Automate Rest call using HTTP client

\*\*No selenium dependency (no front end testing)

Here,our test site is <https://reqres.in/>

Steps:

1.Create a maven project .

2.add dependencies in POM.xml

i) HTTP Client library

ii) HTTP core library

iii)JSON API

iv)TestNG library /dependency

3.Create a POM project

🡪Base package with base class (which is a parent class for all other classes)

🡪Config package with properties file in it .

Initialize all the generic env variables in properties file.

URL=”<https://reqres.in/>” (nothing but endpoint url)

Serviceurl=”/api/users” (nothing but service API url)

4.Create a constructor in base class to initaialize properties file.

**public** testbase() {

String path = "C:\\jansi\_javafiles\_2\\POM\_structure\\src\\main\\java\\com\\freecrm\\qa\\configlayer\\config.properties";

**try** {

FileInputStream ip = **new** FileInputStream(path);

Properties prop = **new** Properties();

prop.load(ip);

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

* FileInputStream-🡪 to create a stream btw file and java code
* Properties=new properties()🡪 to read the properies file
* Prop.load 🡪to load the input file.
* Try..catch block🡪to get the exceptions in constructor.

5.Create a package for client and create a class ‘Rest\_client’ in it to store all the client methods.

i) Get()

🡪for hitting Getcall in API

Function:When we pass the service API url to get method,we get JSON response.

So,we need one get method with url parameter

// Get method to hit GET API

**public** **void** get(String url) **throws** ClientProtocolException, IOException {

//httpclients class is available within httpclient library in POM.xml and createdefault () creates default client connection

//it will return closablehttpclient object

//httpclient - ref variable for CloseableHttpClient

CloseableHttpClient httpclient =HttpClients.*createDefault*();

//httpget request hits the url

HttpGet httpget=**new** HttpGet(url);

//execute method exceutes a request which is a httpget request-->similar to send button in get call

//execute method returns CloseableHttpResponse object

CloseableHttpResponse httpresponse=httpclient.execute(httpget);

//all the response will be stored in httpresponse

**int** statuscode=httpresponse.getStatusLine().getStatusCode();

System.***out***.println("Statuscode:"+statuscode);

//to get the complete JSON response object,we need to use EntityUtils class

//pass 2 parameters to tostring()-->entity ,characterset-->entity is available within httpresponse object

//UTF-8-->character format

//JSON response wil be converted to a string here

String responsestring=EntityUtils.*toString*(httpresponse.getEntity(),"UTF-8");

//since,we do not need String and we need a json object ,we need to use JSONObject class from JSON library/dependency

//here,it will convert the string responsestring to a JSON object,and store in responsejson ref variable

JSONObject responsejson=**new** JSONObject(responsestring);

System.***out***.println("Response json from API"+responsejson);

//getallheaders() returns all headers in array format

Header[] header=httpresponse.getAllHeaders();

//convert header array to hashmap because it is easy to iterate hashmap and stores data in the form of key,vaue pair.

HashMap<String,String> mapheaders=**new** HashMap<String, String>();

//iterate header array and store the key,values in hashmap

**for**(Header data:header)

{

mapheaders.put(data.getName(), data.getValue());

}

System.***out***.println("All headers"+mapheaders);

}

6.Add a package in src/test/java and create a test class

i) test class shd extend base class.

ii)Create a constructor of the test class and invoke the constructor of base class suing super keyword

**public** Rest\_clienttest() **throws** IOException {

**super**();

}

iii) globally declared variables

Rest\_client rc;

String endurl;

String apiurl;

String URL;

@BeforeMethod

**public** **void** setup() **throws** ClientProtocolException, IOException {

endurl = prop.getProperty("url");

apiurl = prop.getProperty("serviceurl");

// append both url

URL = endurl + apiurl;

}

@Test

**public** **void** gettest() **throws** ClientProtocolException, IOException {

// call the get () in Rest\_client class by creating object

rc = **new** Rest\_client();

rc.get(URL);

}

What httpresponse do we get when we hit get call()?

1.Status code

2.JSON string

3.Headers.

Since,we get the JSON object in a single line,copy the entire string and paste in

JSON validator and get the json object

Now,we can compare the o/p with postman output

🡪sequence may be different from postman o/p

Note:

We have to get the CloseableHttpResponse in the get() method and we have to define inside the test case only .

