ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

ECHO HTTP SERVER

LENGUAJES DE PROGRAMACIÓN

Tarik Saghouani Ben-Khalek

Pablo Sánchez Gómez

Eva María Hoyo de la Cruz

José Ignacio Manso LLanes

Contenido

Introducción

En esta práctica realizaremos las modificaciones necesarias en el servidor de la práctica 1 para que sea capaz de procesar peticiones http get desde un browser. Arrancaremos el servidor en el puerto 5005 y desde un browser haremos la petición:

http://127.0.0.1:5005/echo?message="Hello World!!!!"

El servidor responderá un mensaje http response que se envía desde el browser y el browser mostrará en pantalla el contenido de ese mensaje. Así mismo, el servidor sacará por pantalla el diccionario correspondiente a la petición get de acuerdo a la práctica 2.

Por ejemplo, para la petición anterior el servidor contestará con el mensaje siguiente.

"HTTP/1.1 200 OK\r\n
Cache-Control: no-store\r\n
Content-Length: 15\r\n
Content-Type: text/plain; charset=utf-8\r\n
Connection: close\r\n
\r\n
Hello World!!!!"

Hay que Tener en cuenta que el campo Content-Length habrá de modificarse para ajustarlo al tamaño del cuerpo del mensaje. Existe abundante información sobre el protocolo HTTP en Internet.

Entrega

Tal y como indica el enunciado de la practica esta entrega consta de dos partes. En la primera, en un archivo adjunto a este documento se encuentra el código del servidor, que además consta en el ANEXO I más adelante.

Adicionalmente, y para una mejor comprensión del código desarrollado por este grupo, una explicación de este.

Procedimiento de conexión

Inicialmente se debe configurar el socket que se va a utilizar y establecer los parámetros con sus valores iniciales. Estos valores están indicados en el enunciado de la práctica.

Siendo el siguiente fragmento de código el resultado:

```
import socket

servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = 'localhost'
port = 5005
print("starting up on %s port %d" % (ip, port))
servidor.bind((ip, port))
servidor.listen(1)
```

De este código queremos destacar la última línea, en la que se deja al servidor escuchando a la espera que de que se le haga una petición y que algún cliente se conecte a él.

En cuanto al parámetro que recibe la instrucción ***servidor.listen(1)***, indica el número de conexión paralelas que va a atender el servidor a la vez. En nuestro caso, una.

Respuesta del servidor

El primer paso tras configurar el servidor es prepararlo para que se quede esperando a que un cliente le haga una solicitud. Hemos escogido la espera activa que nos proporciona el bucle infinito ***while (true)***.

Una vez que un cliente establece conexión, el servidor debe atenderlo, y proporcionarle una respuesta a dicha solicitud.

Tal y como se pide en el enunciado de la práctica, se capturan los parámetros de la conexión con los que el cliente realiza la solicitud y se muestra por pantalla junto con algún mensaje más que sirve como control para saber que hace el código en cada instante. De esta manera se muestra cuando el servidor espera para realizar la conexión con el cliente, que ip tiene este, etc.

Después de esto, tal y como indica el enunciado, el servidor manda al cliente el siguiente mensaje:

"HTTP/1.1 200 OK\r\n
Cache-Control: no-store\r\n
Content-Length: 15\r\n
Content-Type: text/plain; charset=utf-8\r\n
Connection: close\r\n
\r\n
Hello World!!!!"

```
while True:
    print("Waiting for a connection.")
    cliente, direccion = servidor.accept()
    print("Conection from: %s %d" % (direccion[0], direccion[1]))
    resp = cliente.recv(44)
    respuesta=resp.decode()
    print("Received: %s" % respuesta)
    print("Sending data back to the client")

    mensajeget = respuesta.split('D')
    mensajegetimp = mensajeget[1].split('%20')
    mensajegetimp2 = mensajegetimp[1].split('%')
    mensajemandar=mensajegetimp[0]+" "+ mensajegetimp2[0]

    diccionarioget =  " HTTP/1.1 200 OK\r\n Cache-Control no-store\r\n
Content-Length 15\r\n Content-Type text/plain;charset=utf-8\r\n Connection
close\r\n "+mensajemandar


    print(diccionarioget)
```

```
    cliente.send(diccionarioget.encode())
    print("no more data from: %s %d" %(direccion[0], direccion[1]))
```

Del anterior fragmento de código queremos destacar como la siguiente instrucción modifica el tamaño del cuerpo del mensaje para ajustarlo y que se imprima por pantalla tal y como solicita el enunciado de la práctica.

```
diccionarioget =  " HTTP/1.1 200 OK\r\n Cache-Control no-store\r\n Content-
Length 15\r\n Content-Type text/plain;charset=utf-8\r\n Connection close\r\n
"+mensajemandar
```

Por último, hay que cerrar la conexión para poder seguir atendiendo más, en el caso de que las haya. Antes de esto se envía un mensaje al cliente diciendo que no hay más datos que enviar.

```
cliente.send(diccionarioget.encode())
print("no more data from: %s %d" %(direccion[0], direccion[1]))
cliente.close()
```

Capturas de las salidas

```
starting up on localhost port 5005
Waiting for a connection.
Conection from: 127.0.0.1 58862
Received: GET /echo?message=%E2%80%9DHello%20World!!!!
Sending data back to the client
 HTTP/1.1 200 OK
 Cache-Control no-store
 Content-Length 15
 Content-Type text/plain;charset=utf-8
 Connection close
 Hello World!!!!
no more data from: 127.0.0.1 58862
Waiting for a connection.
```

ANEXO I

```
import socket

servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = 'localhost'
port = 5005
print("starting up on %s port %d" % (ip, port))
servidor.bind((ip, port))
servidor.listen(1)

while True:
    print("Waiting for a connection.")
    cliente, direccion = servidor.accept()
    print("Conection from: %s %d" % (direccion[0], direccion[1]))
    resp = cliente.recv(44)
    respuesta=resp.decode()
    print("Received: %s" % respuesta)
    print("Sending data back to the client")

    mensajeget = respuesta.split('D')
    mensajegetimp = mensajeget[1].split('%20')
    mensajegetimp2 = mensajegetimp[1].split('%')
    mensajemandar=mensajegetimp[0]+" "+ mensajegetimp2[0]

    diccionarioget =  " HTTP/1.1 200 OK\r\n Cache-Control no-store\r\n
Content-Length 15\r\n Content-Type text/plain;charset=utf-8\r\n Connection
close\r\n "+mensajemandar


    print(diccionarioget)

    cliente.send(diccionarioget.encode())
    print("no more data from: %s %d" %(direccion[0], direccion[1]))
    cliente.close()
```