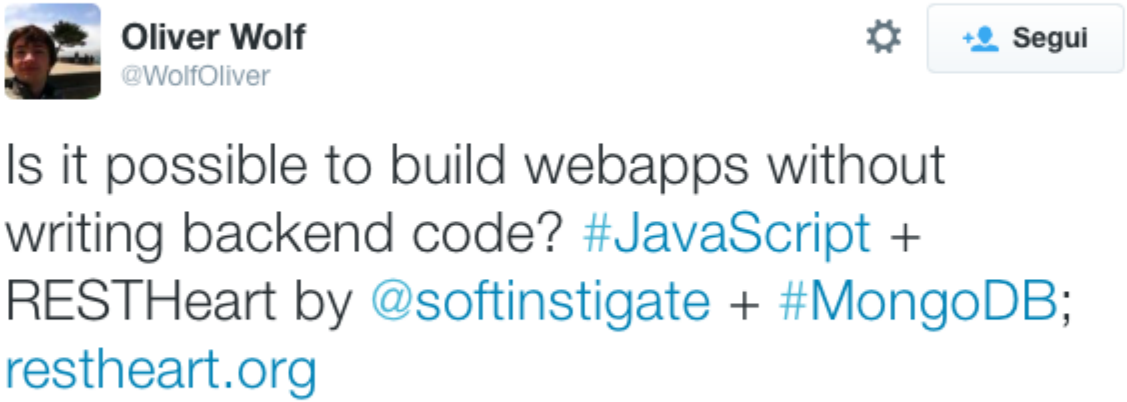
RESTHeart

by SoftInstigate

Agenda

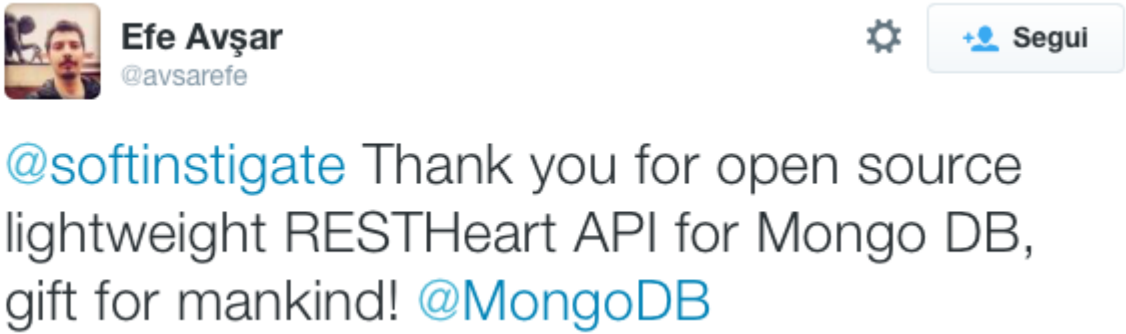
1. Introduzione & Architettura
2. Demo
3. Roadmap
4. Q&A

BTW, we have fans :)



Oliver Wolf
@WolfOliver

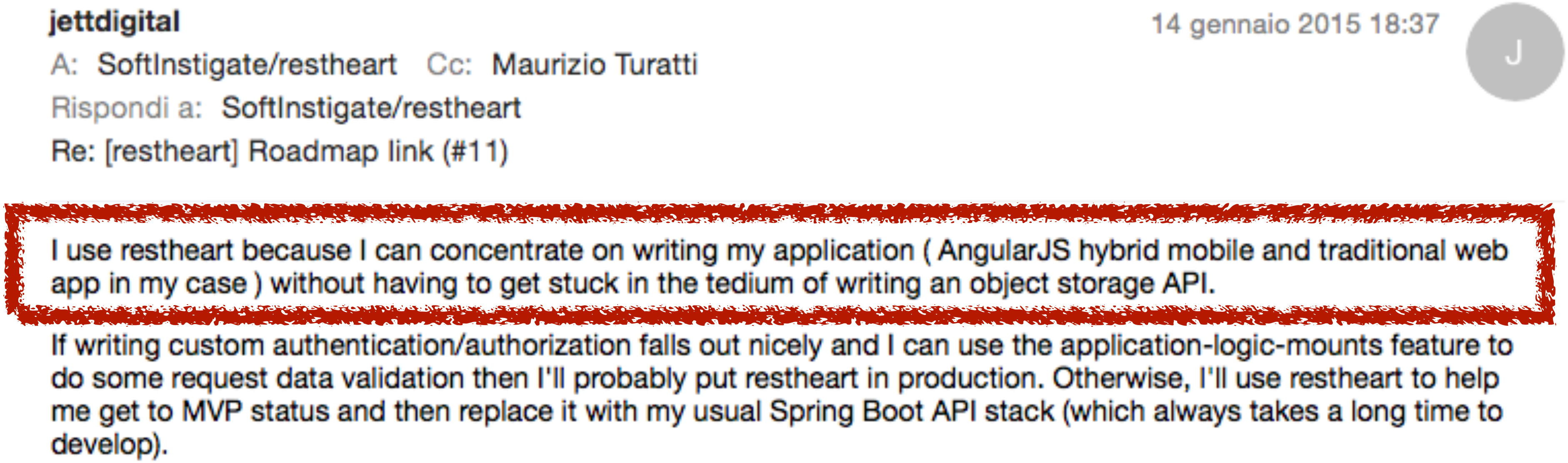
Is it possible to build webapps without writing backend code? #JavaScript + RESTHeart by @softinstigate + #MongoDB; restheart.org



