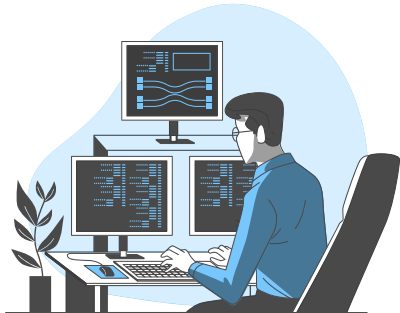
Advanced

JS

Khóa học Backend

Bài 12: Javascript nâng cao (Tiết 2)



Nội dung

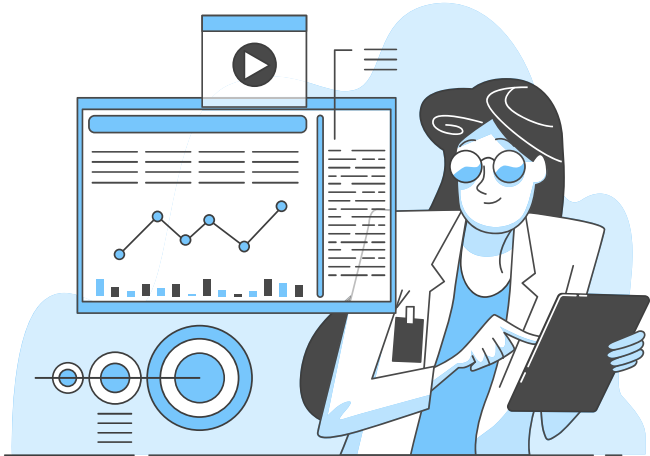
01 Callbacks

02 Fetch API

03 JSON server và Postman

04 Promise

05 Async/Await



01

Callbacks

01. Callbacks

- **Hàm callback** (gọi lại) là một hàm được truyền vào **đối số** của **một hàm khác**.
- Hàm callback có thể được chạy sau khi những chức năng khác kết thúc.

02

Promise

02. Promise

- **Promise** dùng để **giải quyết** vấn đề **callback hell**.
- **Callback hell** là có **nhiều hàm lồng nhau** gây ra khó chịu.
- **Promise** có cách **viết đơn giản và dễ nhìn hơn** so với **callback**.


```
3
4  a(function (resultsFromA) {
5      b(resultsFromA, function (resultsFromB) {
6          c(resultsFromB, function (resultsFromC) {
7              d(resultsFromC, function (resultsFromD) {
8                  e(resultsFromD, function (resultsFromE) {
9                      f(resultsFromE, function (resultsFromF) {
10                         console.log(resultsFromF);
11                     })
12                 })
13             })
14         })
15     })
16 });
17
```

02. Promise

- Cú pháp:

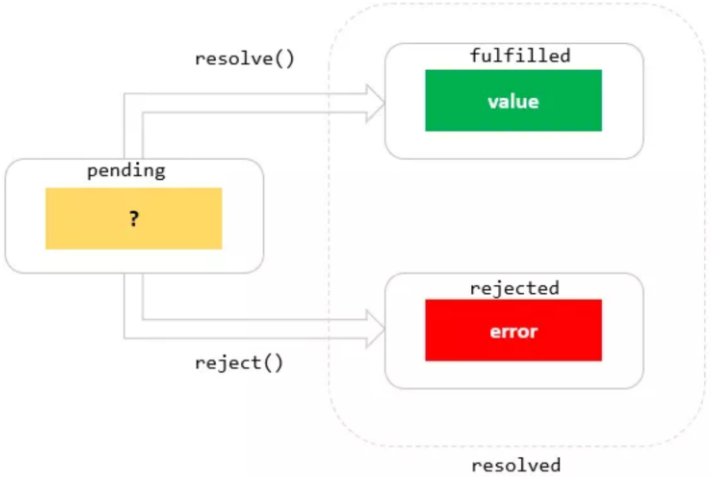
```
var promise = new Promise((resolve, reject) => {
  // resolve(): Nếu thành công chạy vào hàm này
  // reject(): Nếu thất bại chạy vào hàm này
});

promise
  .then((success) => {
    // Nếu thành công chạy vào đây
  })
  .catch((error) => {
    // Nếu thất bại chạy vào đây
  })
  .finally(() => {
    // Luôn luôn chạy vào đây
  })
```

- Trong đó:
  - **new Promise**: khởi tạo promise.
  - **resolve**: là một hàm callback xử lý cho hành động thành công.
  - **reject**: là một hàm callback xử lý cho hành động thất bại.
  - **.then**: Thành công chạy vào đây.
  - **.catch**: Thất bại chạy vào đây.
  - **.finally**: Luôn luôn chạy vào đây.

02. Promise

- Promise có **3 trạng thái**:
  - **Pending**:
    - Khi promise **đang chạy** thì sẽ ở trạng thái này.
    - Kết quả là undefined.
  - **Fulfilled**:
    - Khi promise **đã chạy xong** thì sẽ ở trạng thái này.
    - