Fork me on GitHub

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

17

part I will "translate" the SOAPui 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

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

OR

COPY
```
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

COPY
```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
  0   631    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--     0
HTTP/1.1 200 OK
Date: Tue, 25 Nov 2014 12:54:56 GMT
Server: Jetty(9.0.7.v20131107)
Access-Control-Allow-Headers: X-extra-header
Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia
Allow: OPTIONS
Content-Type: application/xml
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE, PUT
Vary: Accept-Encoding
Content-Length: 631
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<podcast>
   <id>1</id>
   <title>- The Naked Scientists Podcast - Stripping Down Science</title>
   <linkOnPodcastpedia>http://www.podcastpedia.org/podcasts/792/-The-Naked-Scientists-Podcast-Stripping-Down-Science</linkOnPodcastpedia>
   <feed>feed_placeholder</feed>
   <description>The Naked Scientists flagship science show brings you a lighthearted look at the latest scientific breakthroughs, interviews with the world top scient
   <insertionDate>2014-10-29T10:46:02.00+0100</insertionDate>
</podcast>
```

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

Response

```
{
  "id": 1,
  "title": "- The Naked Scientists Podcast - Stripping Down Science",
  "linkOnPodcastpedia": "http://www.podcastpedia.org/podcasts/792/-The-Naked-Scientists-Podcast-Stripping-Down-Science",
  "feed": "feed_placeholder",
  "description": "The Naked Scientists flagship science show brings you a lighthearted look at the latest scientific breakthroughs, interviews with the world top scientis
  "insertionDate": "2014-10-29T10:46:02.00+0100"
}
```

If you want to have it displayed prettier, you can use the following command, provided you have Python installed on your machine.

Request

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

Response

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   758  100   758    0     0   6954      0 --:--:-- --:--:-- --:--:--  6954
[
    {
        "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"
    }
]
```

1.4. Curl request with multiple headers

As you've found out in my latest post, How to compress responses in Java REST API with GZip and Jersey, 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 `Accept-encoding:gzip`.

Curl options

- `-v, --verbose` – make the operation more talkative

To achieve that you need to simply **add another** -H option with the corresponding value. Of course in this case you would get some unreadable characters in the content, if you do not redirect the response to a file:

COPY

```
* Adding handle: conn: 0x28ddd80
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x28ddd80) send_pipe: 1, recv_pipe: 0
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0* About to connect() to proxy vldn680 port 19001 (#0)
*   Trying 10.32.142.80...
* Connected to vldn680 (10.32.142.80) port 19001 (#0)
> GET http://localhost:8888/demo-rest-jersey-spring/podcasts/ HTTP/1.1
> User-Agent: curl/7.30.0
> Host: localhost:8888
> Proxy-Connection: Keep-Alive
> Accept:application/json
> Accept-encoding:gzip
>
< HTTP/1.1 200 OK
< Date: Tue, 25 Nov 2014 16:17:02 GMT
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
< Content-Type: application/json
< 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-Encoding: gzip
< Content-Length: 413
< Via: 1.1 vldn680:8888
<
{ [data not shown]
100   413  100   413    0     0   2647      0 --:--:-- --:--:-- --:--:--  2647▒QKo▒0▒+▒▒g▒▒R▒+{▒V▒Pe▒▒□c▒▒    n▒▒▒▒fæHH▒"▒▒g▒/?2▒eM▒gl
▒{=`7▒Eϖ▒▒c▒ZM

n8▒i▒▒▒}H▒▒i1▒3g▒▒▒▒▒  ;▒E▒0O▒n▒R*▒g/E▒▒n=▒▒▒▒)▒U▒▒▒ld▒Φ▒h▒6▒▒▒_>w▒▒-▒▒:▒▒▒!▒Bb▒Z▒▒tO▒N@'= |▒▒C
▒u▒a▒9hO     #▒▒h▒i▒gq▒▒$▒▒|Ň     ▒▒▒08>#▒0b!▒▒'▒G▒^▒Iↄ.TU▒▒▒z▒\▒i^]e▒▒▒▒2▒▒▒▒▒?▒:/▒m▒▒▒▒▒Y▒h▒▒▒_齟'
* Connection #0 to host vldn680 left intact
```

2. SOAPui test suite translated to curl requests

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

Request

```
curl -i -X DELETE http://localhost:8888/demo-rest-jersey-spring/podcasts/
```

