Flask

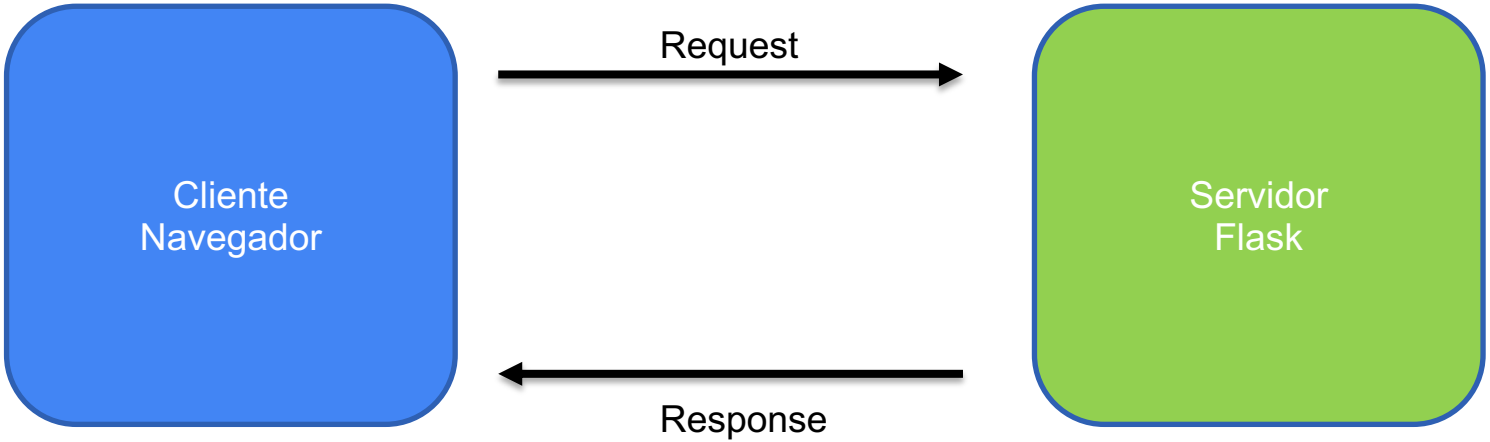
Microframework em Python

O que significa “micro” ?

- O “micro” em microframework significa que o Flask visa manter o núcleo simples, mas extensível.
- O Flask não tomará muitas decisões por você.
- Para que o Flask possa ser tudo que você precisa e nada do que você não precisa.

fonte: https://flask.palletsprojects.com/en/2.0.x/foreword/

Flask



Instalação

- Versão do Python
  - O Flask suporta Python 3.6 e mais recente.

- Dependências:
  - **Werkzeug** - implementa WSGI, a interface Python padrão entre aplicativos e servidores.
  - **Jinja** - é uma linguagem de template que renderiza as páginas (HTML).

Ambientes virtuais

- Use um ambiente virtual para gerenciar as dependências do seu projeto, tanto no desenvolvimento quanto na produção.

- Que problema um ambiente virtual resolve?
  - Ambientes virtuais são grupos independentes de bibliotecas Python, uma para cada projeto. Os pacotes instalados para um projeto não afetarão outros projetos ou os pacotes do Sistema Operacional.

Criando um ambiente virtual

> mkdir meuprojeto

> cd meuprojeto

- py -3 -m venv venv (criando um ambiente virtual com o nome venv)
- **python –m venv venv**

Ative o ambiente virtual

> venv\Scripts\activate

Desative o ambiente

> deactivate

Instalando o Flask

- pip install flask (instalando o flask com o gerenciador pip)
  - Com o python3.x utilize pip3 ...

