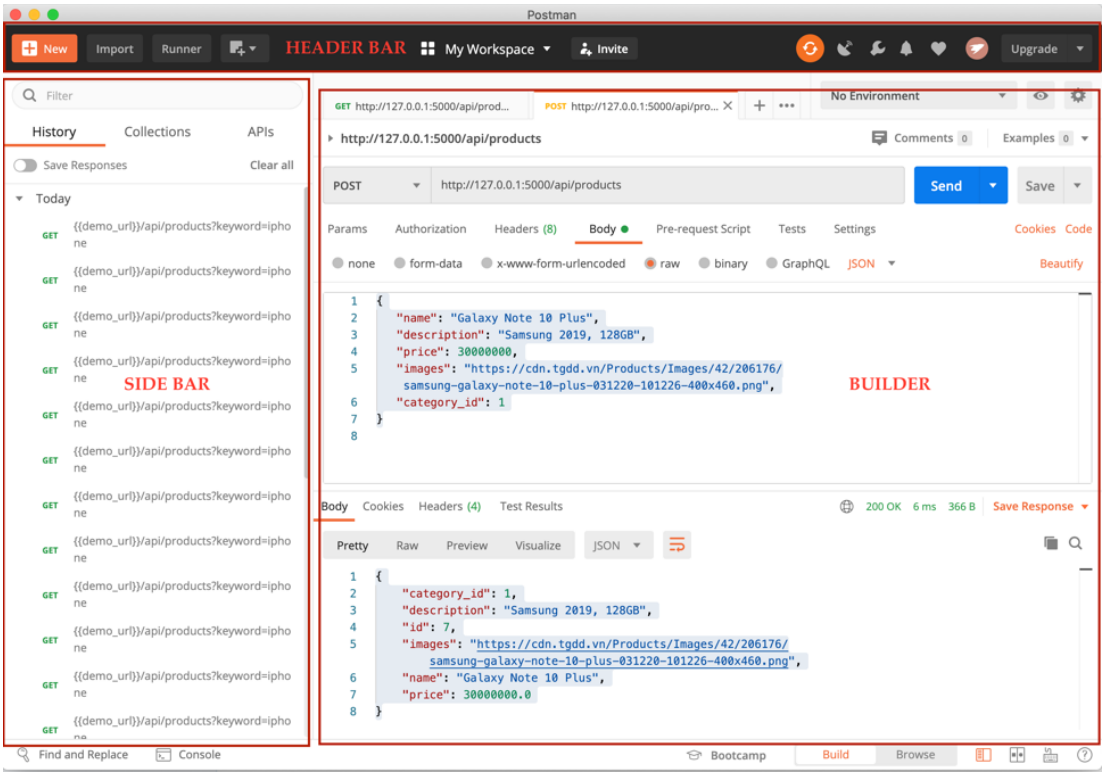
SỬ DỤNG POSTMAN

Postman là một công cụ phổ biến được sử dụng rộng rãi trong trong kiểm thử API. Postman hỗ trợ trên nhiều nền tảng hệ điều hành khác nhau như Mac, Windows (32-bit/64-bit), Linux (32-bit/64-bit). Truy cập https://www.postman.com/downloads/ để tải phiên bản mới nhất (tài liệu này sử dụng phiên bản 7.29.1) và cài đặt postman.



Header bar chứa các nút chức năng để tạo và nhập (Import) các request và collection, truy cập collection runner, di chuyển hoặc mời cộng tác workspace, mở các thiết lập cho postman, v.v.

Slide bar cung cấp các truy cập vào History, Collections và APIs. Trong đó History chứa các request đã được thực hiện trước đó, Collections chứa danh sách các collection, collection chứa nhiều request đã thực hiện và APIs để thiết kế các API trực tiếp trong postman sử dụng API Builder.

Builder xây dựng và làm việc với các request, một số thông tin quan trọng của vùng này:

- HTTP Request: chọn phương thức thực hiện request từ danh sách như GET, POST, PUT, DELETE.
- Request URL: chỉ định URL liên kết API sẽ tương tác.
- Params: chỉ định các tham số cho request.
- Authorization: thiết lập quyền truy cập API.

