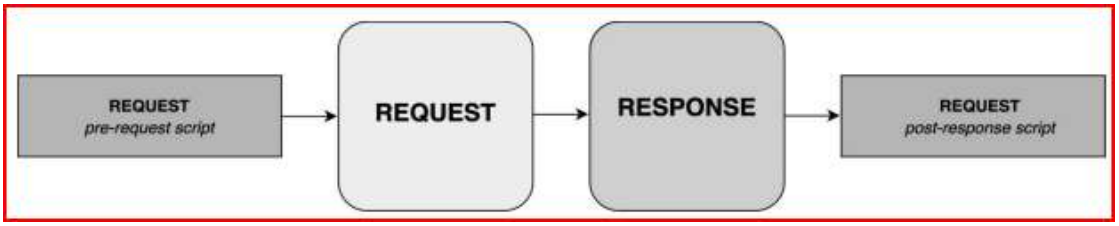
4.2. Scripts in Postman

Postman Scripts

1. Postman scripts can be written at different stages
    a. **Pre-request scripts:** Run before the request is sent.
        i. Used to create variables before sending requests
    b. **Post-response scripts:** Run after receiving the response.
        i. Used to create Assertions or validations after recieving responses.
        ii. Used to Delete variables after recieving responses.

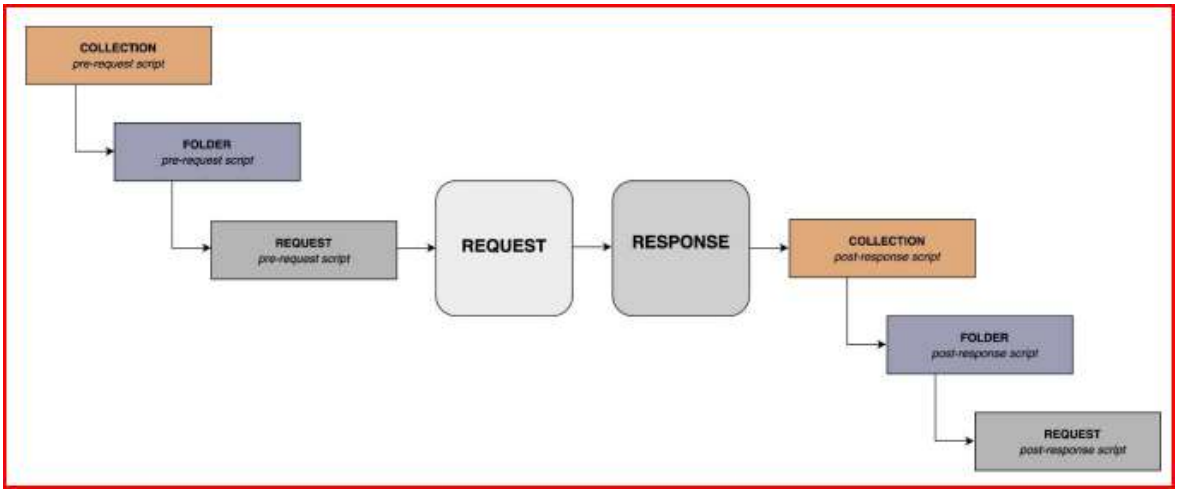
Workflow of scripts

➔ **Pre-request Script → Request → Response → Post-response**



Scripts Execution Order

➔ Scripts will be executed in 3 levels



- **Collection:** A group of requests.
- **Folder:** Subgroups within a collection.
- **Request:** Individual HTTP requests (GET, POST, etc.).

➔ A pre-request script associated with a collection will run prior to every request in the collection.

