Seminario 3 DDSI

20/12/2020

Víctor J. Rubia López
Pablo M. Moreno Mancebo
Antonio Galera Gazquez
Juan Andrés Peña Maldonado
Grupo 3

Parte 1

NOMBRE ASIGNATURA	TÉRMINOS
Diseño y Desarrollo de Sistemas de Información	SQL
Fundamentos de Redes	HTML
Aprendizaje Automático	Python
Ingeniería del Conocimiento	Python
Modelos Avanzados de Computación	Python
Desarrollo de Sistemas Distribuidos	Python
Desarrollo de Software	Java
Sistemas de Información Basados en Web	HTML, CSS, JavaScript, PHP, JavaServer, XML, AJAX
Administración de Bases de Datos	SQL
Ingeniería de Sistemas de Información	SQL
Programación Web	AJAX,CSS,JAVASCRIPT,PHP,XML,DOM,JDOM,JAXP
Sistemas de Información para Empresas	XML, Java EE, MySQL, PHP, Python
Sistemas Multimedia	Java
Tecnologías Web	HTML, CSS, PHP, JavaScript

Parte 2

Para esta parte vamos a usar python como lenguaje de programación con flask y como base de datos mongodb, aparte de la librería request de python.

Requests es una biblioteca HTTP con licencia de Apache2, escrita en Python. Está diseñado para ser utilizado por humanos para interactuar con el idioma. Esto significa que no tiene que agregar manualmente cadenas de consulta a las URL ni codificar el formulario de sus datos POST.

En los últimos años, **REST** (REpresentational State Transfer) (Transferencia de estado representativo) ha surgido como el diseño arquitectónico estándar para servicios web y API web.

Por comodidad usamos en este seminario un entorno python con la versión 3.8

Toda aplicación Flask es una instancia WSGI de la clase Flask. Se importa dicha clase y creamos una instancia llamada app. Para crear dicha instancia, debemos pasar como primer argumento el nombre del módulo o paquete de la aplicación. Para estar seguros de ello, utilizaremos la palabra reservada __name__. Esto es necesario para que Flask sepa, por ejemplo, donde encontrar las plantillas de nuestra aplicación o los ficheros estáticos.

Procedimiento

Crear el entorno

```
python3.8 -m venv env
source env/bin/activate
```

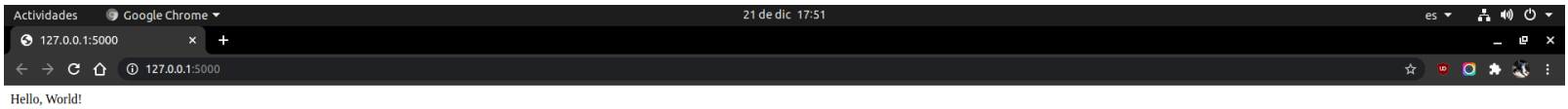
instalar dependencias en el entorno , sacadas con pip freeze

```
pip install fast_json
pip install pymongo
pip install Flask
pip install flake8
pip install requests
```

Ejecutar flask

```
export FLASK_APP=hello.py
flask run
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
(env) pablo@pablomoreno:~/Documentos/ddsi/s3$ export FLASK_APP=hello.py
(env) pablo@pablomoreno:~/Documentos/ddsi/s3$ flask run
 * Serving Flask app "hello.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Hello, World!

Hacemos la petición

```
(env) pablo@pablomoreno:~/Documentos/ddsi/s3$ python making_request.py
200
{'tasks': [{'id': 1, 'pablo': 'gd_pmoreno', 'know_use_that': False}, {'id': 2, 'raul': 'gd_rruiz', 'know_use_that': True}]}
200
{'task': {'id': 3, 'title': 'Nueva tarea', 'description': 'Tarea que mola', 'done': False, '_id': '5fe0d2b42ff2316781bf69e5'}}
(env) pablo@pablomoreno:~/Documentos/ddsi/s3$
```

Accedemos a http://127.0.0.1:5000/tareas

{"tasks": [{"id": 1, "pablo": "gd_pmoreno", "know_use_that": false}, {"id": 2, "raul": "gd_rruiz", "know_use_that": true}, {"id": 3, "title": "Nueva tarea", "description": "Tarea que mola", "done": false, "_id": "5fe0d2b42ff2316781bf69e5"}]}

Código del servicio web

```
from flask import Flask, request, abort

import complemento_mongo

import fast_json
from bson import ObjectId


@fast_json.convert.register(ObjectId)
def _id(object_id):
    return str(object_id)


app = Flask(__name__)


@app.route('/')  # Ruta
def hola():  # Funcion que se ejecuta en la ruta de arriba
    return "Hello, World!"


# Base de datos en memoria
tareas = [
    {
        'id': 1,
        'pablo': 'gd_pmoreno',
        'know_use_that': False
    },
    {
        'id': 2,
        'raul': 'gd_rruiz',
        'know_use_that': True
    }
```

```
]


@app.route("/tareas", methods=['GET'])
def listar_tareas():
    return fast_json.dumps({"tasks": tareas})
# La respuesta de este metodo no es texto  si no un JSON data,
# Con fast_json lo convertimos a string para imprimirlo en vez del
ObjectId(14)


@app.route("/tareas", methods=['POST'])
def nueva_tarea():
    complemento_mongo.estudiantes.delete_many({})
    if not request.json or 'title' not in request.json:
        abort(400)
    task = {
        'id': tareas[-1]['id'] + 1,
        'title': request.json['title'],
        'description': request.json.get('description', ""),
        'done': False
    }
    # Conecto a la BD de mongo y lo introduzco
    complemento_mongo.insertDatosNotebook(task)
    tareas.append(task)
    return fast_json.dumps({'task': task}), 201


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

Mongo

```
from pymongo import MongoClient
# Conexión
mongoClient = MongoClient('localhost', 27017)
db = mongoClient.Prueba
estudiantes = db.estudiantes


# Función insertar
def insertDatosNotebook(NBDataconFecha):
   estudiantes.insert_one(NBDataconFecha)
```

Código de hacer una petición

```
import requests


# Para obtener una pág web
url = "http://127.0.0.1:5000/tareas"
r = requests.get(url)
print(r.status_code)
if (r.status_code == requests.codes.ok):
   print(r.json())

h = requests.post(
       url, json={'title': 'Nueva tarea', 'description': 'Tarea que
mola'})
print(h.status_code)
print(h.json())
```

Bibliografía

Librería Requests

Básico de Flask

Flask v2

Mongo