



Flask

Flask

Programadores Python que desejam aprender a programar nesta linguagem utilizando o Flask.

Pré-requisitos: Python.

Índice

Copyright	4
Equipe	5
Histórico de edições	5
Capítulo 01 - Conhecendo o Flask	6
Características do Flask	6
Capítulo 02 - Configurando o ambiente	7
Capítulo 03 - Comunicação com banco de dados	9

Copyright

As informações contidas neste material se referem ao curso de Lógica de Programação e estão sujeitas às alterações sem comunicação prévia, não representando um compromisso por parte do autor em atualização automática de futuras versões.

A Apex não será responsável por quaisquer erros ou por danos acidentais ou consequenciais relacionados com o fornecimento, desempenho, ou uso desta apostila ou os exemplos contidos aqui.

Os exemplos de empresas, organizações, produtos, nomes de domínio, endereços de email, logotipos, pessoas, lugares e eventos aqui apresentados são fictícios. Nenhuma associação a empresas, organizações, produtos, nomes de domínio, endereços de email, logotipos, pessoas, lugares ou eventos reais é intencional ou deve ser inferida.

A reprodução, adaptação, ou tradução deste manual mesmo que parcial, para qualquer finalidade é proibida sem autorização prévia por escrito da Apex, exceto as permitidas sob as leis de direito autoral.

Equipe

Conteúdos	Diagramação	Revisão
Gustavo Rosauro	Fernanda Pereira	Fernanda Pereira

Histórico de edições

Edição	Idioma	Edição
1ª	Português	Março de 2020

Capítulo 01 - Conhecendo o Flask

Flask é um micro-framework multiplataforma escrito em Python que provê um modelo simples para o desenvolvimento web.

Micro-Framework são Frameworks modularizados que possuem uma estrutura inicial muito mais simples quando comparado a um Framework convencional e são muito utilizados para a criação de microsserviços, como APIs RESTful. Lançado em 2010 e desenvolvido por Armin Ronacher, Flask é destinado principalmente a pequenas aplicações com requisitos mais simples, como por exemplo, a criação de um site básico.

Possui um núcleo simples e expansível que permite que um projeto possua apenas os recursos necessários para sua execução (conforme surja a necessidade, um novo pacote pode ser adicionado para incrementar as funcionalidades da aplicação).

Características do Flask

Simplicidade: Por possuir apenas o necessário para o desenvolvimento de uma aplicação, um projeto escrito com Flask é mais simples se comparado aos frameworks maiores, já que a quantidade de arquivos é muito menor e sua arquitetura é muito mais simples.

Rapidez no desenvolvimento: Com o Flask, o desenvolvedor se preocupa em apenas desenvolver o necessário para um projeto, sem a necessidade de realizar configurações que muitas vezes não são utilizadas.

Projetos menores: Por possuir uma arquitetura muito simples (um único arquivo inicial) os projetos escritos em Flask tendem a ser menores e mais leves se comparados a frameworks maiores.

Aplicações robustas: Apesar de ser um micro-framework, o Flask permite a criação de aplicações robustas, já que é totalmente personalizável, permitindo, caso necessário, a criação de uma arquitetura mais definida.

Capítulo 02 - Configurando o ambiente

Vamos criar o repositório de pacotes utilizando o seguinte comando `python -m venv env`.

```
(venv) C:\Users\Acer\PycharmProjects\CrudFlaskPython>python -m venv env
```

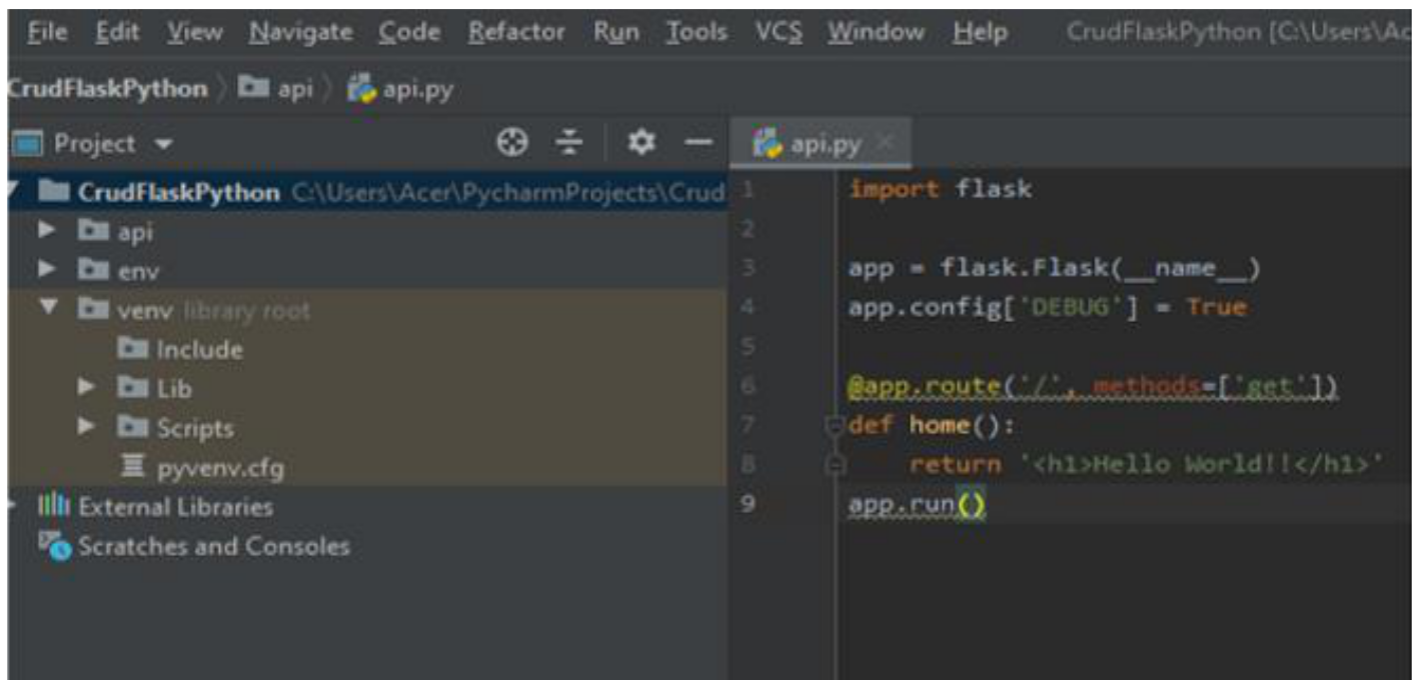
Vá na pasta scripts e execute o comando `activate`

```
(venv) C:\Users\Acer\PycharmProjects\CrudFlaskPython\env\Scripts>activate
```

