



FOR THE ON GHAND



# How to test a REST api from command line with curl



\*\*Promotion\*\* - Efficiently manage your coding bookmarks, aka #codingmarks, on www.codingmarks.org\* and share your hidden gems with the world. They will be published weekly on Github. Please help us build

THE programming-resources location - 🖈 Star

If you want to quickly test your REST api from the command line, you can use curl. In this post I will present how to execute GET, POST, PUT, HEAD, DELETE HTTP Requests against a REST API. For the purpose of this blog post I will be using the REST api developed in my post Tutorial - REST API design and implementation in Java with Jersey and Spring

## Contents

- 1. Introduction
  - 1.1. What is curl?
  - 1.2. HEAD requests
  - o 1.3. GET request
  - 1.4. Curl request with multiple headers
- 2. SOAPui test suite translated to curl requests
  - o 2.1. Create podcast(s) resource
    - 2.1.1. Delete all podcasts (preparation step)
    - 2.1.2. POST new podcast without feed 400 (BAD\_REQUEST)
    - 2.1.3. POST new podcast correctly 201 (CREATED)
    - 2.1.4. POST same podcast as before to receive 409 (CONFLICT)
    - 2.1.5. PUT new podcast at location 201 (CREATED)
  - o 2.2. Read podcast resource
    - 2.2.1. GET new inserted podcast 200 (OK)
    - 2.2.2. GET podcasts sorted by insertion date DESC 200 (OK)
  - 2.3. Update podcast resource
    - 2.3.1. PUT not "complete" podcast for FULL update 400 (BAD\_REQUEST)
    - 2.3.2. PUT podcast for FULL update 200 (OK)
  - 2.3.3. POST (partial update) for not existent podcast 404 (NOT\_FOUND)
    - 2.3.4. POST (partial update) podcast 200 (OK)
  - 2.4. DELETE resource
    - 2.4.1. DELETE second inserted podcast 204 (NO\_CONTENT)
    - 2.4.2. GET deleted podcast 404 (NOT\_FOUND)
  - o 2.5. Bonus operations
    - 2.5.1. Add podcast from application form urlencoded
- Resources















part I will "translate" the SOAPul test suite developed for the REST API tutorial to curl requests

#### 1.1. What is curl?

Curl is a command line tool and library for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, Kerberos...), file transfer resume, proxy tunneling and more.[1]

As mentioned, I will be using curl to simulate HEAD, GET, POST, PUT and DELETE request calls to the REST API.

## 1.2. HEAD requests

If you want to check if a resource is serviceable, what kind of headers it provides and other useful meta-information written in response headers, without having to transport the entire content, you can make a HEAD request. Let's say I want to see what I would GET when requesting a Podcast resource. I would issue the following HEAD request with curl:

#### Request

curl -I http://localhost:8888/demo-rest-jersey-spring/podcasts/1



OR

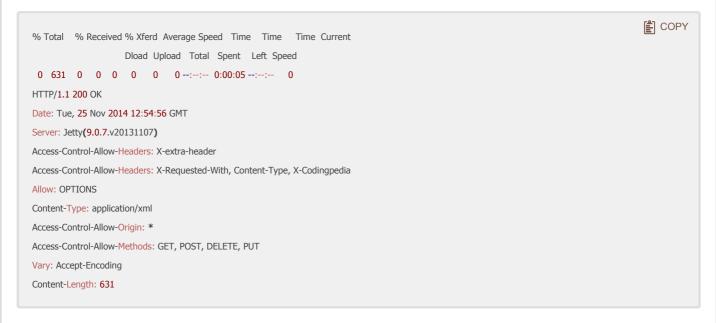
curl -i -X HEAD http://localhost:8888/demo-rest-jersey-spring/podcasts/1



## **Curl options**

- [-i, --include] include protocol headers in the output (H/F)
- -X, --request specify request COMMAND (GET, PUT, DELETE...) to use

#### Response



Note the following headers

Access-Control-Allow-Headers: Content-Type



They've been added to support Cross-Origing Resource Sharing (CORS). You can find more about that in my post How to add CORS support on the server side in Java with Jersey.