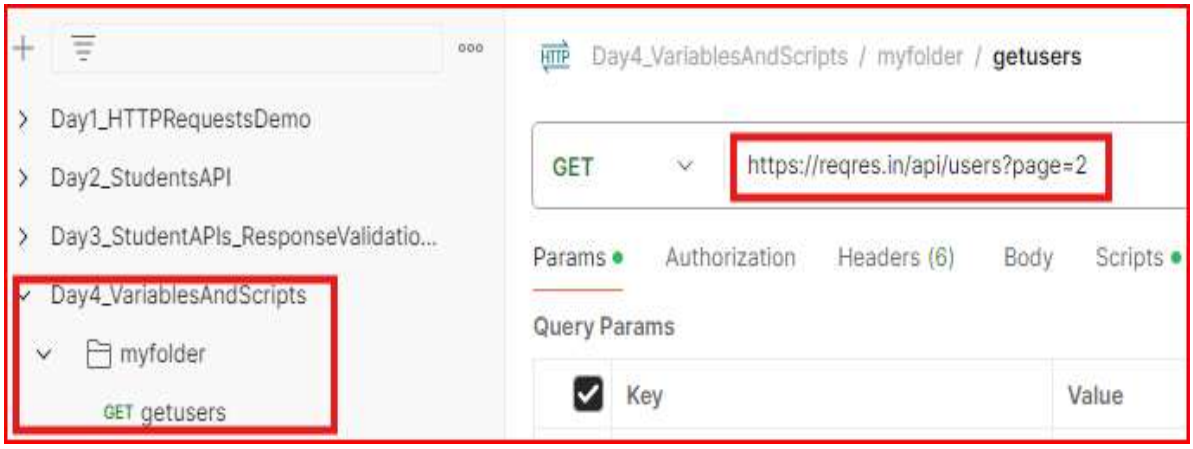
➔ A pre-request script associated with a folder will run prior to every direct child request in the folder.

➔ A post-response script associated with a collection will run after every request in the collection.

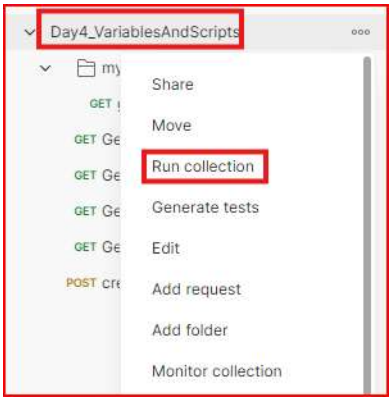
➔ A post-response script associated with a folder will run after every direct child request in the folder.

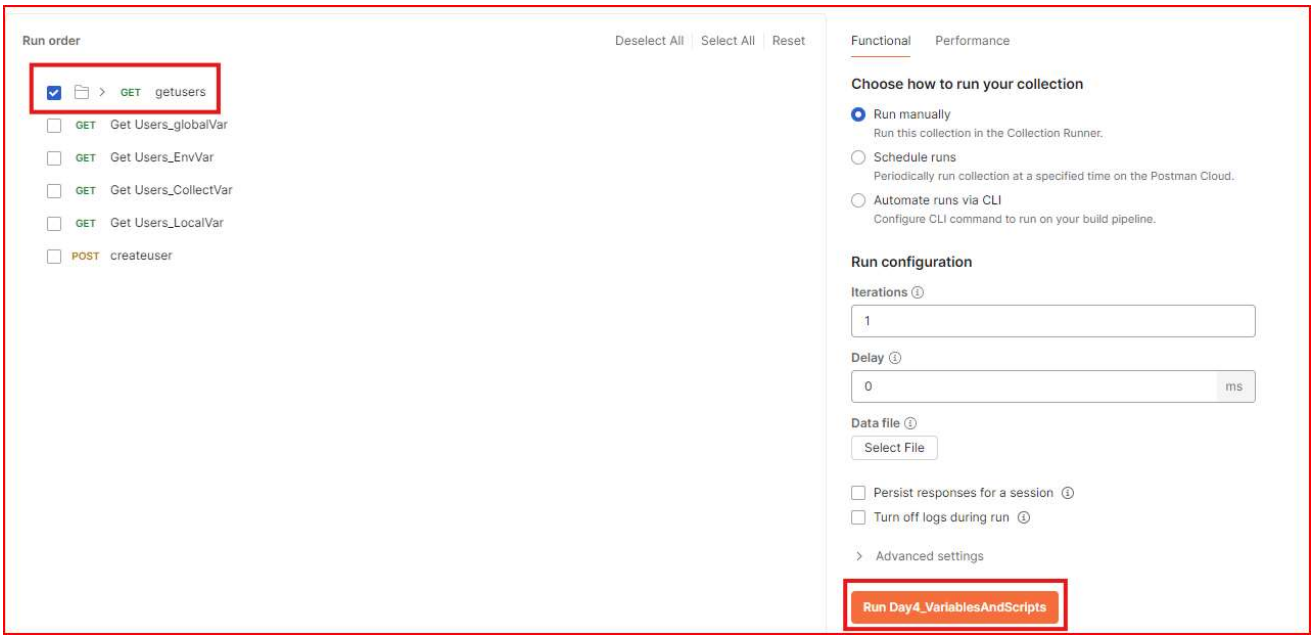
Workout in Postman Tool

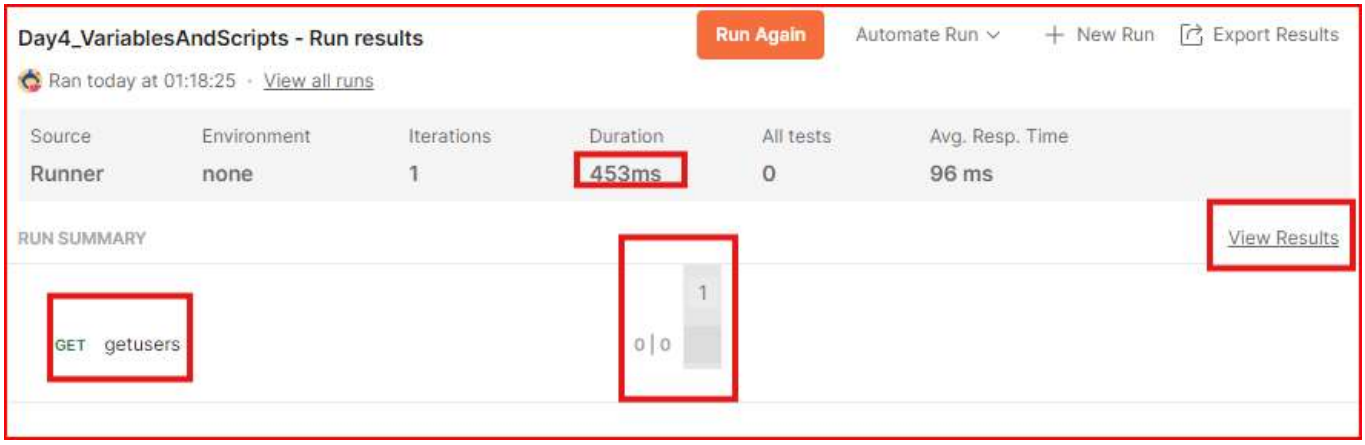
1. Create a Hierarchy as below
    a. Collection (Day4_Variables Collection) → Folder (myfolder) → Request (getusers)

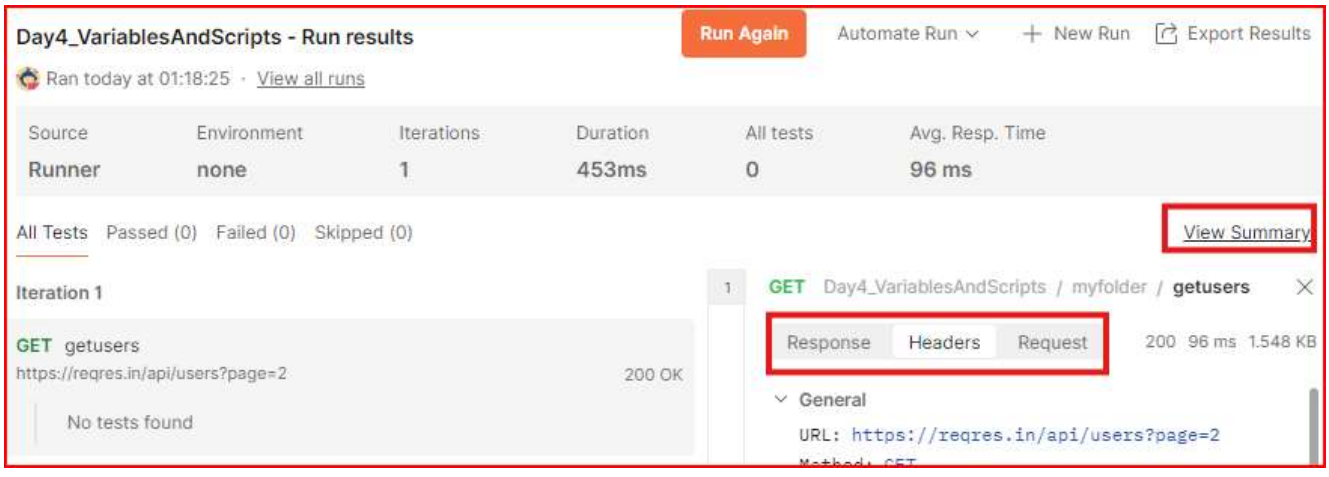


2. Every Level will be having Scripts and inside it we will have Pre-request and Post-response
3. Add Scripts as below for every level
    a. **Collection Level**
        i. **Pre-request** → console.log("Pre-request script at collection level");
        ii. **Post-response** → console.log("Post-response script at collection level");
    b. **Folder Level**
        i. **Pre-request** → console.log("Pre-request script at Folder level");
        ii. **Post-response** → console.log("Post-response script at Folder level");
    c. **Request Level**
        i. **Pre-request** → console.log("Pre-request script at Request level");
        ii. **Post-response** → console.log("Post-response script at Request level");
4. Click on Collection → **Run collection** and observe output in Console




Run order
Deselect All
Select All
Reset
GET getusers
GET Get Users_globalVar
GET Get Users_EnvVar
GET Get Users_CollectVar
GET Get Users_LocalVar
POST createuser
Functional
Performance
Choose how to run your collection
Run manually
Run this collection in the Collection Runner.
Schedule runs
Periodically run collection at a specified time on the Postman Cloud.
Automate runs via CLI
Configure CLI command to run on your build pipeline.
Run configuration
Iterations
1
Delay
0
ms
Data file
Select File
Persist responses for a session
Turn off logs during run
Advanced settings
Run Day4_VariablesAndScripts


Day4_VariablesAndScripts - Run results
Run Again
Automate Run
New Run
Export Results
Ran today at 01:18:25 · View all runs
Source
Runner
Environment
none
Iterations
1
Duration
453ms
All tests
0
Avg. Resp. Time
96 ms
RUN SUMMARY
View Results
GET getusers
1
0 | 0


Day4_VariablesAndScripts - Run results
Run Again
Automate Run
New Run
Export Results
Ran today at 01:18:25 · View all runs
Source
Runner
Environment
none
Iterations
1
Duration
453ms
All tests
0
Avg. Resp. Time
96 ms
All Tests
Passed (0)
Failed (0)
Skipped (0)
View Summary
Iteration 1
GET getusers
https://reqres.in/api/users?page=2
200 OK
No tests found
1
GET Day4_VariablesAndScripts / myfolder / getusers
Response
Headers
Request
200 96 ms 1.548 KB
General
URL: https://reqres.in/api/users?page=2


Online
Find and replace
Console
Import Complete
"Pre-request script at collection level"
"Pre-request script at folder level"
"Pre-request script at request level"
GET https://reqres.in/api/users?page=2
"Post-response script at collection level"
"Post-response script at folder level"
"Post-response script at request level"

Creating and Managing Variables with Scripts

Creating Variables in Pre-request Scripts

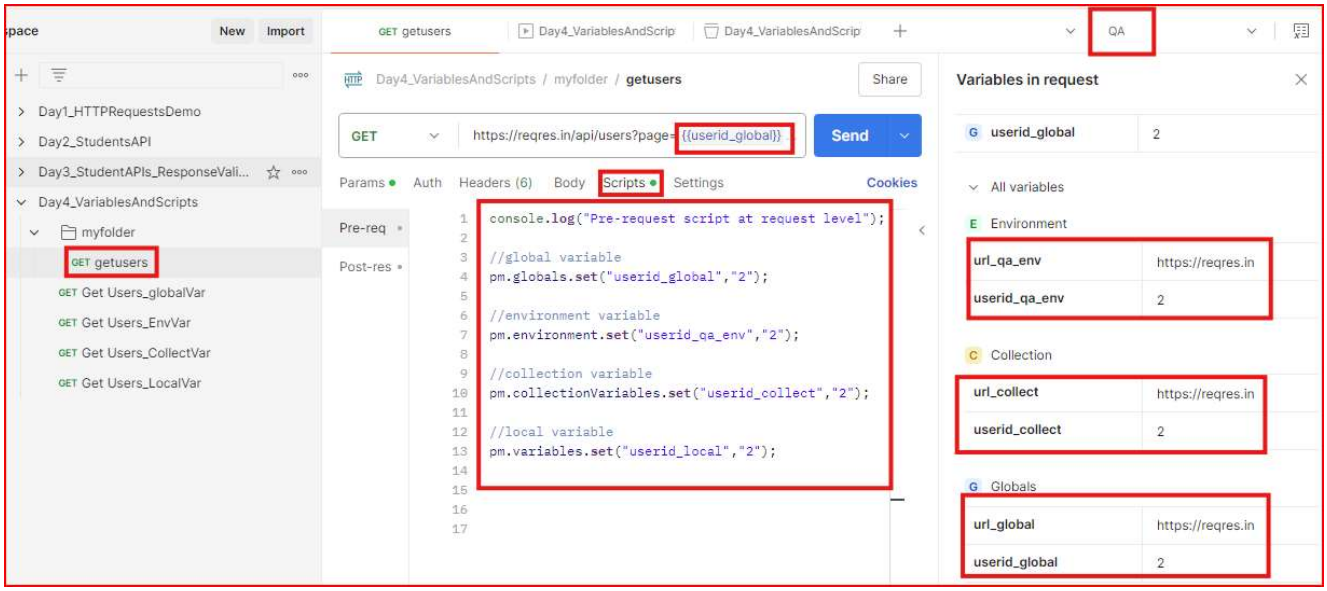
1. We can set variables before sending a request using JavaScript.
    a. **Global Variable** → pm.globals.set("userid_global", "2");
    b. **Environment Variable** → pm.environment.set("userid_qa_env", "2");
    c. **Collection Variable** → pm.collectionVariables.set("userid_collect", "2");
    d. **Local Variable** → pm.variables.set("userid_local", "2");

Workout in Postman Tool

1. In **getusers requests** pre-request script create variables using JavaScript and replace respective variable (in the pic only global) in the url.

Pre-request script in request

```
//global variable
pm.globals.set("userid_global","2");
//environment variable
pm.environment.set("userid_qa_env","2");
//collection variable
pm.collectionVariables.set("userid_collect","2");
//local variable
pm.variables.set("userid_local","2");
```



2. Environmental variable will only create when we select existing environment. If we dont select Environmental variable will not be created.
3. In the Variables Section we can see variables got created through script also.
4. Creating variables at requests level through JS is recommended than folder or collection level.

Note

➔ If we don't know JavaScript we can use **Snippets**

- **Click on ›_ → use existing snippet Snippet**

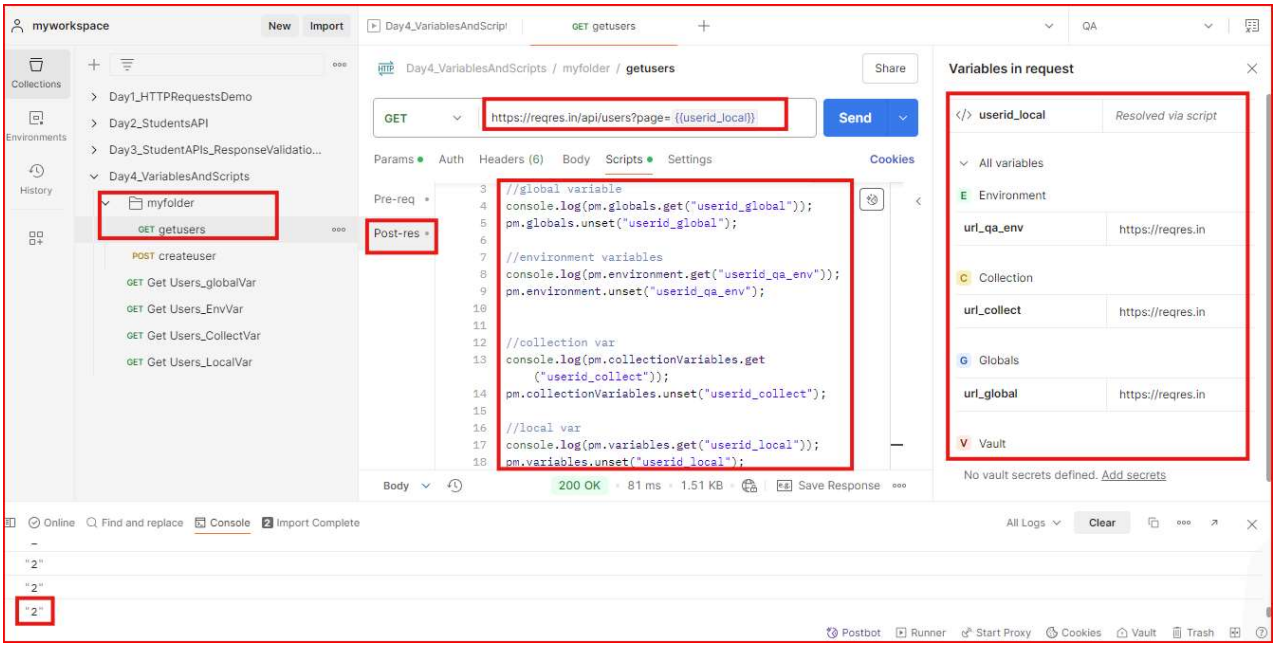
Retrieving and Removing Variables in Post-response Scripts

