Java-Success.com

Register | Login | Logout | Contact Us

Industrial strength Java/JEE Career Companion to open more doors

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

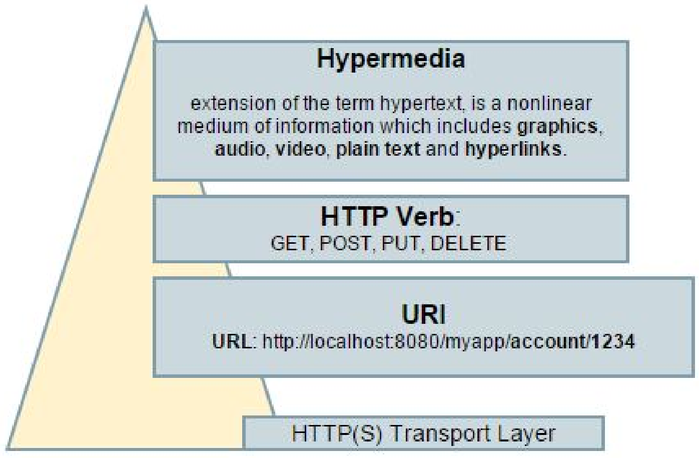
Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › Pressed for time? Java/JEE Interview FAQs › FAQ Java Web Services Interview Q&A Essentials › 06: RESTful Web services and HATEOAS Q&A

06: RESTful Web services and HATEOAS Q&A

Posted on June 16, 2015 by Arulkumaran Kumaraswamipillai

Q1. What is HATEOAS? How does it provide state transition, scalability, and loose coupling?
A1. HATEOAS (Hypermedia as the Engine of Application State) is considered the final level of REST. This means that each link is presumed to implement the standard REST verbs of GET, POST, PUT, and DELETE (or a subset).


Hypermedia
extension of the term hypertext, is a nonlinear medium of information which includes graphics, audio, video, plain text and hyperlinks.
HTTP Verb:
GET, POST, PUT, DELETE
URI
URL: http://localhost:8080/myapp/account/1234
HTTP(S) Transport Layer

REST Layers

Layers of REST & where does HATEOAS sit?

Level 0: Protocol– Transporting protocol (normally HTTP, but it doesn't have to be).

Level 1: Resources – This level uses multiple URIs, where every URI is the entry point to a specific resource.

Level 2: HTTP verbs – This level indicates that your API should use the protocol properties in order to deal with scalability and failures. Don't use a single POST method for all, but make use of GET when you are requesting resources, and use the DELETE method when you want to delete a resources.

Level 3: Hypermedia controls – uses HATEOAS to deal with discovering the possibilities of your API towards the clients. The point of hypermedia controls is that they tell us what we can do next, and the URI of the resource we need to manipulate to do it.

Benefits?

1) One obvious benefit of hypermedia controls is that it allows the server to change its URI scheme **without breaking**

16 Technical Key Areas

80+ step by step Java Tutorials

clients . So, loosely couples the clients from the server.

2) The entire service is discover-able starting from the root URI, hence the documentation is not required. The links give client developers a hint as to what may be possible next. So, it is a form of documentation.

HATEOAS with JSON example

The key to implementing HATEOAS is quite simple by including **links** in responses that go from the server to the client.

JSON Response without **HATEOAS**

```
{ Book  {
     "id" : 1234,
     "title" : "Java/JEE Job Interview Companion"
   }
}

```

JSON Response with **HATEOAS**

```
{  Book {
     "id" : 1234,
     "title" : "Java/JEE Job Interview Companion"
     "links": [ {
         "rel": "BookInfo",
         "href": "http://localhost:8080/estore/cat
     } ]
   }
}
```

rel: means relationship. Link gives more info about the book with id = 1234.

href: means absolute URL that uniquely defines the resource.

Here are more links to get book reviews and ratings.

```
{  Book {
```

100+ Java pre-interview coding tests

How good are your .....?

```
    "id" : 1234,
    "title" : "Java/JEE Job Interview Companion"
    "links": [ {
        "rel": "BookInfo",
        "href": "http://localhost:8080/estore/ca
    }, {
        "rel": "BookReview",
        "href": "http://localhost:8080/estore/re
    }, {
        "rel": "BookRating",
        "href": "http://localhost:8080/estore/ra
    }, {
        "rel": "CheckOut",
        "href": "http://localhost:8080/order/boo
    }]
  }
}
```

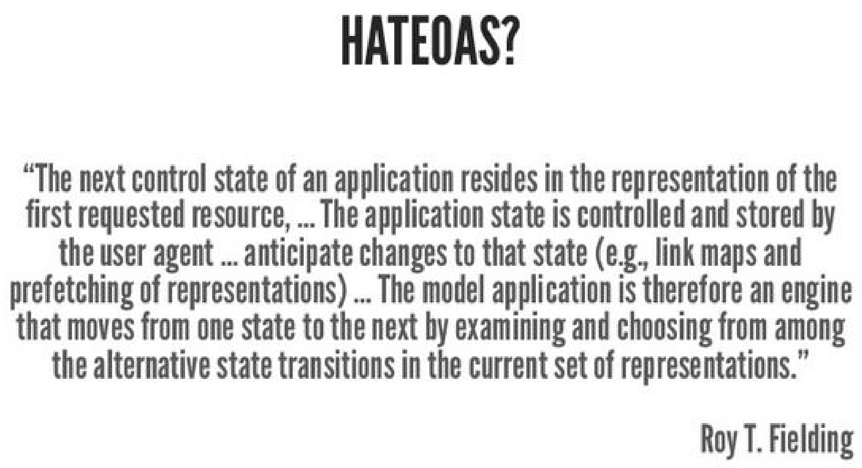
State & Behavior

So, HATEOAS refers to allowing clients to navigate through appropriate application states using hyperlinks. The client can simply look at the response for presence of tags “BookInfo”, “BookReview”, and “BookRating” to navigate and get additional information. It is also important to note that the client of the service doesn’t have to figure out possible “states” or “outcomes”of a request. The possible next states are all captured with hyperlinks by the server.

For example, in an an order processing system, the “rel” and “href” will give you the possible outcomes of placing an order. When browsing a book, you have the choices of “BookInfo”, “BookReview”, “BookRating”, “CheckOut”, etc. Once you have checked out, you will have the options of “OrderStatus”, “CancelOrder” , and “WriteReview”.

So, by adding hyperlinks to control next steps on the server side and NOT making the clients construct URIs, the application becomes a lot **more flexible**. This promotes **loose coupling**, and makes the order processing service to **evolve with new business rules without breaking the existing clients**. The state transitions and business rules are responsibilities of the server. In a service oriented architecture, a service may have many clients. Making the changes in every client is not flexible.

HATEOAS is a design principle that states: “clients should only interact with network applications using hypermedia controls. Neither the client software, and its developer, require specialized knowledge about how to interface with the server beyond a URL and a general understanding of hypertext protocols such as http. This improves the flexibility, security, and scalability of REST as an application architecture.


HATEOAS?

“The next control state of an application resides in the representation of the first requested resource, … The application state is controlled and stored by the user agent … anticipate changes to that state (e.g., link maps and prefetching of representations) … The model application is therefore an engine that moves from one state to the next by examining and choosing from among the alternative state transitions in the current set of representations.”

Roy T. Fielding

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

766 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ Part 3- Adding plugins to multi-module MVN project to perform specific tasks

05: RESTFul Web Service URI conventions with Spring MVC examples ›

Posted in FAQ Java Web Services Interview Q&A Essentials, member-paid, WebService

Tags: Architect FAQs

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer