

Rest

Tutti lo nominano ma
pochi lo conoscono



Rest

- ◎ **REPRESENTATIONAL STATE TRANSFER**
- ◎ **È UN PARADIGMA**
- ◎ **NON È UN PROTOCOLLO!**

- ◎ **NASCE GRAZIE ALLA TESI DI Roy Fielding del 2000**

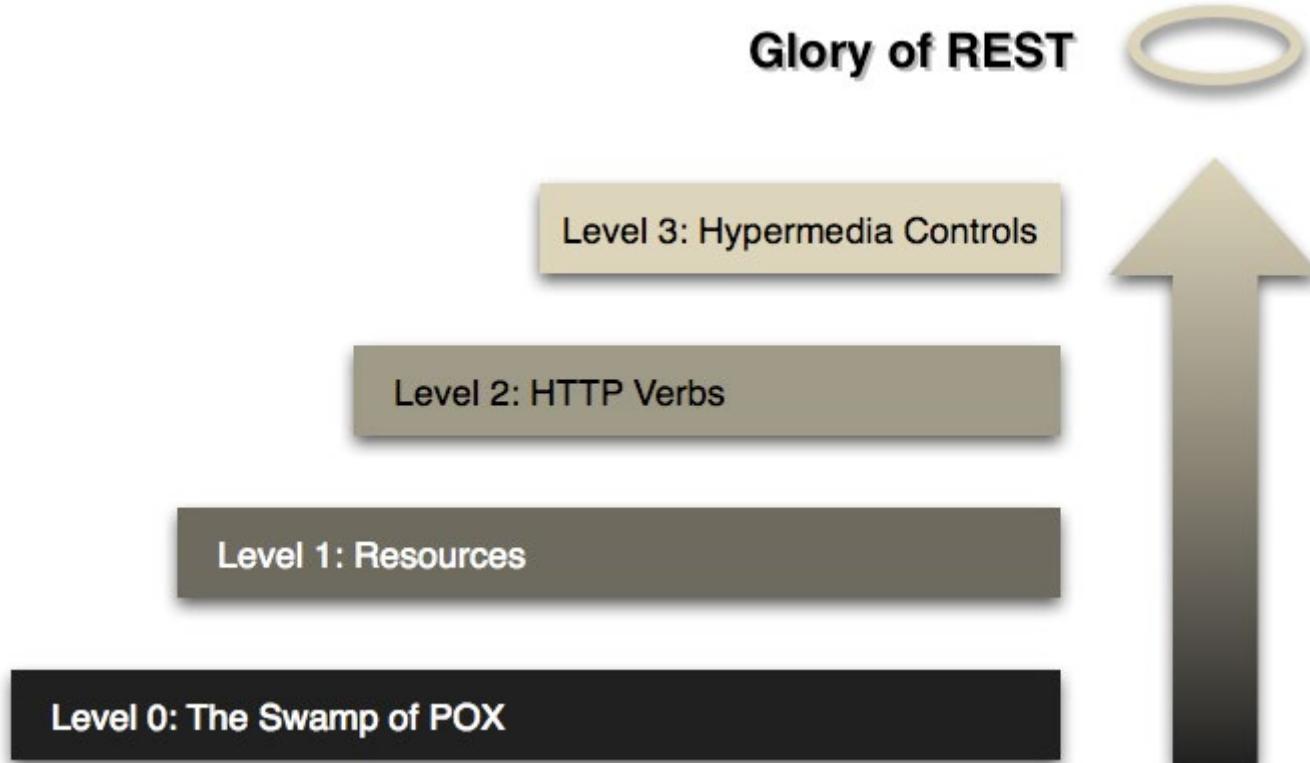
- ◎ https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf (**PAGINA 75**)

Principi del Rest

- REST is a **client-server** architecture
- REST is **stateless**
- REST is **cacheable**
- REST provides a **uniform interface** between components
- REST is a **layered system**
- REST optionally provides **code on demand**

Principi del Rest

- Richardson Maturity Model di **Martin Fowler**
- <https://martinfowler.com/articles/richardsonMaturityModel.html>



Esempio di rest

Risorsa	GET read	POST create	PUT update	DELETE
/books	Ritorna una lista di libri	Crea un nuovo libro	Aggiorna i dati di tutti i libri	Elimina tutti i libri
/books/145	Ritorna uno specifico libro	metodo non consentito (405)	Aggiorna uno specifico libro	Elimina uno specifico libro

```
GET /books/411/authors/ Restituisce la lista degli autori del libro 411
GET /books/411/authors/1 Restituisce l'autore #1 del libro 411
```