How to validate status code response?

It can be done by using Assertion class.

Assertion :

System.***out***.println("Statuscode-->"+statuscode);

//import Assert to testng

//Assertequals(boolean actual,boolean expected"

// Response\_status\_code value is taken from base class thru inheritance concept

Assert.*assertEquals*(statuscode, Response\_status\_code,"Status code is not valid");

How to validate JSON object response?

Steps:

1)

Create a Util package under src/main/java🡪create a Testutil class to parse the JSON object.

2)JSON object has both simple values and Array values.

e.g.

**Simple values:**

"per\_page": 3,

"total": 12,

**Array values:**

**🡪If Jason object starts with [ square bracket,then it is a array object**

//data [] array

"data": [{

"last\_name": "Bluth",

"id": 1,

"avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/calebogden/128.jpg",

"first\_name": "George",

"email": "george.bluth@reqres.in"

}, {

"last\_name": "Weaver",

"id": 2,

"avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg",

"first\_name": "Janet",

"email": "janet.weaver@reqres.in"

}, {

"last\_name": "Wong",

"id": 3,

"avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/olegpogodaev/128.jpg",

"first\_name": "Emma",

"email": "emma.wong@reqres.in"

}],

3)How to parse JSON objects?

Create a utiity function in util class to parse both simple and array values.

**public** **static** String getvaluefromjasonobject(JSONObject responsejson, String jpath) { Object obj = responsejson;

**for** (String s : jpath.split("/"))

**if** (!s.isEmpty())

**if** (!(s.contains("[") || s.contains("]")))

obj = ((JSONObject) obj).get(s);

**else** **if** (s.contains("[") || s.contains("]"))

obj = ((JSONArray) ((JSONObject) obj).get(s.split("\\[")[0]))

.get(Integer.*parseInt*(s.split("\\[")[1].replace("]", "")));

**return** obj.toString();}

4)call the function and pass responsejson and attribute name as jpath

For simple values:

String perpagevalue=Testutil.*getvaluefromjasonobject*(responsejson, "/per\_page");

System.***out***.println("Value of Per\_page-->"+perpagevalue);

//here,entire JSON object is stored in responsejson ref var.

//and the attribute “/per\_page” that we need to fetch is stored in jpath

//use ‘/’ single slash to take the attribute from the path

For array values:

String avatar\_value=Testutil.*getvaluefromjasonobject*(responsejson,"/data[0]/avatar");

System.***out***.println(avatar\_value);

Here,/avatar attribute is inside data[0] array

Data[0]🡪since the attribute is present in 0 th index of the data[].

How to automate Get API with headers?

Sometimes,it is necessary to pass headers along with send request in GET call.One of the mandatory header request is:

Key: "Content-Type"

Value: "application/json"

Other Headers may be username,password,authentication\_token etc and headers will be in key..value pair.

Steps:

1)Create a hashmap and add all the headers in it.

HashMap<String,String> headermap=**new** HashMap<String,String>();

headermap.put("Content-Type", "application/json");

2)create a @test case with headers and call the GETwithheader() call in it.

**public** **void** gettestwithheaders() **throws** ClientProtocolException, IOException

{

rc = **new** Rest\_client();

HashMap<String,String> headermap=**new** HashMap<String,String>();

headermap.put("Content-Type", "application/json");

System.***out***.println("datas of Get call with headers:");

httpresponse=rc. getwithheaders (URL,headermap);

2)Method overriding-create one more Get ()method with headers and pass URL and the hashmap

//here,iterate mapheaders entryset and store in map.entryset

//now add all these headers to httpget using addheader method

**public** CloseableHttpResponse get (String URL, HashMap<String, String> mapheaders)

**throws** ClientProtocolException, IOException {

CloseableHttpClient httpclient = HttpClients.*createDefault*();

HttpGet httpget = **new** HttpGet(URL);

**for** (Map.Entry<String, String> entry : mapheaders.entrySet()) {

httpget.addHeader(entry.getKey(), entry.getValue());

}

CloseableHttpResponse httpresponse = httpclient.execute(httpget);

**return** httpresponse;

}

Errors:

1. The field Baseclass.prop is not visible

Reason:

Public keyword is miising

solution:

When we access the variables defined in the base class in a child class,we have to

i)define it globally

ii)we have to use public keyword