02: ♦ 6 Java RESTful Web services Interview Q&A you must know

Posted on August 20, 2014 by Arulkumaran Kumaraswamipillai

Q1. What is RESTful Web service, and why is it favored over SOAP Web service?

A1. REST stands for REpresentational State Transfer (REST), which is a stateless software architecture that reads webpages containing XML, JSON, Plain text, etc. REST is a simpler alternative to Simple Object Access Protocol (SOAP) and Web Services Description Language Web services (WSDL), and has become a popular Web application program interface (API) model. A RESTful API, or RESTful Web service, uses both HTTP and REST.

1) REST is very lightweight, and does not have all the complexity of SOAP
2) REST is a very simple in that it uses HTTP GET, POST, PUT, and DELETE methods to update resources on the server.
3) REST typically is best used with Resource Oriented Architecture (ROA). In this mode of thinking everything is a resource, and performs CRUD (Create, Read, Update, and Delete) operations on those resources. HTTP POST to Create a resource on the server, HTTP GET to Read a resource from the server, HTTP PUT to Update a resource on the server, and HTTP DELETE to delete a resource on the server. The resource could be account, transaction, etc.

For example:

```
1 http(s)://myserver.com:8080/app-name/{version-no}
2
3 # to list all the accounts:
4 http(s)://myserver.com:8080/accounting-services/1
5
6 # creates a new transaction for account 123
7 http(s)://myserver.com:8080/accounting-services/1
8
```

RESTful Web service is easy, straightforward, supports multiple data formats like XML, JSON, etc and easily testable. For example,

4) It can be tested by

a. Directly invoking the service from the browser by typing a URL if the RESTFul service supports GET request with query parameters. For example,

```
1 http://localhost:8080/executionservices/execution
```

b. You could use a Firefox plugin like “poster” to post an XML request to your service.

c. You could write a Web Service client to consume the Web service from a test class or a separate application client. You

16 Technical Key Areas

80+ step by step Java Tutorials

could use libraries like HttpClient, CXF Client, URLConnection, etc to connect to the RESTful service.

Q2. What is the RESTFul Web Service URI conventions? Can you discuss verbs and nouns and singular Vs plural resources?

A2. The high level pattern for the RESTful URI is

```
1 http(s)://myserver.com:8080/app-name/{version-no}
2
```

to list all the accounts:

```
1 http(s)://myserver.com:8080/accounting-services/1
2
```

This is a **plural** resource returning a collections of accounts. The URI contains **nouns** representing the resources in a hierarchical structure. For example, if you want a to get a particular transaction value under an account you can do

```
1 http(s)://myserver.com:8080/accounting-services/1
```

Where 123 is the account number and 567 is the transaction number or id. This is a singular resource returning a single transaction.

if you want to list a collection of transactions that are greater than a particular date?

```
1 http(s)://myserver.com:8080/accounting-services/1
2
```

The **verbs** are defined via the HTTP methods GET, POST, PUT, and DELETE. The above examples are basically GET requests returning accounts or transactions. If you want to create a new transaction request, you do a POST with the following URL.

```
http(s)://myserver.com:8080/accounting-services/1
```

The actual transaction data will be sent in the body of the request as JSON data. The above URI will be used with a PUT http method to modify an existing transaction record.

You can also control which method gets executed with the help of HTTP headers or host names in the URL. Here is a Spring MVC example:

```
@Controller
@RequestMapping("/forecasting")
public class CashForecastController

{
    @RequestMapping(
            value = "/accounts,
            method = RequestMethod.GET,
            produces = "application/json")
    @ResponseBody
    public AccountResult getAllAccounts(HttpServ
    {
        //get the accounts via a service and a d
    }

    @RequestMapping(
            value = "/account/transaction",
            method = RequestMethod.POST)
    public @ResponseBody Transaction addTransact
            throws Exception
    {
        //logic to create a new Transaction reco
    }

  @RequestMapping(
            value = "/account/transaction",
            method = RequestMethod.PUT)
    public @ResponseBody Transaction modifyTrans
            throws Exception
    {
        //logic to modify a Transaction record v
    }
}

```

Q. Dos and Don’ts:

— Don’t use query parameters to alter state. Use query parameters for sub-selection of a resource like pagination, filtering, search queries, etc
— Don’t use implementation-specific extensions in your URIs (.do, .py, .jsf, etc.). You can use .csv, .json, etc.

— Don't ever use GET to alter state. Use GET for as much as possible. Favor POST over PUT when in doubt.
— Don't perform an operation that is not idempotent with PUT.
— Do use DELETE in preference to POST to remove resources.
— Don't clutter your URL with verbs or stuff that should be in a header or body. Move stuff out of the URI that should be in an HTTP header or a body. Whenever it looks like you need a new verb in the URL, think about turning that verb into a noun instead. For example, turn 'activate' into 'activation', and 'validate' into 'validation'.

