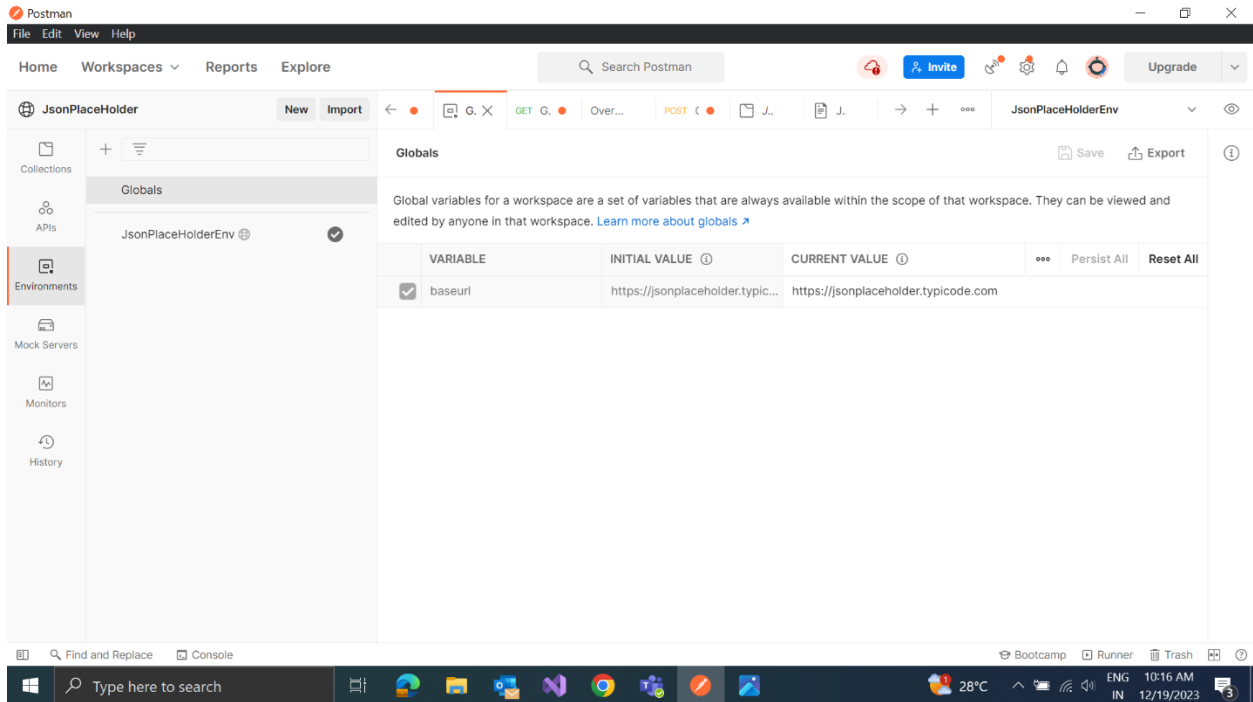
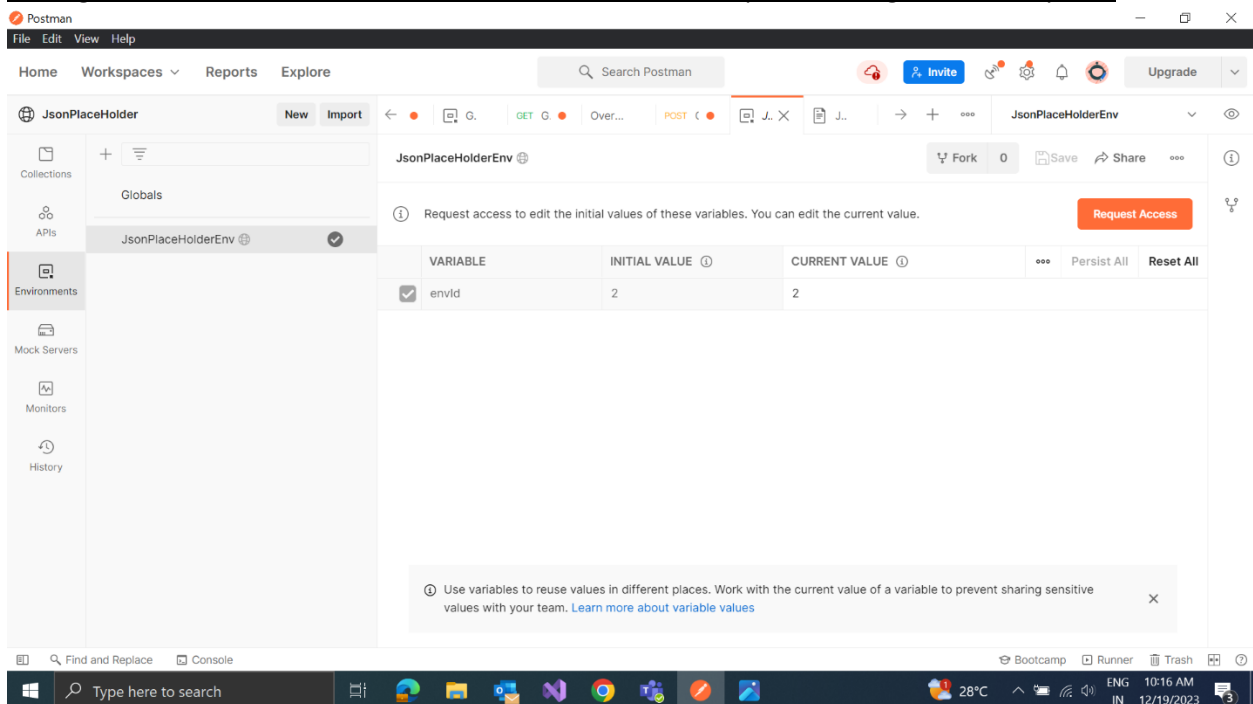


