Automate Post call using HTTPClient:

\*to automate POST call,we need to send **URI with request Payload and Hashmap header.**

Steps:

1.Create a Post method and pass the following parameters:

\*URI

\* Request JSON payload

\* Header map

**public** **void** post(String URL,String JSONentity,HashMap<String,String>mapheaders) **throws** ClientProtocolException, IOException

{

CloseableHttpClient httpclient =HttpClients.*createDefault*();

HttpPost httppost=**new** HttpPost(URL);

//add headers to httppost

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

{

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

}

//add json request payload

httppost.setEntity(**new** StringEntity(JSONentity));

CloseableHttpResponse httpresponse=httpclient.execute(httppost);

}

//here, JSONentity is the json request payload and it is available within setentity method

2.Create a test class for Post\_API test.

\*call the base class constructor using super ()

\* set your @beforemethod and set the value to URI

3.create Test () method and access the post method created within rest\_client java class using obj ref variable.

3.Create a hashmap for headers and all the values in it.

4.Create a jsonrequest payload

How to create a jsonrequest payload?

When we observe the Post api here,

/api/users🡪it is for creating new users with a unique id

Steps:

1.Create a package com.qa.data and create a user class equivalent to request json payload in it and define all the attributes to the users in this class.

2.Define the variables/attributes that u want to add for that user info.

**public** **class** users {

String name;

String job;

}

3.create default constructor of that class and a user constructor and pass all the attributes to it.

4.define getter and setter the variables

How to add getter and setter:

Rightclick🡪src🡪generate getters and setters

Purpose of getter and setter:

Once the variables are initialized,at the run time,we can get any value using getname() method.

5.POJO-Plain old java object

This user class which is equivalent to json request payload is now POZO i.e.java object.

6.How to convert user class/java object to Json object?

i) It is done by marshelling (converting java object to json)

Marshelling is done by adding a Jackson API dependency in POM.xml

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.8</version>

</dependency>

ii)create Objectmapper class and it is available within Jackson-API

ObjectMapper mapper=**new** ObjectMapper();

iii)create an object for users class and pass the values to the variables in the constructor so that variables in the user class will be initialized.

users User=**new** users("morpheus", "leader");

iv)to convert the java object to json ,it is done by writevalue method available within mapper class

we need to create a json file and also pass the user class object to it.

mapper.writeValue(**new** File("C:\\jansi\_javafiles\_2\\RestAPI\_automation\\src\\main\\java\\com\\qa\\data\\users.json"), User);

now,Jason object will be created within user.json file .

v)convert java object to jsonstring .

it is done by wrtievalueAsString methos available within mapper class and pass user class object ref variable.

String jsonstring= mapper.writeValueAsString(User);

7.Call the Post method in test class with three arguments

rc.post(URI,jsonstring , mapheaders);

now,we can get the following from httpresponse

1.status code

2.jsonresponse string

//convert it to json object also

3.convert jason object to java object by unmarshelling

🡪using readvalue method in mapper class and pass response string and user class

users userresobj=mapper.readValue(jsonresponsestring, users.**class**);

System.***out***.println(userresobj);

4.validate the expected response ibject and actual created user object

System.***out***.println(userresobj.getName().equals(User.getName()));

Summary:

Steps:

1.Create a post call and pass the following

1.URI

2.headermap

3.request jsonentity

2.create a java class equilvalent to that of request payload in postman

1.create a user class and define the variables in the constructor

2.generate getter and setter for variables declared.

3.create java object for that class and pass the arguments/variables in the constructor

4.now,we need to pass this java object as json entity,for this,we need to convert it to json object in the Jason string format and then pass it.it is done by marshelling and achieved by mapper class with the help of Jackson API dependency.

5.we can also save this json object in a file

6.next is httpresponse validation.we need to validate the following,

i)status code

ii)json response

we get the json response in the raw format,so first we need to convert it to string format with the help of entityutils class.

Now,we need to convert jsonresponse string to java object through unmarshelling with the help of mapper class.

Now,validation can be done with assertion by comparing two references of java objects. i.e.

Comparing expected java object(that we have created in user class) and response java object that we got in http response.

7.we can also convert the jsonresponse string into json object thru jsonobject class.

Errors :

1.Cannot construct instance of `com.qa.data.user\_pojo` (no Creators, like default construct, exist): cannot deserialize from Object value (no delegate- or property-based Creator)

Reason:

If we create a user defined constructor without a default constructor

Solution:

Default cons. Creation Is must if we need to create a parameterized constructor

2.Unrecognized field id and created at

We have to initialize the complete response attributes