Making a request to a RESTful API using python

Ask Question

64

32

I have a RESTful API that I have exposed using an implementation of Elasticsearch on an EC2 instance to index a corpus of content. I can can query the search by running the following from my terminal (MacOSX):

```
curl -XGET 'http://ES_search_demo.com/document/record/_search?pretty=true' -d '{
  "query": {
    "bool": {
      "must": [
        {
          "text": {
            "record.document": "SOME_JOURNAL"
          }
        },
        {
          "text": {
            "record.articleTitle": "farmers"
          }
        }
      ],
      "must_not": [],
      "should": []
    }
  },
  "from": 0,
  "size": 50,
  "sort": [],
  "facets": {}
}'
```

How do I turn above into a API request using `python/requests` or `python/urllib2` (not sure which one to go for - have been using urllib2, but hear requests is better...)? Do I pass as a header or otherwise?

python api rest elasticsearch

share improve this question

edited Dec 10 '16 at 6:33
Saeed Zhiany
1,382 3 6 25

asked Jun 25 '13 at 15:51
user7289
4,228 13 43 68

add a comment

4 Answers

active oldest votes

Using requests:

110

```
import requests
url = 'http://ES_search_demo.com/document/record/_search?pretty=true'
data = '{
  "query": {
    "bool": {
      "must": [
        {
          "text": {
            "record.document": "SOME_JOURNAL"
          }
        },
        {
          "text": {
            "record.articleTitle": "farmers"
          }
        }
      ],
      "must_not": [],
      "should": []
    }
  },
  "from": 0,
  "size": 50,
  "sort": [],
  "facets": {}
}'
response = requests.post(url, data=data)
```

Depending on what kind of response your API returns, you will then probably want to look at `response.text` or `response.json()` (or possibly inspect `response.status_code` first). See the quickstart docs here, especially this section.

share improve this answer

edited Dec 15 '16 at 19:43
Eduardo Cuomo
6,021 1 44 44

answered Jun 25 '13 at 19:55
andersschuller
5,184 1 16 26

3 i think, it should be: response = requests.post(url, data=data) – KhoaNC Jul 2 '15 at 10:19

3 "requests.get" does not take "data" parameter. It could take optional "params" parameter which is usually a dict carrying query string. If a payload is necessary to fetch data (such as the example posted in question), then "requests.post" needs to be used. Additionally using "json" library makes it easier to parse json response. – HVS Jan 26 '16 at 11:00

2 Does it work with python 3 ? – Parveen Shukhala Aug 25 '16 at 11:29

2 @ParveenShukhala "Requests officially supports Python 2.6–2.7 & 3.3–3.5, and runs great on PyPy." -- pypi.python.org/pypi/requests – danio Dec 15 '16 at 9:10

add a comment

29

Using requests and json makes it simple.

1. Call the API
2. Assuming the API returns a JSON, parse the JSON object into a Python dict using `json.loads` function
3. Loop through the dict to extract information.

Requests module provides you useful function to loop for success and failure.

`if(Response.ok)` : will help help you determine if your API call is successful (Response code - 200)

`Response.raise_for_status()` will help you fetch the http code that is returned from the API.

Below is a sample code for making such API calls. Also can be found in github. The code assumes that the API makes use of digest authentication. You can either skip this or use other appropriate authentication modules to authenticate the client invoking the API.

```
#Python 2.7.6
#RestfulClient.py
```

```
import requests
from requests.auth import HTTPDigestAuth
import json

# Replace with the correct URL
url = "http://api_url"

# It is a good practice not to hardcode the credentials. So ask the user to enter credential
myResponse = requests.get(url,auth=HTTPDigestAuth(raw_input("username: "), raw_input("Passwc
#print (myResponse.status_code)

# For successful API call, response code will be 200 (OK)
if(myResponse.ok):

    # Loading the response data into a dict variable
    # json.loads takes in only binary or string variables so using content to fetch binary c
    # Loads (Load String) takes a Json file and converts into python data structure (dict or
    jData = json.loads(myResponse.content)

    print("The response contains {0} properties".format(len(jData)))
    print("\n")
    for key in jData:
        print key + " : " + jData[key]
else:
  # If response code is not ok (200), print the resulting http error code with description
    myResponse.raise_for_status()
```

share improve this answer

answered Sep 22 '15 at 16:23
HVS
571 2 6 15

1 Last portion with iteration over keys will not always work because JSON document may have array as a top level element. So, it would be an error to try to get `jData[key]` – Denis The Menace Jun 28 '16 at 11:25

@DenisTheMenace if it is an array, how would I loop around it? – qasimalbaqali Mar 12 at 19:51

@qasimalbaqali the same way you loop over dictionary. But array elements will be simply `jData` , not `jData[key]` – Denis The Menace Mar 13 at 15:41

add a comment

So you want to pass data in body of a GET request, better would be to do it in POST call. You can

3 achieve this by using both Requests.

Raw Request

```
GET http://ES_search_demo.com/document/record/_search?pretty=true HTTP/1.1
Host: ES_search_demo.com
Content-Length: 183
User-Agent: python-requests/2.9.0
Connection: keep-alive
Accept: */*
Accept-Encoding: gzip, deflate

{
  "query": {
    "bool": {
      "must": [
        {
          "text": {
            "record.document": "SOME_JOURNAL"
          }
        },
        {
          "text": {
            "record.articleTitle": "farmers"
          }
        }
      ],
      "must_not": [],
      "should": []
    }
  },
  "from": 0,
  "size": 50,
  "sort": [],
  "facets": {}
}
```

Sample call with Requests

```
import requests

def consumeGETRequestSync():
data = '{
  "query": {
    "bool": {
      "must": [
        {
          "text": {
            "record.document": "SOME_JOURNAL"
          }
        },
        {
          "text": {
            "record.articleTitle": "farmers"
          }
        }
      ],
      "must_not": [],
      "should": []
    }
  },
  "from": 0,
  "size": 50,
  "sort": [],
  "facets": {}
}'
url = 'http://ES_search_demo.com/document/record/_search?pretty=true'
headers = {"Accept": "application/json"}
# call get service with headers and params
response = requests.get(url,data = data)
print "code:"+ str(response.status_code)
print "******************"
print "headers:"+ str(response.headers)
print "******************"
print "content:"+ str(response.text)

consumeGETRequestSync()
```

You can check out more calls using requests in [http://stackandqueue.com/?p=75]

share improve this answer