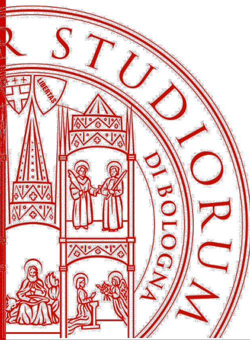
API REST

Angelo Di Iorio

Università di Bologna

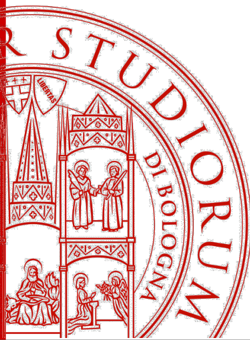
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



API Web

- Un API Web descrive un'interfaccia HTTP che permette ad applicazioni remote di utilizzare i servizi di dell'applicazione
- Queste possono essere:
  - Applicazioni automatiche che utilizzano i dati dell'applicazione
  - Applicazioni Web che mostrano all'utente un menù di opzioni, magari anche un form, e gli permettono di eseguire un'azione sui dati dell'applicazione
- Un esempio: Twitter API v1.1https://developer.twitter.com/en/docs/twitter-api/v1/

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



REST

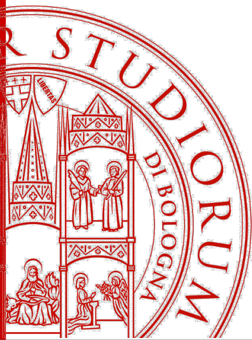
- REST è l'acronimo di ***REpresentional State Transfer***, ed è il **modello architetturale** che sta dietro al World Wide Web e in generale dietro alle applicazioni web "ben fatte" secondo i progettisti di HTTP.

- Applicazioni non REST si basano sulla generazione di un API che specifica le funzioni messe a disposizione dell'applicazione, e alla creazione di un'interfaccia ***indipendente*** dal protocollo di trasporto e ad essa completamente ***nascosta***.

- Viceversa, un'applicazione REST si basa fondamentalmente sull'uso dei protocolli di trasporto (HTTP) e di naming (URI) per generare interfacce ***generiche*** di interazione con l'applicazione, e **fortemente connesse** con l'ambiente d'uso.

- <u>L'obiettivo è creare API consistenti, predicibili e facili da capire e usare</u>

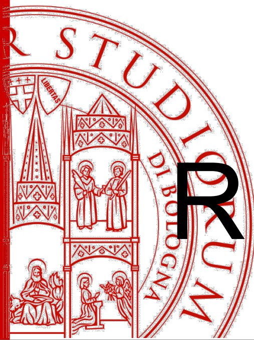
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Il modello CRUD

- Un pattern tipico delle applicazioni di trattamento dei dati
- Ipotizza che tutte le operazioni sui dati siano una di:
  - **Create** (inserimento nel database di un nuovo record)
    - Crea un cliente il cui nome è "Rossi SpA", il telefono "051 654321", la città "Bologna" e restituisce il codice identificatore che è 4123.
  - **Read** (accesso in lettura al database)
    - ***individuale***: dammi la scheda del cliente con id=4123,
    - ***contenitore***: dammi la lista dei clienti la cui proprietà *città* è uguale al valore "Bologna"
  - **Update**
    - Cambia il numero di telefono del cliente il cui id=4123 in "051 123456"
  - **Delete**
    - Rimuovi dal database il cliente con id=4123

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



REpresentational State Transfer

L’architettura REST si basa su quattro punti :

1. Definire **risorsa** ogni concetto rilevante dell’applicazione Web
2. Associargli un **URI** come l’**identificatore** e selettore primario
3. Usare i verbi HTTP per esprimere ogni **operazione** dell’applicazione secondo il modello CRUD:
    - creazione di un nuovo oggetto (metodo PUT)
    - visualizzazione dello stato della risorsa (metodo GET)
    - cambio di stato della risorsa (metodo POST)
    - cancellazione di una risorsa (metodo DELETE)
4. Esprimere in maniera parametrica ogni **rappresentazione dello stato interno della risorsa**, personalizzabile dal richiedente attraverso un **Content Type** preciso

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL’UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Esempio REST: crea cliente

Il metodo specifica l’operazione eseguita

```
PUT clients/1234 HTTP/1.1
Host: http://www.sito.com:80
Content-Type: text/xml; charset=utf-8
Content-length: 474

<client xmlns:m="http://www.myAp
    <nome>Rossi S.p.A.</nome>
    <tel>051 654321</t
    <citta>Bologna</citta>
</client>
```

L' URI dell'oggetto coinvolto

Il body contiene una rappresentazione (in questo caso XML) dell’oggetto da creare

ALMA MATER STUDIORU
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

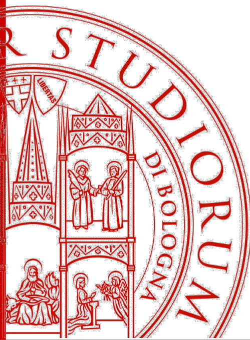
Esempio REST: aggiorna cliente

```
PUT clients/1234 HTTP/1.1
Host: http://www.sito.com:80
Content-Type:application/json; charset=utf-8
Content-length: 176

{
   "nome": "Rossi S.p.A.",
   "tel" : "051 654321",
   "citta" : "Bologna"
}
```

Il body contiene una rappresentazione JSON dell'oggetto da sovrascrivere

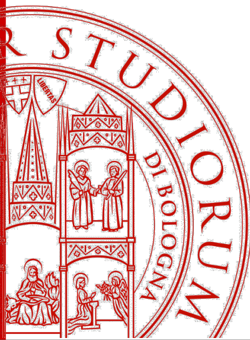
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Individui e collezioni

- REST identifica due concetti fondamentali: **individui** e **collezioni**
  - un cliente vs. l'insieme di tutti i clienti
  - un esame vs. l'insieme di tutti gli esami superati
  - …
- Fornisce URI ad entrambi
- <u>Ogni operazione avviene su uno e uno solo di questi concetti.</u>
- Su entrambi si possono eseguire operazioni CRUD. A seconda della combinazione di verbi e risorse otteniamo l'insieme delle operazioni possibili.
- Ciò che passa come corpo di una richiesta e/o risposta NON E' la risorsa, ma una *rappresentazione* della risorsa, di cui gli attori si servono per portare a termine l'operazione.

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Gerarchie

- Le collezioni possono "contenere" individui o altre collezioni
- E' consigliabile strutturare gli URI in modo gerarchico, per esplicitare queste relazioni
- API più leggibile e *routing* semplificato in molti framework di sviluppo
- Ad esempio:
  - Tutti i clienti

    /clients/
  - Il cliente 1234

  /clients/1234
  - Tutti gli ordini del cliente 1234

  /clients/1234/orders/

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Linee guida degli URI in REST

Le **collezioni** sono intrinsecamente **plurali**. Gli **individui** sono intrinsecamente **singolari**.

Le collezioni debbono essere visivamente distinguibili dagli individui. Per questo usiamo un termine plurale **e** uno slash in fondo all'URI

URI	Rappresentazione
`/customers/`	collezione dei clienti
`/customers/abc123`	cliente con id=abc123
`/customers/abc123/`	collezione delle sotto-risorse del cliente con id=abc123 (es. indirizzi, telefoni,ecc.)
`/customers/abc123/addresses/1`	primo indirizzo del cliente con id=abc123

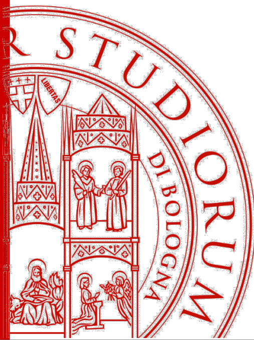
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Filtri e search negli URI REST

Un filtro genera un **sottoinsieme** specificato attraverso una regola di qualche tipo. La gerarchia permette di specificare i tipi più frequenti e rilevanti di filtro.