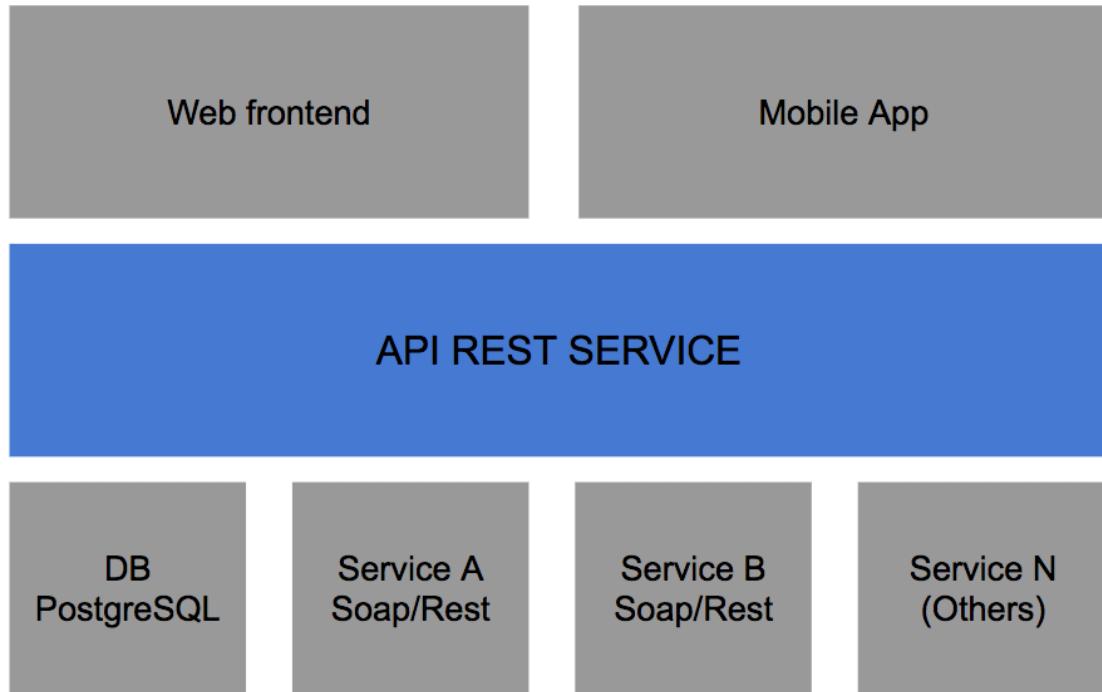
<https://developers.speaker.com/api/>

<https://developer.twitter.com/en/docs/api-reference-index>

<https://api.nasa.gov/>

Perché tutti vogliono REST?

- **Facile**
- **Comprensibile**
- **Flessibile**



NodeJs – ExpressJS – API Rest

```
var express = require('express');
var app = express();
app.get('/', function(req, res, next) {
  next();
})
app.listen(3000);
```

The diagram illustrates the flow of arguments for an Express.js middleware function. It starts with the code snippet above, which defines an application object `app` using the `express` module. The `get` method is called on `app` with a route path `/` and a middleware function as arguments. This middleware function takes three arguments: `req` (HTTP request), `res` (HTTP response), and `next` (callback). The `next` argument is passed back to the application's middleware stack via a call to `next()`. The `listen` method is then called on `app` to start the server on port 3000.

- HTTP method for which the middleware function applies.
- Path (route) for which the middleware function applies.
- The middleware function.
- Callback argument to the middleware function, called "next" by convention.
- HTTP response argument to the middleware function, called "res" by convention.
- HTTP request argument to the middleware function, called "req" by convention.

