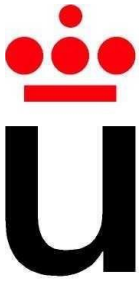
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA



Universidad
Rey Juan Carlos

HTTP FILE 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 3 para que sea capaz de servir ficheros en lugar de texto ante un http get. Suponiendo que tenemos arrancado el servidor en el puerto 5005 y desde un browser haremos la petición:

http://127.0.0.1:5005/form.html

El servidor responderá un mensaje http response con la cabecera de la práctica 3 donde el contenido del mensaje será el fichero form.html que estará en el directorio html dentro del directorio donde se encuentre el fichero del servidor (i.e: el servidor está en ~/projects/server.py, ~/projects/html/form.html).

Así la respuesta a la petición get generada por http://127.0.0.1:5005/form.html tendrá la forma:

"HTTP/1.1 200 OK\r\n
Cache-Control: no-store\r\n
Content-Length: <longitud del fichero>\r\n
Content-Type: text/html ; charset=utf-8\r\n
Connection: close\r\n
\r\n
<contenido del fichero>"

Asegurarse de que el Content-Type está puesto a text/html. Se puede encontrar el fichero form.html en el aula virtual. La práctica de esta semana concluye cuando el browser muestre el contenido del fichero.

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

Tal y como explicamos en la memoria de la practica 3, el primer paso es configurar el servidor para poder atender solicitudes/peticiones que realizar los clientes. Esto no ha cambiado con respecto a la anterior práctica. Lo primero es importar el módulo de los sockets, instanciar uno con los parámetros indicados en el enunciado de la práctica y dejar el servidor escuchando a la espera de las peticiones que le puedan llegar con una espera activa (tal y como explicamos en la anterior práctica, eso se hace con una espera activa implementada con un bucle ***While(True)***).

Fundamentalmente, lo que cambia en el nuevo código es precisamente lo que hay dentro del bucle while.

Antes de analizar el bucle, hay que destacar que se van a manipular ficheros. Para ello hay que importar la librería correspondiente, esto se realiza junto a la importación de la librería de sockets. Además de esto, antes del bucle ***WHILE***, se declaran dos variables ***ARRAY*** que usaremos más adelante.

Analizando ya el código que hay dentro del bucle, encontramos lo siguiente:

```
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)
respArray = respuesta.split()
conex = (respArray[2], "200", "OK")
respArray[0] = ' '.join(conex)
```

De nuevo el servidor espera a que un cliente (y solo uno) le haga una solicitud, y una vez que la haga, captura los parámetros de la conexión y muestra los mensajes que se indican en enunciado de la práctica. Como elemento nuevo, hay que destacar que se crea una variable llamada ***conex*** y una **respArray** en las que se guardan los parámetros de estado de la conexión.

En concreto el código 200 de http, esto es la respuesta del servidor http que devuelve el estatus de la petición correcta estándar para indicar que se va responde a la petición sin problemas.

Una vez que se ha realizado la conexión cliente-servidor correctamente, el servidor va a proporcionar con un fichero llamado ***form.html*** que estará en el directorio html dentro del directorio donde se encuentre el fichero del servidor.

Nota El parámetro para abrir este archivo, correspondiente a la ruta donde se encuentra este, se pasa como una ruta relativa que depende a la raíz de cada equipo. Una posible mejora es hacer esta ruta para que se pueda abrir en cualquier equipo sin tener que modificarla (como es el caso para esta práctica).

```
ruta = '../prac4/html'
filename =  ruta + respArray[1]
ruta2= ruta + '/form.html'
tam = os.path.getsize(ruta2)
file = open(ruta2, 'rb')
text = file.read(tam)
file.close()
```

Una vez se busca y se abre el archivo se guarda la información que necesitamos en la variable ***text*** y se cierra para evitar errores y efectos laterales en futuras ejecuciones del código.

