Целью работы было создать многопоточный Web-сервер на языке Python. Работает он следующим образом:

Вначале мы подключаемся к нужному порту, и прослушиваем на нем все соединения. Порт был выбран 8080, так как для стандартного порта 80 нужны права администратора, а они не всегда есть.

# Подключаемся к порту 8080

sock = socket.socket()

sock.bind(('', 8080))

sock.listen(5)

Далее запускаем бесконечный цикл, в котором будем отвечать на запросы:

# В бесконечном цикле

while True:

Как только нашлось соединение, с помощью конструкции start\_new\_thread запускаем новый обработчик (функцию ourserver):

# Если нашлось подключение

conn, addr = sock.accept()

print("Connected", addr)

# Плюс один поток

ThreadCount += 1

# Запускаем обработчик потока

start\_new\_thread(ourserver, (conn,ThreadCount))

Функция ourserver вначале читает данные, и проверяет, есть ли они:

# Функция для каждого потока

def ourserver(conn, threadcount):

print("Starting Thread Number " + str(threadcount))

# Читаем данные

data = conn.recv(8192)

# Если данных нет, ничего не делаем

if not data:

print("No data")

else:

# Если есть, выводим информацию

print("DATA IS: " + str(data))

Если данные есть, нужно а) декодировать их, б) разделить на строки, в) взять первую, г) разделить ее на слова, д) взять второе слово. Это будет название ресурса, которое нужно отдать:

# Получаем данные в виде строки

msg = data.decode()

# Делим на строки

msglines = msg.splitlines()

# Делим первую строку по пробелам

msgfl = msglines[0].split(' ');

# Второе значение - это ресурс, который нужно отобразить

resour = msgfl[1];

Если нас просят / (главную страницу), то это index.html:

# Если это главная страница, то index.html

if resour == '/':

resour = '/index.html'

Все, что мы можем отдать лежит в папке DATA, поэтому добавляем название папки, чтобы получить название файла, который нужно отдать:

# Копия названия ресурса, на случай, если нужно будет вывести ошибку

cresour = resour

# Имя файла, который нужно отобразить

resour = "./DATA/" + resour

Поработаем с заголовками. Во-первых, отдадим текущую дату:

# Заголовки

headers = ""

# Текущая дата

headers = headers + "Date: " + formatdate(timeval=None, localtime=False, usegmt=True) + "\n"

Во-вторых, тип файла (HTML) и его кодировка (UTF-8):

# Тип файла и его кодировка

headers = headers + "Content-type: text/html; charset=UTF-8\n"

В-третьих, описание сервера (любая строка):

# Описание сервера

headers = headers + "Server: Our brand new Python Web Server\n"

В-четвертых, сообщение о том, что соединение нужно сразу же закрыть:

# Заголовок, сообщающий о том, что соединение нужно закрыть

headers = headers + "Connection: close\n"

Теперь пытаемся открыть файл, и прочитать его содержимое, а также получить его размер:

# Пытаемся открыть нужный файл

try:

myfile = open(resour, "r")

# Читаем данные

data = myfile.read()

# Получаем их размер

datasize = len(data.encode())

Если получилось – добавляем заголовок с размером данных:

# Добавляем заголовок с размером данных

headers = headers + "Content-Length: " + str(datasize) + "\n"

Формируем строку, которую нужно отправить – код 200 (ОК), заголовки, и данные файла, а затем закрываем файл:

# Отправляем код 200 + заголовки + данные

resp = "HTTP/1.1 200 OK\n" + headers + "\n\n" + data

# Закрываем файл

myfile.close()

Если же файла такого не найдено, возвращаем код 404 (NOT FOUND), заголовки и заранее определенный текст.

except IOError:

# Если файла нет, выдаем ошибку 404

resp = "HTTP/1.1 404 NOTFOUND\n" + headers + "\n\n" + "FILE " + cresour + " WAS NOT FOUND!"

Отправляем ответ и закрываем соединение:

# Отправляем ответ

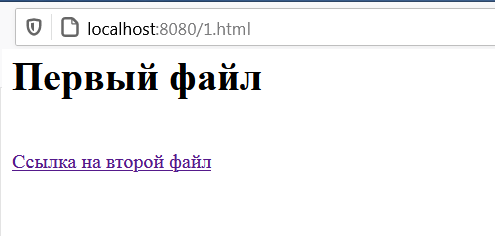
conn.sendall(resp.encode())

# Закрываем соединение

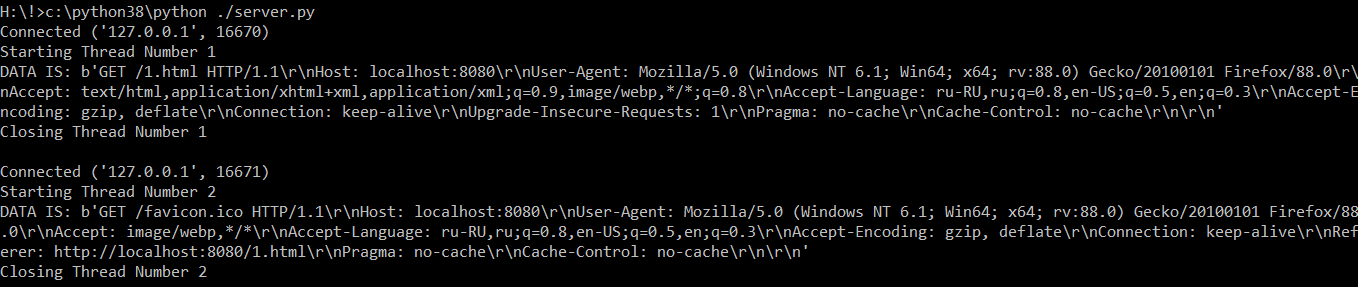
print("Closing Thread Number " + str(threadcount) + "\n")

conn.close()

Пример работы:

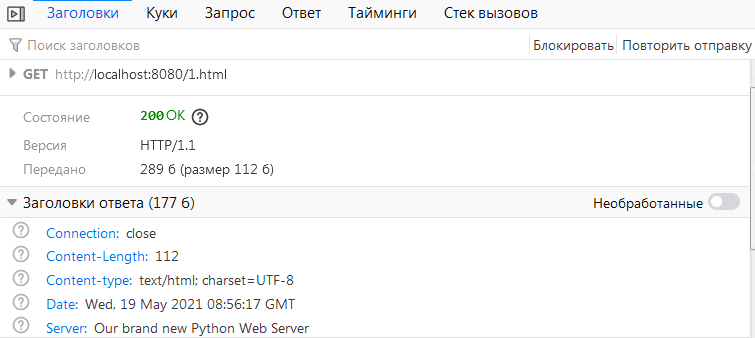


При своей работе сервер выводит в консоль диагностические сообщения:

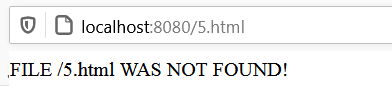


Видно, что браузер запросил главную страницу (/), а затем иконку сайта (/favicon.ico). Последней у нас не было.

Если открыть средства разработчика (F12), то можно увидеть, что необходимые заголовки отправляются:



Кроме того, если попросить несуществующую страницу, выведется ошибка:



А в заголовках будет видно, что код ошибки – 404:



Сервер может раздавать несколько файлов одновременно, так как работает с потоками.

Из улучшений, которые можно реализовать довольно быстро, можно отметить отсутствие поддержки любых типов файлов кроме HTML (так как в любом случае выдается в заголовках тип text/html). Можно добавить тип для CSS/JS/ICO, и других типов файлов.