API in Python con Flask

Cosa è una API?

API: Application Programming Interface ovvero la parte di una programma progettata per essere usata o manipolata da un altro programma, contrariamente alle interfacce disegnate per essere utilizzate da un umano.

Una Web API permette di implementare una funzionalità e renderla accessibile ad altri programmi attraverso la rete.

In queste lezioni useremo i termini API e Web API per indicare lo stesso concetto.

Quando creare una API?

- Per fornire accesso a grandi quantità di dati.
- Quando utenti o programmi devono accedere a dati in tempo reale.
- Per fornire accesso a dati che cambiano spesso.
- Per fornire accesso a piccole parti di grandi dataset.
- Per fornire la possibilità di scrivere, modificare o aggiornare dati, non solo scaricarli.

I data-dump

Se i dati da fornire sono pochi, si può fornire un data-dump nella forma di un file scaricabile di tipo JSON, XML, CSV o SQLite.

API rest?

Le Web API comunemente implementate e che noi implementeremo si ispirano parzialmente alle API REST.

L’acronimo REST sta per **REpresentational State Transfer**, cioè trasferimento della rappresentazione dello stato.

https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Le nostre web API pongono l’accento sulla semplice gestione di risorse remote: CRUD (Create, Retrieve, Update, Delete).

URL

L’accesso alla API avviene tramite una **URL (Uniform Resource Locator)**

La URL è l’indirizzo di una risorsa sul web, ad esempio `https://programminghistorian.org/about`. La URL consiste di un **protocollo** (`http://`), un **dominio** (programminghistorian.org) e un **percorso** opzionale (`/about`).

Una URL descrive dove si trova una specifica risorsa.

Web Service

I servizi di tipo CRUD esposti tramite la web API si chiamano Web Service!

JSON

JSON (JavaScript Object Notation) è un formato standard per file testuali di dati, facile da leggere sia per umani che macchine.

JSON è solitamente il formato più diffuso per far ritornare dati e risposte una una web API. Pensate ai dizionari Python…..

Talvolta si utilizza anche XML.

https://www.json.org/json-en.html

Esempio di JSON

```
"quiz": {
        "sport": {
            "q1": {
                "question": "Which
one is correct team name in NBA?",
                "options": [
                    "New York
Bulls",
                    "Los Angeles
Kings",
                    "Golden State
Warriros",
                    "Huston Rocket"
                ],
                "answer": "Huston
Rocket"
            }
        },
        "maths": {
            "q1": {
                "question": "5 + 7
= ?",
"options": [
                    "10",
                    "11",
                    "12",
                    "13"
                ],
                "answer": "12"
            },
            "q2": {
                "question": "12 - 8
= ?",
                "options": [
                    "1",
                    "2",
                    "3",
                    "4"
                ],
                "answer": "4"
            }
        }
    }
}
```

Buone prassi

Il punto di partenza per un utente di una API è la documentazione. Inoltre URL ben progettate rendono l'utilizzo della API semplice e intuitivo.

Per cui occorre prestare massima attenzione a:

- documentare le API che si implementano
- Progettare URL semplici e intuitive

Sviluppare una API

Svilupperemo una semplice API in Python 3 utilizzando il web framework Flask.

Concetti molto simili a quelli qui affrontati possono essere applicati a web framework più completi e complessi come Django.

Creiamo una semplice API che permetta l'accesso ad un catalogo di libri, ad esempio quelli della biblioteca della scuola.

Il web server

Per prima cosa si crea una normale applicazione Flask:

```
import flask

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


@app.route('/', methods=['GET'])
def home():
    return "<h1>Biblioteca online</h1><p>Prototipo di web
API.</p>"

app.run()
```

I dati

Nella prima versione di web API, la soluzione più semplice per memorizzare di dati è salvarli in un dizionario Python!

```
books = [
    {'id': 0,
     'title': 'Il nome della Rosa’,
     'author': 'Umberto Eco',
     'year_published': '1980'},
    {'id': 1,
     'title': 'Il problema dei tre corpi',
     'author': 'Liu Cixin',
     'published': '2008'},
    {'id': 2,
     'title': 'Fondazione',
     'author': 'Isaac Asimov',
     'published': '1951'}
]
```

Una prima URL per la API

Implementiamo una semplice API che ritorni tutto l'elenco dei libri con titolo, autore e anno.

La classe jsonify presente in Flask permette la conversione di liste e dizionari in formato JSON. Nel nostro caso la lista di dizionari viene convertita in JSON.

```
@app.route('/api/v1/resources/books/all',
methods=['GET'])
def api_all():
    return jsonify(books)
```

Interroghiamo l’API

Semplicemente aprendo la URL http://127.0.0.1:5000/api/v1/resources/books/all con un browser si ottiene la risposta HTTP con header:

HTTP/1.0 200 OK

Content-Type: application/json

Content-Length: 347

Server: Werkzeug/0.16.0 Python/3.7.4

Ricerca per id

Aggiungiamo alla web API la ricerca per id del libro:

```
@app.route('/api/v1/resources/books', methods=['GET'])
def api_id():
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided. Please specify an id."

    results = []

    for book in books:
        if book['id'] == id:
            results.append(book)

    return jsonify(results)
```

Invocare la nuova API

http://127.0.0.1:5000/api/v1/resources/books?id=1

Che succede?

Il browser invia una richiesta GET con la URL sopra indicata!

La classe request di Flask effettua il parsing della parte di URL che segue il ? ovvero dei parametri della query.

La risposta è generata nello stesso modo di prima, ma questa volta la lista di libri è filtrata.

Esercizio

- Invocare la web API e trascrivere la richiesta GET completa e la relativa risposta utilizzando gli strumenti di sviluppo del browser.

- Cambiando i parametri della query, come cambiano request.args?