What I find a little bit intriguing is the response header Content-Type: application/xml , because I would have expected it to be application/json , since in the resource method defined with Jersey this should have taken precedence:

```
@GET
@Path("{id}")
@Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
public Response getPodcastById(@PathParam("id") Long id, @QueryParam("detailed") boolean detailed)
    throws IOException, AppException {
    Podcast podcastById = podcastService.getPodcastById(id);
    return Response.status(200)
        .entity(podcastById, detailed ? new Annotation[]{PodcastDetailedView.Factory.get()} : new Annotation[0])
        .header("Access-Control-Allow-Headers", "X-extra-header")
        .allow("OPTIONS").build();
}
```

## 1.3. GET request

Executing curl with no parameters on a URL (resource) will execute a GET.

## Request

curl http://localhost:8888/demo-rest-jersey-spring/podcasts/1



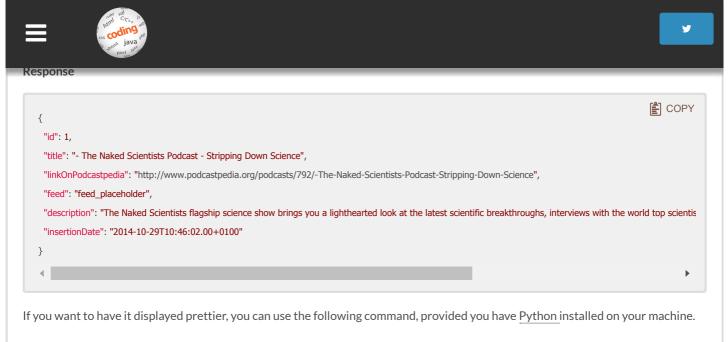
#### Response

Note that as expected from the HEAD request we get an xml document. Anyway we can force a JSON response by adding a header line to our curl request, setting the Accept HTTP header to application/json:

```
curl --header "Accept:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts/1
```

#### **Curl options**

• [-H, --header] – customer header to pass to the server



#### Request

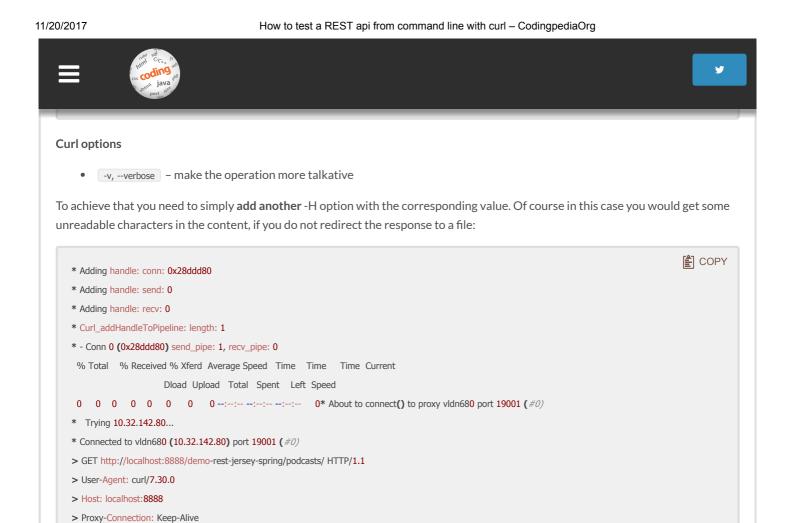
```
curl -H "Accept:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts/1 | python -m json.tool
```

#### Response

```
COPY
% Total % Received % Xferd Average Speed Time Time Current
                     Dload Upload Total Spent Left Speed
100 758 100 758 0 0 6954 0 --:--:-- 6954
I
  {
     "description": "The Naked Scientists flagship science show brings you a lighthearted look at the latest scientific breakthroughs, interviews with the world top sci
     "feed": "feed_placeholder",
     "id": 1,
     "insertionDate": "2014-10-29T10:46:02.00+0100",
     "linkOnPodcastpedia": "http://www.podcastpedia.org/podcasts/792/-The-Naked-Scientists-Podcast-Stripping-Down-Science",
     "title": "- The Naked Scientists Podcast - Stripping Down Science"
  },
  {
     "description": "Quarks & Co: Das Wissenschaftsmagazin",
     "feed": "http://podcast.wdr.de/quarks.xml",
     "id": 2,
     "insert
     ionDate": "2014-10-29T10:46:13.00+0100",
     "linkOnPodcastpedia": "http://www.podcastpedia.org/quarks",
     "title": "Quarks & Co - zum Mitnehmen"
  }
1
```