1. **Retrieve a Variable Value**
   a. console.log(pm.globals.get("userid_global"));
   b. console.log(pm.environment.get("userid_qa_env"));
   c. console.log(pm.collectionVariables.get("userid_collect"));
   d. console.log(pm.variables.get("userid_local"));
2. **Unset (Remove) a Variable** → After getting the response
   a. pm.globals.unset("userid_global");
   b. pm.environment.unset("userid_qa_env");
   c. pm.collectionVariables.unset("userid_collect");
   d. pm.variables.unset("userid_local");

Post-response script in request

```
//global variable
console.log(pm.globals.get("userid_global"));
pm.globals.unset("userid_global");
//environment variables
console.log(pm.environment.get("userid_qa_env"));
pm.environment.unset("userid_qa_env");
```

```
//collection var
console.log(pm.collectionVariables.get("userid_collect"));
pm.collectionVariables.unset("userid_collect");
//local var
console.log(pm.variables.get("userid_local"));
pm.variables.unset("userid_local");
```



Observation

- ➔ Upon using unset variables got delete from variable section also.
- ➔ **console.log()** is used for printing output in the Console

Using Variables Inside the Request Body

- ➔ We can set and use variables in the request body to make dynamic requests.
- ➔ **Create Local Variables in Pre-request Script**
  - pm.variables.set("name", "Madhan Mohan");
  - pm.variables.set("job", "Senior Associate Consultant and Trainer");
- ➔ **POST Request Example**
  - URL: https://reqres.in/api/users
  - **Request Body**

```
{
  "name": "{{name}}",
  "job": "{{job}}"
}
```

➔ Here, {{name}} and {{job}} will be replaced with the values "Madhan Mohan" and "Senior Associate Consultant and Trainer" dynamically.

Workout in Postman Tool

1. Create a **createuser** post request in same Day4_Variables collection
    a. **https://reqres.in/api/users**
    b. Click on Pre-request and create variables
        i. `pm.variables.set("myname","Madhan Mohan");`
        ii. `pm.variables.set("myjob","Senior Associate Consultant and Trainer");`
    c. Click on Body → raw → select JSON from dropdown → create JSON body

```
{
  "name": "{{myname}}",
  "job": "{{myjob}}"
}
```

    d. Now Send the request and we can can see the response as below

    e. Click on Post-response and add assertions and managing variables script before sending the request.

Post-response script in request

```
pm.variables.unset("myname");

pm.variables.unset("myjob");

// Test to check if the status code is 200

pm.test("Status code is 201", () => {

 pm.response.to.have.status(201);

});

// Test to verify the status code name contains "Created"

pm.test("Status code name has string", () => {
```

```
 pm.response.to.have.status("Created");
});
// Parse the JSON response
const jsonData = pm.response.json();
pm.test("Values of fields in response", () => {
  pm.expect(jsonData.name).to.eql("Madhan Mohan");
  pm.expect(jsonData.job).to.eql("Senior Associate Consultant and Trainer");
});
```