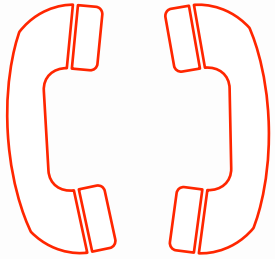
LINKgroup

Distance Learning System



HTTP Protokol i http paket

Web Application Building

Paket http

- **http** implementira HTTP protokol
- **http** sakriva operacije niskog nivoa (konektovanje, tcp pakete i slično)
- **http** je paket iz standardne python biblioteke i sadrži module: **server**, **client**, **cookies** i **cookiejar**

Modul server

- Modul **http.server**, sadrži klase potrebne za kreiranje http servera
- Dve najbitnije klase u ovom paketu su:

HTTPServer i **SimpleHTTPRequestHandler**

Nasleđuje klasu socketserver.TCPServer

Nasleđuje klasu BaseHTTPRequestHandler

Hosting statičkog sadržaja

- Statički sadržaj je sadržaj koji se distribuira klijentu u istom obliku u kome postoji na serveru
- Statički sadržaj su: HTML strane, slike, tekstualni fajlovi i slično
- Za distribuciju sadržaja potrebno je da server ima mogućnost čitanja HTTP zahteva i rada sa fajl sistemom
- Elementi modula http.server imaju podrazumevano ovu mogućnost

```
[# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

- Nakon komande sa slike, startuje se HTTP server koji distribuira fajlove iz direktorijuma u kome je komanda startovana

Hosting statičkog sadržaja
(wab-ex01 myfirstsite)

- Kreirajte fajl pod nazivom index.html i u njega stavite bilo koji tekstualni sadržaj, a zatim u direktorijumu u kome se fajl nalazi, startujte komandu:

  python -m http.server

- Nakon startovanja, pokušajte da, u browseru otvorite adresu iz konzole

index.html

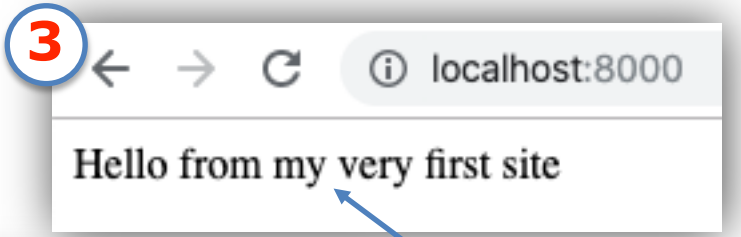
Hello from my very first site

1

2

```
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [07/Nov/2019 13:56:34] "GET / HTTP/1.1" 200 -
```

3

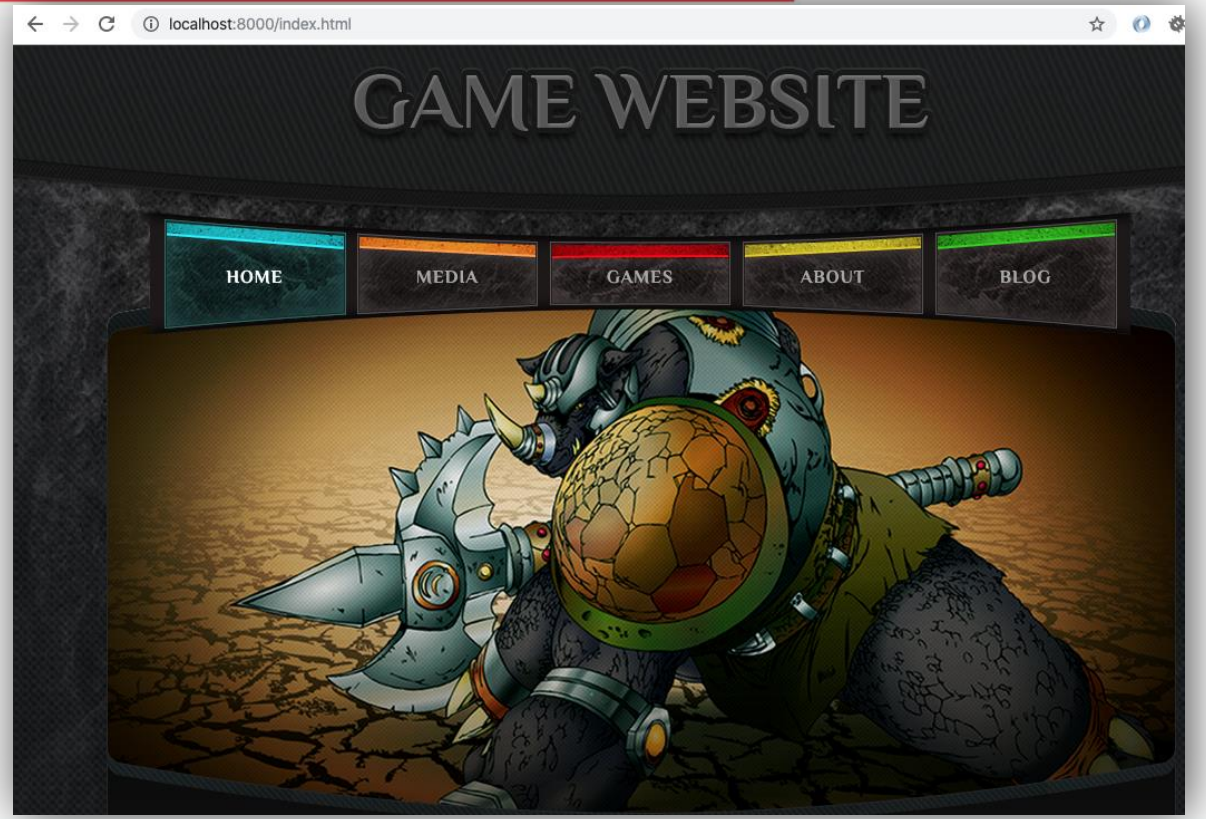


4

Vežba 1 (wab-ex01 gamesite)

- Kreirajte sajt pomoću nekog od šablona sa sledeće adrese

https://freewebsitetemplates.com/



Programabilno kreiranje servera

- Umesto da pustimo Python da uradi sve automatski možemo (i hoćemo) programabilno kontrolisati sve segmente izvršavanja
- Sledeća komanda kreira instancu klase HTTPServer i aktivira slušanje (efekat je isti kao startovanje servera iz konzole)

