

Introdução ao Flask

Usabilidade, desenvolvimento web, mobile e jogos

Prof. Me. Gustavo Torres Custódio
gustavo.custodio@anhembi.br

Conteúdo

Introdução

Flask

Exercícios



Introducao ao Flask

Introdução

Frontend e Backend

- Até o momento, construímos o *frontend* de um site, ou seja, a porção gráfico do site, que é executada do lado do usuário.
- O *backend* (lado do servidor) é responsável por armazenar e organizar dados, além de fornecer informações para o cliente.

Frontend e Backend

- Entre tarefas que o *backend* executa:
 - Gerar páginas customizadas para diferentes usuários.
 - Processar formulários enviados pelo usuário.
 - Consultar e cadastrar informações no banco de dados.
 - Validar formulários de *login* e senha.



Introducao ao Flask

Flask

Flask



Flask

Flask

- O Flask é um micro framework multiplataforma escrito em Python utilizado para desenvolvimento web.
 - Dedicado para pequenas aplicações.
- Ele possui duas dependências principais:
 - O *Web Server Gateway Interface* (WSGI) vem do Werkzeug.
 - Os templates são fornecidos pelo Jinja2.

Flask

- Primeiro criamos um ambiente virtual em Python para trabalhar com o Flask.
 - Isso evita inconsistências de diferentes versões instaladas.
- Dentro de um terminal digite:
 - `virtualenv --version`.
 - Se algum problema for encontrado, primeiro instale o `virtualenv`.
 - `pip install virtualenv`

Flask

- Crie um ambiente virtual:
 - `virtualenv venv`
- Ative o ambiente virtual:
 - Em Windows: `venv\Scripts\activate`
 - Em Linux e Mac: `source venv/bin/activate`

Flask

- Quando terminar de criar o ambiente virtual, instale o Flask.
 - `pip install Flask`.
- Após instalado, vamos criar um arquivo chamado `backend.py`:

Flask

```
from flask import Flask

app = Flask(__name__)
app.config["TEMPLATES_AUTO_RELOAD"] = True

@app.route('/')
def index():
    return '<h1>Olá Mundo!</h1>'
```

Flask

- Utilize o comando:
 - `flask --app backend run.`
- Iniciamos um servidor rodando na porta 5000.
- Acessamos o conteúdo desse servidor pelo caminho:
 - `localhost:5000`

Flask

- O cliente manda um request para o servidor Flask e este envia uma resposta.
 - Neste caso, a resposta é um título `<h1>`.
 - `app.route` define a rota utilizada para executar o método `index()`.
 - Neste caso, a rota é o caminho padrão `“/”`.

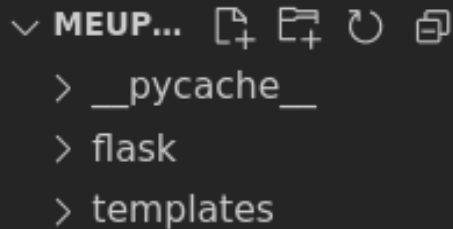
Flask





- Se alterarmos a rota para “/home”, acessamos a página pelo caminho:
 - localhost:5000/home
- Crie um método home() que retorna o texto “<p>Essa é minha Home Page</p>”
 - Acessado pela rota “/home”.

Templates

- Como devolver uma página HTML pronta para um usuário que acessa minha página?
 - Ao invés de escrever o HTML no meu *script* Python.
- Por padrão, o Flask trabalha com arquivos em uma pasta padrão chamada `templates`.
 - Vamos criar essa pasta dentro de nosso projeto e adicionar um arquivo chamado `index.html`.

Templates



✓ **MEUP...**    

- > __pycache__
- > flask
- > templates

Templates

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Página Principal</title>
</head>
<body>
  <h1>Home</h1>
  <p>Bem vindo</p>
</body>
</html>
```

Templates

- Vamos adicionar o template à página principal utilizando o `render_template`.

```
@app.route('/')  
def index():  
    return render_template('index.html')
```

Templates

- É possível adicionar elementos dinâmicos no template.
 - Como um espaço contendo uma **variável nome**.

`<p>Bem vindo, {{nome }}</p>`

- O valor dessa variável pode ser passado por meio de uma rota dinâmica.

Templates

```
@app.route('/profile/<nome>')  
def perfil(username):  
    return render_template('profile.html', nome=nome)
```

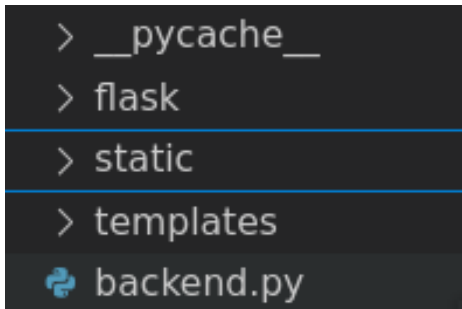
- No exemplo acima temos uma rota dinâmica.
 - Substituímos <nome> na rota da página pelo nome que desejamos exibir.

Arquivos Estáticos

- Como trabalhar com uma página que possui arquivos CSS e JavaScript?
 - Esses elementos são referenciados dentro do template HTML.
 - Mas não serão carregados na página.

Arquivos Estáticos

- Na sua configuração padrão, o Flask procura por arquivos estáticos dentro de uma pasta `static` na raiz do seu projeto.



Arquivos Estáticos

- Coloque todos os seus arquivos CSS, imagens e JavaScript dentro desse diretório.
- Precisamos modificar as páginas HTML e adicionar os elementos estáticos da página utilizando o `url_for`.
 - Exemplo: `{{ url_for('static', filename='estilo.css') }}`.

Arquivos Estáticos

```
<head>
  <meta charset="UTF-8">
  <title>Página Principal</title>
  <link rel="stylesheet"
        href="{{ url_for('static', filename='estilo.css') }}"
        type="text/css">
  <script
        src="{{ url_for('static', filename='comport.js') }}"
        defer>
  </script>
</head>
```

Arquivos Estáticos

- Vamos modificar a página profile para adicionar um CSS e uma imagem.



Introducao ao Flask

Exercícios

Exercício 1

- Tente acessar um o servidor de um dos computadores na rede.
 - Substitua o localhost pelo IP do computador.

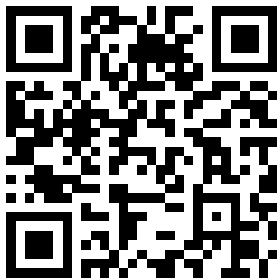
Exercício 2

- Crie um formulário chamado `cadastro1.html`.
 - O formulário contém **nome**, **idade**, **e-mail**, sendo que todos os campos são necessários.
 - use o atributo `required` para tornar o campo obrigatório.
 - No `action` do formulário, coloque a rota de uma página que retorna:
 - “Formulário enviado com sucesso”.

Exercício 3

- Crie uma página `home.html` contendo 3 itens de um menu:
 - Home;
 - Login;
 - Sobre.
- Faça com que cada item do menu tenha uma rota diferente.

Conteúdo



<https://gustavotcustodio.github.io/usabilidade.html>

Obrigado

gustavo.custodio@anhembi.br