Response

```
HTTP/1.1 204 No Content
Date: Tue, 25 Nov 2014 14:10:17 GMT
Server: Jetty(9.0.7.v20131107)
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
Via: 1.1 vldn680:8888
Content-Length: 0
```

2.1.2. POST new podcast without feed – 400 (BAD_REQUEST)

Request

```
curl -i -X POST -H "Content-Type:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts/ -d '{"title":"- The Naked Scientists Podcast - Stripping Do
```

Response

```
HTTP/1.1 400 Bad Request
Date: Tue, 25 Nov 2014 15:12:11 GMT
Server: Jetty(9.0.7.v20131107)
Content-Type: application/json
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: 271
Via: 1.1 vldn680:8888
Connection: close

{"status":400,"code":400,"message":"Provided data not sufficient for insertion","link":"http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementation-
```

2.1.3. POST new podcast correctly – 201 (CREATED)

Request

```
curl -i -X POST -H "Content-Type:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts/ -d '{"title":"- The Naked Scientists Podcast - Stripping Do
```

```
HTTP/1.1 201 Created
Location: http://localhost:8888/demo-rest-jersey-spring/podcasts/2
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: 60
Server: Jetty(9.0.7.v20131107)

A new podcast has been created AT THE LOCATION you specified
```

2.1.4. POST same podcast as before to receive – 409 (CONFLICT)

Request

```
curl -i -X POST -H "Content-Type:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts/ -d '{"title":"- The Naked Scientists Podcast - Stripping Do
```

Response

```
HTTP/1.1 409 Conflict
Date: Tue, 25 Nov 2014 15:58:39 GMT
Server: Jetty(9.0.7.v20131107)
Content-Type: application/json
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: 300

{"status":409,"code":409,"message":"Podcast with feed already existing in the database with the id 1","link":"http://www.codingpedia.org/ama/tutorial-rest-api-desi
```

2.1.5. PUT new podcast at location – 201 (CREATED)

Request

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

Response

```
HTTP/1.1 201 Created
Location: http://localhost:8888/demo-rest-jersey-spring/podcasts/2
Content-Type: text/html
Access-Control-Allow-Origin: *
```

```
Server: Jetty(9.0.7.v20131107)

A new podcast has been created AT THE LOCATION you specified
```

2.2. Read podcast resource

2.2.1. GET new inserted podcast – 200 (OK)

Request

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

Response

```
< HTTP/1.1 200 OK
< Access-Control-Allow-Headers: X-extra-header
< Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia
< Allow: OPTIONS
< Content-Type: application/json
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET, POST, DELETE, PUT
< Vary: Accept-Encoding
< Content-Length: 192
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
{ [data not shown]
* STATE: PERFORM => DONE handle 0x600056180; line 1626 (connection #0)
100   192  100   192    0     0   2766      0 --:--:-- --:--:-- --:--:--  3254
* Connection #0 to host localhost left intact
* Expire cleared
{
    "feed": "http://podcast.wdr.de/quarks.xml",
    "id": 1,
    "insertionDate": "2014-06-05T22:35:34.00+0200",
    "linkOnPodcastpedia": "http://www.podcastpedia.org/quarks",
    "title": "Quarks & Co - zum Mitnehmen"
}
```

2.2.2. GET podcasts sorted by insertion date DESC – 200 (OK)

Request

```
curl -v -H "Accept:application/json" http://localhost:8888/demo-rest-jersey-spring/podcasts?orderByInsertionDate=DESC | python -m json.tool
```

Response

```
< Access-Control-Allow-Methods: GET, POST, DELETE, PUT
< Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-Codingpedia
< Vary: Accept-Encoding
< Content-Length: 419
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
  0  419    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{ [data not shown]
* STATE: PERFORM => DONE handle 0x600056180; line 1626 (connection #0)
100   419  100   419    0     0   6044      0 --:--:-- --:--:-- --:--:--  6983
* Connection #0 to host localhost left intact
* Expire cleared
[
  {
    "feed": "http://podcast.wdr.de/quarks.xml",
    "id": 1,
    "insertionDate": "2014-06-05T22:35:34.00+0200",
    "linkOnPodcastpedia": "http://www.podcastpedia.org/quarks",
    "title": "Quarks & Co - zum Mitnehmen"
  },
  {
    "feed": "http://www.dayintechhistory.com/feed/podcast-2",
    "id": 2,
    "insertionDate": "2014-06-05T22:35:34.00+0200",
    "linkOnPodcastpedia": "http://www.podcastpedia.org/podcasts/766/Day-in-Tech-History",
    "title": "Day in Tech History"
  }
]
```

