GET REQUEST – Expect Response 200 Ok

Post REQUEST – Expect Response 201 created

Put REQUEST(when the resource is not present) – Expect Response 201 created

Put REQUEST(when the resource is present) – Expect Response 200 Ok

Patch REQUEST(when the resource is present) – Expect Response 200 Ok

Delete REQUEST – Expect Response 200 Ok

**Note:** we can duplicate the request by click and select the duplicate option

|  |  |
| --- | --- |
| Authentication | Authorization |
| Authentication is a process of identifying who someone is with a help of credentials | Authorization is a process of identifying to which and all resources an authenticated person can access |

AUTHORIZATION TYPES

No Auth🡪 No Authorization is required

API key🡪 value of appid, it can be given either in headers/query parameters

API key is representing an authorized person who has access to an application which should not share to others.

Bearer Token🡪 Also known as “token Authentication”

Bearer token is a cryptic string(not easily understandable)

Bearer token will be generated when a server sending response back to the login request

Bearer token should be mentioned in application headers not as a query param. Only if the bearer token is present then only the request can be successfully made to protected resources

After authentication whatever protected resource, the user is trying to access there, and all the token authentications place the major role silently

Basic Auth🡪 Only Username and password is required (E.g., Gmail / any public application)

Digest Auth🡪 Similar to Basic Auth. It also requires Only Username and password (E.g., Gmail / any public application).

Digest authentication does the same thing as Basic authentication, but it **provides a security improvement in the way in which a user's credentials are sent across the network**. Using Digest authentication, credentials are transmitted across the network as an MD5 (message digest) hash.

If the requested resource is protected then user should get 401 Un Authorized error with realm, nonse and qop values (one-time values). so, postman will re send the login request with proper credentials along with these details without throwing any errors intermittently

Incase if we disable the checkbox then we should do the re-request part once after the 401-status code returned as a response from a server

OAuth 2.0🡪 (OAuth 1.0 deprecated on 2012 itself, so OAuth 2.0 is live now)

Authorization between Service 1 and Service 2

Here between Grow application and Gmail (two different services) authorization is happening. Which is OAuth

**Gmail uses the OAuth 2.0 protocol** for authenticating a Google account and authorizing access to user data. You can also use Google Sign-in to provide a "sign-in with Google" authentication method for your app

Variables—>

Global Variable 🡪 This variable is applicable for all the collections and environments available within the workspace

Environment variable 🡪 This variable is applicable for individual environment

Collection variable 🡪 This variable is applicable for individual collection

Global Variables🡪 Global variable is applicable for all the collections present inside our workspace

Step 1: click on eye icon on the right corner  
step 2:click on add button available next to Global  
step 3: enter the variable and give the values

Step 4: in the URL tab provide the global variable within {{ Global Variable }}

Environment variable 🡪 This variable is applicable for individual environment {{ Env Variable }}

Step 1: click on eye icon on the right corner  
step 2:click on add button available next to Environment  
step 3: enter the variable and give the values

Collection Variables 🡪{{ Collection Variable }}

Click on set as a new variable

Enter the name and select the scope as collection

Another way to create collection variable🡪click on the collection name and select the last option called variables and add the required variable

**Writing a test in postman**

**Test should be written using Java script**

Built in postman Object🡪pm (which contains both request and response)

**Syntax  
Pm.test(“name”, javascript function(){**

**}) or**

**Pm.test(“name”, ()=>{**

**})**

**Types of doing Assertions (**Postman using Chai Assertion library for validation) 1.Use method chaining (BDD style)

Pm.response.to.have.status(200);

Pm.response.to.have.status(ok);

2.Use Expect function

Pm.expect(pm.response.code).eql(200);

**We can also put 2 assertions inside a single test**

If any one of the assertions fails, then whole test will be a failed one **JSON RESPONSE VALIDATION**

Creating a variable to store json response ( var JsonResponse = pm.response.json() )

pm.expect(JsonResponse[6].lastname).eql(“Kirubha”);

**JSON PATH🡪**

JsonResponse[6].lastname

**Validate the presence one text without mentioning the json path**

Include() is a function to check whether the string contains the required value

Note: have is not a function…include() is a function

pm.expect(pm.response.text()).to.include(“Kanmani”)

**Writing Test using snippets**

From snippets we can directly use the frequently used codes instead of writing it

**To print the values in the console🡪**

Console.log(“condition/print statement”);

Old Method🡪Instead of hardcoding the value to be checked we can use data.key (data is a built-in object which stores the values present in json/csv file)

New Method🡪Instead of data.value pm.iterationData.get(“key”)

**Same will be applicable for console.log also**





**Data Driven Method**

**Data from External files**

From Json file

From CSV file

Instead of running each method separately we can run oncollection level or folder level

Step 1:Create a json file/csv file in local machine

Note: For json file data should be present inside an Array

Step 2: Click on Collection/Folder

Step 3: Click on the run button which is present in the right side

Step 4: Import your json/csv file by clicking “Select file” option present under Data

Step 5: Click on preview before running the request to validate whether all values are populated correctly

Step 6: Before Running we should make sure the hardcoded values inside the request body are replaced with variables present inside the json

Step 7: we should make sure the varibale type should be mentioned properly inside the request body

For example.

For String “{{variable name}}”

For Integer/long {{Age}}

Step 8: Run the request and validate the response