

# Java-Success.com

[Register](#) | [Login](#) | [Logout](#) | [Contact Us](#)

Industrial strength Java/JEE Career Companion to open more doors


[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](#) › 05: RESTful Web Service URI conventions with Spring MVC examples

## 05: RESTful Web Service URI conventions with Spring MVC examples

Posted on [June 16, 2015](#) by [Arulkumaran Kumaraswamipillai](#)



The high level pattern for the **RESTful URI** is

- `http(s)://myserver.com:8080/app-name/{version-no}/{domain}/{rest-resource-convention}`

**For example:**

- `http(s)://myserver.com:8080/accounting-services/1.0/forecasting/accounts`

### 600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[JEE Interview Q&A \(3](#)

[JEE Overview \(2\)](#)

[Web basics \(8\)](#)

[WebService \(11\)](#)

[01: ♥♦ 40+ Java](#)

[02: ♦ 6 Java RE](#)

[03: ♥ JAX-RS hc](#)

[04: 5 JAXB inter](#)

[05: RESTful We](#)

[06: RESTful Wel](#)

[07: HATEOAS R](#)

[08: REST constr](#)

[09: 11 SOAP We](#)

[10: SOAP Web](#)

[11: ♥ JAX-WS hc](#)

[JPA \(2\)](#)

[JTA \(1\)](#)

[JDBC \(4\)](#)

[JMS \(5\)](#)

to list all the accounts. 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

- `http(s)://myserver.com:8080/accounting-services/1.0/forecasting/account/123/transaction/567`

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

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

- `http(s)://myserver.com:8080/accounting-services/1.0/forecasting/account/123/transactions/searchtxn-date=20120201`

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/1.0/forecasting/account/123/transaction`

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.

Finally, you can also control which method gets executed with the help of HTTP headers or host names in the URL. Let's see some **Spring MVC** examples in a controller class as to how it maps a request URI, headers, etc to execute the relevant method on the server side.

Here is the sample code with GET requests

- [JMX \(3\)](#)
- [JNDI and LDAP \(1\)](#)
- [Pressed for time? Java](#)
- [SQL, XML, UML, JSC](#)
- [Hadoop & BigData Int](#)
- [Java Architecture Inte](#)
- [Scala Interview Q&As](#)
- [Spring, Hibernate, & I](#)
- [Testing & Profiling/Sa](#)
- [Other Interview Q&A 1](#)
- [Free Java Interview](#)

## 16 Technical Key Areas

[open all](#) | [close all](#)

- [Best Practice \(6\)](#)
- [Coding \(26\)](#)
- [Concurrency \(6\)](#)
- [Design Concepts \(7\)](#)
- [Design Patterns \(11\)](#)
- [Exception Handling \(3\)](#)
- [Java Debugging \(21\)](#)
- [Judging Experience In](#)
- [Low Latency \(7\)](#)
- [Memory Management](#)
- [Performance \(13\)](#)
- [QoS \(8\)](#)
- [Scalability \(4\)](#)
- [SDLC \(6\)](#)
- [Security \(13\)](#)
- [Transaction Managen](#)

## 80+ step by step Java Tutorials

[open all](#) | [close all](#)

- [Setting up Tutorial \(6\)](#)

```

1 @Controller
2 @RequestMapping("/forecasting")
3 public class CashForecastController
4 {
5
6
7     @RequestMapping(
8         value = "/accounts",
9         method = RequestMethod.GET,
10        produces = "application/json")
11     @ResponseBody
12     public AccountResult getAllAccounts(HttpServ
13     {
14         //get the accounts via a service and a d
15     }
16
17     @RequestMapping(
18         value = "/accounts.csv",
19         method = RequestMethod.GET,
20        produces = "text/csv")
21     @ResponseBody
22     public void getAllAccounts(HttpServletResponse
23     {
24         //produces a CSV download file
25     }
26
27
28     @RequestMapping(
29         value = "/account/{accountCd}",
30        method = RequestMethod.GET,
31        produces = "application/json")
32     @ResponseBody
33     public Account getAccount(
34         @PathVariable(value = "accountCd") S
35     {
36         //get the accounts via a service and a d
37     }
38 }
39
40 //accept only if there is a special header
41 @RequestMapping(
42     value = "/account/{accountCd}",
43     method = RequestMethod.GET,
44     headers =
45     {
46         "operation=special"
47     }
48     produces = "application/json")
49     @ResponseBody
50     public Account getAccountSpecial(
51         @PathVariable(value = "accountCd") S
52     {
53         //get the accounts via a service and a d
54         //special handling based
55     }
56
57     @RequestMapping(
58         value = "/account/{accountCd}/transa
59         method = RequestMethod.GET,
60        produces = "application/json")
61     @ResponseBody
62     public Transaction getTransaction(
63         @PathVariable(value = "accountCd") S
64         @PathVariable(value = "transactionId") String
65         HttpServletResponse response) throws Exceptio

```

- [Tutorial - Diagnosis \(2](#)
- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2](#)
- [Hadoop & Spark Tuto](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate Ti](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

## 100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(](#)
- [♦ Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

## How good are your .....?

[open all](#) | [close all](#)

- [Career Making Know-](#)
- [Job Hunting & Resum](#)

```

66     {
67         //get the accounts via a service and a d
68         //accountCode and txnId can be used here
69     }
70
71     @RequestMapping(
72         value = "/account/{accountCd}/transa
73         method = RequestMethod.GET,
74         produces = "application/json")
75     @ResponseBody
76     public TransactionResult getTransactions(
77         @PathVariable(value = "accountCd") S
78         @RequestParam(value = "txn-date", required =
79         HttpServletResponse response) throws Exceptio
80     {
81         //get the accounts via a service and a d
82         //accountCode and txnDate can be used here
83     }
84 }
85
86

```

Here is the sample code with POST and PUT requests

```

1  @Controller
2  @RequestMapping("/forecasting")
3  public class CashForecastController
4  {
5      @RequestMapping(
6          value = "/account/transaction",
7          method = RequestMethod.POST)
8      public @ResponseBody Transaction addTransact
9          throws Exception
10     {
11
12         //logic to create a new Transaction reco
13
14     }
15
16
17
18     @RequestMapping(
19         value = "/account/transaction",
20         method = RequestMethod.PUT)
21     public @ResponseBody Transaction modifyTrans
22         throws Exception
23     {
24
25         //logic to modify a Transaction record v
26
27     }
28 }
29
30

```

- 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'.

## 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](http://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](http://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](#)

◀ 06: RESTful Web services and HATEOAS Q&A

08: REST constraints (i.e. design rules) interview Q&A ▶

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

## 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

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.