#### 1.4. Curl request with multiple headers

As you've found out in my latest post, <u>How to compress responses in Java REST API with GZip and Jersey</u>, all the responses provided by the REST api are being compressed with GZip. This happens only if the client "suggests" that it accepts such encoding, by setting the following header <u>Accept-encoding:gzip</u>.



100 413 100 413 0 0 2647 0-:--:-- --:--:-- 2647 QKo 0 + g R +{ V Pe c c n fæHH " g g //22 eM g|

| 08># 0b! | G ^ I.i.TU | z \ i^]e | 2 | ? :/ m | Y h | <u>雌</u>

# 2. SOAPui test suite translated to curl requests

Accept:application/jsonAccept-encoding:gzip

< Date: Tue, 25 Nov 2014 16:17:02 GMT

< Server: Jetty(9.0.7.v20131107)
< Content-Type: application/json
< Access-Control-Allow-Origin: \*

< Vary: Accept-Encoding < Content-Encoding: gzip < Content-Length: 413 < Via: 1.1 yldn680:8888

{ [data not shown]

{=`7 Εω c ZM

u a 9hO

\* Server Jetty(9.0.7.v20131107) is not blacklisted

< Access-Control-Allow-Methods: GET, POST, DELETE, PUT

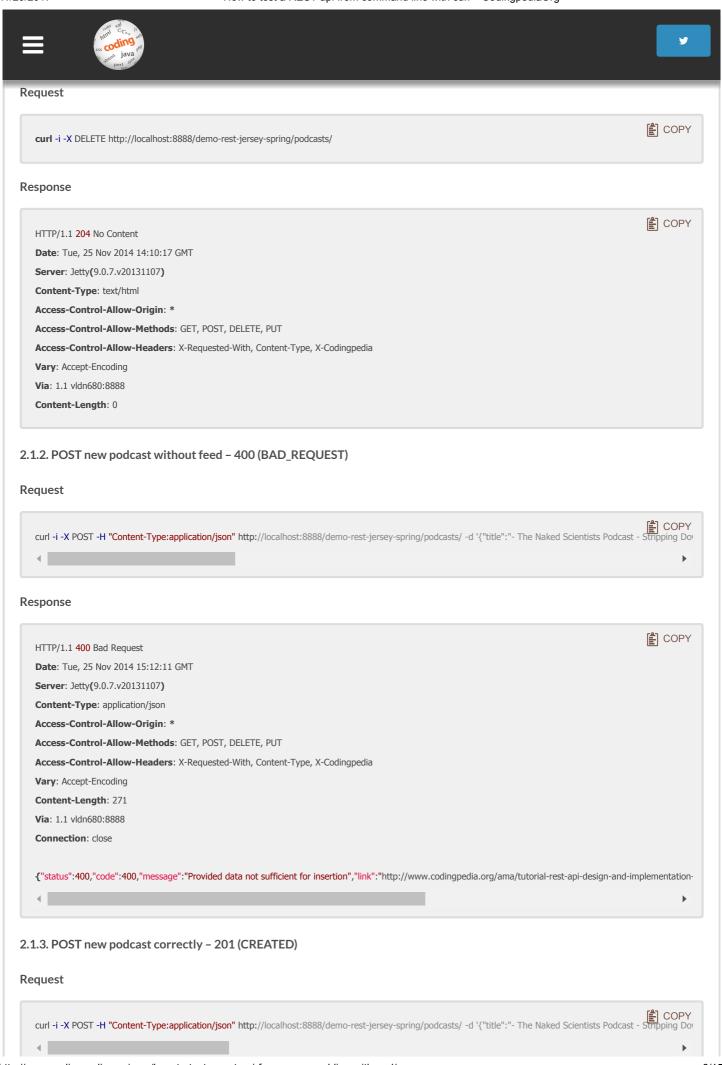
# h i gq \$ |Ň

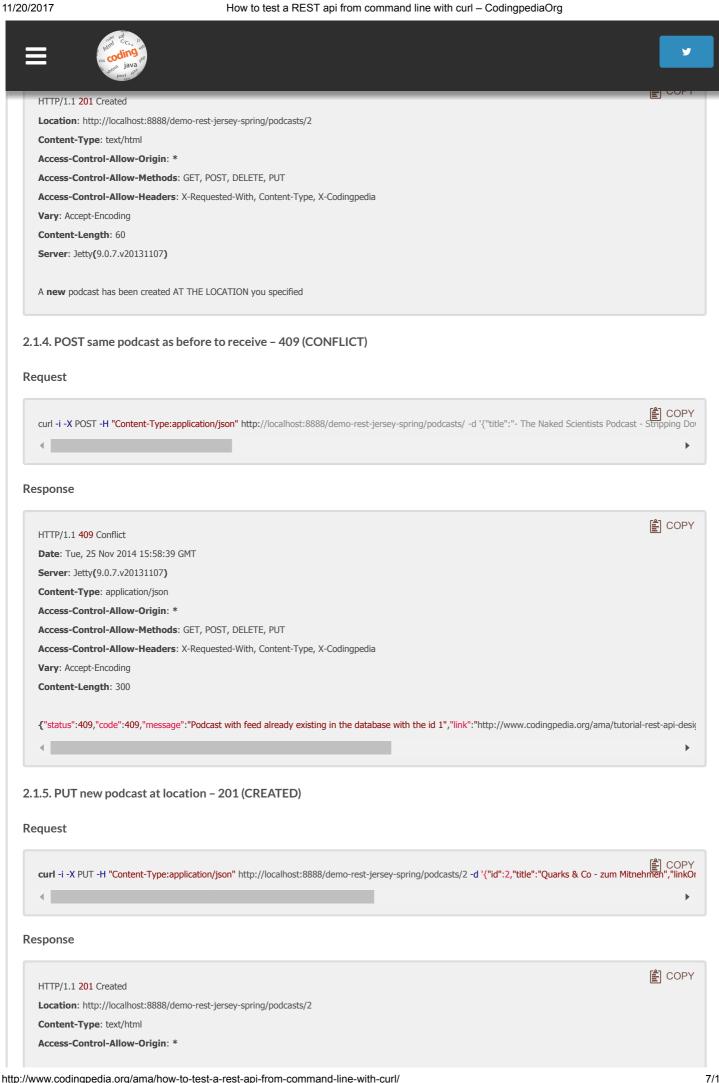
\* Connection #0 to host vldn680 left intact

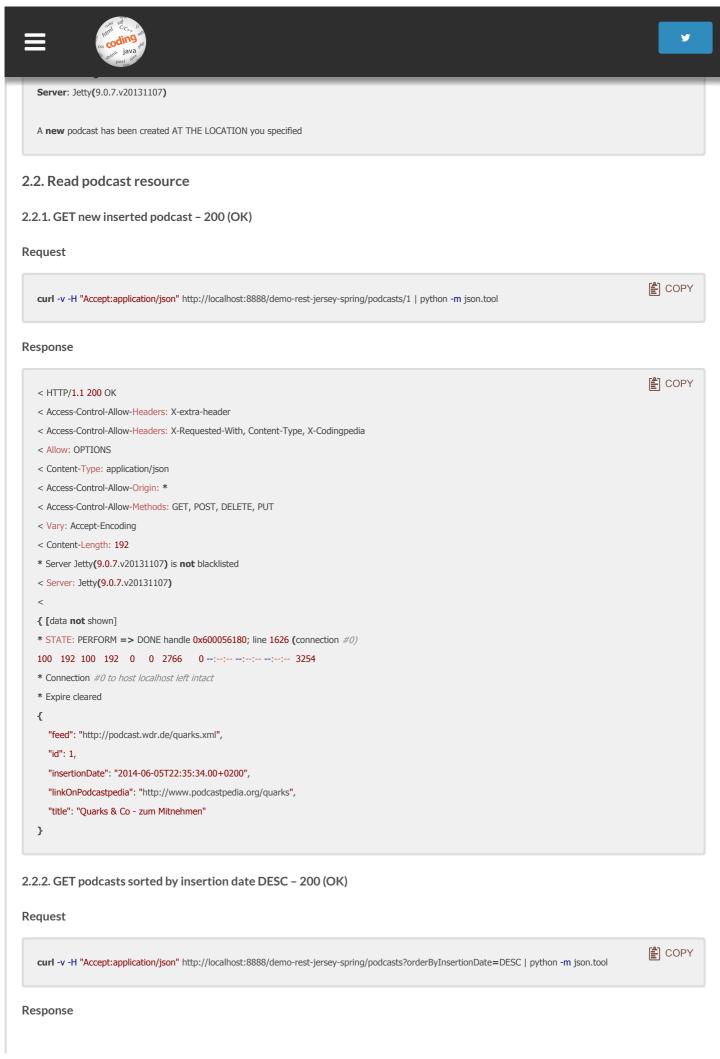
< Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia

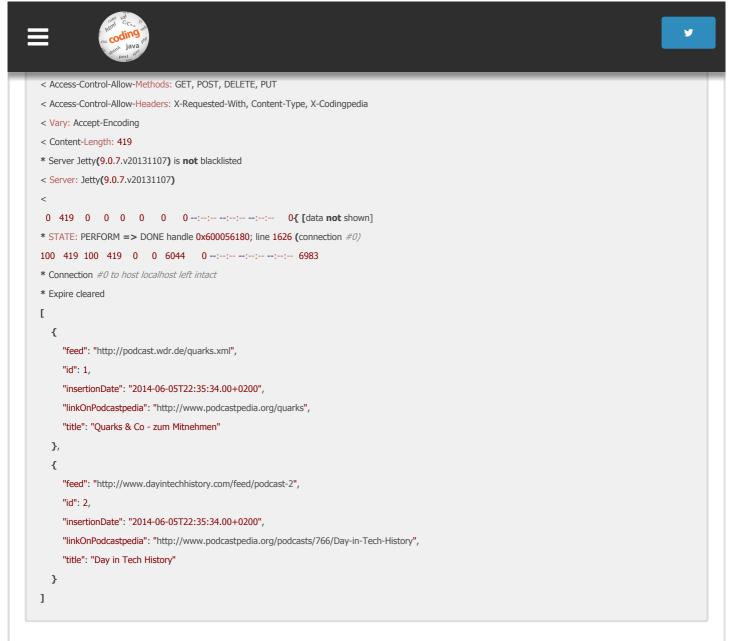
< HTTP/1.1 200 OK

As mentioned, in this second part I will map to curl requests the SOAPui test suite presented here.









## 2.3. Update podcast resource

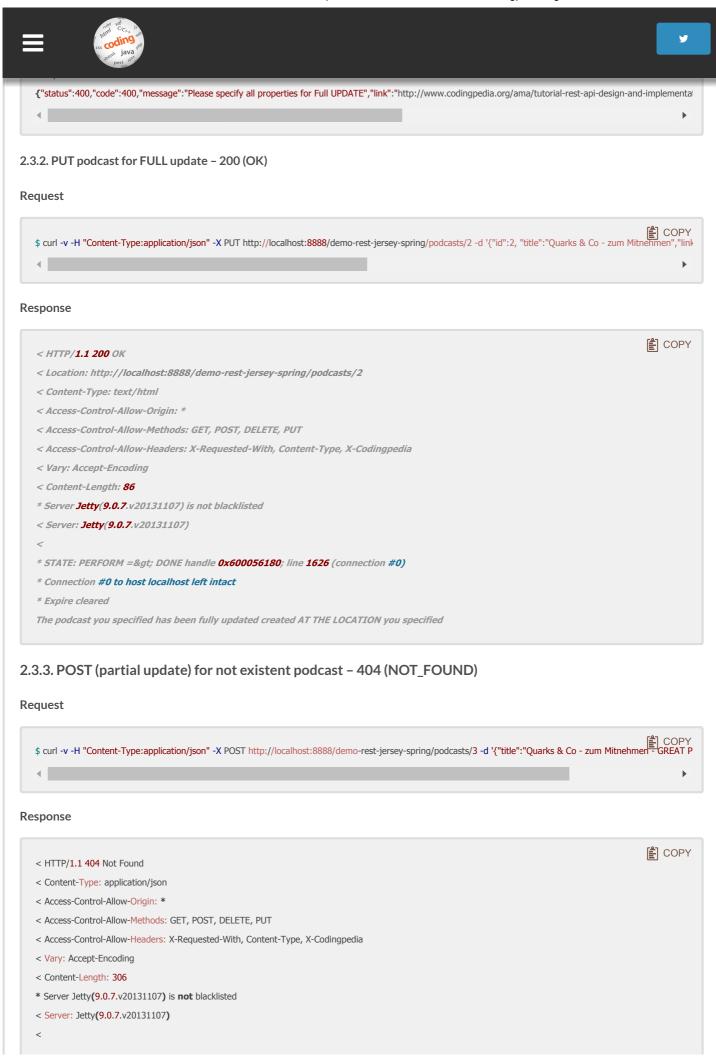
## 2.3.1. PUT not "complete" podcast for FULL update - 400 (BAD\_REQUEST)

# Request

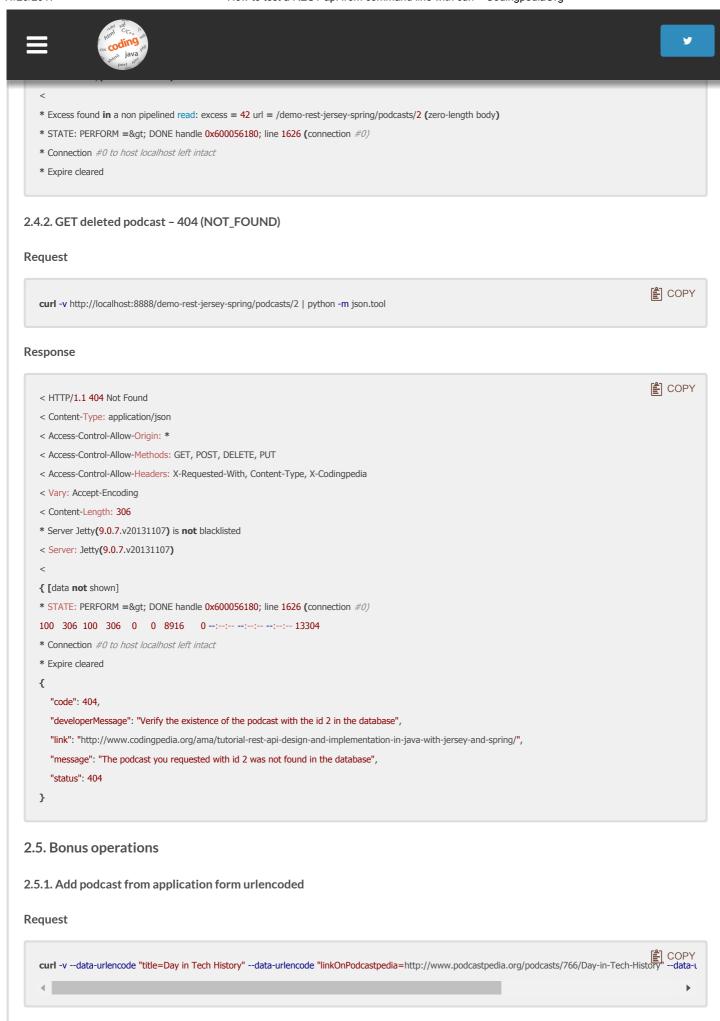
```
curl -v -H "Content-Type:application/json" -X PUT http://localhost:8888/demo-rest-jersey-spring/podcasts/2 -d '{"id":2, "title":"Quarks & Co - zum Mitnehmen","link(
```

## Response

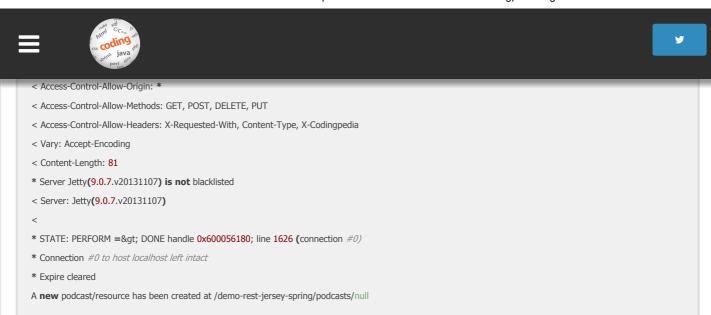
```
< HTTP/1.1 400 Bad Request</p>
< Content-Type: application/json</p>
< Access-Control-Allow-Origin: *</p>
< Access-Control-Allow-Methods: GET, POST, DELETE, PUT</p>
< Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia</p>
< Vary: Accept-Encoding</p>
< Content-Length: 290</p>
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)</p>
```



```
* Expire cleared
      "code": 404,
     "developerMessage": "Please verify existence of data in the database for the id - 3",
     "link": "http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementation-in-java-with-jersey-and-spring/",
     "message": "The resource you are trying to update does not exist in the database",
      "status": 404
   }
2.3.4. POST (partial update) podcast - 200 (OK)
Request
   $ curl -v -H "Content-Type:application/json" -X POST http://localhost:8888/demo-rest-jersey-spring/podcasts/2 -d '{"title":"Quarks & Co - zum Mitnehmen - GREAT P
Response
                                                                                                                                               COPY
   < HTTP/1.1 200 OK
   < Content-Type: text/html
   < Access-Control-Allow-Origin: *
   < Access-Control-Allow-Methods: GET, POST, DELETE, PUT
   < Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia
   < Vary: Accept-Encoding
   < Content-Length: 55
   * Server Jetty(9.0.7.v20131107) is not blacklisted
   < Server: Jetty(9.0.7.v20131107)
   * STATE: PERFORM => DONE handle 0x600056180; line 1626 (connection #0)
   * Connection #0 to host localhost left intact
   * Expire cleared
   The podcast you specified has been successfully updated
2.4. DELETE resource
2.4.1. DELETE second inserted podcast - 204 (NO_CONTENT)
Request
                                                                                                                                               COPY
   $ curl -v -X DELETE http://localhost:8888/demo-rest-jersey-spring/podcasts/2
Response
                                                                                                                                               COPY
   < HTTP/1.1 204 No Content
   < Content-Type: text/html
   < Access-Control-Allow-Origin: *
   < Access-Control-Allow-Methods: GET, POST, DELETE, PUT
```



Response



## Note:

I am still at the beginning of using curl, so please if you have any suggestions leave a comment. Thank you.

## Resources

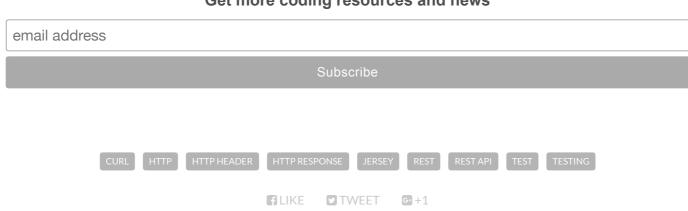
- Curl
- Cygwin
- How to Set Up a Python Development Environment on Windows
- Python.org

