REST ASSURED API

**RestAssured.baseURI = “…URI…”; = To Store The Base URI**

**RequestSpecification obj1 = RestAssured.given(); = used to send the http request and store the request in obj1**

**Response obj2 = obj1.request ( Method . Http\_request\_type , ”path parameter”); = used to store the response & Execute the http request with his mentioned type**

1. **obj2.getBody().asPrettyString(); = Return response body in Well managed string format**
2. **obj2.getStatusCode(); = return response Status code in integer format**
3. **obj2.getStatusLine(); = return response Status line In string format**
4. **obj2.getTime(); = return response time in long format**
5. **obj2.header(“header key”); = return header value corresponding to mentioned key**
6. **obj2.headers(); = capture all the headers Keys And Value OR Point out all the headers Keys And Value**
7. **obj2.jsonPath().get(“ response body key ”) = return response body value corresponding to mentioned key in String format**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TO GET THE RESPONSE BODY PARAMETER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Response obj2 = obj1.request ( Method.http\_request\_type , “path parameter” );**

**String obj3 = obj2.getBody().asPrettyString();**

**obj3.contains (“ value in response body ”); = verify whether mentioned value is present in response body or not & return Boolean value**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SEND THE POST REQUEST \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**To send the post request we need to add body with it, so most of time it is in json format with help of json object we can achieve it.**

**RequestSpecification obj = RestAssured.given();**

**JSONObject obj1 = new JSONObject ();**

**obj1 . put (“ key ”, “ value ”) ; = create body to be send**

**obj . header (“ key ” , ” value ”); = set header part ex\_\_\_\_\_\_\_\_\_\_ .header("Content-Type", "application/json");**

**obj . body ( obj1.toJSONString () ); = convert into json format (String into json object =)**

**Response obj2 = obj. request (Method.POST, "path parameter"); = To send the post request & store response in object**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* HTTP REQUEST WITH AUTHENTICATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**To pass authentication set username and password before sending the request**

**PreemptiveBasicAuthScheme obj = new PreemptiveBasicAuthScheme ();**

**obj.setUserName (“ username ”); = Used set username**

**obj.setPassword(“ password ”); = Used to set password**

**obj.generateAuthToken(“ token ”); = Used to set token**

**RestAssured.authentication = obj ; = Used to allow execution with pre-defined authentication**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**POSTMAN**

**HTTP methods:**

**GET: To retrieve the data (idempotent)**

**POST: To create the new data (not idempotent=409[conflict])**

**PUT: To updated data (Follow upsert operation) (idempotent)**

**PATCH: To updated Specific data (not idempotent=409[conflict])**

**DELETE: To delete the data (idempotent)**

**---------------------------------------------**

**Upsert Operation (Update or Create) = If data is already exist then update it, and if not then create it**

**Idempotent = Idempotent means that you can repeat the request and the result should be the same**

**----------------------------------------------**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| **Simple Object Access Protocol** | **Representational State Transfer** |
| **It is a protocol** | **It is a design pattern** |
| **Work only with XML language** | **Work With Different language(json,xml,javascript)** |
| **Require more bandwidth** | **Required less band-Width** |
| **Used where security is high** | **Used where security is low** |

**---------------------------------------------------------**

**Authentication = Verify the user**

**Authorization (Access Control) = verify the Access of the user**

**-------------------------------------------------------------------------**

**Oauth 2.0 = Authorization protocol permits a user to share access to specific resource with a service provider (Allow limited Access)**

**Bearer Token = it one kind of access key which generated by server with expire time based ..username + password + expire time duration**

**-------------------------------------------------------------------**

**API SNIPPETS**

**For validation purpose:-**

**1. To verify specific value in response body**

**var obj-name =JSON.parse (responseBody);**

**tests[“ Test name ”] = obj-name . json path for response data key == 2;**

**Ex: var response =JSON.parse(responseBody);**

**Tests [“page no”] = response.page == 2;**

**2. To verify Status Code**

**pm.test( “ Test name ” function () { pm.response.to.have.status(Status code)});**

**Ex: pm.test( “Status code is 200”, function () { pm.response.to.have.status(200)});**

**3. To verify Response Time**

**pm.test (“ Test Name ” , function () { pm.expect (pm.response.responseTime) .to .be .below (Response time); });**

**Ex: pm.test(“response time is less than 200 ” , function () { pm.expect (pm.response.responseTime) .to .be .below (200); });**

**4. To verify Header Part**

**pm.test(" Test name",function() { pm.response.to.be.header("Header key "," respected Header key value");});**

**Ex: pm.test("check Content-Type header",function() { pm.response.to.be.header("Content-Type","application/json");});**

**TO SET VARIABLE**

**Used For response chaining process , verification Purpose**

1. **Pm.enviroment.set (“variable key” , “variable value” ); = To set environment variable**
2. **pm.globals.set (“ variable key ”,” variable value ”); = To set Global variable.**
3. **pm.collectionVariables.set ( “variable key”, “ variable value”); = To set collection variable**

**TO GET VARIABLE**

**Used For response chaining process , verification Purpose**

**pm.enviroment .get(“variable key”); = To get environment variable with mentioned value of key**

**pm.globals.get( “variable key” ); = To get globals variable with mentioned value of key**

**pm.collectionVariable.get (“variable key”); = To get collection variable with mentioned value of key**

**Response Code**

**200 = ok……. (Successful)**

**201 = Create…… (Data are created successfully)**

**202 = Request Accept…( Request Accepted for Processing but Action are pending )**

**204 = No content ….(Response Body is Empty)**

**300 = Multiple choice**

**301 = move Permanently… (Url move permanently another address)**

**302 = move Temporally … (Url move Temporally another address)**

**400 = bad Request … error from client side**

**401 = Un-authorized …(need Authorization)**

**403 = forbidden …(Access reject to request from server side)**

**404 = not found address (Request can not find the address)**

**405 = method not allowed (Specific method not allowed)**

**409 = conflict ( request constraint from server side)**

**500 = internal server issue… (Service un available due to error From the server side)**

**502 = Bad Gateway…. ( proxy url )**

**503 = Service Unavailable…( The server is not ready to handle the request. Common causes are a server that is down for maintenance or that is overloaded.)**

**504 = Gateway Timeout…..( gateway response time out)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***