**Postman Scripting, Cookies, cURL, and Advanced API Features - Tutorial Document**

**Overview**

In this document we will explore **Postman Scripting**, **Cookies**, and how to use **cURL** for making API requests. These are advanced features that allow for enhanced testing, request automation, and understanding of API behaviour.

* **Learning Objectives**:
  + Understand and use Postman scripting for pre-request and test scripts.
  + Work with cookies in Postman and understand their role in API authentication.
  + Learn how to use cURL commands for testing APIs and exporting Postman requests.
  + Learn about the Advanced API features.

**1. Postman Scripting: Introduction**

* **What is Postman Scripting?**
  + Postman allows you to write scripts in JavaScript for automation, validation, and processing within the **Pre-request Script** and **Tests** tabs.
  + **Pre-request Script**: Executes before the request is sent. Used for tasks like setting variables or preparing data for the request.
  + **Tests**: Executes after the response is received. Used for validating response data, status codes, and response times.

**1.1 Postman Scripting Examples**

1. **Pre-request Script Example**:
   * **Step 1: Open Postman and Create a New Request**
     1. Open Postman.
     2. Click + to open a new request tab.
     3. Set the request method to GET.
   * **Step 2: Add a Pre-request Script**
     1. Go to the Scripts tab and select Pre-request.
     2. Use Postman’s snippet library to add pre-written code
     3. On the right side of the Pre-request Script editor, find the Snippets section.
     4. Click on **"Set an environment variable"**. This snippet will add a variable to the environment.
     5. The snippet will look like this: pm.environment.**set**("variable\_key", "variable\_value");
     6. Modify the snippet to include authentication logic: pm.environment.**set**("baseUrl", "https://jsonplaceholder.typicode.com");  
        pm.environment.**set**("apiKey", "12345");
     7. Enter the URL: {{baseUrl}}/posts?apiKey={{apiKey}}
     8. Click **Send**.
     9. **Expected Response**:
        1. **Status**: 200 OK.
        2. **Body**: List of posts.
2. **Tests Script Example**:
   * Add the following code to the post-response.
   * **Objective**: Validate if the response status code is 200 and the response body contains specific data.
   * **Script**:

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

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

});

pm.test("Response contains user data", function () {

pm.response.to.have.jsonBody("qui est esse");

});

* + **Explanation**: This script runs after the response is received. It checks if the status code is 200 and that the response body contains a qui est esse key.

**2. Working with Cookies in Postman**

* **What are Cookies?**
  + Cookies are small pieces of data that the server sends to the client (your Postman) and are returned by the client with each subsequent request.
  + In API testing, cookies are often used for authentication, maintaining sessions, or tracking user data across requests.
* **Handling Cookies in Postman**:

**2.1 Adding Cookies in Postman**

1. Create a collection named **Cookies in Postman**
2. Add a request **Exploring cookies in postman**
3. **Set Up the API Endpoint**
   * HTTP Method: **GET**
   * URL: http://postman-echo.com/cookies/set
4. **Add a Cookie**
   * Click the **"Cookies"** button below the Send button.
   * Copy the domain (postman-echo.com).
   * Paste the domain into the cookie manager.
   * Click **"Add Domain"**.
5. **Specify Cookie Data**
   * Under the added domain, click **"+" (Add Cookie)**.
   * Add key-value pairs for cookies. Just replace Cookie\_=value with:
     + Example:
       - **Cookie\_1**: skill1=**value**: Selenium using Java
       - **Cookie\_2**: skill2=**value**: API testing using POSTMAN
       - **Cookie\_3**: skill3=**value**: API testing using Selenium
   * Click **Save** after each entry.
6. **Send the Request**
   * Click **Send**.
   * Verify:
     + Response Status Code: **200**
     + Cookies in Response Body: Your added cookies appear under the cookies JSON object.

**2.2 Editing Cookies in Postman (Practice at home)**

1. **Access Cookies**
   * Click the **"Cookies"** button again.
   * Select the domain (postman-echo.com).
2. **Modify a Cookie**
   * Click on the specific cookie key to edit.
   * Update the value (e.g., change skill1 value to Selenium using Java and Python).
   * Save the changes.
3. **Re-Send the Request**
   * Click **Send** again.
   * Verify the updated value in the response and Postman console logs.

**2.3 Deleting Cookies in Postman (Practice at home)**

1. **Delete Specific Cookies**
   * Click **"Cookies"**.
   * Under the domain, click the **"X"** button next to the cookie key you want to delete.
2. **Delete All Cookies**
   * Click **"Cookies"**.
   * Use the delete icon for the entire domain to remove all cookies at once.
3. **Re-Send the Request**
   * Click **Send**.
   * Verify that the cookies are no longer present in the response.

**2.4 Managing Cookies with APIs**

**For a clear understanding, make sure that you’ve deleted all the cookies before executing these.** You can manage cookies programmatically using APIs:

1. Add a request **Managing Cookies with APIs**
2. **Setting Cookies**
   * API Endpoint: http://postman-echo.com/cookies/set?key=value
   * Example: http://postman-echo.com/cookies/set?skill1=selenium&skill2=postman
   * Use the **GET** method to set cookies dynamically.
3. **Getting Cookies**
   * API Endpoint: http://postman-echo.com/cookies
   * Use the **GET** method to retrieve cookies set for the domain.
4. **Deleting Cookies**
   * API Endpoint: Depends on server implementation (e.g., http://postman-echo.com/cookies/delete?skill1).
   * Set the HTTP method to **GET** as it is a mock API. Send a request to the endpoint designed to clear the skill1 cookie only.