- Headers: thiết lập các thông tin header cho request, mỗi thiết lập dưới dạng key/value.
- Body: chứa thông tin chi tiết của request.

Làm việc với GET request

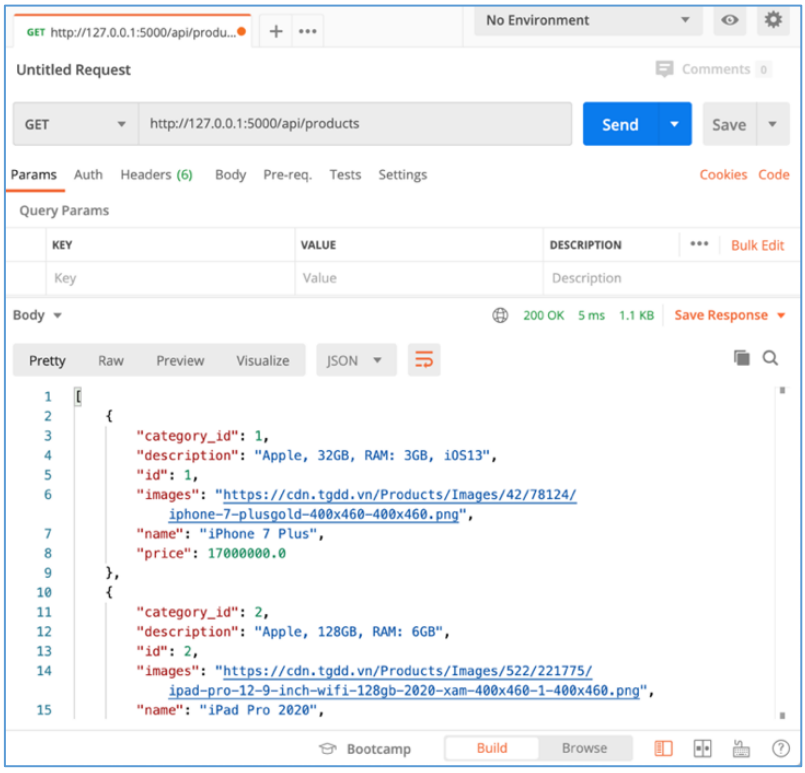
Giả sử ta có một API: /api/products để lấy danh sách các sản phẩm của một hệ thống bán hàng trực tuyến bằng phương thức GET. API này được phép nhận tham số get param để lọc dữ liệu sản phẩm như sau /api/products?keyword=iphone

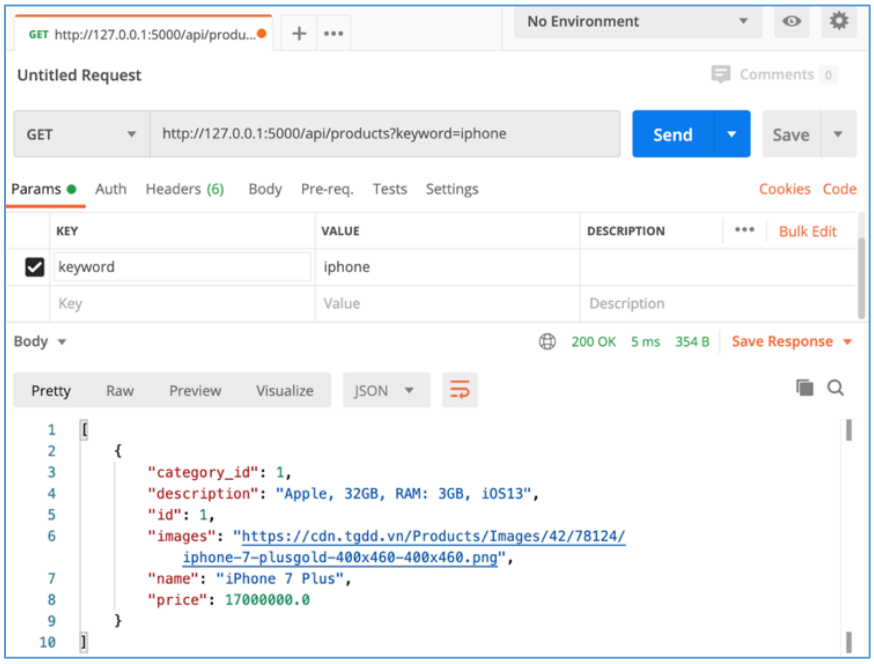
Dữ liệu response của API này có dạng:

```
[{
   "id": <int>,
   "name": <string>,
   "description": <string>,
   "price": <float>,
   "images": <string>,
   "category_id": <int>
}]
```