Efe Avşar
@avsarefe

@softinstigate Thank you for open source lightweight RESTHeart API for Mongo DB, gift for mankind! @MongoDB

...real users...



jettdigital

14 gennaio 2015 18:37

J

A: SoftInstigate/restheart Cc: Maurizio Turatti

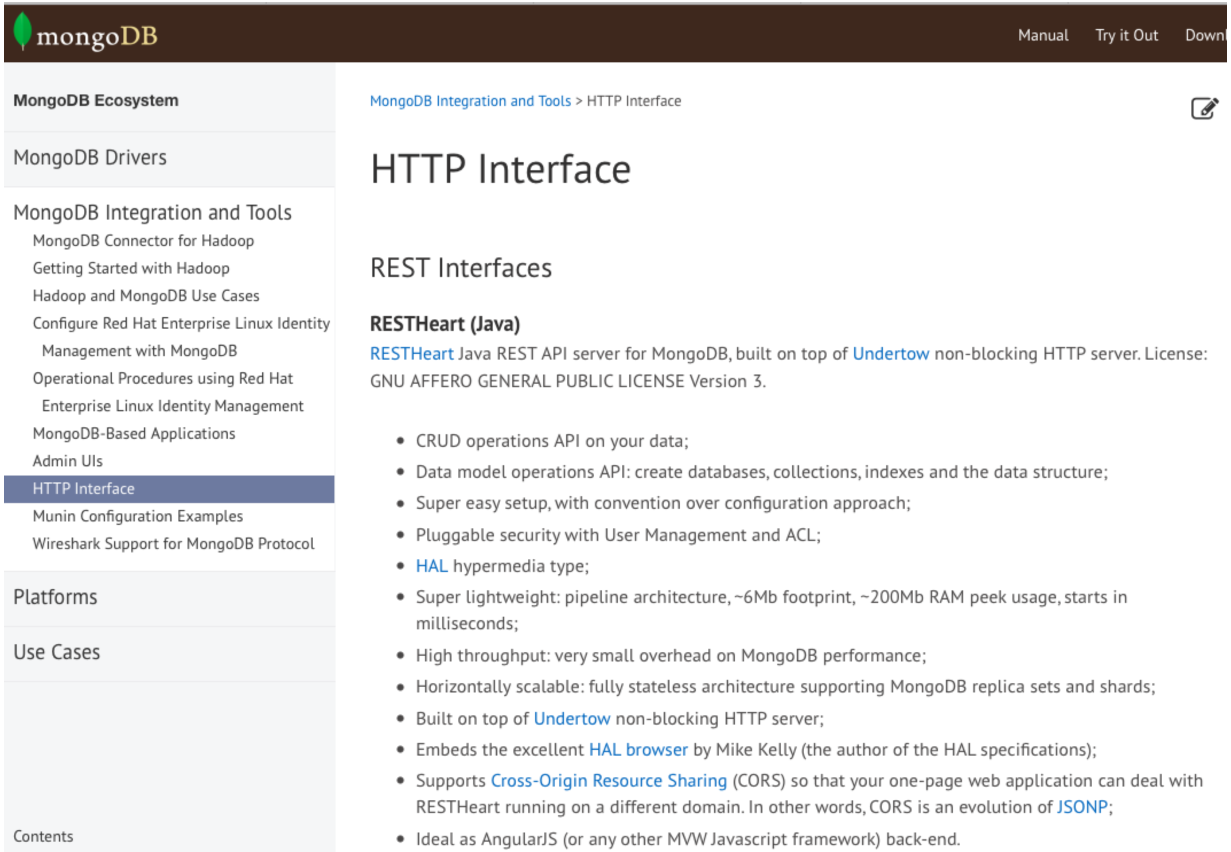
Rispondi a: SoftInstigate/restheart

Re: [restheart] Roadmap link (#11)

I use restheart because I can concentrate on writing my application (AngularJS hybrid mobile and traditional web app in my case) without having to get stuck in the tedium of writing an object storage API.

If writing custom authentication/authorization falls out nicely and I can use the application-logic-mounts feature to do some request data validation then I'll probably put restheart in production. Otherwise, I'll use restheart to help me get to MVP status and then replace it with my usual Spring Boot API stack (which always takes a long time to develop).

...and visibility



MongoDB Integration and Tools > HTTP Interface

HTTP Interface

REST Interfaces

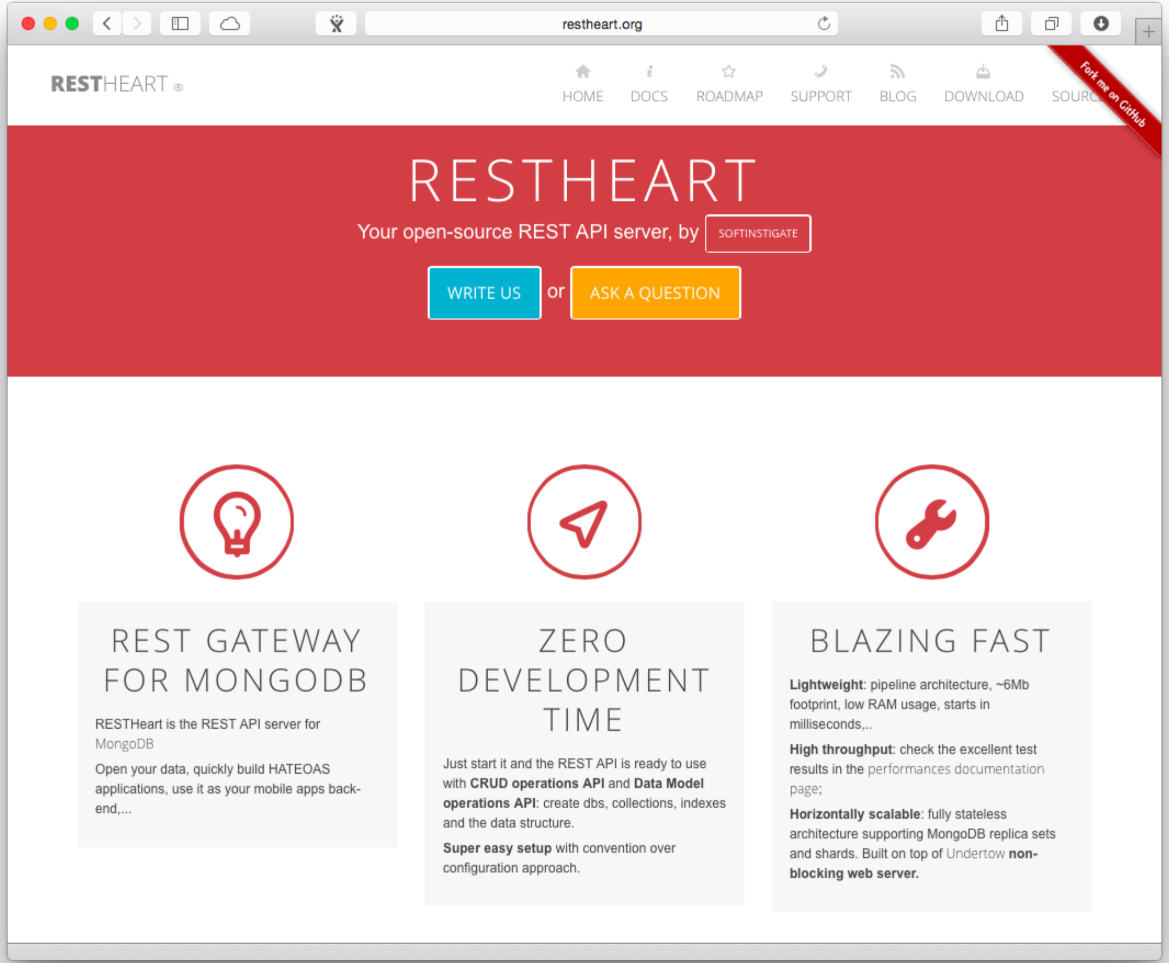
RESTHeart (Java)

RESTHeart Java REST API server for MongoDB, built on top of Undertow non-blocking HTTP server. License: GNU AFFERO GENERAL PUBLIC LICENSE Version 3.

- CRUD operations API on your data;
- Data model operations API: create databases, collections, indexes and the data structure;
- Super easy setup, with convention over configuration approach;
- Pluggable security with User Management and ACL;
- HAL hypermedia type;
- Super lightweight: pipeline architecture, ~6Mb footprint, ~200Mb RAM peek usage, starts in milliseconds;
- High throughput: very small overhead on MongoDB performance;
- Horizontally scalable: fully stateless architecture supporting MongoDB replica sets and shards;
- Built on top of Undertow non-blocking HTTP server;
- Embeds the excellent HAL browser by Mike Kelly (the author of the HAL specifications);
- Supports Cross-Origin Resource Sharing (CORS) so that your one-page web application can deal with RESTHeart running on a different domain. In other words, CORS is an evolution of JSONP;
- Ideal as AngularJS (or any other MVW Javascript framework) back-end.

Summary

- RESTHeart is the **REST API** server for **MongoDB**
- Build **HATEOAS** applications with the **HAL** standard
- Leverages the WWW as an (in)finite state machine
- **CRUD** operations and Data Model operations
- Fully **stateless** => horizontally scalable
- Use it as your Web and mobile apps back-end



RESTHEART

Your open-source REST API server, by SOFTINSTIGATE

REST GATEWAY FOR MONGODB

RESTHeart is the REST API server for MongoDB

Open your data, quickly build HATEOAS applications, use it as your mobile apps back-end,...

ZERO DEVELOPMENT TIME

Just start it and the REST API is ready to use with **CRUD operations API** and **Data Model operations API**: create dbs, collections, indexes and the data structure.

Super easy setup with convention over configuration approach.

BLAZING FAST

Lightweight: pipeline architecture, ~6Mb footprint, low RAM usage, starts in milliseconds,..

High throughput: check the excellent test results in the performances documentation page;

Horizontally scalable: fully stateless architecture supporting MongoDB replica sets and shards. Built on top of Undertow **non-blocking web server.**

Open source project

- Home: http://restheart.org
- Source code is on Github:
  - https://github.com/SoftInstigate/restheart
- GNU AFFERO GENERAL PUBLIC LICENSE V3
- Vagrant boxes available:
  - https://github.com/SoftInstigate/restheart-ansible

Technologies

Java 8
mongoDB
{JSON}
maven
VAGRANT
undertow

Objectives

RESTHeart is a RESTful API Server for data.

The original idea was:

- Q: “What’s the fastest and most direct path to publish some data to the Web?”
- A: “Just put the data in **MongoDB** and enjoy your RESTful, navigable API for free”.

Why MongoDB ?

- **Flexible Data Model**
  - The document data model makes it easy for you to store data of any structure and dynamically modify the schema.
- **Highly Scalable**
  - Scale up or scale out horizontally, from a single server to thousands of nodes.
- **Expressive Query Language**
  - MongoDB's query language provides field-level operators, data types and in-place updates.

Specifications

- **Lightweight**: pipeline architecture, ~7Mb footprint, low RAM usage, starts in milliseconds
- Horizontally **scalable**: fully stateless pipeline
- Pluggable **security** with User Management and ACL
- Use **GridFS** for file management
- Built in **Java 8** on top of **Undertow** non-blocking Web server.

Implemented REST Concepts

Following the REST mantra, you transfer **resource states** back and forth by the means of representations.

- **HAL** - Hypertext Application Language
- **HATEOAS** - Hypermedia as the Engine of Application State
- Embeds the **HAL browser** by Mike Kelly (the author of the HAL specifications)
- Support Cross-origin resource sharing **CORS** so that your single page Web application can deal with RESTHeart running on a different domain.

HATEOAS… WTF?!?

Hypermedia as the Engine of Application State

- The principle is that a client interacts with a network application entirely **through hypermedia** provided dynamically by application servers.
- A REST client needs **no prior knowledge** about how to interact with any particular application or server beyond a generic understanding of hypermedia => no contract!
- By contrast, in a **SOA**, clients and servers interact through a fixed interface
- The HATEOAS constraint decouples client and server in a way that allows the server functionality to evolve independently.

HAL - Hypertext Application Language

RESTHeart uses the **HAL+json** hypermedia format. HAL stands for Hypermedia Application Language and it is simple, elegant and powerful.

Only two simple concepts: Resources and Links.

