



Creare Web API RESTful con il linguaggio Python

Webinar



Marco Breveglieri

*Sviluppatore software
consulente e trainer*

👉 <https://www.breveglieri.it>



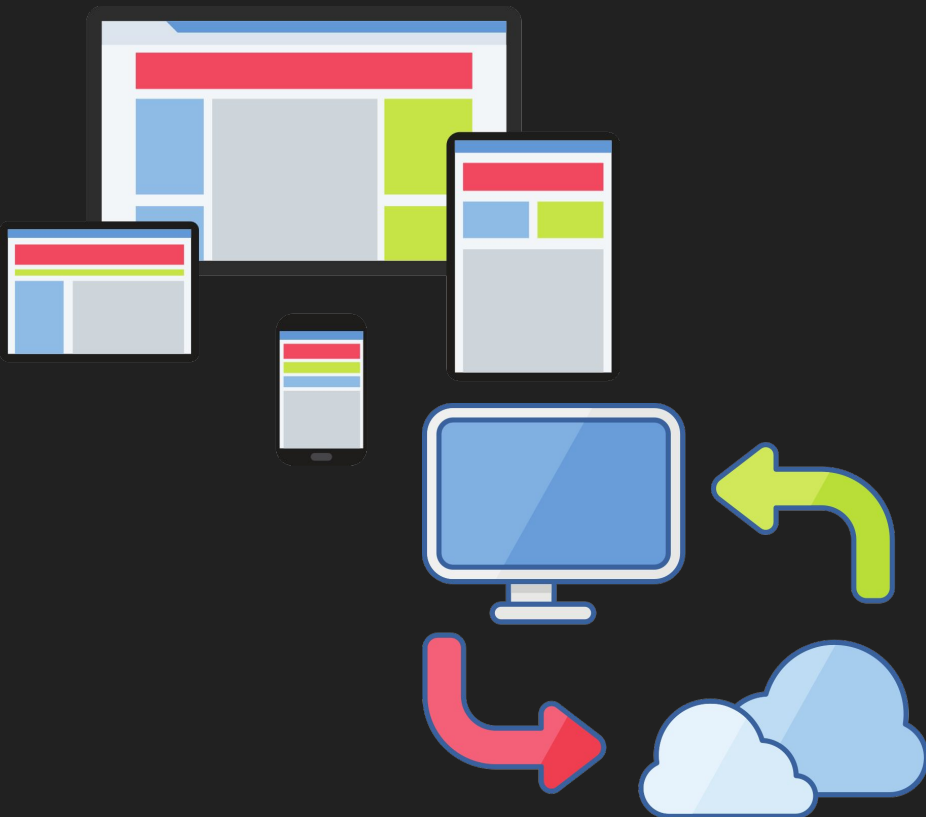


Introduzione



Lo scenario di oggi

- Applicazioni “Internet-Scaled”
- Applicazioni “Cloud-Based”
- App su tipologie diverse di device, OS, form factor, ecc.
- Sistemi operativi eterogenei





Web API: perché?

La creazione di una Web API può aiutare a migliorare l'efficienza, la flessibilità e la sicurezza di un sistema software, nonché consentire l'accesso a funzionalità e dati ad altre applicazioni e piattaforme.





Web API: le caratteristiche

- **Accessibilità**
 - Consente l'accesso a dati in mobilità e a funzionalità di un sistema software
- **Flessibilità**
 - Permette di aggiungere nuove feature senza cambiare il sistema principale
- **Interoperabilità**
 - Consente di creare applicazioni che possono funzionare su sistemi differenti (applicazioni Web, desktop e mobile, servizi cloud) per maggiore versatilità
- **Sicurezza**
 - Limita l'accesso a dati e funzionalità del sistema solamente a chi è autorizzato
- **Scalabilità**
 - Il sistema si può adattare ai volumi di traffico con maggiore efficacia di gestione



Web API in dettaglio



Una Web API *NON* è un Web Service

POST <http://www.myservice.com/getApplePrices>

POST /item HTTP/1.1

Host: 189.123.255.239

Content-Type: text/plain

Content-Length: 200

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Body>
    <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
    </m:GetPrice>
  </soap:Body>
</soap:Envelope>
```




Una Web API *NON* è (solo) JSON

POST <http://www.myservice.com/getGlossaryTerms>

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to...",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

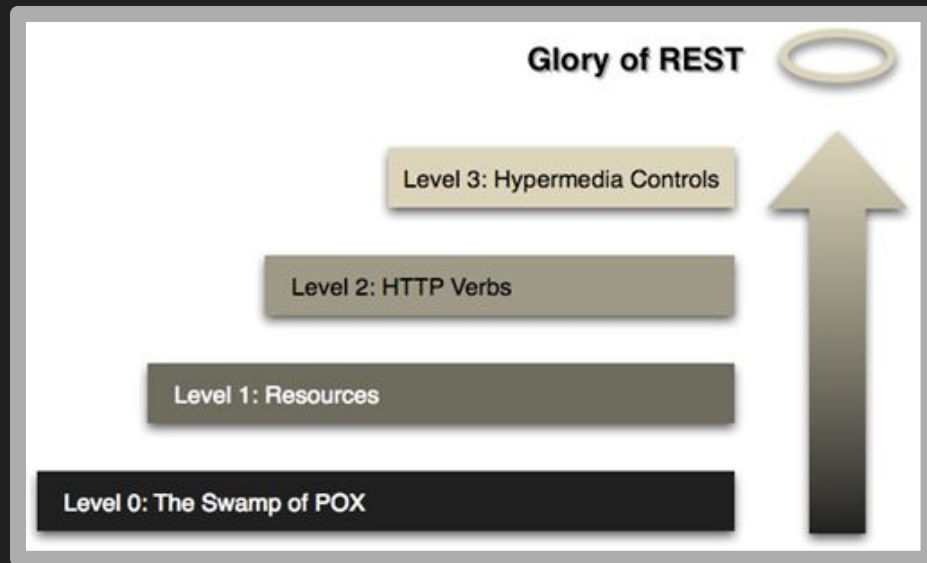
```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<glossary>
  <title>example glossary</title>
  <GlossDiv><title>S</title>
    <GlossList>
      <GlossEntry ID="SGML" SortAs="SGML">
        <GlossTerm>Standard Generalized Markup Language</GlossTerm>
        <Acronym>SGML</Acronym>
        <Abbrev>ISO 8879:1986</Abbrev>
        <GlossDef>
          <para>A meta-markup language, used to...</para>
          <GlossSeeAlso OtherTerm="GML">
            <GlossSeeAlso OtherTerm="XML">
          </GlossDef>
          <GlossSee OtherTerm="markup">
        </GlossEntry>
      </GlossList>
    </GlossDiv>
  </glossary>
```



Ma cosa si intende per *RESTful*?

Richardson Maturity Model

- *Level 0: uso di HTTP*
- *Level 1: uso di risorse*
- *Level 2: uso dei verbi HTTP*
- *Level 3: Hypermedia Controls*

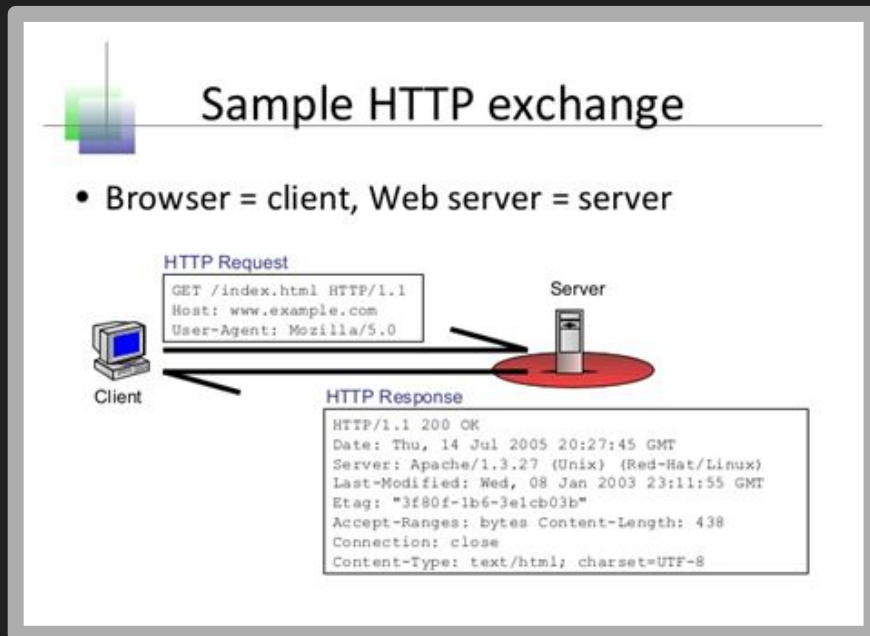


<https://martinfowler.com/articles/richardsonMaturityModel.html>



Il protocollo HTTP

- HTTP è un “application layer protocol” di prima classe
- Nasce per lo scambio di una singola informazione (a differenza di altri protocolli)
- E' facilmente leggibile poiché richiesta e risposta sono in formato testuale
- Su questo protocollo si fonda il servizio più usato nel mondo: il Web!





Accesso a risorse

- HTTP non deve essere usato come protocollo RPC (Remote Procedure Call)
- HTTP usa gli URI per identificare le risorse

`http://theagency.com:8080/agents?id=1`

Schema Host Port Resource Query



Verbi HTTP

- HTTP supporta più verbi (oltre ai classici GET e POST)
- I verbi rappresentano una “azione” da eseguire sulla risorsa
- Nel pacchetto della richiesta, il verbo precede l'URI della risorsa
- I comandi si possono “rimappare” sulle note operazioni basilari CRUD

GET
POST
PUT
DELETE
PATCH

HEAD
OPTIONS
TRACE
CONNECT



Status Code

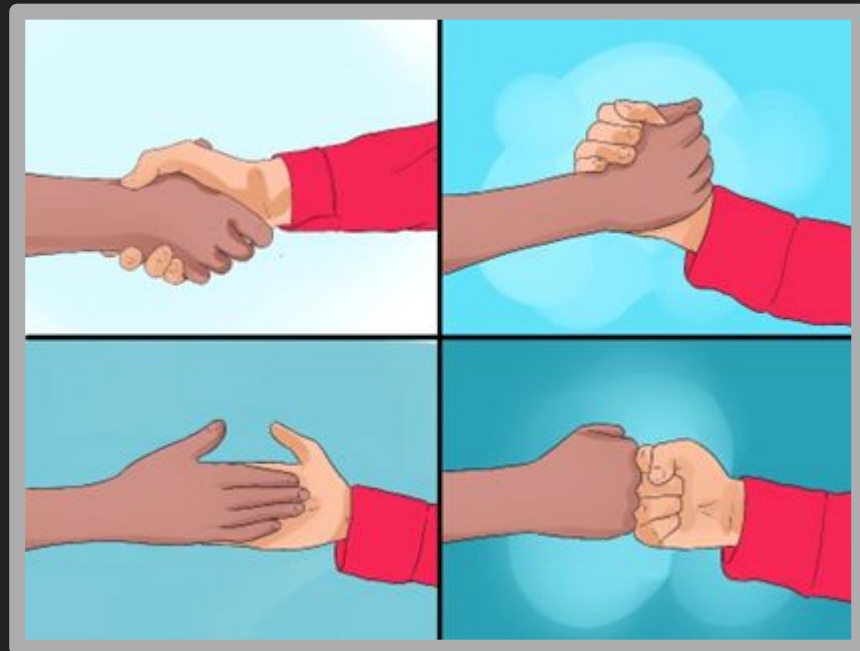