Các bước thực thi API

- Thiết lập HTTP request là GET.
- Nhập vào endpoint của API tại thanh URL.
- Nhập giá trị tham số keyword tại tab Params nếu có.
- Bấm nút Send





Làm việc với POST request

Giả sử ta có API thêm sản phẩm: /api/products

Phương thức sử dụng POST

Dữ liệu gửi lên có định dạng json như sau:

```
{
   "name": <string>,
   "description": <string>,
   "price": <float>,
   "images": <string>,
   "category_id": <int>
}
```

Dữ liệu response của API:

- Nếu API thực thi thêm sản phẩm thành công sẽ trả về sản phẩm được thêm vào

```
{
    "id": <int>,
    "category_id": <int>,
    "description": <string>,
    "images": <string>,
    "name": <string>,
    "price": <float>
}
```

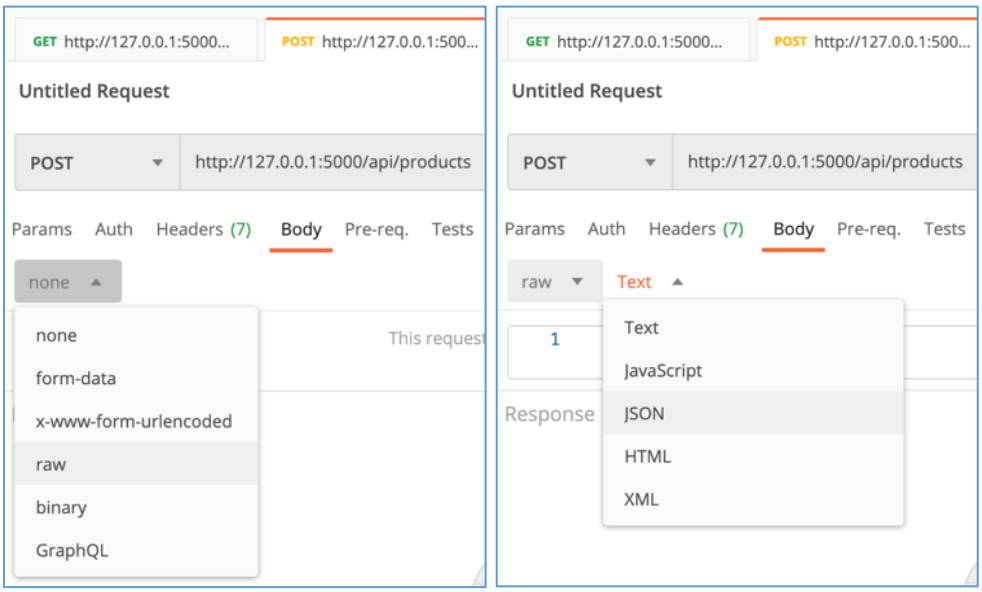
Nếu API có lỗi sẽ gửi về một thông điệp lỗi

```
{
   "error_message": string
}
```

Các bước thực hiện

- Chọn phương thức HTTP request là POST.
- Nhập địa chỉ endpoint của API vào ô URL.