Q3. What are the various implementations of JAX-RS available to choose from in Java?
A3. When you're developing with REST in Java, you have a lot of options to choose from in terms of the frameworks. There's Jersey, the reference implementation from Oracle, then you have RestEasy, the JBoss choice, and there is CXF, the Apache choice.

Q4. How would you go about implemnting the RESTful Web service using the framework of your choice?
A4.

Step 1: In the pom.xml define the JAX-RS library

```
<dependencies>
        <dependency>
            <groupId>org.jboss.resteasy</groupId>
            <artifactId>resteasy-jaxrs</artifactI
            <version>2.2.1.GA</version>
        </dependency>
</dependencies>

```

Step 2: Implement the RESTful Web service

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.core.Response;

@Path("/hello")
```

```
7  public class HelloServiceImpl {
8
9      @GET
10     @Path("/{name}")
11     public Response printMessage(@PathParam("nam
12         String result = "Restful hello: " + name
13         return Response.status(200).entity(resul
14     }
15 }
16
```

Step 3: Bootstrap the jboss resteasy via web.xml deployment descriptor

```
1  <web-app id="WebApp_ID" version="2.4"
2      xmlns="http://java.sun.com/xml/ns/j2ee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
4      xsi:schemaLocation="http://java.sun.com/xml/
5      http://java.sun.com/xml/ns/j2ee/web-app_2_4.
6      <display-name>Restful Web Application</displ
7
8      <!-- Auto scan REST service -->
9      <context-param>
10         <param-name>resteasy.scan</param-name>
11         <param-value>true</param-value>
12     </context-param>
13
14     <!-- this need same with resteasy servlet ur
15     <context-param>
16         <param-name>resteasy.servlet.mapping.pre
17         <param-value>/rest</param-value>
18     </context-param>
19
20     <listener>
21         <listener-class>
22                org.jboss.resteasy.plugins.server
23         </listener-class>
24     </listener>
25
26     <servlet>
27         <servlet-name>resteasy-servlet</servlet-
28         <servlet-class>
29              org.jboss.resteasy.plugins.server.s
30         </servlet-class>
31     </servlet>
32
33     <servlet-mapping>
34         <servlet-name>resteasy-servlet</servlet-
35         <url-pattern>/rest/*</url-pattern>
36     </servlet-mapping>
37
38 </web-app>
39
```

Step 4: The URL to hit the resource

```
1 http://localhost:8080/RESTfulExample/rest/hello/P
```

Step 5: The output:

"Restful hello: Peter"

Q5. What happens if RestFul resources are accessed concurrently by multiple clients ?
A5.Since a new Resource instance is created for every incoming Request, there is no need to make it thread-safe or add synchronization. Concurrent clients can safely access the RestFul resources.

Q6. What are some of the annotations used in JAX-RS?
A6.

@GET, @POST, @PUT, @DELETE to specify what type of verb this method (or web service) will perform

```
1 @DELETE
2 @Produces("application/json")
3 @Path("{accountId}")
4 public RestResponse<Account> delete(@PathParam("a
5 ...
6 }
7
```

@Produces to specify the type of output this method (or web service) will produce.

```
1 @GET
2 @Produces("application/json")
3 public Account getAccount() {
4   ...
5 }
6
```

@Consumes to specify the MIME media types a REST resource can consume

```
1 @PUT
2 @Consumes("application/json")
3 @Produces("application/json")
4 @Path("{accountId}")
5 public RestResponse<Account> update(Account accou
6 ...
7 }
```

```
8
```

@Path to specify the URL path on which this method will be invoked

```
1 @GET
2 @Produces("application/xml")
3 @Path("accounting-services/{accountName}")
4 public Account getAccount() {
5   ...
6 }
7
```

@PathParam to bind REST style parameters to method arguments. For example http://localhost:8080/context/accounting-services/PeterAndCo

```
1 @GET
2 @Produces("application/xml")
3 @Path("accounting-services/{accountName}")
4 public Acount getAccount(@PathParam("accountName"
5   Account account = accountService.findByAccountN
6   return account;
7 }
8
```

@QueryParam to access parameters in query string (http://localhost:8080/context/accounting-services?accountName=PeterAndCo).

```
1 @GET
2 @Produces("application/xml")
3 @Path("accounting-services")
4 public Acount getAccount(@QueryParam("accountName
5   Account account = accountService.findByAccountN
6   return account;
7 }
8
```

@FormParam to read parameters sent in a POST request. REST resources usaually consume XML/JSON, but at times you want to read the paramewters in POST.

```
1 @POST
2 public String save(@FormParam("accountName") Stri
3     @FormParam("accountNumber") String accountNum
4       ...
5 }
```

6

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**

‹ 09: 11 SOAP Web service interview questions and answers

04: 5 JAXB interview Questions & Answers ›

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

Tags: Java/JEE FAQs, JEE 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

 ↑