Curso de introducción al desarrollo web con Flask

Rodolfo Ferro
rodolfo.ferro@leon.gob.mx
LAB León

Módulo 3:

Introducción a Flask

¿Qué es Flask?

Flask es un *framework* ligero para web. Está diseñado para que comenzar sea rápido y fácil, con la capacidad de escalar a aplicaciones complejas. Comenzó como un simple *wrapper* de **Werkzeug** y **Jinja** y se ha convertido en uno de los frameworks de Python más populares para el desarrollo web.

Setup del curso

Para el desarrollo de las prácticas contenidas en el taller, estaremos trabajando sobre scripts de Python, por lo que necesitaremos instalar algunos requerimientos.

La lista de requerimientos es la siguiente:

- Python y PIP
- MongoDB
- Flask y pymongo
- Atom / Sublime Text

Instalación de PIP

Para instalar PIP, el gestor de paquetes de Python, descargaremos el archivo `get-pip.py`. Una vez descargado, navegamos en nuestra terminal hasta llegar al folder que contiene el archivo y procedemos a instalarlo de manera muy sencilla, como si ejecutáramos un script de Python:

```
$ cd <path_al_folder_que_contiene_el_archivo>
$ python get-pip.py
```

Instalación de Flask

Para instalar Flask, haremos uso de PIP, por lo que sólo hará falta instalarlo corriendo un comando en la terminal.

```
$ pip install flask
```

Para comprobar que Flask ha sido correctamente instalado, deberíamos poder importarlo sin problemas:

```
$ python
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import flask
```

Introducción a Flask – Hello world!

Crearemos un script llamado `app.py` donde definiremos las rutas de la aplicación web:

```
from flask import Flask

# Let's instantiate a Flask app:
app = Flask(__name__)

@app.route('/')    # We define the routes as decorators
def index():       # And the content as functions
    return "<h1>¡Hola, curso de Flask!</h1>"

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

Recursos técnicos

Para comprender un poco más sobre las variables especiales como `__name__` o `__main__`, así como un poco más sobre los decoradores, puedes revisar las siguientes referencias:

- Python Special Variables and Methods
- What's in a (Python's) __name__?
- Primer on Python Decorators
- Decorators in Python

Rutas

Utilizaremos los decoradores para definir rutas en nuestra aplicación web:

```
# ...
@app.route('/acerca-de')
def about():
    return "<h1>Acerca de...</h1>"


@app.route('/material')
def material():
    return "<h1>Material del curso</h1>"
# ...
```

Rutas – Error 404!

Podemos incluso especificar qué sucede con las rutas no definidas, como el famoso error 404 (*not found!*):

```
from flask import request

# ...
@app.errorhandler(404)
def not_found(error=None):

    return f“<h2>Ruta no encontrada: {request.url}</h2>”
# ...
```

Estructura de un proyecto

Normalmente se define una estructura estándar para un proyecto con Flask:

```
$ tree
$ .
  ├── app.py
  ├── requirements.txt
  ├── static
  |   ├── css
  |   └── js
  └── templates
      ├── base.html
      └── index.html
```

Plantillas

Una vez conocida la estructura de un proyecto, podemos comenzar a definir las platillas HTML + CSS a utilizar. Dichas plantillas deben estar contenidas en su correspondiente ruta y deben ser importados en sus respectivos archivos de referencia:

```
from flask import render_template

# ...
@app.route('/acerca-de')
def about():
    return render_template("about.html")
# ...
```

Argumentos en plantillas

Flask trabaja su HTML de manera dinámica con Jinja, por lo que podemos pasar objetos a nuestras plantillas:

```
# ...
@app.route('/saludo')
def greetings():
    name = "Rodolfo"
    return render_template("greetings.html", object_to_pass=name)
# ...
```

```
<!-- ... -->
  <p>¡Saludos, {{ object_to_pass }}!</p>
<!-- ... -->
```

Rutas dinámicas

La manera en la que especificamos rutas dinámicas es definiendo el tipo de variable junto con un nombre de variable en la misma ruta:

```
# ...
@app.route('/saludo/<str:name>')
def greetings(name):
    return render_template("greetings.html", name=name)
# ...
```

```
<!-- ... -->
  <p>¡Saludos, {{ name }}!</p>
<!-- ... -->
```

¿Qué es Jinja?

Jinja2 es un motor de plantillas completo para (y hecho con) Python, el cual permite insertar datos procesados y texto predeterminado dentro de un documento de texto.

Aunque se instala automáticamente al instalar Flask, Jinja no sólo es utilizado por Flask.

Introducción a Jinja

Jinja permite añadir bloques con declaraciones o expresiones programáticas en las plantillas de Flask:

```
<!-- ... -->

  <!-- Ejemplo con un expresión -->
  <p>¡Saludos, {{ <expresion> }}!</p>

  <!-- Ejemplo con un declaración -->
  {% <declaration> %}
     <p>¡Saludos, {{ expresion }}!</p>
  {% endof<type of declaration> %}

<!-- ... -->
```

Introducción a Jinja - Extensión de plantillas

Jinja permite realizar extensiones de plantillas de manera muy sencilla, para no definir un montón de código base una y otra vez.

```
<!-- En archivo “base.html” -->
  {% block <blockname> %}
  {% endblock %}
<!-- ... -->
```

```
<!-- En archivo “otro.html” -->
  {% extends “base.html” %}
  {% block <blockname> %}
    <!-- Contenido -->
  {% endblock %}
<!-- ... -->
```


4 BETA
Bootstrap

Siguiente sesión...

Módulo 4:

Diseño de una *web app*