```
import http.server as server

s = server.HTTPServer(("localhost",8005),server.SimpleHTTPRequestHandler)
s.serve_forever()
```

Server nikada ne prekida petlju

Na svaki zahtev se pozivaju odgovarajuće metode ove klase

Bitniji članovi klase BaseHTTPRequestHandler

https://docs.python.org/3/library/http.server.html

- Klasa BaseHTTPRequestHandler je osnovna klasa za obradu HTTP zahteva u modulu http.server
- Ova klasa nasleđuje klasu socketserver.StreamRequestHandler i osim nasleđenih osobina, ima i sopstvene
- Za nas su interesantni članovi:

Svojstva

- **path** (putanja zahteva)
- **headers** (zaglavlja zahteva)
- **wfile** (izlazni tok)
- **rfile** (ulazni tok)

Metode

- **do_GET** (get zahtev)

Dinamički sadržaj (wab-ex01 timeserver)

- Problem kod dosadašnjeg pristupa je nemogućnost emitovanja dinamičkog sadržaja (na primer, na dostavljenoj stranici ne možemo prikazati tačno vreme)
- Ovaj problem se prevazilazi distribucijom dinamički generisanog sadržaja
- U trenutnom kontekstu, ovo se može postići posebnim hendlerom za CGI ili **sopstvenim hendlerom (primer u prilogu)**

```
import http.server as server
import datetime

class TimeHandler(server.SimpleHTTPRequestHandler):
    def do_GET(self):
        current_time = datetime.datetime.now()
        self.wfile.write(f"HTTP/1.1 200 OK\r\nConnection:close\r\n\r\n".encode("utf-8"))
        self.wfile.write(f"{current_time}".encode("utf-8"))

ss = server.HTTPServer(("0.0.0.0",8005),TimeHandler)
ss.serve_forever()
```

Dinamički sadržaj

- Iako je konačni efekat isti, možemo umesto ručno, zaglavlja i opis odgovora, upisati i pomoću metoda klase

```
import http.server as server
import datetime

class TimeHandler(server.SimpleHTTPRequestHandler):
    def do_GET(self):
        current_time = datetime.datetime.now()
        self.send_response(200)
        self.send_header("Connection","Close")
        self.send_header("Content-Type","text/html")
        self.end_headers()
        self.wfile.write(f"{current_time}".encode("utf-8"))

ss = server.HTTPServer(("0.0.0.0",8005),TimeHandler)
ss.serve_forever()
```

Common Gateway Interface skripte

https://docs.python.org/3.7/library/cgi.html

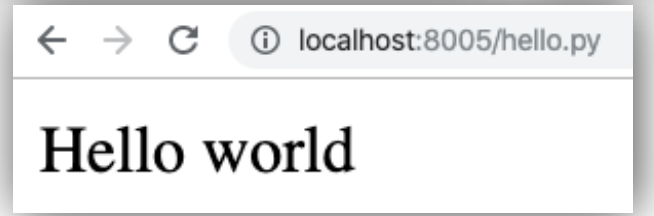
- Common Gateway Interface (CGI) skripte su programi koji se izvršavaju na serveru a inicirani su nekim HTTP zahtevom
- CGI skripta ima na raspolaganju kompletan kontekst zahteva (ulazni tok, zaglavlja, adresu pošiljaoca i slično)

main.py

```
handler = server.CGIHTTPRequestHandler
handler.cgi_directories = ["/"]
ss = server.HTTPServer(("0.0.0.0",8005),UploadHandler)
ss.serve_forever()
```

hello.py

```
#!/usr/bin/env python3
print("Content-type:text/html")
print("")
print("Hello world")
```



Dinamičke skripte (wab-ex01 dynscripts)

- Ako bismo u metodi hendlera obrađivali sve slučajeve, on bi ubrzo postao nepregledan

- Umesto toga, hendler može služiti samo kao dispečer koji će učitavati odgovarajuće tipove dokumenata ili startovati skripte, na osnovu nekog kriterijuma

- U primeru, hendler analizira dokument iz opisa zahteva i na osnovu njega pokušava da učita modul. Ukoliko ne uspe, prepušta obradu dokumenta roditeljskoj metodi do_GET

```
import http.server as server
import importlib

class DynamicRequestHandler(server.SimpleHTTPRequestHandler):
    def do_GET(self):
        page = self.path.replace("/","")
        page = f"{page}_mod"
        try:
            module = importlib.import_module(page)
            self.send_response(200)
            self.send_header("Content-Type","text/html")
            self.send_header("Connection","Close")
            self.end_headers()
            module.do_GET(self)
        except:
            return super().do_GET()

ss = server.HTTPServer(("0.0.0.0",8005),DynamicRequestHandler)
ss.serve_forever()
```