# REST API DESIGN

## Laboratorio 2

# REST API DESIGN

- Qual è il codice di stato della risposta HTTP appropriato per un determinato scenario?

  `101 200 201 302 400 404 418 500…`

- Quando dovrebbero essere nominati con sostantivi plurali, i segmenti di percorso URI?

  `/users/userId`

  `/leagues/leagueName`

- Quale metodo di richiesta deve essere utilizzato per interagire con la risorsa?

  `GET POST PUT DELETE PATCH…`

# REST API DESIGN

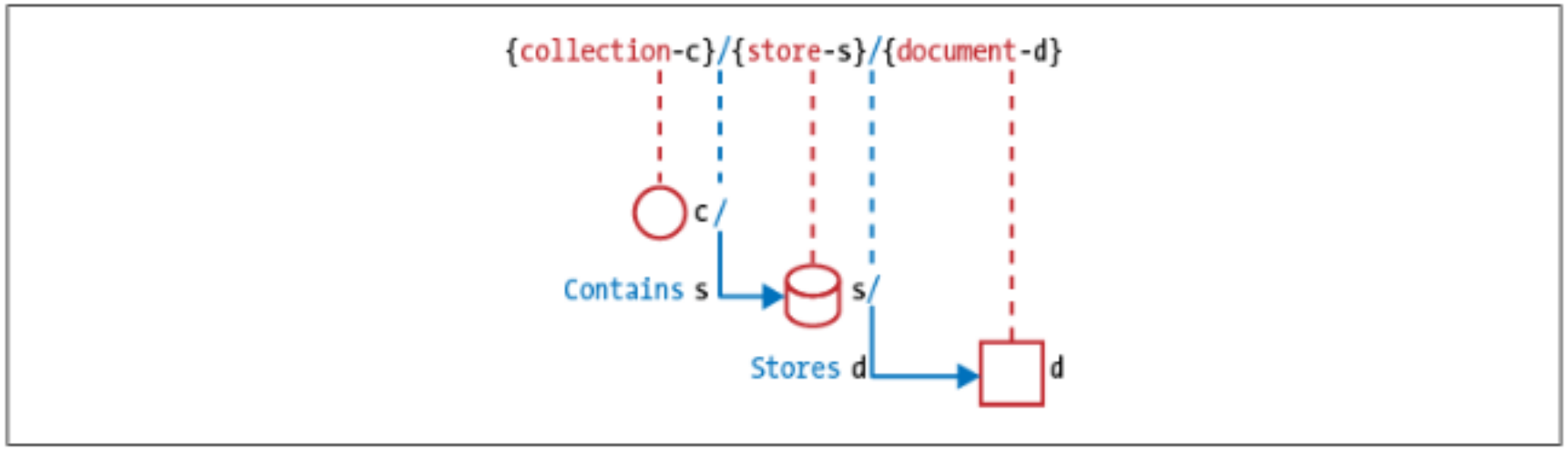
- Come posso mappare le operazioni non CRUD ai miei URI?

  `/message/messageId/send`

- Come posso gestire le versioni delle rappresentazioni di stato di una risorsa?
  - Url: `/api/v1/leagues`
  - Query: `/api/leagues?v=1`
  - HTTP headers
  - …
- Come devo strutturare un collegamento ipertestuale in JSON?

# Resource Modelling

- Quando modelliamo le risorse di un'API, si può iniziare con alcuni *archetipi di risorse* di base.
- Un'API REST è composta da quattro distinti archetipi di risorse: **document**, **collection**, **store**, e **controller**.



# Documents

Una risorsa **documento** è un concetto simile a *un'istanza di oggetto* o ad un *record di database*.

La rappresentazione dello stato di un documento, in genere, include sia *valori* che *collegamenti* ad altre risorse collegate.

Il tipo di documento è l'archetipo di base per gli altri archetipi di risorse.

# Documents

Ciascun URI di seguito identifica una risorsa documento:

```
http://api.soccer.restapi.org/leagues/seattle
http://api.soccer.restapi.org/leagues/seattle/teams/trebuchet
```

Un documento può avere *risorse figlio* che rappresentano i suoi specifici concetti subordinati. Con la sua capacità di riunire molti tipi di risorse diversi sotto un unico genitore, un documento è un candidato logico per la risorsa radice di un'API REST, nota anche come `docroot`. Per esempio, l'URI riportato di seguito identifica il docroot, che è il punto di ingresso dell'API Soccer REST:

```
http://api.soccer.restapi.org
```

# Collection

Una **Collection** è una repository di risorse gestita dal **server**.

I *client* possono proporre nuove risorse da aggiungere ad una collezione. Spetta alla collezione scegliere se creare o meno una nuova risorsa, cosa vuole contenere e decide anche gli URI delle risorse contenute.