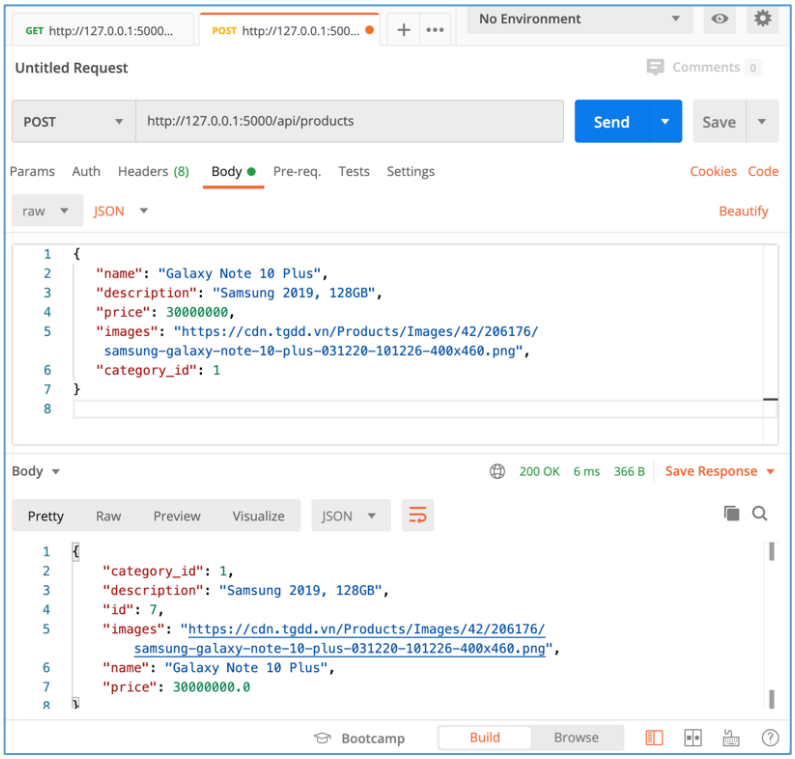
- Nhập dữ liệu gửi lên server tại tab Body, chọn raw >> json.



- Nhập dữ liệu cho body như sau:

```
{
   "name": "Galaxy Note 10 Plus",
   "description": "Samsung 2019, 128GB",
   "price": 30000000,
   "images": "/images/galaxy-note-10-plus.png",
   "category_id": 1
}
```

- Bấm nút Send.

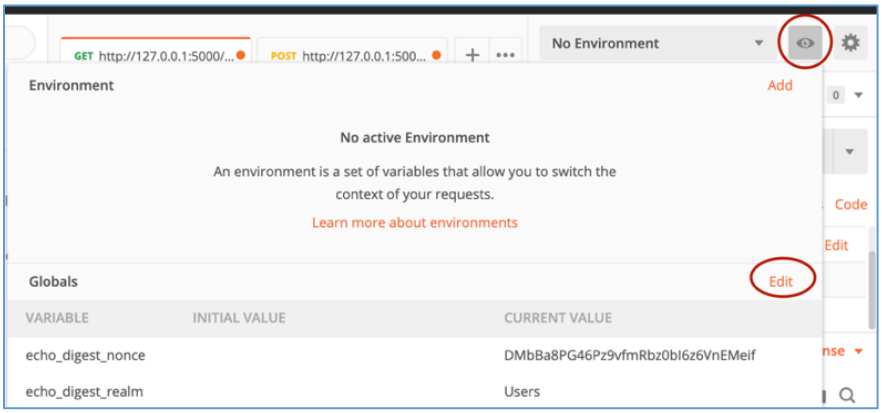


Thiết lập biến môi trường trong postman

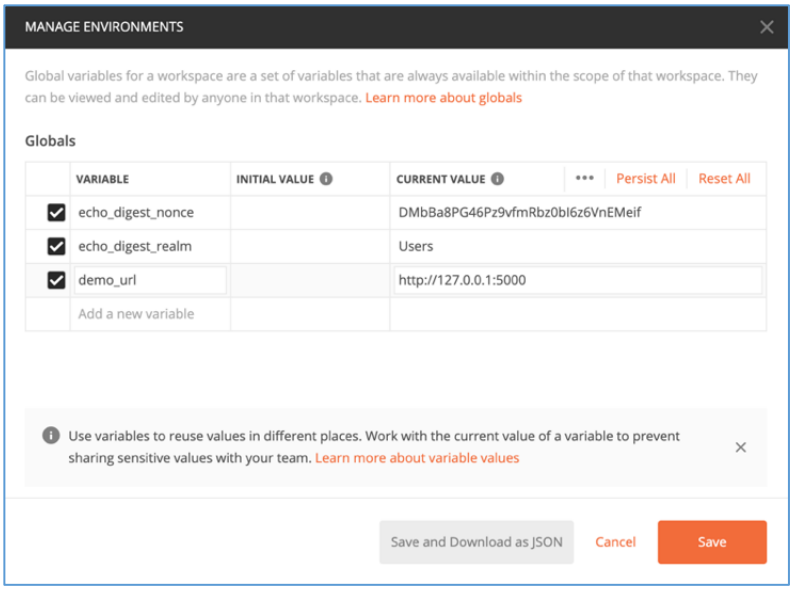
Đây là một trong các tính năng hữu ích của postman cho các biến dữ liệu giống nhau để có thể tái sử dụng nhiều chỗ. Việc thiết lập chúng thành các biến dùng chung tránh tình trạng lặp lại trong quá trình kiểm thử, cũng như giúp dễ dàng thay đổi giá trị của chúng khi thực hiện kiểm thử các giá trị khác nhau.

Các bước thiết lập:

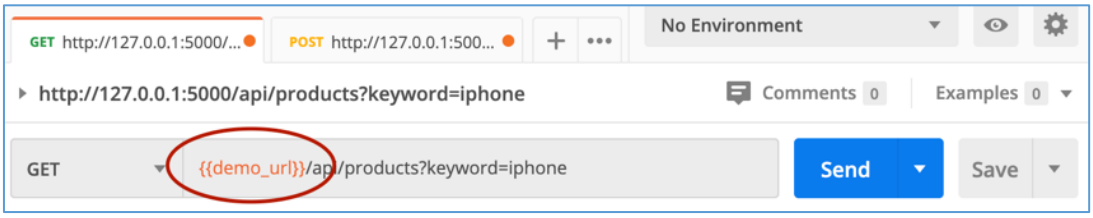
- Click biểu tượng mắt ở trên phải. Ở vùng Globals, click liên kết Edit.



- Bổ sung biến demo_url, sau đó đóng cửa sổ.



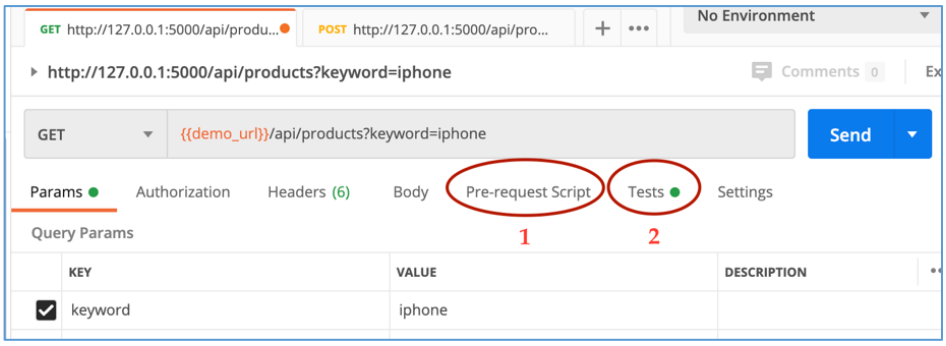
- Sử dụng biến demo_url theo cú pháp {{demo_url}}.



Viết test script trong postman

Postman cung cấp cơ chế thực thi dựa trên NodeJS cho phép thêm các xử lý vào request và collection. Cơ chế này giúp ta có thể viết test suite, tạo các request chứa các tham số động, truyền dữ liệu giữa các request, v.v. Ta có thể thêm mã nguồn Javascript để thực thi trong 2 sự kiện:

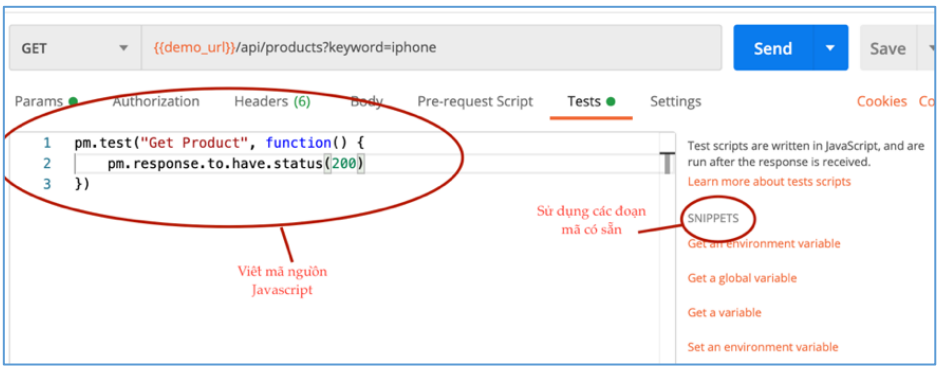
- Script được thực thi trước khi request được gửi lên server, sự kiện này được đặt tại tab Pre-request Script.
- Script được thực thi sau khi request đã hoàn thành và nhận được response, sự kiện này được đặt tại tab Tests.



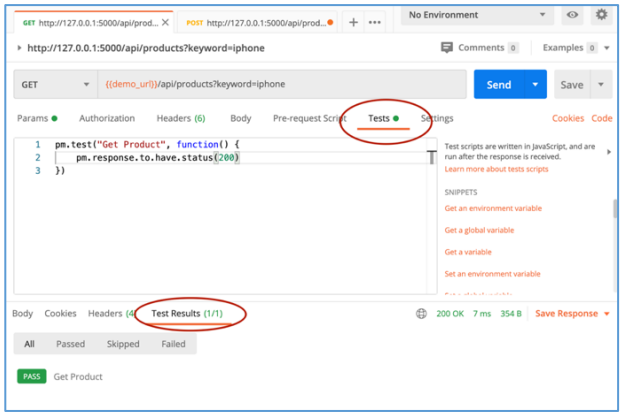
Trong các trình bày sau tập trung vào test script được thực thi sau request, cách viết test script thực thi trước request tương tự.

Test có thể được thêm vào request, thư mục hoặc các collection. Test script liên kết trong request sẽ được thực thi sau mỗi request, test script liên kết trong collection sẽ được thực thi sau mỗi request trong collection. Tương tự, một test script kết hợp trong folder sẽ được thực thi sau mỗi request trong folder.

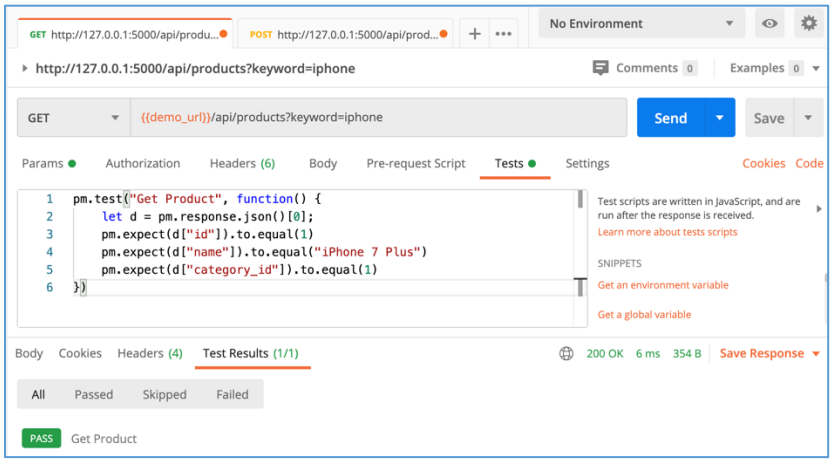
Để thêm test cho một request, ta mở request và nhập mã nguồn Javascript vào tab Tests hoặc sử dụng một số đoạn mã nguồn có sẵn trong vùng Snippets, các test sẽ được thực thi sau khi request chạy xong, kết quả sẽ hiển thị trong tab Test Results.