Una vez realizada esta tarea, empezamos a generar la información que devuelve el código con la ayuda de la que hemos extraído del fichero ***form.html.***

Elementos nuevos

A continuación, se prepara la información que se va a enviar y se imprime por pantalla de la siguiente manera de la siguiente manera:

```
if "fname=" in respuesta:
    nombre = respuesta.split('fname=')
    ap = nombre[1].split('&lname=')
    ap1=ap[1].split(' HTTP')

contenido = {
    'Cache-Control': 'no-store',
    'Content-Length': tam,
    'Content-Type': 'text/html; charset=utf8',
    'Connection': 'close',
    'Contenido del fichero': ap[0] + " " + ap1[0],
}
respCont = ''.join('%s: %s\n' % (k, v) for k, v in contenido.items())
print(" ")

print(respCont)
print(text)
```

De este fragmento de código destacamos la siguiente línea de código:

```
respCont = ''.join('%s: %s\n' % (k, v) for k, v in contenido.items())
```

Con esta línea se da formato la información que se imprimirá continuación para que cumpla las restricciones impuestas en el enunciado de la práctica. Dicha información se obtenido en el if anterior para ser asignada en la variable en la variable ***contenido*** tal y como ha sido mostrado em la captura de pantalla de más arriba.

```
print(respCont)
print(text)
cliente.send(respArray[0].encode())
cliente.send(respCont.encode())
cliente.send('\n'.encode())
cliente.send(text)
cliente.close()
```

Por último, se imprime por pantalla en la salida de la terminal del entorno de programación la información de control que indica el enunciado de la practica y de la que hemos hablado más arriba. También se muestra en el explorador web la información del fichero. Y se finaliza enviado la información a proceso cliente que realizado la solicitud y se cierra el socket para poder atender otras peticiones, de la misma manera que se explico en la memoria de la anterior práctica.

Además se ha añadido un icono con formato favicon.ico para que se quite un error que saldría del formulario si no se añade, también para quitarlo se podrian quitar unas íneas de código del html pero se ha optado por añadir el icono, dentro de la carpeta html del proyecto.

Capturas de las salidas

Como hemos hablado más arriba, en esta practica hay dos salidas. La primera de ella es la que se produce por la salida de la terminal de nuestro entorno de desarrollo. En nuestro caso ***PyCharm***.

Haciendo un poco de zoom para que se vea mejor:

```
starting up on localhost port 5005
Waiting for a connection.
Conection from: 127.0.0.1 59319
Received: GET /form.html HTTP/1.1
Host: 127.0.0.1:500

Cache-Control: no-store
Content-Length: 470
Content-Type: text/html; charset=utf8
Connection: close
Contenido del fichero:

b'<html><head></head><body>\r\n\r\n<h2>HTML Forms</h2>\r\n
Waiting for a connection.
```

La otra salida que produce nuestro código es la que se produce en nuestro navegador web, en este caso ***Firefox.***

HTML Forms

First name:
John
Last name:
Doe

Submit

If you click the "Submit" button, the form-data will be sent to a page called "/register".

ANEXO I

```
import os
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)

ap = [" ", " "]
ap1 = [" ", " "]
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)
    respArray = respuesta.split()
    conex = (respArray[2], "200", "OK")
    respArray[0] = ' '.join(conex)

    ruta = '../prac4/html'
    filename =  ruta + respArray[1]
    ruta2= ruta + '/form.html'
    tam = os.path.getsize(ruta2)
    file = open(ruta2, 'rb')
    text = file.read(tam)
    file.close()


    if "fname=" in respuesta:
        nombre = respuesta.split('fname=')
        ap = nombre[1].split('&lname=')
        ap1=ap[1].split(' HTTP')

    contenido = {
        'Cache-Control': 'no-store',
        'Content-Length': tam,
        'Content-Type': 'text/html; charset=utf8',
        'Connection': 'close',
        'Contenido del fichero': ap[0] + " " + ap1[0],
    }
    respCont = ''.join('%s: %s\n' % (k, v) for k, v in contenido.items())
    print(" ")

    print(respCont)
    print(text)
    cliente.send(respArray[0].encode())
    cliente.send(respCont.encode())
    cliente.send('\n'.encode())
    cliente.send(text)
    cliente.close()
```