Flask

A unicorn is here to see you.

Introdução

- Flask é um micro framework escrito em Python para a web
- Micro porque a sua base de funcionalidades é extremamente reduzida, apesar das suas inúmeras extensões
- Utilizado para criar sítios na Web e **API's**
- Exemplos de outros frameworks:
  - ASP.net (C#), Play (Java), Ruby-on-Rails (Ruby), Django (Python)

Introdução

- Flask é baseado nos projectos Werkzeug, Jinja 2 e boas intenções ☺
- Porque Flask ?
  - Fácil de aprender
  - Pythonico (adopta os princípios e estilos de programação do Python)
  - Pequeno/Leve mas escalável para grandes aplicações
  - Rotas definidas através de decoradores
  - Multitude de Plugins

Ambiente de desenvolvimento

- Criação de um virtualenv

```
mkdir projecto
cd projecto
mkdir static
mkdir templates
touch app.py
chmod +x app.py
python3 –m venv venv
source venv/bin/activate
pip install Flask
```

Hello World

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
@app.route("/index")
def index():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

- Executar projecto

```
python projecto/app.py
```

Routing/Redirect

```
@app.route("/user/<name>")
def say_hi(name):
      return f"Hi {name}"

@app.route("/index.html")
def index2():
      return redirect(url_for('index’)) #index é nome da função
```

Static

- Ficheiros na pasta `static` são servidos directamente (sem necessidade de qualquer código)
  - Util para colocar imagens, JavaScripts, CSS, etc

Templates com Jinja2

- Flask configura o sistema de templates Jinja2 automaticamente
  - Render_template
- Jinja2 é um motor de templates para Python
  - Permite mapear variaveis python no template
  - Permite implementar alguma lógica e ciclos (`if, for`, etc)

Templates com Jinja2

hello.html

```
<!doctype html>
<title>Hello</title>
{% if name %}
    <h1>Hello {{ name }}</h1>
{% else %}
    <h1>Hello World</h1>
{% endif %}
```

app.py

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/user")
@app.route("/user/<name>")
def say_hi(name=None):
    return render_template("hello.html",
                           name=name)
if __name__ == "__main__":
    app.run()
```

Templates com Jinja2

users.html

```
<!doctype html>
<title>Hello</title>
{% for user in users %}
    <h1>Hello {{ user }}</h1>
{% endfor %}
```

app.py

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/users")
def list_users():
    users = [“Diogo”, “Gomes”]
    return render_template(”users.html",
                           users=users)
if __name__ == "__main__":
    app.run()
```

Metodos HTTP

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST’:
        return f"Welcome { request.form['username'] }"
    else:
        return "curl -X POST -F 'username=dgomes' http://localhost:5000/login"
```

Referências

- https://pepa.holla.cz/wp-content/uploads/2015/11/Flask-Web-Development.pdf

Python Requests

- Biblioteca de alto nível para interacção com servidores HTTP
- Baseada na biblioteca urllib3
- Porque usar ?
  - Keep-Alive & Connection Pooling
  - Sessões com persistência de cookies
  - Autenticação Básica/Digest
  - Upload de ficheiros usando Multipart

Pedido GET

```
import requests

if __name__ == "__main__":
    r = requests.get("https://www.ua.pt/manifest.json")
    print(r.text)
```

- Ou melhor:

```
if __name__ == "__main__":
      r = requests.get("https://www.ua.pt/manifest.json")
      data = r.json()
      print(data["name"])
```

Pedido POST

```
if __name__ == "__main__":
    url = “https://httpbin.org/post”
    payload = {"name": "Diogo"}
    response = requests.post(url, json = payload)
    print(response.text)
```