**CHAI ASSERTION LIBRARY**

Assertion Library scripts – for Response validations.

In Postman, you can use the **"Tests"** tab to write scripts to validate various aspects of the API response, including status code, headers, cookies, response time, and body. Below is a list of scripts for each of these checks:

**1. Status Code**

To verify the status code returned by the API:

pm.test("Status code is 200", function () {

pm.response.to.have.status(200); // Change '200' to the expected status code

});

**2. Headers**

To check for a specific header in the response:

pm.test("Content-Type is present", function () {

pm.response.to.have.header("Content-Type");

});

// To verify the value of a specific header

pm.test("Content-Type is application/json", function () {

pm.expect(pm.response.headers.get("Content-Type")).to.eql("application/json");

});

**3. Cookies**

To check if a specific cookie exists and to validate its value:

pm.test("Cookie is present", function () {

pm.expect(pm.cookies.has('cookieName')).to.be.true;

});

// To validate the value of a cookie

pm.test("Cookie value is correct", function () {

let cookie = pm.cookies.get('cookieName');

pm.expect(cookie.value).to.eql("expectedValue");

});

**4. Response Time**

To check the response time of the API:

pm.test("Response time is less than 500ms", function () {

pm.expect(pm.response.responseTime).to.be.below(500); // Set your threshold in milliseconds

});

**5. Response Body**

To verify the structure or specific values in the response body:

// Check if the response body contains a particular value

pm.test("Response body contains expected value", function () {

pm.expect(pm.response.text()).to.include("expectedValue");

});

// To check if the response body is valid JSON and contains certain fields

pm.test("Response body contains certain fields", function () {

let jsonData = pm.response.json();

pm.expect(jsonData).to.have.property("fieldName");

pm.expect(jsonData.fieldName).to.eql("expectedValue");

});

**7] Json Schema Validation –**

//Define the Json schema ou want to validate

var schema **=**

{

  "type": "object",

  "properties": {

    "id": {

      "type": "integer"

    },

    "name": {

      "type": "string"

    },

    "email": {

      "type": "string"

    },

    "gender": {

      "type": "string"

    },

    "status": {

      "type": "string"

    }

  },

  "required": [

    "id",

    "name",

    "email",

    "gender",

    "status"

  ]

};

//Response Json Schema Validation

pm.**test**("Response Body JSON Schema Valid", **function** () {

    pm.response.to.have.jsonSchema(schema);

});

**6) Full Example**

If you want to combine all the checks into a single script:

// Status code

pm.test("Status code is 200", function () {

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

});

// Headers

pm.test("Content-Type is present and correct", function () {

pm.response.to.have.header("Content-Type");

pm.expect(pm.response.headers.get("Content-Type")).to.eql("application/json");

});

// Cookies

pm.test("Cookie is present and has correct value", function () {

pm.expect(pm.cookies.has('cookieName')).to.be.true;

let cookie = pm.cookies.get('cookieName');

pm.expect(cookie.value).to.eql("expectedValue");

});

// Response time

pm.test("Response time is less than 500ms", function () {

pm.expect(pm.response.responseTime).to.be.below(500);

});

// Response body

pm.test("Response body is as expected", function () {

let jsonData = pm.response.json();

pm.expect(jsonData).to.have.property("fieldName");

pm.expect(jsonData.fieldName).to.eql("expectedValue");

});

8) Example –

\* GET Request Example –

pm.**test**("Status code is 200", **function** () {

    pm.response.to.have.status(200);  // Change '200' to the expected status code

});

pm.**test**("Content-Type is present", **function** () {

    pm.response.to.have.header("Content-Type");

});

// To verify the value of a specific header

pm.**test**("Content-Type is application/json", **function** () {

    pm.expect(pm.response.headers.**get**("Content-Type")).to.eql("application/json; charset=utf-8");

});

pm.**test**("Response time is less than 1000 ms", **function** () {

    pm.expect(pm.response.responseTime).to.be.below(1000);  // Set your threshold in milliseconds

});

// Check if the response body contains a particular value

pm.**test**("Response body contains Agniprava Sethi", **function** () {

    pm.expect(pm.response.text()).to.include("Agniprava Sethi");

});

// To check if the response body is valid JSON and contains certain fields

pm.**test**("Response body contains id", **function** () {

    let jsonData **=** pm.response.json();

    pm.expect(jsonData).to.have.property("id");

    pm.expect(jsonData.fieldName).to.eql("7467708");

});

* Post Request Example – Json Schema
* //Define the Json schema ou want to validate
* var schema **=**
* {
* "type": "object",
* "properties": {
* "id": {
* "type": "integer"
* },
* "name": {
* "type": "string"
* },
* "email": {
* "type": "string"
* },
* "gender": {
* "type": "string"
* },
* "status": {
* "type": "string"
* }
* },
* "required": [
* "id",
* "name",
* "email",
* "gender",
* "status"
* ]
* };
* //Response Json Schema Validation
* pm.**test**("Body JSON Schema Validation", **function** () {
* pm.response.to.have.jsonSchema(schema);  // Change '200' to the expected status code
* });

API Requests Validation Points –

1)GET Request –

Status Code –

Response Time –

Header –

Body Json Fields –

Body Json Schema –

2)POST Request –

Status Code –

Response Time –

Header –

Json Schema –

Body Json Fields -