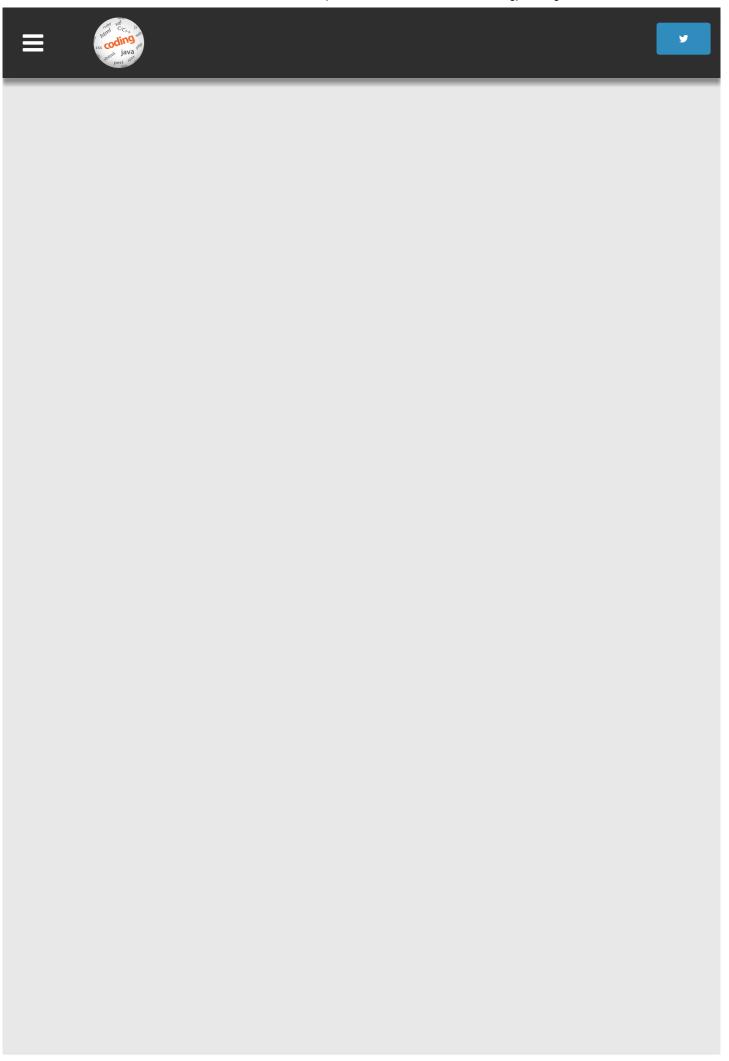


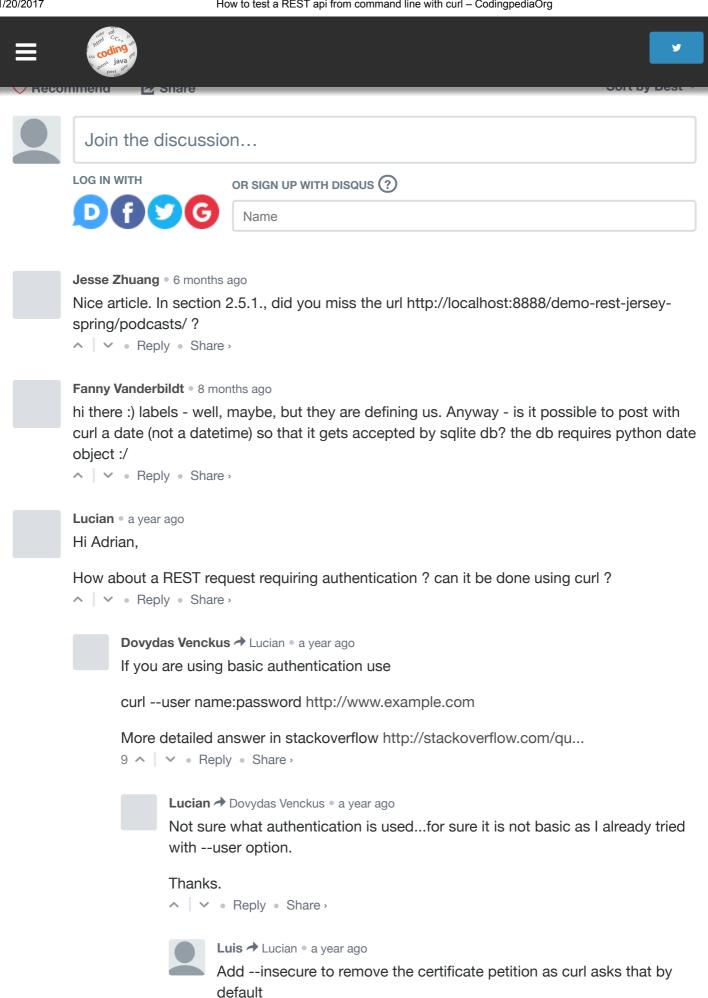
#### Adrian Matei

Creator of Podcastpedia.org and Codingpedia.org, computer science engineer, husband, father, curious and passionate about science, computers, software, education, economics, social equity, philosophy - but these are just outside labels and not that important, deep inside we are all just consciousness, right?

# Get more coding resources and news







A Renly Share