02

Primeira aplicação

Transcrição

Nesse curso de Flask, iremos criar nossa primeira aplicação — e isso se dará bem rapidamente, com poucas linhas de código.

O Flask é considerado um **microframework**, o que basicamente significa que ele não vem com muitas soluções embutidas para resolver problemas que não foram apresentados ainda. Dessa forma, o Flask é mais flexível: à medida em que os problemas vão aparecendo, nós os resolvemos com soluções que escolhemos no caminho, sem nos prendermos a soluções fixas.

Para começarmos a estudar o Flask, primeiro precisamos entender como funciona a Web. Quando digitamos um endereço no navegador, ele é enviado (a partir do HTTP) para um servidor, por meio de uma requisição (*Request*). Essa requisição é processada no servidor e enviada de volta ao navegador na forma de uma resposta (*response*), também por meio do HTTP.

Com essa resposta, o navegador irá renderizar o HTML ou até uma mensagem. O importante é sabermos que, no nosso caso, todo o processamento se dará no Flask.

Isso posto, vamos iniciar o Pycharm, no qual criaremos o primeiro projeto, que terá o nome da aplicação que criaremos durante o curso, **jogoteca**. Como interpretador, selecionaremos o **Python 3**.

Clicando com o botão direito do mouse no projeto **jogoteca** e, em seguida, em "New > Python file". Criaremos um arquivo Python no qual escreveremos nossa aplicação, também chamado "jogoteca".

Para essa aplicação funcionar, inicialmente, precisaremos importar o pacote do Flask, o que é feito com `import flask`. A palavra `flask` será sublinhada em vermelho, pois ele não está instalado nessa versão do Python, nem na nossa IDE.

Como podemos importar um pacote da internet para o nosso ambiente Python? Bom, podemos simplesmente clicar na lâmpada que indica o erro e selecionar "Install package flask", mas não faremos isso no momento, pois queremos observar como os mecanismos dessa linguagem funcionam.

Portanto, abriremos o Terminal e usaremos uma ferramenta disponibilizada pelo Python para instalar pacotes, chamada **PIP 3**, e o comando `install`.

Em seguida, uma das possibilidades é utilizarmos o comando `pip 3 install flask`. Se fizermos isso, a IDE fará o download e a instalação da versão mais recente do Flask. Mas, será que isso é recomendado?

Estar atualizado pode ser interessante, mas também traz pontos negativos à execução e manutenção do nosso projeto, já que versões diferentes podem quebrar o nosso código, gerando a necessidade de corrigi-lo a todo momento.

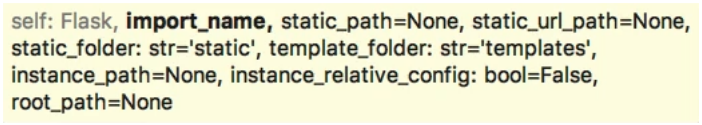
Portanto, utilizaremos o comando `pip3 install flask==0.12.2`, que é a versão que utilizaremos nesse curso. O Flask será instalado com sucesso e poderemos voltar ao nosso arquivo Python.

O comando `import flask` importa todo o pacote Flask e, no momento, precisamos somente da classe `Flask`, não do pacote inteiro. Para importarmos uma classe de um pacote, usaremos `from flask import Flask` — ou seja, "importar a classe Flask do pacote flask".

Agora, precisaremos criar um objeto do tipo `Flask` que representará nossa aplicação — no caso, uma variável `app` que receberá o objeto a ser instanciado:

```
app = Flask()
```

A IDE já nos indicará quais parâmetros podem ser utilizados para instanciar essa classe, como `import_name`, `static_path=None`, `static_url_path=None`, entre outros.



Dentre eles, o único obrigatório é `import_name`, que nada mais é do que o nome do pacote/módulo que estamos executando, até mesmo para o Flask saber onde buscar os arquivos. Todos os outros são parâmetros nomeados e têm valores pré-definidos.

Para passarmos o nome do nosso módulo, utilizaremos `__name__`:

```
app = Flask(__name__)
```

Com isso, já temos uma aplicação. Para rodar essa aplicação é muito simples: basta digitarmos `app.run()` (para chamar o método `run()` do objeto `app`, clicar com o botão direito do mouse no nosso arquivo e em seguida em "Run 'jogoteca'". O console exibirá a seguinte mensagem:

- Running on http://127.0.0.1:5000/ (http://127.0.0.1:5000/) (Press CTRL+C to quit)

A porta `5000` é a porta padrão na qual a IDE sobe a nossa aplicação. Para visualizarmos essa aplicação na web, basta clicarmos no próprio link no console. O navegador exibirá a seguinte mensagem:

> **Not Found**
> The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

Isso significa que o navegador não encontrou conteúdos para mostrar. Então, a partir desse momento, precisamos definir algum conteúdo para ser mostrado. Dessa forma, quando enviarmos uma requisição, teremos uma resposta.

No Flask, é necessário informar à nossa aplicação que temos algum caminho definido. Por exemplo, em vez do caminho definido http://127.0.0.1:5000/ (http://127.0.0.1:5000/), passaremos o caminho http://127.0.0.1:5000/inicio (http://127.0.0.1:5000/inicio), esperando que esse caminho nos retorne uma mensagem.

Para configurarmos uma mensagem a ser mostrada, precisamos primeiro criar uma função. No caso, criaremos a função `ola()`, que simplesmente nos retornará um texto. Esse texto será um HTML:

```
def ola():
    return '<h1>Olá Flask!</h1>'
```

Observação: Escrever HTML em um código Python não é uma boa ideia, mas melhoraremos isso no futuro!

Para o Flask conhecer uma função e saber que deve executá-la quando alguém chamar uma rota, precisamos indicar isso para a aplicação. Isso pode ser feito por meio da seguinte diretiva:

```
@app.route().
```

Dentro dela, colocaremos a configuração da nossa rota, que é `/inicio` :

```
from flask import Flask

app = Flask(__name__)

@app.route('/inicio')
def ola():
    return '<h1>Olá Flask!</h1>'

app.run()
```

Para testarmos se nossa aplicação funciona, vamos reiniciar a aplicação clicando no botão "Rerun" (ou pressionando "Ctrl + F5"). No navegador, acessaremos a URL http://127.0.0.1:5000/inicio (http://127.0.0.1:5000/inicio). O resultado será a mensagem "Olá Flask!", do jeitinho que esperávamos. Note que conseguimos fazer isso utilizando somente 6 linhas de código.

Como você pode perceber, é bem rápido fazer uma aplicação com Flask. Nos vídeos seguintes aprenderemos a mostrar um conteúdo mais dinâmico na página. Até lá!