



Ciências
ULisboa

| **Informática**

Servidores HTTP em Python

Pedro Ferreira, Vinicius Cogo

AD - TP05 | ©DI, Ciências, ULisboa



1. Servidores de HTTP em Python

2. Módulo *http.server*

3. Módulo *http.client*

1. Servidores de HTTP em Python

Opções clássicas:

- Sockets e implementação do HTTP
 - muito baixo nível!
- Biblioteca *standard* do Python, módulo *http.server*
 - pouco versátil para escrever código que responde aos pedidos
- Mecanismo *Common Gateway Interface* (CGI)
 - scripts invocados a cada pedido do cliente: pouco eficiente
- Usando o servidor WEB *Apache*: módulo *mod_python*
 - depende do software *Apache*

Todas estas soluções dependem de mecanismos diferentes de interface aplicação/servidor

2. Módulo *http.server*

Define a classe **SimpleHTTPRequestHandler**

- Por omissão serve ficheiros do diretório corrente (se não existir *index.html*), e dos seus diretórios descendentes.

```
import http.server
import socketserver

PORT = 8888
HOST = ""

http_handler = http.server.SimpleHTTPRequestHandler

http_server = socketserver.TCPServer((HOST,PORT), http_handler)

http_server.serve_forever()
```

Aceder a *http://localhost:8888*

2. Módulo *http.server*

É possível programar respostas aos pedidos

- Podemos redefinir funções dos métodos HTTP HEAD, GET, e os restantes métodos.

```
class MyHTTPHandler(http.server.SimpleHTTPRequestHandler):  
    def _set_headers(self):  
        self.send_response(200)  
        self.send_header('Content-type', 'text/html')  
        self.end_headers()  
  
    def do_GET(self):  
        self._set_headers()  
        print(self.path)  
        self.wfile.write("<html><body><p>Hello World!</p></body></html>")  
  
    def do_HEAD(self):  
        self.set_headers()  
  
    def do_POST(self):  
        # código para tratar método POST
```


2. Módulo *http.server*

É possível programar respostas aos pedidos

- Servidor usa uma classe baseada em SimpleHTTPRequestHandler
 - Classe implementa métodos do HTTP em funções **do_<MÉTODO>()**

```
import http.server
import socketserver

PORT = 8888
HOST = ""

# Colar aqui a classe definida no slide anterior

http_server = socketserver.TCPServer((HOST, PORT), MyHTTPHandler)
http_server.allow_reuse_address = True
http_server.serve_forever()
```

Aceder a <http://localhost:8888>

3. Módulo *http.client*

Cria uma conexão com o Servidor

- Executa pedidos num servidor HTTP.

```
import http.client

PORT = 8888
HOST = 'localhost'

http_conn = http.client.HTTPConnection(HOST, PORT)
body = 'Example of a message'
http_conn.request('POST', '/path/sub1/sub2', body)
resposta = http_conn.getresponse()
print(resposta.status, resposta.reason)
print(resposta.read().decode())
http_conn.close()
```

- Brandon Rhodes and John Goerzen.
Foundations of Python Network Programming, second edition, Apress.
- Python online documentation: http.server – HTTP servers
(<https://docs.python.org/3/library/http.server.html>)
- Python online documentation: SimpleHTTPServer – Simple HTTP request handler
(<https://docs.python.org/3/library/http.server.html#http.server.SimpleHTTPRequestHandler>)
- Python online documentation: http.client – HTTP protocol client
(<https://docs.python.org/3/library/http.client.html>)