Carlo Vassallo

Claudio Agostino Ardagna

Applicazioni Web e Cloud

https://gentle-earth-39991.herokuapp.com/

IlVolo REST

Strumenti utilizzati:

Per lo sviluppo dell'applicazione IlVolo ho scelto di utilizzare il framework Spring nel linguaggio Java in combinazione con Hibernate, un ORM, connesso ad un database PostgreSQL per la persistenza dei dati. Uno dei principali problemi che sorgono durante lo sviluppo di applicazioni con interfaccia REST è che il codice risulta duplicato e/o ricorrente. Per ovviare a questo problema Spring mette a disposizione un modulo chiamato Spring-Data-Rest che mette a permette la creazione dinamica degli endpoint REST.

Spring-Data-Rest

Spring-Data-Rest è un modulo di Spring basato su Spring-MVC, esso permette la creazioni di interfacce REST a partire dai dati, ovvero, dopo aver definito le classi di persistenza dei dati (JPA, e.g. la classe Aircraft) e un repository contenente i metodi necessari per l'accesso e la selezione dei dati, Spring mappa i metodi delle varie repository a degli appositi endpoint REST. Il risultato di questo setup è un interfaccia conforme alle specifiche HATEOAS.

HATEOAS

HATEOAS (Hypermedia as the Engine of Application State) definisce un design standard per interface REST in modo hypermedia-driven, mettendo quindi a disposizione delle informazioni che permettono all'utente, o client, di esplorare gli endpoint senza averne una conoscenza precedente.

```
GET /aircrafts
cache-control: no-cache
user-agent: PostmanRuntime/7.1.1
accept: */*
host: gentle-earth-39991.herokuapp.com
accept-encoding: gzip, deflate

HTTP/1.1 200
status: 200
content-type: application/hal+json;charset=UTF-8
transfer-encoding: chunked
date: Sat, 09 Jun 2018 09:05:21 GMT

{
  "_embedded": {
    "aircrafts": [
      {
        "model": "jet",
        "seats": 200,
        "haul": "SHORT",
        "maxSpeed": 200,
        "wingspan": 200,
        "engine": "TURBOJET",
        "_links": {
          "self": {
            "href": "http://localhost:8080/aircrafts/1"
          },
          "aircraft": {
            "href": "http://localhost:8080/aircrafts/1"
          },
          "routes": {
            "href": "http://localhost:8080/aircrafts/1/routes"
          }
        }
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/aircrafts{?page,size,sort}",
      "templated": true
    },
    "profile": {
      "href": "http://localhost:8080/profile/aircrafts"
    }
  }

  [...]
```

Esempio:

L'esempio sopra riportato ci permette di ottenere la lista di Aircraft. Come si può notare, l'header di risposta "Content-type" riporta come MIME tipe "application/hal+json". Di fatto il contenuto della risposta è un JSON valido, ma in questo modo specifichiamo che il contenuto del JSON (o XML nel caso in cui il formato scelto sia tale) è conforme alle specifiche HAL che definiscono la semantica con cui esprimere i link di esplorazione. Sempre nell'esempio possiamo vedere la presenza dei link di cui sopra, in particolare ogni entità ha un oggetto "__links" che contiene le informazioni necessarie, ovvero un URL, per potersi muovere all'interno delle risorse esposte, come per esempio il link a se stesso e/o a oggetti ad esso relazionati.

ALPS

Analizzando più a fondo notiamo anche la presenza di un link "profile" che se seguito ci fornisce le informazioni (o documentazione) sul endpoint in questione

```
GET /profile/aircrafts
accept: */*
cache-control: no-cache
user-agent: PostmanRuntime/7.1.1
host: gentle-earth-39991.herokuapp.com
accept-encoding: gzip, deflate

HTTP/1.1 200
status: 200
content-type: application/alps+json;charset=UTF-8
transfer-encoding: chunked
date: Sat, 09 Jun 2018 09:36:11 GMT
```

```
{
   "alps": {
      "version": "1.0",
      "descriptors": [
         {
            "id": "aircraft-representation",
            "href": "http://localhost:8080/profile/aircrafts",
            "descriptors": [
               {
                  "name": "model",
                  "type": "SEMANTIC"
               },
              [...]
               {
                  "name": "routes",
                  "type": "SAFE",
                  "rt": "http://localhost:8080/profile/routes#route-representation"
               }
            ]
         },
         {
            "id": "get-aircrafts",
            "name": "aircrafts",
            "type": "SAFE",
            "rt": "#aircraft-representation",
            "descriptors": [
               {
                  "name": "page",
                  "doc": {
                     "value": "The page to return.",
                     "format": "TEXT"
                  },
                  "type": "SEMANTIC"
               },
               {
                  "name": "size",
                  "doc": {
                     "value": "The size of the page to return.",
                     "format": "TEXT"
                  },
                  "type": "SEMANTIC"
               },
               {
                  "name": "sort",
                  "doc": {
                     "value": "The sorting criteria to use to calculate the content
                               of the page.",
                     "format": "TEXT"
                  },
                  "type": "SEMANTIC"
               }
            ]
         },
```