2.3. Update podcast resource

2.3.1. PUT not "complete" podcast for FULL update – 400 (BAD_REQUEST)

Request

COPY

```
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

COPY

```
< HTTP/1.1 400 Bad Request
< Content-Type: application/json
< 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: 290
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
```

```
{"status":400,"code":400,"message":"Please specify all properties for Full UPDATE","link":"http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementa
```

2.3.2. PUT podcast for FULL update – 200 (OK)

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/1.1 200 OK
< Location: http://localhost:8888/demo-rest-jersey-spring/podcasts/2
< 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: 86
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
* STATE: PERFORM =&gt; DONE handle 0x600056180; line 1626 (connection #0)
* Connection #0 to host localhost left intact
* Expire cleared
The podcast you specified has been fully updated created AT THE LOCATION you specified
```

2.3.3. POST (partial update) for not existent podcast – 404 (NOT_FOUND)

Request

```
$ curl -v -H "Content-Type:application/json" -X POST http://localhost:8888/demo-rest-jersey-spring/podcasts/3 -d '{"title":"Quarks & Co - zum Mitnehmen - GREAT P
```

Response

```
< HTTP/1.1 404 Not Found
< Content-Type: application/json
< 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: 306
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
```

```
* 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

COPY

```
$ 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 =&gt; 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
```

```
<
* Excess found in a non pipelined read: excess = 42 url = /demo-rest-jersey-spring/podcasts/2 (zero-length body)
* STATE: PERFORM =&gt; DONE handle 0x600056180; line 1626 (connection #0)
* Connection #0 to host localhost left intact
* Expire cleared
```

2.4.2. GET deleted podcast – 404 (NOT_FOUND)

Request

```
curl -v http://localhost:8888/demo-rest-jersey-spring/podcasts/2 | python -m json.tool
```

Response

```
< HTTP/1.1 404 Not Found
< Content-Type: application/json
< 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: 306
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
{ [data not shown]
* STATE: PERFORM =&gt; DONE handle 0x600056180; line 1626 (connection #0)
100   306  100   306    0     0  8916      0 --:--:-- --:--:-- --:--:-- 13304
* Connection #0 to host localhost left intact
* Expire cleared
{
    "code": 404,
    "developerMessage": "Verify the existence of the podcast with the id 2 in the database",
    "link": "http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementation-in-java-with-jersey-and-spring/",
    "message": "The podcast you requested with id 2 was not found in the database",
    "status": 404
}
```

2.5. Bonus operations

2.5.1. Add podcast from application form urlencoded

Request

```
curl -v --data-urlencode "title=Day in Tech History" --data-urlencode "linkOnPodcastpedia=http://www.podcastpedia.org/podcasts/766/Day-in-Tech-History" --data-u
```

Response

```
< 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: 81
* Server Jetty(9.0.7.v20131107) is not blacklisted
< Server: Jetty(9.0.7.v20131107)
<
* STATE: PERFORM =&gt; DONE handle 0x600056180; line 1626 (connection #0)
* Connection #0 to host localhost left intact
* Expire cleared
A new podcast/resource has been created at /demo-rest-jersey-spring/podcasts/null
```

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

email address

Subscribe

CURL HTTP HTTP HEADER HTTP RESPONSE JERSEY REST REST API TEST TESTING

LIKE TWEET +1

Recommend Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Jesse Zhuang • 6 months ago

Nice article. In section 2.5.1., did you miss the url http://localhost:8888/demo-rest-jersey-spring/podcasts/ ?

Reply • Share ›

Fanny Vanderbildt • 8 months ago

hi there :) labels - well, maybe, but they are defining us. Anyway - is it possible to post with curl a date (not a datetime) so that it gets accepted by sqlite db? the db requires python date object :/

Reply • Share ›

Lucian • a year ago

Hi Adrian,

How about a REST request requiring authentication ? can it be done using curl ?

Reply • Share ›

Dovydas Venckus → Lucian • a year ago

If you are using basic authentication use

curl --user name:password http://www.example.com

More detailed answer in stackoverflow http://stackoverflow.com/qu...

9 • Reply • Share ›

Lucian → Dovydas Venckus • a year ago

Not sure what authentication is used...for sure it is not basic as I already tried with --user option.

Thanks.

Reply • Share ›

Luis → Lucian • a year ago

Add --insecure to remove the certificate petition as curl asks that by default

Reply • Share ›