I seguent URI identificano una collezione:

```
http://api.soccer.restapi.org/leagues
http://api.soccer.restapi.org/leagues/seattle/teams
```

# Store

Uno **Store** è una repository di risorse gestito dal **client**.

Consente ad un client di inserire risorse, recuperarle ed eliminarle. Da soli non creano nuove risorse e non generano nuovi URI. Invece, ogni risorsa memorizzata ha un URI che è stato scelto da un client quando è stato inizialmente inserito nello store.

Di seguito si mostra un utente (con ID 1234) di un programma client che utilizza un'API REST Soccer fittizia per inserire una risorsa documento denominata alonso nel proprio archivio di preferiti:

```
PUT /users/1234/favorites/alonso
```

# Controller

Un **Controller** modella un concetto procedurale, funzioni eseguibili, con parametri e valori restituiti, di input e di output.

Un'API REST si basa sulle risorse del controller per eseguire azioni specifiche dell'applicazione che non possono essere mappate logicamente a uno dei metodi standard (CRUD). In genere, vengono visualizzati come ultimo segmento in un percorso URI, senza risorse figlio che li seguono nella gerarchia.

L'esempio seguente mostra una risorsa controller che consente a un client di inviare nuovamente un avviso (num. 245743) a un utente:

```
POST /alerts/245743/resend
```

# URIs

Le API REST utilizzano gli Uniform Resource Identifier (**URI**) per indirizzare le risorse.

Sul Web di oggi, i designer degli URI spaziano tra soluzioni diverse che comunicano il modello di risorse dell'API come:

```
http://api.restapi.org/paris/louvre/leonardo-da-vinci/mona-lisa
http://api.restapi.org/68dd0-a9d3-11e0-9f1c-0800200c9a66
```

RFC 3986 definisce la sintassi URI generica come mostrato di seguito:

**URI** = **scheme** "://" **authority** "/" **path** [ "?" **query** ] [ "#" **fragment** ]

# Best practises

- La **barra** (/) deve essere utilizzato per indicare una relazione gerarchica. Non dovrebbe essere inclusa negli URI

- I **trattini** (-) dovrebbero essere usati per migliorare la leggibilità degli URI
  `http://api.example.restapi.org/blogs/mark-masse/entries/this-is-my-first-post`

- Le **lettere minuscole** dovrebbero essere preferite nei percorsi URI
  `http://api.example.restapi.org/my-folder/my-doc`

- Le **estensioni** dei file non devono essere incluse negli URI
  `http://api.college.restapi.org/students/3248234/transcripts/2005/fall`

# Best practises

- Un **sostantivo singolare** deve essere utilizzato per i nomi dei **documents**
  `http://api.soccer.restapi.org/leagues/seattle/teams/trebuchet/players/claudio`
- Un **sostantivo plurale** deve essere utilizzato per i nomi delle **collection**
  `http://api.soccer.restapi.org/leagues/seattle/teams/trebuchet/players`
- Un **sostantivo plurale** dovrebbe essere usato per i nomi degli **store**
  `http://api.music.restapi.org/artists/mikemassedotcom/playlist`**`s`**
- Un **verbo** o una **frase** per i nomi dei **controller**
  `http://api.college.restapi.org/students/morgan/`**`register`**
  `http://api.build.restapi.org/qa/nightly/`**`runTestSuite`**

# Best practises

- I **segmenti** possono essere sostituiti con valori basati sull'identità

```
http://api.soccer.restapi.org/leagues/:leagueId/teams/:teamId
```

- I **nomi delle funzioni** CRUD non devono essere utilizzati negli URI

```
DELETE /users/1234
```

- Esempi di **antipattern**

```
GET /deleteUser?id=1234
GET /deleteUser/1234
DELETE /deleteUser/1234
POST /users/1234/delete
```

# Best practises

- Il componente **query** di un URI può essere utilizzato per filtrare **collection** o **store**

```
GET /users?role=admin
```

- Il componente **query** di un URI deve essere utilizzato per impaginare i risultati da **collection** o **store**

```
GET /users?pageSize=25&pageStartIndex=50
```

# Message Body Format

Un'API REST utilizza comunemente il body di un messaggio di risposta per trasmettere lo **stato della risorsa** identificata di un messaggio di richiesta.

Le API REST utilizzano spesso un formato basato su testo per rappresentare lo stato di una risorsa come un insieme di campi significativi. Oggigiorno, i formati di testo più comunemente usati sono **XML** e **JSON**.

Di seguito, un esempio di JSON ben formato per la rappresentazione delle risorse

```
{
    "firstName" : "Osvaldo",
    "lastName" : "Alonso",
    "firstNamePronunciation" : "ahs-VAHL-doe",
    "number" : 6,  ❶
    "birthDate" : "1985-11-11"  ❷
}
```

# Riferimenti


- Masse, M. (2011). *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.".