Resources have:

- Links (to URIs)
- Embedded Resources (i.e. other resources contained within them)
- State (JSON or XML data)

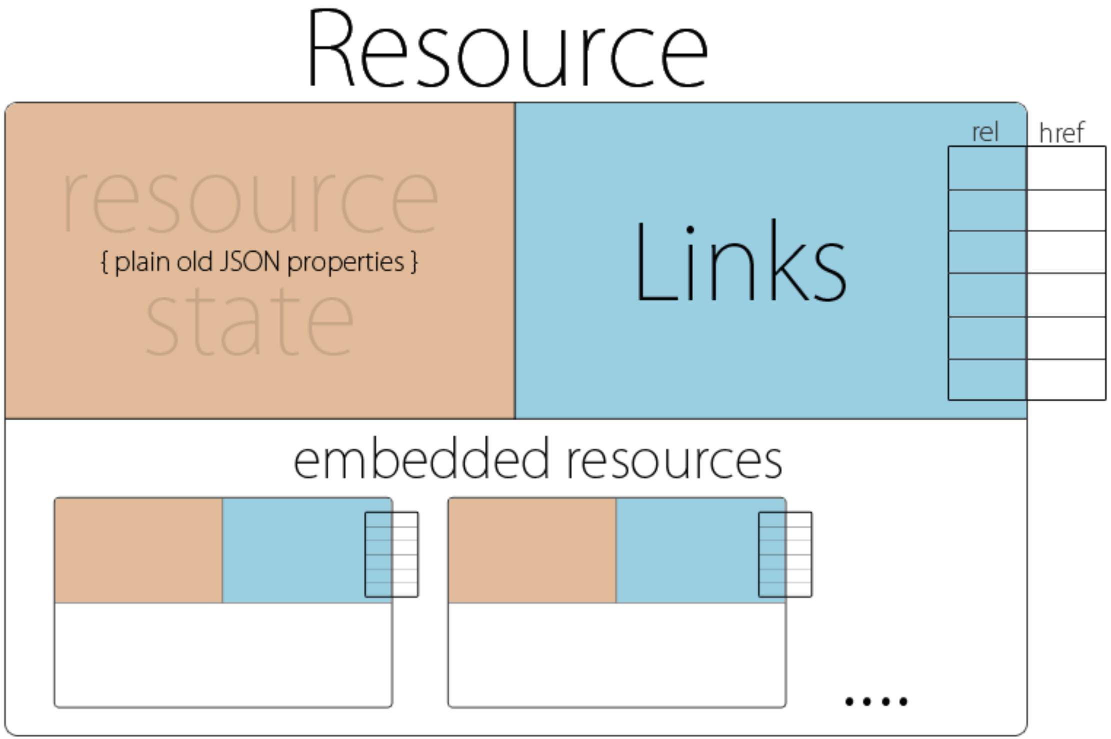
Links have:

- A target (a URI)
- A relation aka. 'rel' (the name of the link)
- A few other optional properties to help with deprecation, content negotiation, etc.

How it looks like?

```
{
    "_links": {
        "self": { "href": "https://api.example.com/player/1234567890" },
        "friends": { "href": "https://api.example.com/player/1234567890/friends" }
    },
    "playerId": "1234567890",
    "name": "Kevin Sookocheff",
    "nickname": "soofaloofa",
    "image": "https://api.example.com/player/1234567890/avatar.png"
}
```

Resources


Resource
resource
{ plain old JSON properties }
state
Links
rel
href
embedded resources
....

DEMO

Roadmap

Next release 0.11

Bug

- [RH-35] - relations on nested document fields

Improvement

- [RH-37] - allow to enable/disable the CORS headers by configuration
- [RH-43] - allow defining write concern
- [RH-54] - consider replacing our AccessManagerHandler with AccessControlListHandler
- [RH-64] - allow file metadata update
- [RH-67] - support projection and fields to return selection

New Feature

- [RH-19] - support array update operators
- [RH-41] - stored queries
- [RH-49] - DB based IDM implementation
- [RH-53] - schema validation/enforcement
- [RH-68] - automatic document properties injection via metadata

Releases 0.12 and 0.13

Improvement

- [RH-20] - exception logging and access to log file via http
- [RH-28] - more system properties in resource representation
- [RH-42] - javadoc documentation

New Feature

- [RH-7] - batch operations
- [RH-22] - provide operation details on response
- [RH-23] - collect and display statistics

New Feature

- [RH-51] - allow plugging in a distributed cache
- [RH-52] - application logic via javascript function metadata - TBD

Q&A

Grazie!

http://restheart.org

Web	http://www.softinstigate.com/
E-mail	info@softinstigate.com
Twitter	@softinstigate