Instituto Tecnológico y de Estudios

Superiores de Occidente.

**Servidor y Aleatorios**

|  |  |
| --- | --- |
| Materia | Seguridad en redes |
| Profesor | Luis Julián Domínguez Pérez |
| Fecha | 15/04/2022 |
| Autor(es) | Rodolfo Luthe Narváez |

# Explicación del funcionamiento

## Cliente

|  |
| --- |
| import requests  import pseudorandom  pload = {'number':'"'+str(pseudorandom.pseudorand())+'"'}  r = requests.post('http://localhost:8080',data = pload)  # r = requests.post('http://localhost:8080')  print(r.text) |

Se importan las librerías de requests y pseudorandom para poder hacer la petición al servidor. Requests es una librería muy utilizada para enviar cosas por http y d esta manera interactuar con recursos web. Pseudorandom es la que se generó como parte de la tarea pasada.

Se crea un json con el número obtenido de pseudorandom y se manda por post al servidor corriendo en el puerto 8080 de localhost.

Al final se imprime la respuesta del servidor.

## Servidor

|  |
| --- |
| from http.server import BaseHTTPRequestHandler, HTTPServer  import logging  class S(BaseHTTPRequestHandler):      def \_set\_response(self):          self.send\_response(200)          self.send\_header('Content-type', 'text/html')          self.end\_headers()      def do\_GET(self):          logging.info("GET request,\nPath: %s\nHeaders:\n%s\n", str(self.path), str(self.headers))          self.\_set\_response()          self.wfile.write("GET request for {}".format(self.path).encode('utf-8'))      def do\_POST(self):          content\_length = int(self.headers['Content-Length']) # <--- Gets the size of data          post\_data = self.rfile.read(content\_length) # <--- Gets the data itself          logging.info("POST request,\nPath: %s\nHeaders:\n%s\n\nBody:\n%s\n",                  str(self.path), str(self.headers), post\_data.decode('utf-8'))          self.\_set\_response()          self.wfile.write("POST request for {}".format(self.path).encode('utf-8'))  def run(server\_class=HTTPServer, handler\_class=S, port=8080):      logging.basicConfig(level=logging.INFO)      server\_address = ('', port)      httpd = server\_class(server\_address, handler\_class)      logging.info('Starting httpd...\n')      try:          httpd.serve\_forever()      except KeyboardInterrupt:          pass      httpd.server\_close()      logging.info('Stopping httpd...\n')  if \_\_name\_\_ == '\_\_main\_\_':      from sys import argv      if len(argv) == 2:          run(port=int(argv[1]))      else:          run() |

El servidor es el que viene incluido en serpy pero desambiguado, puede recibir GET y POST. No le hace nada a las peticiones y solo contesta que tipo de petición fue y a que ruta. Lo que hace es hacer registro de cada petición con la librería de logging y esto lo usamos para evidenciar que el cliente envió el número.

El server usa http como su método de recibir peticiones, por default usa el puerto de http, que es el 8080, pero se puede especificar uno distinto si es necesario.

# Código a repositorio

<https://github.com/Rluthen/serverRandom>

# Bibliografía

|  |  |
| --- | --- |
| [1] | O. Smith, «Python Http Request Tutorial,» DataCamp, 19 09 2019. [En línea]. Available: https://www.datacamp.com/community/tutorials/making-http-requests-in-python. [Último acceso: 15 04 2022]. |
| [2] | mdonkers, «Simple Python 3 HTTP server for logging all GET and POST requests,» GitHub, 17 12 2019. [En línea]. Available: https://gist.github.com/mdonkers/63e115cc0c79b4f6b8b3a6b797e485c7. [Último acceso: 15 04 2022]. |
| [3] | «http.server — HTTP servers,» Python, [En línea]. Available: https://docs.python.org/3/library/http.server.html#module-http.server. [Último acceso: 15 04 2022]. |