Quando executar esse comando ele irá trocar o venv que se encontra no final, pelo env que foi criado, que é o novo repositório. Agora, execute o comando `pip install flask` para instalar o flask, ele permitirá criar uma api rest e fazer requisições.

```
★ (env) C:\Users\Acer\PycharmProjects\CrudFlaskPython>pip install flask  
Collecting flask
```

Agora crie uma pasta chamada api para referenciar que nessa pasta estará os arquivos da api. Finalizado essa parte, crie um método get para testar a api.



Feito essa parte, acesse o caminho do arquivo e execute o comando `python api.py` ou `python <nome do seu arquivo>`

```
(env) C:\Users\Acer\PycharmProjects\CrudFlaskPython\api>python api.py
* Serving Flask app "api" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
```

Capítulo 03 - Comunicação com banco de dados

Instale um framework para fazer a comunicação com o banco de dados mysql pip install flask-mysqldb.

```
(env) C:\Users\Acer\PycharmProjects\CrudFlaskPython>pip install flask-mysqldb
Collecting flask-mysqldb
  Downloading Flask-MySQLdb-0.2.0.tar.gz (2.1 kB)
Requirement already satisfied: Flask>=0.10 in c:\users\acer\pycharmprojects\crudflaskpython\env\lib\site-packages (from flask-mysqldb) (1.1.1)
```

Agora, execute o comando para que depois qualquer api possa comunicar com a aplicação é o comando pip install -U flask_cors

```
(env) C:\Users\Acer\PycharmProjects\CrudFlaskPython>pip install -U flask-cors
Requirement already up-to-date: flask-cors in c:\users\acer\pycharmprojects\crudflaskpython\env\lib\site-packages (3.0.8)
Requirement already satisfied, skipping upgrade: Six in c:\users\acer\pycharmprojects\crudflaskpython\env\lib\site-packages
★ Requirement already satisfied, skipping upgrade: Flask>=0.9 in c:\users\acer\pycharmprojects\crudflaskpython\env\lib\site-packages
Requirement already satisfied, skipping upgrade: Jinja2>=2.10.1 in c:\users\acer\pycharmprojects\crudflaskpython\env\lib\site-packages
```

E configure a aplicação conforme demonstra a imagem abaixo:

```
1 import json
2
3 from flask import Flask, request, Response, jsonify
4 from flask_cors import CORS
5 from flask_mysqldb import MySQL
6 app = Flask(__name__)
7 app.config['DEBUG'] = True
8 app.config['MYSQL_HOST'] = '127.0.0.1'
9 app.config['MYSQL_USER'] = 'root'
10 app.config['MYSQL_PASSWORD'] = ''
11 app.config['MYSQL_DB'] = 'apex'
12 mysql = MySQL(app)
13 cors = CORS(app, resources={r"*": {"origins": "*"}})
14 @app.route('/', methods=['get'])
15 def home():
16     cur = mysql.connect.cursor()
17     cur.execute('SELECT * FROM ALUNO')
18     data = cur.fetchall()
19     cur.close()
20     response = Response(json.dumps(data), mimetype="application/json")
21     return response
22
23 app.run(debug=True)
```


Dessa maneira faz-se uma consulta em sua base. Inserindo um registro na base utilizando flask, para isso faremos um método utilizando POST.

Agora tem uma diferença, terás que utilizar uma transação, para que quando inserir um registro, se não acontecer nenhum erro até o final, daremos um commit isso serve para informar que essa alteração na tabela pode ser salva.

Para deletar um registro, utilize a requisição delete, onde irás passar o id para o método.

```
41
42 @app.route('/<int:id>', methods=['delete'])
43 def deletar(id):
44     db = mysql.connect
45     cur = db.cursor()
46     sqlDelete = "DELETE FROM ALUNO WHERE ID =" + str(id)
47     cur.execute(sqlDelete)
48     db.commit()
49     return jsonify("removido com sucesso !")
50
```

Para alterar um registro, utilize a requisição put onde também irá passar o id para o método

```
51 @app.route('/<int:id>', methods=['put'])
52 def editar(id):
53     db = mysql.connect
54     cur = db.cursor()
55     dados = (request.json['nome'], request.json['idade'])
56     sqlUpdate = "update aluno set nome = %s, idade = %s where id =" + str(id)
57     cur.execute(sqlUpdate, dados)
58     db.commit()
59     return jsonify("Alterado com sucesso!!")
60
61
62 app.run(debug=True)
```