Creando una API REST con Flask

A Python Web Microframework

Diego Francisco Concepción

Co-fundador & CTO en dargo

@costular

costular@gmail.com

¿Qué es Flask?

- Microframework Web
- Con mucha documentación
- Lo necesario para funcionar
- Con facilidad para añadir librerías de terceros

Dependencias

Flask depende fundamentalmente de dos librerías

WSGI utility library

Templating language

¿Cómo funciona?

```
pip install Flask
```

```
1.  from flask import Flask
2.  app = Flask(__name__)
3.
4.  @app.route("/")
5.  def hello():
6.      return "Hello World!"
7.
8.  if __name__ == "__main__":
9.      app.run()
```

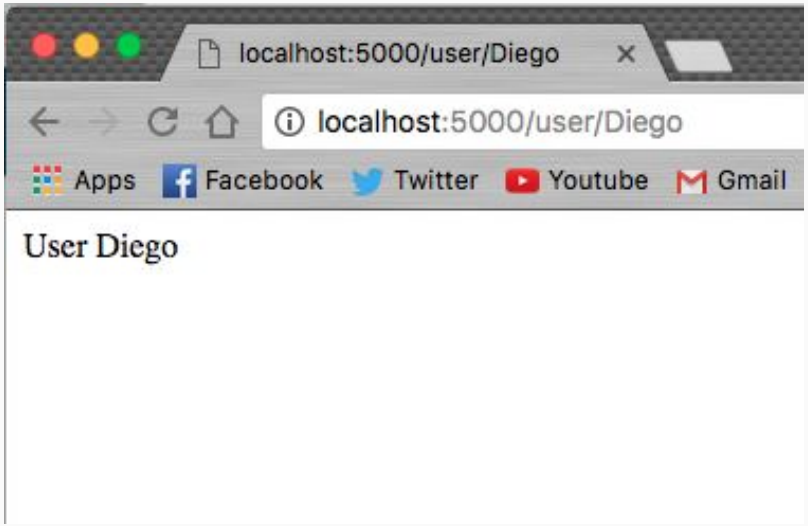
Routing

Utilizando el *decorator* **route** y pasándole un string de la ruta.

Variables:
<int:num> (tipo de dato y nombre de variable)

```
1.  @app.route('/user/<username>')
2.  def show_user_profile(username):
3.      # show the user profile for that user
4.      return 'User %s' % username
```

```
1.  url_for('show_user_profile', args=('Diego',)
```



Blueprints

Los blueprints nos permiten modularizar nuestra aplicación, al estilo Django.

- Puedes asignar URL prefix.
- Subdividir la app en otras y tratarlas de manera distinta
- Reutilizar apps como Django

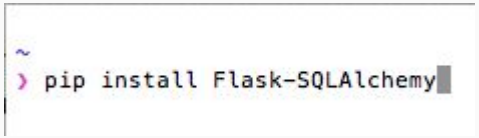
```
1.  api = Blueprint('api', __name__,
2.                          prefix='api')
3.
4.  @api.route('/hello'):
5.  def hello():
6.      return 'Hello :)'
```

```
1.  url_for('api.hello')
```

Blueprint

ORM, tú eliges

MySQL, PostgreSQL, MongoDB, DynamoDB (AWS)... cualquier base de datos encaja a la perfección con **Flask**. Esta es una clara ventaja frente a **Django**.



Si utilizamos MySQL o PostgreSQL instalaremos **SQLAlchemy**

```
pip install SQLAlchemy Flask-SQLAlchemy
```

ORM y modelos -- Flask-SQLAlchemy

```
1.  db = SQLAlchemy()
2.
3.  class Post(db.Model):
4.      __tablename__ = 'posts'
5.      id = db.Column(db.Integer, primary_key=True)
6.      title = db.Column(db.String(55))
7.      comments = db.relationship('Comment', backref=db.backref('post')
8.
9.  class Comment(db.Model):
10.     __tablename__ = 'comments'
11.     id = db.Column(db.Integer, primary_key=True)
12.     post_id = db.Column(db.Integer, db.ForeignKey('posts.id'))
13.     content = db.Column(db.Text)
```

CRUD

```
# Crear
post = Post(title='A simple post')
db.session.add(post)
db.session.commit()

# Obtener todos
posts = Post.query.all()

# Obtener el primero
first_post = Post.query.first()

# Filtrar posts
Post.query.filter_by(title='A simple post').first()
Post.query.filter(Post.title == 'A simple post').first()

# Ordenar, limit...
posts = Post.query.filter_by(title='A simple post').order_by(Post.id.desc()).limit(5).all()
```

CRUD

```
1.  # Eliminar
2.  Post.query.filter_by(title='A simple post').delete() # commit session
    later
3.
4.  db.session.delete(post)
5.  db.session.commit()
```

Contexto

Flask funciona de una forma peculiar. Puedes tener varias apps y cuenta con su propio contexto.

- **flask.request** hace referencia al request actual.
- **flask.current_app** hace referencia a la app actual.
- **flask.g** es el contexto global, podremos guardar variables para toda la app sin importar la petición.

Cuando se necesite el contexto para acceder a la base de datos, el email o cualquier app que requiera del contexto de Flask, deberemos:

```
with current_app.app_context():
    # Do something
```

Librerías: ~~micro~~framework

Hay cientos de librerías para extender la funcionalidad de Flask.

- **Flask-Script**: Conjunto de scripts al estilo manage.py de Django.
- **Flask-Admin**: Interfaz de administración al estilo Django-Admin.
- **Flask-RESTful**: Librería para crear APIs RESTful.
- **Flask-Celery:** Integración de Celery en Flask.
- **Flask-Mail:** Librería para enviar emails con Flask fácilmente.
- Y muchos más.

¡Manos a la obra!

Empezamos con nuestra API REST

Sanic

El clon asíncrono de Flask

¿Por qué Sanic?

Un benchmark vale más que mil palabras

Server	Implementation	Requests/sec	Avg Latency
Sanic	Python 3.5 + uvloop	33,342	2.96ms
Wheezy	gunicorn + meinheld	20,244	4.94ms
Falcon	gunicorn + meinheld	18,972	5.27ms
Bottle	gunicorn + meinheld	13,596	7.36ms
Flask	gunicorn + meinheld	4,988	20.08ms
Kyoukai	Python 3.5 + uvloop	3,889	27.44ms
Aiohttp	Python 3.5 + uvloop	2,979	33.42ms
Tornado	Python 3.5	2,138	46.66ms

Contribuye

http://www.github.com/costular/sanic-rest

¿Alguna pregunta?

¡Muchas gracias!