

Software Installation:

- Postman Desktop application
- Installation of Nodejs
- Newman CLI – Nodejs library
 - ✓ `npm install -g newman`
 - ✓ `htmlReport`
 - ✓ `junit report`

Newman Execution Command:

```
newman run  
  collectionFile.json  
  -e environmentFile.json  
  -g globals.postman_globals.json  
  --export-globals ./globals.postman_globals.json  
  -r htmlextra  
  --reporter-htmlextra-export ./TestReport.html
```

Documentation & Snapshot of collection for quick review:

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Employee_Collection" is selected, and the "Get all Employee Details" request is highlighted. The request is a GET method to the endpoint `((app_url))/employees`. The "Tests" tab is active, showing a JavaScript script that collects the response body, sets a global variable for the first employee ID, and logs it. The "Documentation" panel on the right shows the URL `https://dummy.restapiexample.com/api/v1/employees` and lists different types of variables (Environment, Global, Collection) and validation options (Status Validation, Attributes validations, For Loop, Storing Data).

```
7 //collecting the response body in json
8 var resp = pm.response.json().data;
9
10 // get first record from response data attribute
11 var firstEmpId = resp[0].id
12
13 //Showcasing different types of variable
14 pm.collectionVariables.set("firstEmpId", firstEmpId);
15 pm.globals.set("firstEmpId", firstEmpId)
16 pm.environment.set("firstEmpId", firstEmpId)
17
18 // Looping through the response to verify attributes
19 for(ele in resp)
20 {
21     //printing all the employee ids from the response
22     console.log(ele.id)
23 }
24
25
26 setTimeout(function(){} ,5000);
27
28
29
```

Click Send to get a response

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Employee_Collection" is selected, and the "Create Employee" request is highlighted. The request is a POST method to the endpoint `((app_url))/create`. The "Pre-req" tab is active, showing a JavaScript script that sets a timeout, generates a random name, salary, and age, and sets them as collection variables. The "Documentation" panel on the right shows the URL `https://dummy.restapiexample.com/api/v1/create` and lists various features across request body and prerequest usage, including request body, prerequest to updating the request body, and JavaScript functions (string - replacing the special characters, math - generating random numbers). It also mentions exporting the created record as part of globals and file post executing from newman.

```
1 setTimeout(function(){} ,10000)
2
3 let timestamps = new Date().toString();
4 timestamps = timestamps.replace(/^[a-zA-Z]/g, '_');
5
6 randomName = "randomName" + " " + timestamps
7 randomSalary = Math.floor(Math.random() * 5000) + 1000;
8 randomAge = Math.floor(Math.random() * 100) + 1;
9
10 pm.collectionVariables.set("randomName", randomName)
11 pm.collectionVariables.set("randomSalary", randomSalary)
12 pm.collectionVariables.set("randomAge", randomAge)
13
```

Click Send to get a response

Postman interface showing a GET request to `Employee_Collection / Verify Created Employee Record`. The URL is `{{(app_url)}}/employee/{{(createdEmpId)}}`. The request is configured with the following tests:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Verify record is found", function () {
6   //expect exact match
7   pm.expect(pm.response.json().status).equals("success")
8
9   //partial test assertion
10  pm.expect(pm.response.json().message).contains("Record has been fetched.")
11
12  //null comparison
13  pm.expect(pm.response.json().data).to.be.null
14 });
15
```

The response section shows a placeholder image of an astronaut and the text "Click Send to get a response".

Documentation on the right shows the URL `https://dummy.restapiexample.com/api/v1/employee/` and the title "Showcasing various types of chai assertions on response verification:".

- equals : Exact Match
- contains : Partial match
- to.be.null : Null comparison

Postman interface showing a PUT request to `Employee_Collection / Update the Created Record`. The URL is `{{(app_url)}}/update/{{(createdEmpId)}}`. The request is configured with the following pre-request script:

```
1 setTimeout(function() {{,20000}}
2
3 randomSalary = Math.floor(Math.random() * 5000) + 1000;
4
5 pm.collectionVariables.set("randomSalary", randomSalary)
6
7
```

The response section shows a placeholder image of an astronaut and the text "Click Send to get a response".

Documentation on the right shows the URL `https://dummy.restapiexample.com/api/v1/update/` and the title "Showcasing different request method - PUT".

- Updating the existing record
- using pre-request section for dynamic record update

Body raw (json) shows the following JSON:

```
{
  "name": "test",
  "salary": "",
  "age": "23"
}
```

HomeWorkspacesAPI NetworkExplore

Search Postman

InviteUpgrade

My Workspace

NewImport

GET Get all Employee DetailsGET Verify Employee Record u

Environment

Collections

Employee_Collection

GET Get all Employee DetailsPOST Create EmployeeGET Verify Created Employee RecordPUT Update the Created RecordGET Verify Employee Record updatedDEL Delete the created EmployeeGET Verify Deleted Employee Record

Employee_Collection / Verify Employee Record updated

Save

Send


GET{{app_url}}/employees

ParamsAuthHeaders (6)BodyPre-reqTestsSettingsCookies

1

setTimeout(function() {}, 25000)

Response



Click Send to get a response

Documentation

https://dummy.restapiexample.com/api/v1/employees

Showcasing validation on updated records

- Usage of collection variables
- Looping through response
- applying conditional statement to find the record
- comparing the attributes value

OnlineFind and replaceConsole

RunnerCapture requestsCookiesTrash