JsonPlaceholder Postman API Tests

Setting baseUrl as global variable which can be accessed across the workspace



Setting envId as an environment variable to store the id to be passed along with the requests



GET:

GetPosts Request with baseUrl from global variables

The screenshot shows the Postman interface with a workspace named 'JsonPlaceholder'. The left sidebar lists collections, APIs, environments, mock servers, monitors, and history. The main panel displays a GET request to 'JsonPlaceholderApi / GetPosts' with the URL '({baseUrl})/posts'. The 'Tests' tab is active, showing a JavaScript test script:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Data is not empty", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData).to.not.empty;
7 });
8 pm.test("First post user id is 1", function () {
```

The 'Test Results' section shows four passed tests:

- PASS Status code is 200
- PASS Data is not empty
- PASS First post user id is 1
- PASS First post id is 1

The status bar at the bottom indicates a 200 OK response, 233 ms execution time, and 27.98 KB of data.

GetPostById request with envId from environment variables

The screenshot shows the Postman interface with a workspace named 'JsonPlaceholder'. The left sidebar lists collections, APIs, environments, mock servers, monitors, and history. The main panel displays a GET request to 'JsonPlaceholderApi / GetPostById' with the URL '({baseUrl})/posts/({envId})'. The 'Tests' tab is active, showing a JavaScript test script:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Returned post id is correct", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.id).to.eql(Number(pm.environment.get("envId")));
7 });
```

The 'Test Results' section shows two passed tests:

- PASS Status code is 200
- PASS Returned post id is correct

The status bar at the bottom indicates a 200 OK response, 1251 ms execution time, and 1.38 KB of data.