> flask --version (verificar a versão do flask instalado)

Ambiente de desenvolvimento

$ pip freeze (retorna tudo que foi instalado no ambiente virtual)

$ pip freeze > requirements.txt (gravando a saída do freeze em um arquivo txt)

$ pip install -r requirements.txt (instalando os pacotes em outro ambiente virtual)

Primeiro exemplo em Flask

app.py

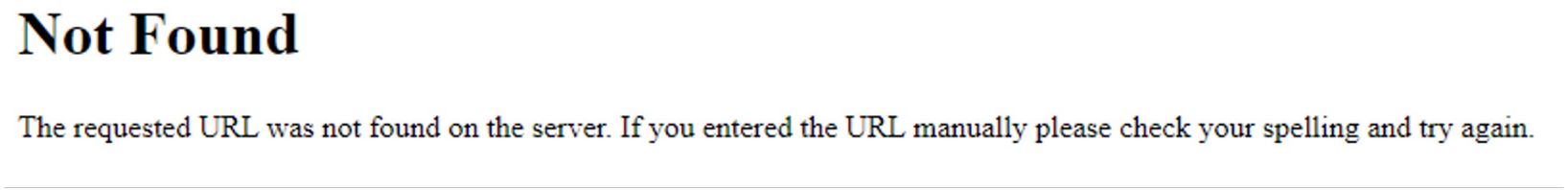
```python
from flask import Flask

app = Flask("__name__")
```

> flask run

```
PS C:\Users\JeanCarlos\Documents\flask> flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [31/Aug/2021 21:25:18] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [31/Aug/2021 21:25:23] "GET / HTTP/1.1" 404 -
```

localhost

navegador -> http://127.0.0.1:5000/



Uma aplicação mínima em Flask

```
from flask import Flask #Importa classe flask

app = Flask(__name__) # cria uma instância dessa classe

@app.route("/") # criando rotas com decorator

def hello_world(): # função para retornar uma mensagem

    return "<p>Hello, Flask!</p>"
```

Primeira aplicação em Flask

Etapas:

1 - Importação da classe Flask

2 - Cria uma instância desta classe. O primeiro argumento é o nome do módulo, (__name__). Isso será necessário para que o Flask sabia onde procurar recursos como templates por exemplo.

3 - Usamos decorator para criamos rotas(), para dizer qual URL deve acionar esse função.

4 - A função retorna uma mensagem que queremos exibir no navegador.

Execução do aplicativo

Save o arquivo com **app.py** ou qualquer nome semelhante.

- Nota: Não salvar com o nome **flask.py,** pois entraria em conflito com o próprio flask.

Execução:

- flask run (caso o nome do arquivo for **app.py**)

Exportando a variável de ambiente

- **BASH**

```
$ export FLASK_APP=hello

$ flask run
```

- **CMD**

```
> set FLASK_APP=hello

> flask run
```

- **Powershell**

```
> $env:FLASK_APP = "hello"

> flask run
```

Exemplo para o arquivo hello.py
Se o arquivo for nomeado app.py ou wsgi.py, você não precisa definir a FLASK_APP

Habilitar recurso de desenvolvimento

Iremos definir a **FLASK_ENV** variável de ambiente como **development** antes de chamar **flask run**

Bash

```
$ export FLASK_ENV=development

$ flask run
```

CMD

```
> set FLASK_ENV=development

> flask run
```

```
* Serving Flask app "hello" (lazy loading)
* Environment: development
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 279-466-287
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Powershell

```
> $env:FLASK_ENV = "development"

> flask run
```

Rotas ou Roteamento

Os aplicativos da web modernos usam URLs significativos para ajudar os usuários.

- Use o **route()** “É um decorador para vincular uma função a um URL”

```
@app.route('/')
def index():
    return 'Página index'
@app.route('/hello')
def hello():
    return 'Hello, World'
```

Modelos de renderização

Gerar HTML a partir do Python não é divertido. Por causa disso, o Flask configura o mecanismo de template Jinja2 para você automaticamente.

Para renderizar um modelo, você pode usar o método **render_template()**. Tudo que você precisa fazer é fornecer o nome do modelo e as variáveis que deseja passar para o mecanismo de modelo como argumentos de palavra-chave.

Modelos de renderização

```
from flask import render_template

@app.route('/hello/')

@app.route('/hello/<name>')

def index():

    return render_template('index.html')
```

Modelos de renderização

O Flask procurará modelos na pasta templates.

```
/app.py
/templates
    /index.html
```

Arquivos estáticos

Os aplicativos da web dinâmicos também precisam de arquivos estáticos, como **Imagens, CSS e JavaScript** durante o desenvolvimento, o Flask também pode fazer isso. Basta criar uma pasta chamada “**static**” seu diretório padrão.

Para gerar URLs para arquivos estáticos, use:

```
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
```

O arquivo deve ser armazenado no sistema de arquivos como `static/style.css`