<https://dev.to/lenmorld/quick-rest-api-with-node-and-express-in-5-minutes-336j>

<https://github.com/gothinkster/node-express-realworld-example-app>

Security

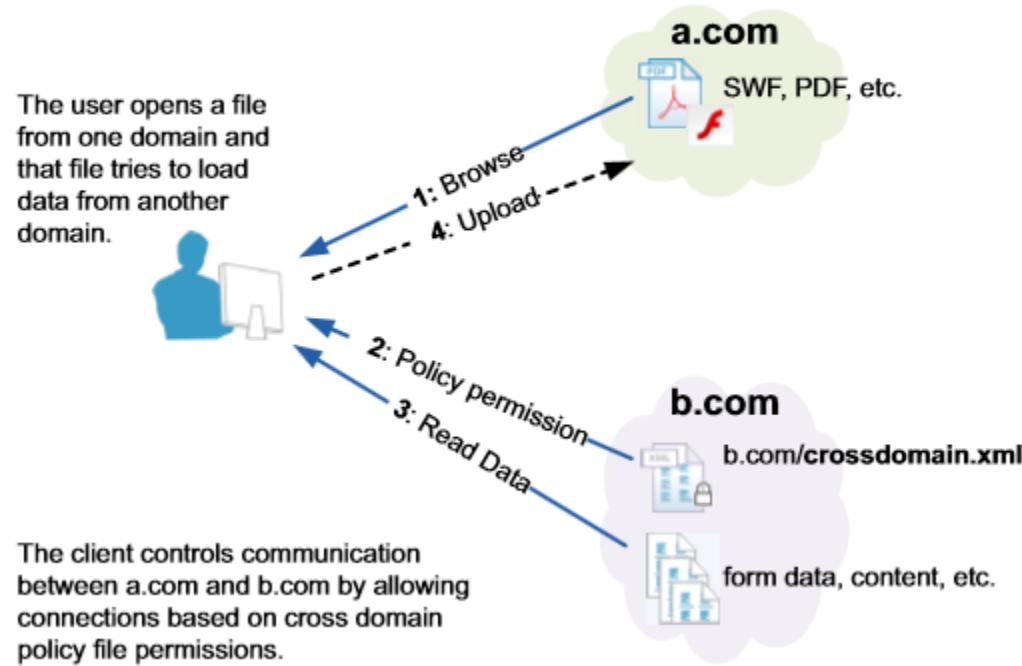
CORS



Cos'è:

CORS: Cross-Origin Resource Sharing

Figure 1 Cross domain workflow



Cos'è:

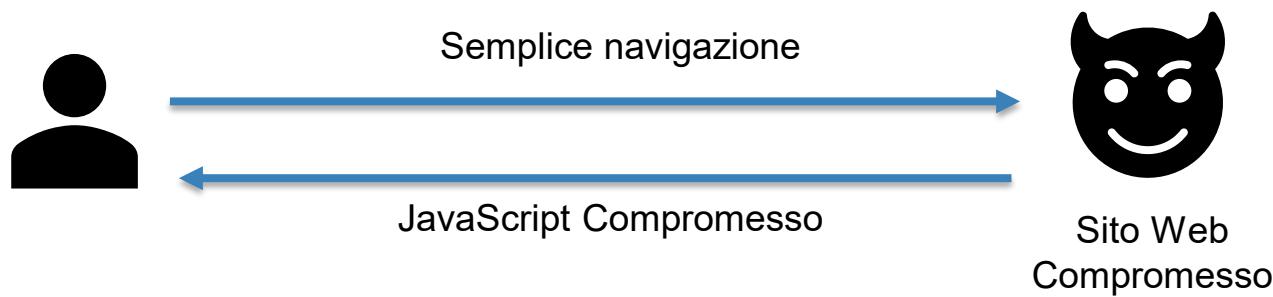
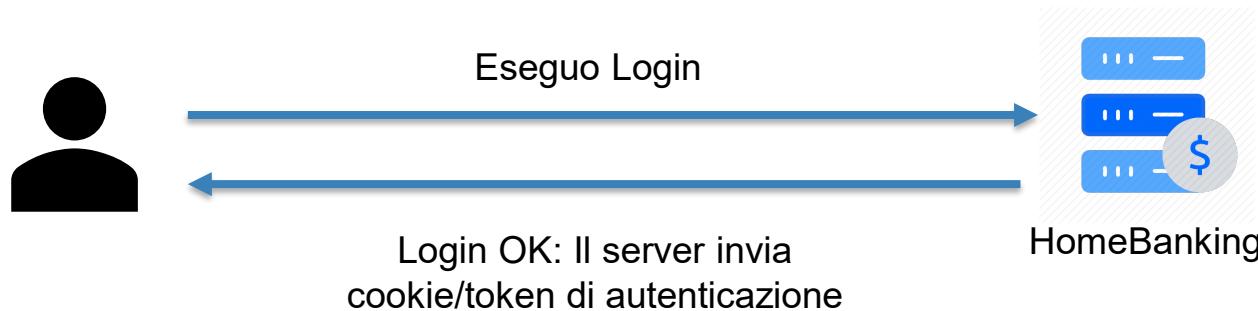
CORS: Cross-Origin Resource Sharing

Il Cross-Origin Resource Sharing (CORS) è un meccanismo che usa header HTTP addizionali per indicare a un browser che un'applicazione Web in esecuzione su un'origine (dominio) dispone dell'autorizzazione per accedere alle risorse selezionate da un server di origine diversa. Un'applicazione web invia una **cross-origin HTTP request** quando richiede una risorsa che ha un'origine (protocollo, dominio e porta) differente dalla propria.

Esempio di cross-origin request: Il codice Javascript di frontend per un'applicazione web servita da `http://domain-a.com` utilizza `XMLHttpRequest` per inviare una richiesta a `http://api.domain-b.com/data.json`.

Importante: per permettere l'accesso su server di origine diversa si deve necessariamente **abilitare** e non disabilitare il CORS.

Perché è importante il CORS e cosa combatte?



Perché è importante il CORS e cosa combatte?

