**POSTMAN TOOL**

--> dummy API : https://reqres.in

->Create a simple maven project

->Add all rest assured jars in class path

->import dependency in project

->three parts of testing API

--> given() : you need to add the end point url/header/body/query param

--> when() : give conditions like type of request/ resource URI

--> then() : validate the response recieved from above given/when parts of API

--> .log().all() : added to log the response in console.

--> if exception occurs : response will be

Exception in thread "main" java.lang.AssertionError: 1 expectation failed.

Expected status code <209> but was <200>.

--> Example response:

Request method: POST

Request URI: https://rahulshettyacademy.com/maps/api/place/add/json?key=qaclick123

Proxy: <none>

Request params: <none>

Query params: key=qaclick123

Form params: <none>

Path params: <none>

Headers: Accept=\*/\*

Content-Type=application/json; charset=UTF-8

Cookies: <none>

Multiparts: <none>

Body:

{

"location": {

"lat": -38.383490,

"lng": 33.427360

},

"accuracy": 50,

"name": "Niwas",

"phone\_number": "(+91) 983 893 3930",

"address": "29-A, side layout, cohen 09",

"types": [

"shoe park",

"shop"

],

"website": "http://google.com",

"language": "IDOKOREAN-IK"

}

HTTP/1.1 200 OK

Date: Wed, 10 Mar 2021 17:21:19 GMT

Server: Apache/2.4.18 (Ubuntu)

Access-Control-Allow-Origin: \*

Access-Control-Allow-Methods: POST

Access-Control-Max-Age: 3600

Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With

Content-Length: 194

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

Content-Type: application/json;charset=UTF-8

{

"status": "OK",

"place\_id": "a893514e153daeae65d9f394f77e4e55",

"scope": "APP",

"reference": "16ad4c8cdec576cf3c60c6dfdff9fb2816ad4c8cdec576cf3c60c6dfdff9fb28",

"id": "16ad4c8cdec576cf3c60c6dfdff9fb28"

}

--> to use assertion add testNG jar file in project build path.

--> use testNG 7.0 version

--> debug in testNG , install it from masket place and add jcommander dependency

--> collections : group of API request that can be stored stored in a logical & organised manner

--> Authentication can be given at collection level

--> monitor collection : run collection periodically/ under run tab

--> add folder inside collections to group requests inside collections

--> generate collection runner --> goto run option and add parameters according to required.

--> you can export result/ check run summary/retry

--> variables in post : any element that can be stored that can take different values. It is use to reuse values at multiple places. Also to avoid repition and re-work in case of value changes.

--> under collection , goto option edit and add variable. Replace the original value in the url with

the key.

--> set current & initial values in variables as the old value.

--> you can also create new Environment ans set variables at environment level.It will be

common to that environment.

--> can also refer to postman console in View Tab in top to check the logs at postman level/

values of variable

--> expanding logs can give further info about the logging details.

--> can also set variables at global level. Common to all environments.

--> Scripting GET/POST variables:

--> Add under Test option write scripts :

console.log("This is logging test");

var jsonData = JSON.parse(responseBody);

postman.setEnvironmentVariable("placeID", jsonData.place\_id);

console.log("Place ID :" +pm.environment.get("placeID"));

pm.variables.get();

pm.variables.set();

pm.globals.get();

pm.globals.set();

pm.environment.get();

pm.environment.set();

pm.collectionVariables.get("variable\_key");

Using data variables:

The Collection Runner lets you import a CSV or a JSON file, and use the values from the data file inside requests and scripts. You cannot set a data variable inside Postman because it is pulled from the data file, but you can access data variables inside scripts, for example using pm.iterationData.get("variable\_name").

Using dynamic variables : Postman provides dynamic variables that you can use in your requests.

Examples of dynamic variables are as follows:

{{$guid}} : A v4 style guid

{{$timestamp}}: The current timestamp (Unix timestamp in seconds)

{{$randomInt}}: A random integer between 0 and 1000

See the Dynamic Variables section for a full list.