Altrimenti si usa la parte query dell'URI di una collezione:

URI	Rappresentazione
`/regions/ER/customers/`	collezione dei clienti dell'Emilia Romagna
`/status/active/customers/`	collezione dei clienti attivi
`/customers/?tel=0511234567` oppure `/customers/?search=tel&value=051 1234567`	collezione dei clienti che hanno telefono = 051 1234567
`/customers/?search=sales&value=100000&op=gt`	collezione dei clienti che hanno comprato più di 100.000€

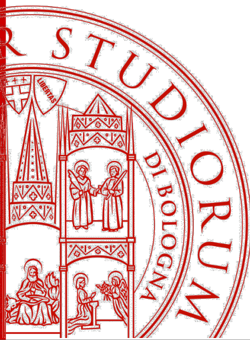


Uso dei verbi HTTP in REST

Attenzione a questa differenza! Operazioni su collezione o individuo

- Elencare tutti i clienti
  `GET /customers/`
- Accedere ai dati del cliente id=abc123
  `GET /customers/abc123`
- Creare un nuovo cliente (il client **non** decide l'identificatore)
  `POST /customers/`
- Creare un nuovo cliente (il client **decide** l'identificatore)
  `PUT /customers/abc123`
- Modificare (tutti) i dati del cliente id=abc123
  `PUT /customers/abc123`
- Modificare alcuni dati del cliente id=abc123
  `POST /customers/abc123/telephones/`
- Cancellare il cliente id=abc123
  `DELETE /customers/abc123`

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Semantica del POST

Nelle vecchie versioni di HTTP (ad es. RFC2616), si diceva:

- *"The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. It essentially means that POSTrequest-URI should be of a collection URI."*

Nel 2014 è stata approvata una modifica e chiarificazione ad alcuni testi del documento di HTTP (RFC7231), e in particolare:

- *"The POST method requests that the target resource process the representation enclosed in the request according to the resource's own specific semantics."*

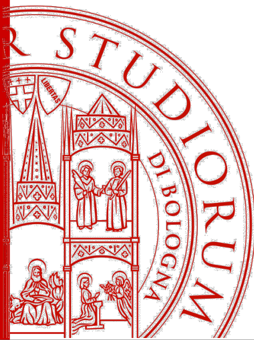
In pratica, il POST può essere usato in una moltitudine di situazioni secondo una semantica decisa localmente, purché non sovrapposta a quella degli altri verbi

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI

Altri consigli e linee guida

- Adottare una convenzione di denominazione coerente e chiara negli URI
- Usare gerarchie ma valutare i livelli necessari (chiarezza vs. carico sul server)
- Evitare di creare API che rispecchiano semplicemente la struttura interna di un database
- Fornire meccanismi – parametri nelle query – per filtrare e paginare le risposte
- Supportare richieste asincrone e in questo caso restituire codice HTTP 202 (Accettato ma non completato) e informazioni (URL) per accedere allo stato della risorsa

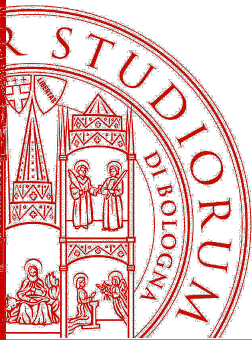
IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Descrivere una RESTful API

- Una API è RESTful se utilizza i principi REST nel fornire accesso ai servizi che offre
- Per documentare un API è necessario definire:
  - **end-point** (*URI / route*) che supporta
    - separando collezioni e elementi singoli
  - **metodi** HTTP di accesso
    - Cosa succede con un GET, un PUT, un POST, un DELETE, ecc.
  - **rappresentazioni** in **Input** e **Output**
    - Di solito non si usa un linguaggio di schema, ma un esempio fittizio e sufficientemente complesso
  - **condizioni di errore** e i **messaggi** che restituisce in questi casi

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI



Conclusioni

- REST considera ogni applicazione come un ambiente di cui si cambia lo stato attraverso un insieme limitato di comandi (i metodi HTTP) applicati alle risorse (espresse attraverso URI) e visualizzate attraverso una esplicita richiesta di rappresentazione (attraverso un content Type MIME).
- REST ha il pregio di sfruttare completamente ed esattamente tutti gli artifici del web, ed in particolare caching, proxying, sicurezza, ecc.
- Inoltre l'aprirsi all'uso sistematico di URI permette ad applicazioni sofisticate basate su logica ed inferenza si sfruttare le tecniche del Semantic Web per creare funzionalità ancora più sofisticate e intelligenti con applicazioni create su architettura REST.

IL PRESENTE MATERIALE È RISERVATO AL PERSONALE DELL'UNIVERSITÀ DI BOLOGNA E NON PUÒ ESSERE UTILIZZATO AI TERMINI DI LEGGE DA ALTRE PERSONE O PER FINI NON ISTITUZIONALI