**3. Using cURL in Postman**

* **What is cURL?**
  + cURL (Client URL) is a command-line tool for transferring data using various network protocols, including HTTP, HTTPS, FTP, and others.
  + Used for testing APIs from the terminal or command prompt..
  + Postman allows you to export API requests as cURL commands, which can be executed in the **terminal or command line**.
* **General cURL syntax:**
  + **curl -X <HTTP\_METHOD> "<URL>" -H "<HEADER>" -d "<DATA>“**
    - **curl:** The command to invoke the cURL utility.
    - **-X:** Specifies the HTTP method (e.g., GET, POST, PUT, DELETE).
    - **<URL>:** The endpoint of the API you want to interact with.
    - **-H**: Sets the headers for the request (e.g., Content-Type, Authorization).
    - **-d:** Sends data (usually in JSON format) in the body of the request.

**3.1 Importing cURL into Postman**

**Step 1: Copy a cURL Command**

* Example cURL command:

curl -X GET "https://jsonplaceholder.typicode.com/posts" -H "Content-Type: application/json"

**Step 2: Open Postman**

1. Go to **File > Import** (or click the "Import" button in Postman).
2. Select the **"Raw Text"** tab.
3. Paste the cURL command and click **Continue**.
4. Click **Import** to convert the cURL into a Postman request.

**Step 3: View the Converted Request**

* Postman will display the request with all headers, method, and URL pre-configured.

**3.2 Generating cURL from Postman**

**Step 1: Create a Request in Postman**

1. Select a method (e.g., GET, POST, PUT).
2. Enter a URL (e.g., https://jsonplaceholder.typicode.com/posts).
3. Add headers or a body (if required).

**Step 2: Generate cURL**

1. Click the **Code** icon (</>).
2. Select **cURL** from the dropdown.
3. Copy the cURL command generated.

**3.3 Working Examples**

**Example 1: GET Request**

**cURL Command:**

curl -X GET "https://jsonplaceholder.typicode.com/posts" -H "Content-Type: application/json"

**Expected Output:** A JSON response with a list of posts:

[

{

"userId": 1,

"id": 1,

"title": "Sample Post Title",

"body": "Sample post content."

},

...

]

**Example 2: POST Request**

**cURL Command:**

curl -X POST "https://jsonplaceholder.typicode.com/posts" \

-H "Content-Type: application/json" \

-d '{"title":"foo","body":"bar","userId":1}'

**Postman Steps:**

1. Import the cURL.
2. Verify the method is POST.
3. Check that the JSON body is present.

**Expected Output:**

{

"title": "foo",

"body": "bar",

"userId": 1,

"id": 101

}

**Example 3: PUT Request**

**cURL Command:**

curl -X PUT "https://jsonplaceholder.typicode.com/posts/1" \

-H "Content-Type: application/json" \

-d '{"id":1,"title":"updated title","body":"updated body","userId":1}'

**Expected Output:**

{

"id": 1,

"title": "updated title",

"body": "updated body",

"userId": 1

}

**Example 4: DELETE Request**

**cURL Command:**

curl -X DELETE "https://jsonplaceholder.typicode.com/posts/1"

**Expected Output:** No content or status 200 OK.

**4. Advanced API Features**

* **Objective:** Learn how to handle pagination, sorting, and continuation tokens using the Postman Echo API.

**4.1 Pagination**

* Pagination divides data into smaller "pages" to simplify navigation and retrieval.

**4.1.1 Add Pagination Query Parameters**

Simulate a paginated API response using the page and limit query parameters.

1. **Endpoint:**

https://postman-echo.com/get?page=1&limit=5

1. **Steps in Postman:**
   * Open Postman and create a new GET request.
   * Enter the URL above into the request URL field.
   * Click **Send**.
2. **Inspect the Response:**  
   The API echoes back the page and limit values you provided in the query string. In a real-world API, this would return the first 5 items.

**4.1.2. Change the Page**

Update the page parameter to retrieve the second page:

https://postman-echo.com/get?page=2&limit=5

Repeat the steps above to test this query.

**4.2 Sorting Example**

Sorting orders the data based on a specific field, such as name or date.

**4.2.1. Add Sorting Query Parameters**

Use a sortBy and order parameter to simulate sorting.

1. **Endpoint:**

https://postman-echo.com/get?sortBy=name&order=asc

1. **Steps in Postman:**
   * Create a new GET request in Postman.
   * Enter the URL above in the request field.
   * Click **Send**.
2. **Inspect the Response:**  
   The API returns the sortBy and order fields. In real APIs, this would order data by name in ascending order.

**4.2.2. Change Sorting Order**

Change the order parameter to desc for descending order:

https://postman-echo.com/get?sortBy=name&order=desc

**4.3 Continuation Token Example**

A continuation token is like a **bookmark** that tells the server where you left off when retrieving data. Continuation tokens allow you to fetch subsequent pages of results when data size is too large.

**4.3.1 Add a Continuation Token**

Simulate a continuation token with the token parameter.

1. **Endpoint:**

https://postman-echo.com/get?token=12345

1. **Steps in Postman:**
   * Create a new GET request in Postman.
   * Enter the URL above in the request field.
   * Click **Send**.
2. **Inspect the Response:**  
   The token is echoed back in the response. In a real API, this token would be required to fetch the next set of results.

**4.3.2 Use a New Token**

Update the token to simulate fetching the next page:

https://postman-echo.com/get?token=67890

**4.4: Combine Pagination, Sorting, and Continuation Token**

Test a URL that includes all three concepts together.

1. **Endpoint:**

https://postman-echo.com/get?page=1&limit=5&sortBy=name&order=asc&token=12345

1. **Steps in Postman:**
   * Create a new GET request in Postman.
   * Enter the URL above in the request field.
   * Click **Send**.
2. **Inspect the Response:**  
   The response will echo all the parameters. This simulates real-world API scenarios.