- I codici di stato (Status Code) descrivono il risultato dell'azione
- Vengono restituiti dal server nel pacchetto di risposta
- Sono accompagnati generalmente da una descrizione o da un contenuto
- Sono suddivisi in intervalli in base alla tipologia

1xx	<i>Informational</i>
2xx	<i>Success</i>
3xx	<i>Redirection</i>
4xx	<i>Client Error</i>
5xx	<i>Server Error</i>



Content Negotiation

- HTTP è nato per trasferire ipertesti, ovvero documenti HTML
- Le pagine possono contenere riferimenti a risorse esterne (immagini, video, ecc.)
- HTTP consente al cliente di specificare nella richiesta il formato (o i formati) attesi per la risposta: lo standard MIME
- HTTP supporta diverse intestazioni (*Accept-**) e diversi modi (es. "query string") per opzionare la risposta





Formato dei dati: JSON

- JSON (JavaScript Object Notation) è lo standard de facto per la serializzazione dei dati (in ingresso e in uscita)
- Viene preferito al linguaggio XML perché più leggero e compatto, meno ridondante
- E' universalmente supportato da qualsiasi linguaggio da una o più librerie
- E' semplice, intuitivo e leggibile

👉 <https://www.json.org/>





Come si crea?





Python + FastAPI



- Performance elevate
- Facile da utilizzare
- Validazione dei dati
- Autenticazione e sicurezza
- Auto-documentazione
- Supporto Dependency Injection
- Programmazione asincrona



Perché FastAPI?

	django	 Flask	 FastAPI
Performance speed	Normal	Faster than Django	The fastest out there
Async support	YES with restricted latency	NO needs Asyncio	YES native async support
Packages	Plenty for robust web apps	Less than Django for minimalistic apps	The least of all for building web apps faster
Popularity	The most popular	The second popular	Relatively new
Learning	Hard to learn	Easier to learn	The easiest to learn

Fonte: <https://quintagroup.com/services/python/fastapi>



Let's code! 





Conclusioni



Risorse e approfondimenti

- Sito ufficiale di **Python**
<https://www.python.org/>
- Sito ufficiale di **FastAPI**
<https://fastapi.tiangolo.com/>
- **Richardson Maturity Model**
by *Martin Fowler*
<https://martinfowler.com/articles/richardsonMaturityModel.html>
- Sito ufficiale di **Flask**
<https://flask.palletsprojects.com/>
- Sito ufficiale di **Django REST Framework**
<https://www.django-rest-framework.org/>
- Sito del formato dati **JSON**
<https://json.org>
- Portale **Reteinformaticavoro** su **formazione e corsi**
<https://reteinformaticavoro.it/corsi>



Q & A





Grazie!