POST:

CreatePost request

The screenshot shows the Postman interface for a POST request to `JsonPlaceholderApi / CreatePost`. The request is successful, returning a 201 status code. The response body is a JSON object representing a new post. The Tests tab shows four passing tests: "Status code is 201", "User Id should be present", "Title should not be empty", and "Body should be new body".

```
1 var jsonData = pm.response.json();
2 pm.test("Status code is 201", function () {
3   pm.response.to.have.status(201);
4 });
5 pm.test("User Id should be present", function(){
6   pm.expect(jsonData.id).to.not.null;
7 })
```

Test Results (4/4):

- PASS Post created
- PASS User Id should be present
- PASS Title should not be empty
- PASS Body should be new body

CreatePost request is sent with a Json body containing new post details

The screenshot shows the Postman interface for a POST request to `JsonPlaceholderApi / CreatePost`. The request is configured with a JSON body containing the following details:

```
1 {
2   "userId": 100,
3   "title": "new post",
4   "body": "new body"
5 }
```

The Tests tab shows four passing tests: "Status code is 201", "User Id should be present", "Title should not be empty", and "Body should be new body".

Test Results (4/4):

- PASS Post created
- PASS User Id should be present
- PASS Title should not be empty
- PASS Body should be new body

PUT:

UpdatePost request using envId

The screenshot shows the Postman interface with a PUT request to `JsonPlaceholderApi / UpdatePost` at the endpoint `{{baseurl}}/posts/{{envId}}`. The request is configured with the following details:

- Method:** PUT
- URL:** `{{baseurl}}/posts/{{envId}}`
- Params:** None
- Authorization:** None
- Headers (9):** None
- Body:** None
- Pre-request Script:** None
- Tests:** A JavaScript test script is provided:

```
1 var jsonData = pm.response.json();
2 pm.test("Status code is 200", function () {
3   pm.response.to.have.status(200);
4 });
5 pm.test("Id is correct", function () {
6   pm.expect(jsonData.id).to.eql(Number(pm.environment.get("envId")));
7 });
8 pm.test("Title updated", function () {
```
- Settings:** None

The response is a 200 OK status with a response time of 507 ms and a size of 1.17 KB. The test results show four passed tests:

- PASS: Post created
- PASS: Id is correct
- PASS: Title updated
- PASS: Post Id correct

UpdatePost request is sent with json body containing updated post details

The screenshot shows the Postman interface with a PUT request to `JsonPlaceholderApi / UpdatePost` at the endpoint `{{baseurl}}/posts/{{envId}}`. The request is configured with the following details:

- Method:** PUT
- URL:** `{{baseurl}}/posts/{{envId}}`
- Params:** None
- Authorization:** None
- Headers (9):** None
- Body:** A JSON body is provided:

```
1 {
2   "userId": 2,
3   "title": "updated title",
4   "body": "updated body"
5 }
```
- Pre-request Script:** None
- Tests:** None
- Settings:** None

The response is a 200 OK status with a response time of 507 ms and a size of 1.17 KB. The test results show four passed tests:

- PASS: Post created
- PASS: Id is correct
- PASS: Title updated
- PASS: Post Id correct

DELETE:

DeletePost request with envId

The screenshot displays the Postman interface for a DELETE request. The left sidebar shows the 'JsonPlaceholder' collection with the 'DeletePost' endpoint selected. The main panel shows the request details for 'DeletePost' with the URL 'DELETE {{baseurl}}/posts/{{envId}}'. The 'Tests' tab is active, showing two test scripts: 'Status code is 200' and 'Body is correct'. The 'Test Results' tab shows two passed tests: 'PASS Status code is 200' and 'PASS Body is correct'. The status bar at the bottom indicates a 200 OK response with 1015 ms execution time and 1.06 KB body size.

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

JsonPlaceholder

New Import

JsonPlaceholderApi / DeletePost

DELETE {{baseurl}}/posts/{{envId}}

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Body is correct", function () {
5   pm.response.to.have.body("{}");
6 });
```

Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)

SNIPPETS

[Get an environment variable](#)

[Get a global variable](#)

Body Cookies Headers (23) Test Results (2/2)

All Passed Skipped Failed

PASS Status code is 200

PASS Body is correct

200 OK 1015 ms 1.06 KB Save Response

Find and Replace Console

Type here to search

Bootcamp Runner Trash

28°C 10:15 AM 12/19/2023