REST APIs

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

2016-12-09

Disclaimer: Big Data software is constantly updated, code samples may be outdated.

Outline

REST [11]

- Representational state transfer (REST): System API utilizing HTTP / TCP
- **RESTful**: Term indicates the system is conforming to REST constraints
- Advantages (due to HTTP)
  - Simplicity of the interfaces
  - Portability
  - Cachable
  - Tracable: Communication can be inspected

Semantics of HTTP request verbs [13]

- GET: retrieve a representation of a resource (no updates)
- PUT: store the enclosed data under the given URI
- POST: transfer an entity/data as subordinate of the web resource
- DELETE: remove the given URI
- PUT and DELETE are idempotent and GET (w/o concurrent updates)

HTTP 1.1 [13]

- The Hypertext Transfer Protocol (HTTP) is a stateless protocol
- Request via TCP $\Rightarrow$ Response (status and content)
- Request/Response are encoded in ASCII
- Include a header with standardized key/value pairs [14]
- Non-standard key/value pairs can be added
  - Usually prefixed with X for eXtension
- One data section (at the end) according to the media type
- Separation between header and data via one newline

Example HTTP Request

```
1 GET /dir/file HTTP/1.1
2 Host: www.test.de:50070
3 User-Agent: mozilla
4 Cache-Control: no-cache
5 Accept: */*
```

HTTP 1.1

Media types [15]

- Based on Multipurpose Internet Mail Extensions (MIME) types
- Media type is composed of type, subtype and optional parameters
  - e.g., image/png
  - e.g., text/html; charset=UTF-8
- Media types should be registered by the IANA[1]

Example HTTP Response

```
1   HTTP/1.1 200 OK
2   Date: Sun, 06 Dec 2015 16:41:16 GMT
3   Expires: -1
4   Cache-Control: private, max-age=0
5   Content-Type: text/html; charset=ISO-8859-1
6   Server: gws
7   X-XSS-Protection: 1; mode=block
8   X-Frame-Options: SAMEORIGIN
9   Set-Cookie: PREF=ID=11111:FF=0:TM=1449420076:LM=1444476:V=1:S=doDl; expires=Thu, 31-Dec-2015 16:02:17 GMT; path=/;
        ↪ domain=.test.de
10  Set-Cookie: NID=74=UNTSNZy expires=Mon, 06-Jun-2016 16:41:16 GMT; path=/dir; domain=.test.de; HttpOnly
11  Accept-Ranges: none
12  Vary: Accept-Encoding
13  Transfer-Encoding: chunked
14
15  DATA formatted according to content type
```

[1] Internet Assigned Numbers Authority

REST Semantics

- Depends on the service implementation
- Behavior usually depends on URI type
  - Collections/Directories, e.g., http://test.de/col/
  - Items/Files, e.g., http://test.de/col/file

Typical semantics [11]

Resource	GET	PUT	POST	DELETE
Collection	List the collection	Replace the collection with new data.	Create a new entry in the collection, return the URI of the created entry	Delete the collection
Item	Retrieve the data	Replace the element or create it	Not widely used.	Delete the element in the collection

Direct Access via TCP

- Connect to the service IP address and port
- Use any API or tool
  - UNIX sockets for C, Python, ...
  - Netcat (`nc`)
  - `curl`
  - Python
  - Browser

CURL

- `curl` transfers data from/to a server
- Useful for scripting / testing of webservers
- Supports many protocols, standards for proxy, authentication, cookies, ...

```
# -i: include the HTTP header in the output for better debugging
# -L: if the target location has moved, redo the request on the new location
curl -i -L "http://xy/bla"
# Send data in myFile using HTTP PUT, use "-" to read from STDIN
curl -i --request PUT "http://xy/bla?param=x&y=z" -d "@myFile"
# To put a binary file use --data-binary
curl -i --request POST --data-binary "@myFile" "http://xy/bla?param=x&y=z"
# Delete a URI
curl -i -request DELETE "http://xy/bla?param=x&y=z"
```

Python

- The requests package supports HTTP requests quite well

Transferring JSON data

```
1  import json, requests
2
3  params = {'parameters' : [ 'testWorld' ] }
4
5  s = requests.Session() # we use a session in this example
6  resp = s.post(url='http://localhost:5000/compile',
7     data=json.dumps(params),
8     headers={'content-type': 'application/json'},
9     auth=('testuser','my secret'))
10 print(resp.status_code)
11 print(resp.headers)
12
13 # assume the response is in JSON
14 data = json.loads(resp.text, encoding="utf-8")
15
16 # retrieve another URL using HTTP GET
17 resp = s.get(url='http://localhost:5000/status', auth=('testuser','my secret'))
```

Example: WebHDFS [12]

- Full access to file system via `http://$host/webhdfs/v1/`*`FILENAME`*`?op=`*`OPERATION`*

```
$ host=10.0.0.61:50070
$ curl -i -L "http://$host/webhdfs/v1/foo/bar?op=OPEN"
HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Sun, 06 Dec 2015 16:06:11 GMT
Date: Sun, 06 Dec 2015 16:06:11 GMT
Pragma: no-cache
Content-Type: application/octet-stream
Location: http://abu1.cluster:50075/webhdfs/v1/foo/bar/file?op=OPEN&namenoderpcaddress=abu1.cluster:8020&offset=0
Content-Length: 0
Server: Jetty(6.1.26.hwx)

HTTP/1.1 200 OK
Access-Control-Allow-Methods: GET
Access-Control-Allow-Origin: *
Content-Type: application/octet-stream
Connection: close
Content-Length: 925
DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA DATA

$ curl -i "http://$host/webhdfs/v1/?op=GETFILESTATUS"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 06 Dec 2015 16:11:14 GMT
Date: Sun, 06 Dec 2015 16:11:14 GMT
Pragma: no-cache
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)
{"FileStatus":{"accessTime":0,"blockSize":0,"childrenNum":7,"fileId":16385,"group":"hdfs","length":0,"modificationTime":
1444759104314,"owner":"hdfs","pathSuffix":"","permission":"755","replication":0,"storagePolicy":0,"type":"DIRECTORY"}}
```

Bibliography

11 https://en.wikipedia.org/wiki/Representational_state_transfer

12 http://hortonworks.com/blog/webhdfs-%E2%80%93-http-rest-access-to-hdfs/

13 https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

14 https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

15 https://en.wikipedia.org/wiki/Media_type