To use dynamic variables in pre-request or test scripts, you need to use pm.variables.replaceIn(), e.g. pm.variables.replaceIn('{{$randomFirstName}}').

--> Setting Environment: it is a key value pair used to refer common values among all API service

requests.

--> Snippets: They used to create quick scripts in postman.

--> can individually add scripts under a particular request/ or at collection level / folder level

--> pre-request scripts : js code that is executed before response

--> test : to execute after response.

--> Test : js code that is executed after receiving response back from the server.

--> can individually add scripts under a particular request/ or at collection level / folder level

--> Example:

pm.test("Verify response time ::", function(){

pm.expect(pm.response.responseTime).to.be.below(200);});

--> Debug: using console window / under view -> developer->show view.

--> Add data file : set data into json/.csv file and using variable concept pass new key. In collection runner add file and run the case.

--> to test : FAILED

tests["conains email"]= responseBody.has(data.email);

tests["conains password"]= responseBody.has(data.password);

--> Authorization : in postman it is termed as authorization and not authentication.

--> authentication : valid credentials and you are allowed in an environment.

--> authorization : what all you can access in that environment.

--> since API is like the endpoint+resource, it basically approving you to access a resource, hence

termed as authorization.

--> how to add : learn

--> CommandLine runner and Jenkins :

--> always install node.js on C:\ else it does not recognise npm/node commands.

--> install newman : CLI runner for postman

--> export collection to a location

--> to to that location in cmd and type : newman run File\_Name.json-- shows all collection data

--> fir jenkins: run your jenkins on browser

--> goto new Item, add item name, select freestyle project and OK.

--> under configuration , add build steps- execute windows batch command

--. Under commands add same command as of cmd. :

--> C:\Users\apurva.misra\PostmanFile>newman run CollectionRunner.json

--> then build the job and under view as plain text, one can see the whole collection data.

--> workspace : in postman its an area where you can group, organise and manage collection. Only in

v6.0 and above.

--> it can be TEAM/ Individual, can view workspace under ... view tab

--> under browse , it show details of workspace/ environment/ collections/ to duplicate or add

workspace/ collection / environment.

--> also show details of the workspace / add /delete

--> monitors : help to run collections periodically to check the performance and response of the API.

--> create using new option/ collections-add monitor / directly from monitor window in

browser.

--> open monitor/ add collection / add environment / batch time /other fields and submit.

--> on clicking monitor , popup window opens which shows the monitor logs.

--> can edit / pause monitor settings.

--> by default get 1000 monitoring calls per month free.

--> documentation : lets you share API information in a beautifully formatted web page.

--> create new API doc , add collection, add description- then submit, it gives you a link to access

the API.

--> can also make it as public/ private.

--> can also publish the doc, using publish option and select environment./ can directly publish

using collection option.

--> values of that environment will be populated in doc.

--> can share that published url to anyone for public use.

--> remove any private data like password before publish. / can also unpublish.

https://documenter.getpostman.com/view/10861769/Tz5p6dfD

--> share collection : under share opyion get the link.

https://www.getpostman.com/collections/69bcc574c1434dea6c2f

--> using newman in CLI you can run the collection : newman run "url"

--> API chaining : using values from response of one API into body or paramaters of another.

--> MOCK API : api that imitates a real API by providing realistic response to requests.

--> required to run the test / complete scenarios in cases when API are not fully developed.

--> 3rd party API response required for testing but there is no access.

--> first create a mock server with icon or under any collection :

--> in url add example.com, and give response body/query as { "name" : "Mock"}

--> submit- then select name/environment and others and then create. Copy the mock

url provide used to hit mock response.

--> replace the url tab in request with the mock url provided and send e.g.

https://8f71733c-1875-47ea-875a-9aa9cd9b0c41.mock.pstmn.io