Engenharia - Tecnologias Web
Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

HANDOUT AULA 1

Os objetivos de aprendizado dessa aula são:

- Criar um site web simples em um framework web mínimo (Flask).
- Listar as principais tecnologias envolvidas no desenvolvimento web.

Resumo:
Para essa aula será usado o Flask, um framework de desenvolvimento web em Python minimalista baseado no Werkzeug e Jinja2. O objetivo das tarefas é fazer o aluno passar pelo principais passos para o desenvolvimento de um site simples. Ao final o aluno já conhecerá alguns termos e deverá conseguir portar as ferramentas aprendidas para o primeiro projeto do curso.

1. INSTALAÇÃO DO FLASK:

Para o uso do Flask é necessário ter o Python instalado no computador. Caso ainda não tenha instale o Python, para simplificar o processo é recomendado o uso do Anaconda (https://www.continuum.io/downloads), tanto a versão 2 como a 3 do Python devem funcionar adequadamente.

Para instalar o Flask é possível usar os comandos *conda* ou *pip* do Python, todas as dependências devem ser resolvidas nesse processo:

```
$ conda install Flask
ou
$ pip install Flask
```

Detalhes em: http://flask.pocoo.org/docs/0.11/installation/ (Não se preocupe em instalar o virtualenv nesse momento)

2. HELLO WORLD:

Todo primeiro contato com uma ferramenta de programação começa melhor com um Hello World! Assim, vamos começar com o básico. Crie um diretório no seu computador e coloque lá o seguinte arquivo [hello.py]:

Engenharia - Tecnologias Web
Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

```
# -*- coding: utf-8 -*-

from flask import Flask
app = Flask(__name__)

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

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

Para rodar o programa, basta chamar a aplicação pelo interpretador python. Outra opção é usar a rotina nativa do Flask:

Linux / MacOS

```
$ python hello.py
ou
$ export FLASK_APP=hello.py
$ flask run
```

Windows

```
$ python hello.py
ou
$ set FLASK_APP=hello.py
$ flask run
```

Agora abra o seu navegador (Chrome, Firefox, Safari, o que seja) e digite o seguinte endereço e veja o que acontece: http://127.0.0.1:5000/

Experimente agora incluir na string "Hello World!" algo como "Ola Mundo!". O que acontece?

Detalhes em: http://flask.pocoo.org/ e http://flask.pocoo.org/docs/0.11/server/

3. Roteamento e Variáveis:

Uma das formas de acessar os diferentes serviços de um site, é pelos endereços informados na URL. Por exemplo, é possível retornar uma outra informação se digitarmos um endereço como http://127.0.0.1:5000/about. Para fazer o roteamento de forma adequada, são usados *decorators* em seu código. por exemplo inclua as seguintes linhas em seu programa Python:

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

```
@app.route("/about")
def about():
   return "Programa desenvolvido por: <em> Seu Nome </em>"
```

Um recurso ainda mais prático é recuperar partes da linha de endereço, por exemplo buscar a página de alguém. Para isso o valor da linha é armazenado em uma variável para posteriormente ser usado. Veja o exemplo abaixo:

```
@app.route("/user/<name>")
def user(name):
        if name=="fulano":
                return "Esta é a página de fulano"
        else:
                return "Usuário {0} não encontrado".format(name)
```

Originalmente as barras da linha de comando eram referenciadas a arquivos, porém hoje isso não é a regra, porém pode gerar confusões. Por exemplo, digite: http://127.0.0.1:5000/about/ na linha de comando e veja o que acontece. Por que será?

4. PÁGINAS HTML:

Agora crie uma página estática, ou seja, um arquivo HTML com algum conteúdo e coloque em um subdiretório chamado *static*. Por exemplo, crie um arquivo [index.html] como apresentado abaixo:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Título da página</title>
    </head>
    <body>
        Arquivo exemplo de HTML.
    </body>
</html>
```

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

Agora acesse o endereço: http://127.0.0.1:5000/static/index.html

Porém, um dos motivos de se estar usando um servidor é a possibilidade de se gerar conteúdo automaticamente no código rodando no servidor. Porém criar páginas todas em código pode ser complicado. Por isso são usados templates. Crie um novo diretório chamado *templates*, e coloque lá o arquivo [people.html]:

```
<!doctype html>
<title>Exemplo de Template</title>
{% if name %}
  <h1>Oi {{ name }}!</h1>
{% else %}
  <h1>Não te conheço!</h1>
{% endif %}
```

Agora adicione o seguinte código no seu arquivo em Python:

```
from flask import render_template

@app.route('/people/')
@app.route('/people/<nick>')
def hello(nick=None):
   return render_template('people.html', name=nick)
```

5. CSS (Cascading Style Sheets)

CSS é usado para definir as propriedades visuais das páginas HTML. Para isso propriedades são definidas para cada tipo de marcador (tag). Por exemplo, os textos dentro de um marcador *h1* podem ser definidos para terem sempre a cor azul. Crie o arquivo [color.css] abaixo descrito e coloque no diretório *static/css*.

```
h1 {
        color:blue;
}
```

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

Agora adicione a seguinte linha no seu arquivo HTML para invocar o CSS:

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

Veja se ao acessar a página algo mudou. Cuidado que algumas vezes o navegador pode fazer cache das páginas. Sabe o que é cache?

6. JAVASCRIPT

Rotinas de JavaScript são usadas para dar funcionalidade as páginas do lado da aplicação cliente, ou seja, rodando diretamente no navegador. Os programas podem estar em um arquivo separado, mas aqui vamos mostrar um programa que está dentro do próprio HTML. Crie o arquivo [convert.html] e coloque ele no diretório *static*.

```
<!DOCTYPE html>
<html>

<head>
	<script>
	function convert() {
		var C = document.getElementById("c").value;
		document.getElementById("f").innerHTML = C * 9/5 + 32;
	}
	</script>
</head>

<body>
	<p>Insira e temperatura em Celsius:</p>
	<input id="c" onkeyup="convert()"> Celsius igual a
	<span id="f">---</span> Fahrenheit
</body>

</html>
```

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

Quer tentar algo, crie agora uma rotina para converter de Fahrenheit para Celsius.

7. Passando Informações:

Um recurso importante é transmitir dados para o servidor. Existem duas formas mais convencionais e simples de se fazer isso. Uma é passar dados diretamente na linha de comando. Outra possibilidade é usar um recurso REST do protocolo HTTP chamado de POST.

Nesse primeiro exemplo uma mensagem pode ser colocada diretamente na linha de endereço. Adicione o seguinte trecho de código no seu programa e verifique o resultado.

```
from flask import request
@app.route("/product")
def product():
        productId = request.args.get('productId')
        if productId=='1':
                return "Produto: carro"
        elif productId=='2':
                return "Produto: casa"
        else:
                return "Produto não encontrado"
```

Agora coloque no seu navegador o seguinte endereço:
http://127.0.0.1:5000/product?productId=1

Embora simples, essa abordagem é um pouco limitada para alguns tipos de dados, além de deixar muito explicita a exibição do que você esta transmitindo. Uma alternativa é enviar os dados por um POST. Para isso primeiramente se cria uma página com campos para os dados serem preenchidos, nesse exemplo, o código html fica completamente no código. Quando o botão for pressionado, a página faz uma requisição de POST, onde os dados serão tratados.

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        if request.form['username'] != 'admin' or \
                request.form['password'] != '1234':
            return "Usuário não encontrado ou senha incorreta"
        else:
            return "Bem vindo admin"
    return '''
        <form action="" method="post">
            <p>usuário: <input type=text name=username>
            <p>senha: <input type=password name=password>
            <p><input type=submit value=Login>
        </form>
    '''
```

Experimente colocar o seguinte endereço no navegador:
http://127.0.0.1:5000/login

SUGESTÃO, colocar o comando de redirect:
from flask import redirect
return redirect("/static/login.html")

8. DICAS:

Para não precisar ficar reiniciando o servidor a cada vez que fizer algum mudança nos seus arquivos, ou acontecer algum problema, inicie o servidor no modo debug. Para isso faça a seguinte chamada na linha de comando:

```
$ export FLASK_DEBUG=1
```

Profs: Luciano Soares <lpsoares@insper.edu.br> e Camila Achutti <camilafa1@insper.edu.br>

Para a sua aplicação ser visível de outros computadores, você terá que liberar o acesso para isso. Para habilitar essa funcionalidade, faça a chamada do seu programa da seguinte forma:

```
$ flask run --host=0.0.0.0
ou
app.run(host="0.0.0.0")
```

Detalhes das últimas sessões em: http://flask.pocoo.org/docs/0.11/quickstart/

9. PROJETO:

Escolha um dos temas propostos para o seu primeiro projeto. Vocês estarão só praticando a ferramenta nessa aula, porém usem o contexto do seu projeto, para já verificar possíveis limitações técnicas e testar ideias de como poderia implementar. Não perca tempo em detalhes pois não é o objetivo nesse momento.

Referência:

http://rest.elkstein.org/