
            rfile.close()


    class Response:
        def __init__(self, status, reason, headers=None, body=None):
            self.status = status
            self.reason = reason
            self.headers = headers
            self.body = body


    if __name__ == '__main__':
        serv = MyHTTPServer('127.0.0.1', 8000)
        serv.serve_forever()
```

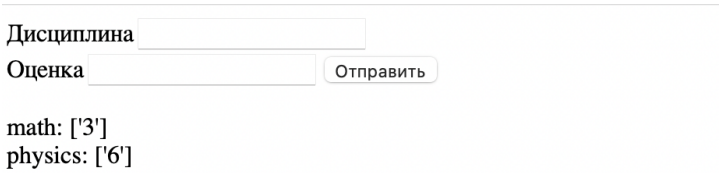

Рисунок 1 - Интерфейс добавления оценок

4. Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов.

Файл server.py

```
import socket, threading

host = socket.gethostbyname(socket.gethostname())
port = 1106

server = (host, port)

s = socket.socket()
s.bind(server)
s.listen()

print("[ Server Started ]")

clients = []

def trd(conn):
    while True:
        msg=conn.recv(1024)

        for client, addr in clients:
            client.send(msg)

while True:
```

```
    conn, addr = s.accept()

    if addr not in clients:
        clients.append((conn, addr))

    msg = conn.recv(1024)
    print(msg.decode("utf-8"))

    for client, addr in clients:
        client.send(msg)

    threading.Thread(target=trd, args=(conn, )).start()

else:
    print('\n[ Server Stopped ]')
    s.close()
```

Файл client.py

```
import socket, threading, time

host = socket.gethostbyname(socket.gethostname())
port = 1106

server = (host, port)

s = socket.socket()
s.connect(server)

shutdown = False

username = input("Name: ")

def receving(name, sock):
    while not shutdown:
        try:
            while True:
                data, addr = sock.recvfrom(1024)
                print(data.decode("utf-8"))

                time.sleep(0.2)
        except:
            pass


rT = threading.Thread(target=receving, args=("RecvThread", s))
rT.start()
s.sendto(("[" + username + "] => join chat ").encode("utf-8"),
server)

while not shutdown:
    try:
        message = input()

        if message != "":
            s.sendto(("[" + username + "] :: " +
message).encode("utf-8"), server)

```

```
38.         time.sleep(0.2)
39.     except:
40.         s.sendto(("[" + username + "] <= left chat
   ").encode("utf-8"), server)
41.         shutdown = True
42.
43.
44.rT.join()
45.s.close()
```

Вывод

В результате данной работы был изучены основы клиент-серверного взаимодействия, работа с сокетами, протокол HTTP, потоки в Python.