Hateoas e alps ci permettono duque di costruire un interfaccia auto esplicativa e dinamica. Da una parte abbiamo una documentazione dinamica e standard, dando la possibilità agli sviluppatori di comprendere la struttura dell'interfaccia REST, dall'altra una serie di endpoint strutturati in maniera tale da rilassare la forte dipendenza del client dal server, permettendo un evoluzione parallela, infatti mettendo a disposizione i link corrispondenti alle risorse direttamente nelle risposte HTTP il client ha unicamente bisogno di sapere il nome con cui le entità di cui ha bisogno sono identificate, e seguire il link corrispondente. Nel caso in cui il server cambi struttura alle APIs mantenendo gli standard di cui sopra, il client non è affetto dal cambiamento.

IlVolo

Lo sviluppo dell'applicativo ha seguito gli standard sopra descritti definendo degli endpoint CRUD per ogni risorsa. In linea con le specifiche HATEOAS, chiamando la root del servizio otteniamo i link di tutte le entità messe a disposizione.

```
GET /
cache-control: no-cache
user-agent: PostmanRuntime/7.1.1
accept: */*
host: gentle-earth-39991.herokuapp.com
accept-encoding: gzip, deflate

HTTP/1.1 200
status: 200
connection: keep-alive
content-type: application/hal+json;charset=UTF-8
transfer-encoding: chunked
date: Tue, 12 Jun 2018 20:12:22 GMT
```

```
{
   "_links": {
      "aircrafts": {
         "href": "https://gentle-earth-39991.herokuapp.com/aircrafts{?page,size,sort}",
         "templated": true
      },
      "slots": {
         "href": "https://gentle-earth-39991.herokuapp.com/slots{?page,size,sort}",
         "templated": true
      },
      "routes": {
         "href": "https://gentle-earth-39991.herokuapp.com/routes{?page,size,sort}",
         "templated": true
      },
      "airlineCompanies": {
         "href": "https://gentle-earth-39991.herokuapp.com/airlineCompanies{?
page,size,sort}",
         "templated": true
      },
      "profile": {
         "href": "https://gentle-earth-39991.herokuapp.com/profile"
      }
   }
}
```


Aircrafts

Questo endpoint è dedicato agli aeromobili e permette le usuali operazioni CRUD. In oltre vi è la possibilità, per ogni aeromobile, di operare sulle rotte associate.

Routes

Analogo a quello per gli aircraft, ma speculare, questo endpoint espone le rotte e, pero ognuna, gli aeromobili abilitati.

Airline Companies

Espone un interfaccia CRUD per la gestione delle compagnie aeree e le loro relazioni.

Convenzioni

In aggiunta a quanto già detto sugli standard, le APIs in questione sono stare implementate rispettando la filosofia REST nella sua interezza. Per ogni endpoint CRUD è stato rispettato il seguente schema (riportato anche nelle route /profiles)

- GET: utilizzato per ottenere la lista delle entità o la signora specificata nell'url
- POST: per la creazione di un entità o per l'aggiunta di una relazione
- PUT: per la modifica di un entità o per la modifica di una relazione
- DELETE: per la cancellazione di un entità

Sono anche stati utilizzati espressivamente i status code http corrispettivi ai risultati degli endpoint, come ad esempio 200 per il successo delle chiamate get, 404 per le risorse non esistenti, 201 per il successo e 400 per l'errore degli endpoint di creazione, etc..

Deploy

Il deploy dell'applicazione è avvenuto mediante la piattaforma Heroku e Heroku-postgress, e disponibile all'indirizzo https://gentle-earth-39991.herokuapp.com/ .