HTTP / RESTful API Calls with Python Requests Library

POSTED ON MARCH 30, 2015 UPDATED ON FEBRUARY 25, 2016

3 Votes

The objective of this post is to give a brief introduction to HTTP and RESTful APIs. Also develop an RESTful client in Python using the "***requests***" library and "***json***" library. I intentionally did not use the urllib2 or any other standard Python library, since I want to explain the power of the "***requests***" library, which is a simple and straight forward library for developing RESTful Clients.

HTTP & RESTful APIs

HTTP is a request / response protocol and is similar to client-server model. In the internet world, generally the web browser sends the HTTP request and the web server responds with HTTP response. Also it is not necessary that the client is always a browser. The client can be any application which can send a HTTP request.

We have used so many application level communication protocols. Starting from RPC (Remote Procedure Call), Java RMI (Remote Method Invocation), XML/RPC, SOAP/HTTP. In this lineage RESTful API is the current application level client-server protocol.

RESTful API is an application level protocol. It is heavily used in internet (WWW) and distributed systems. It is recommended by Services Oriented Architecture (SOA) to communicate between loosely coupled distributed components. The RESTful API is a form of HTTP protocol is the de facto standard for Cloud communications.

The two properties of RESTful which makes suitable for modern internet and cloud communication is **stateless** and **cache-less**. The protocol does not enforce any state-machine, it means there is no order of protocol messages enforced. Also the protocol will not remember any information across requests or responses. Each and every request is unique and it has no relation with previous or next request which may come. To understand more on HTTP protocol look at the references below. Hence forth we will move along with Python Requests library to learn and develop RESTful API.

Request Library

The Requests python library is simple and straight forward library for developing RESTful Clients. Python has a built in library called urllib2, it is bit complex and old style when compared to Requests. After writing couple of programs using the urllib2, I am completely convinced by the below statement issued by the developers of ***Requests***. Also refer the Reference[4] for comparing the code segments written using urllib2 and requests library.

> *Python's standard **urllib2** module provides most of the HTTP capabilities you need, but the API is thoroughly **broken**. It was built for a different time — and a different web. It requires an enormous amount of work (even method overrides) to perform the simplest of tasks.*

Please refer the URL http://docs.python-requests.org/en/latest/user/install/#install to install the requests library before proceeding.

The Structure of HTTP / RESTful API

Following are points to remember while developing RESTful API:

1. URL (Universal Resource Locator)

2. Message Type
3. Headers
4. Parameters
5. Payload
6. Authentication

1. URL

The URL is the core of RESTful API. Generally the URL refers a web page, but it can also refer a service or a resource.

For example : http://graph.facebook.com/v2.3/{photo-id}

The above URL is a resource which holds the photo with id photo-id. As per the above syntax the value for the photo-id must be replaced with {photo-id}.

Python code snippet to store a URL in a Python object:

```
>>> url = 'http://graph.facebook.com/v2.3/123435'
```

2. Message Types

HTTP supports GET/POST/PUT/DELETE message types. There are few more types as well. Please take a look at the reference[1] to understand them in detail.

GET – to retrieve resource. Eg. GET http://graph.facebook.com/v2.3/1234345 will retrieve the photograph stored in that location.

```
>>> import requests
>>> ret = requests.get(url)
>>> ret.staus_code
200
```

POST – to update a resource . POST http://graph.facebook.com/v2.3/123435 will update the existing photo with the new photograph supplied in the message payload. POST will also create resource, if the resource is not available.

```
>>> import requests
>>> ret = requests.post(url)
>>> ret.status_code
200
```

PUT – to create a resource. PUT http://graph.facebook.com/v2.3/123435 will create a resource by uploading the photograph sent on the message payload.

```
>>> import requests
>>> ret = requests.put(url)
>>> ret.status_code
201
```

DELETE – to delete a resource – DELETE http://graph.facebook.com/v2.3/123435 will delete the photograph present in that location.

```
>>> import requests
>>> ret = requests.delete(url)
>>> ret.status_code
200
```

3. Headers

The HTTP header generally contains information used to process the request and responses. The headers are colon separated key value pairs. For example "Accept: text/plain". The http request & response may be have multiple headers. Since it is a key value pair, we can use Python's dictionary data type to store these values.

Single Header & Multiple headers:

```
>>> head = {"Content-type": "application/json"}
```

```
>>> head= {"Accept":"applicaiton/json",
        "Content-type": "application/json"}
```

Make the API call with the above header:

```
>>> ret = requests.get(url,headers=head)
>>> ret.status_code
200
```

In the above statement, "**headers**" is the name of argument. So we have used the Python feature of passing **named arguments** to a function.

4 Parameters

Sometimes we may want to pass values in the URL parameters. For example, the URL http://www.abc.com/abc.php?name=Saravanan&designation=Technical Leader . This URL expects the user to send the value for the keyword “name” and “designation”. The below code snippet helps to you accomplish this tasks. The “params” argument is used to set the value for parameters.

```
>>> parameters = {'name':'Saravanan',
          'designation':'Technical Leader'}
>>> head = {'Content-Type':'application/json'}
>>> ret = requests.post(url,params=parameters,header=head)
>>> ret.status_code
200
```

5 Payload

The payload contains the data to be sent on the requests. In this we will see how to send a JSON object in the payload.

```
empObj = {'name':'Saravanan', 'title':'Architect','Org':'Cisco Systems'}
```

As in the previous examples, we cannot send the JSON object which is a dictionary data type in Python. In the above snippet we created a empObj which is a dictionary data type of Python. This must be converted into JSON object before send the request.

The json library in Python helps here .

```
>>> import json
>>> emp = json.dumps(empObj)
```

The json.dumps converts the dictionary object into a JSON object.

The complete code snippet is below:

```
>>> import json
>>> import requests
>>>
>>> url='http://graph.facebook.com/v2.3/123123
>>> head = {'Content-type':'application/json',
             'Accept':'application/json'}
>>> payload = {'name':'Saravanan',
               'Designation':'Architect',
               'Orgnization':'Cisco Systems'}
```

```
>>> payld = json.dumps(payload)
>>> ret = requests.post(url,header=head,data=payld)
>>> ret.status_code
200
```

6 Authorization

The “requests” library supports various forms of authentication, which includes Basic, Digest Authentication, OAuth and others. The value for authentication can be passed using “auth” parameter of the requests method.

```
>>>
>>> from requests.auth import HTTPBasicAuth
>>> url = 'http://www.hostmachine.com/sem/getInstances'
>>> requests.get(url, auth=HTTPBasicAuth('username','password')
200
```

The “**auth**” argument can take any function, so if you want to define your own custom authentication and pass it to “**auth**“.

Summary

The above code snippet is a sample to explain the simplicity of Python and requests library. You can take a look at the official website of Requests and learn advanced concepts in RESTful API developments.

References

[1] HTTP Wiki : http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[2] History of HTTP by W3 Org : http://www.w3.org/Protocols/History.html

[3] Requests – http://docs.python-requests.org/en/latest/

[4] Requests and Urllib2 Comparison : https://gist.github.com/kennethreitz/973705

[5] Installation of Requests library : http://docs.python-requests.org/en/latest/user/install/#install

[6] HTTP Headers – http://en.wikipedia.org/wiki/List_of_HTTP_header_fields