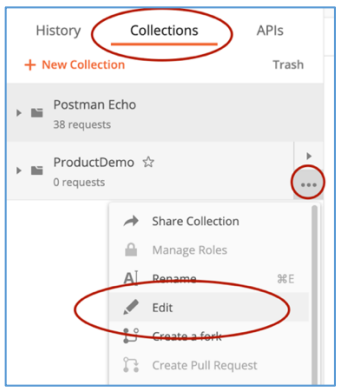
Để kiểm tra dữ liệu trả về sử dụng đối tượng pm.response, định nghĩa các test sử dụng hàm pm.test() gồm hai tham số là tên và một hàm trả về giá trị luận lý cho biết test case đó là pass hay fail.



Sử dụng hàm pm.expect() để kiểm tra kết quả mong muốn.



Để thêm test script vào collection, ta click vào tab Collections, và tới collection muốn thêm test để edit collection đó như sau:



Tại tab Tests, viết test script cho collection đó.

Một số thao tác thông dụng với test script

Thiết lập và lấy giá trị biến môi trường

```
pm.environment.set("variable_key", "variable_value")
pm.environment.get("variable_key")
```

Xoá biến môi trường

```
pm.environment.unset("variable_key")
```

Thiết lập và lấy giá trị biến collection

```
pm.collectionVariables.set("variable_name",
                           "variable_value")

pm.collectionVariables.get("variable_name")
```

Xoá biến collection

```
pm.collectionVariables.unset("variable_name")
```

Thiết lập và lấy giá trị biến toàn cục

```
pm.globals.set("variable_key", "variable_value")
pm.globals.get("variable_key")
```

Xoá biến toàn cục

```
pm.globals.unset("variable_key")
```

Lấy giá trị biến có trong globals hoặc một môi trường có hiệu lực

```
pm.variables.get("variable_key")
```

Xử lý response

- Kiểm tra response body bằng một chuỗi

```
pm.test("Response Handling", function() {
    pm.response.to.have.body("str")
})
```

- Kiểm tra response body chứa một chuỗi nào đó

```
pm.test("Response Handling", function() {
   pm.expect(pm.response.text()).to.include("str")
})
```

- Kiểm tra dữ liệu trả về

```
pm.test("Response Handling", function() {
    var d = pm.response.json()
    pm.expect(d.value).to.equal(1)
})
```

- Kiểm tra header có thuộc tính Content-Type không.

```
pm.test("Response Handling", function() {
    pm.response.to.have.header("Content-Type")
})
```

- Kiểm tra thời gian phản hồi dưới 200 mili giây

```
pm.test("Response Handling", function() {
    pm.expect(pm.response.responseTime)
      .to.be.below(200)
})
```

- Kiểm tra status code của POST request là thành công

```
pm.test("Response Handling", function() {
    pm.expect(pm.response.code)
      .to.be.oneOf([200, 201, 202])
})
```

Gửi request bất đồng bộ (chỉ dùng trong pre-request test script)

```
pm.sendRequest("https://postman-echo.com/get",
               function (err, response) {
    console.log(response.json());
});
```

Một số phương thức assertion thông dụng khác

- Kiểm tra kiểu dữ liệu

```
pm.expect("postman").to.be.a("string")
pm.expect({a: 1}).to.be.an("object")
pm.expect(undefined).to.be.an("undefined")
```

- Kiểm tra chuỗi rỗng hoặc mảng rỗng

```
pm.expect([]).to.be.empty
pm.expect("").to.be.empty
```

- Kiểm tra kết quả chứa/không chứa key nào đó

```
pm.expect({a: 1, b: 2}).to.have.all.keys("a", "b")
pm.expect({a: 1, b: 2}).to.have.any.keys("a")
pm.expect({a: 1, b: 2}).to.not.any.keys("c", "d")
```

- Kiểm tra chiều dài của chuỗi hoặc mảng

```
pm.expect("foo").to.have.lengthOf(3)
pm.expect([2, 1, 3]).to.have.lengthOf(3)
```

- Kiểm tra hai mảng có các phần tử giống nhau

```
pm.expect([1, 2, 3]).to.have.members([3, 1, 2])
```

- Kiểm tra kết quả chứa một phần tử nào đó

```
pm.expect([1, 2, 3]).to.include(2)
pm.expect({a: 1, b: 2}).to.include({a: 1})
```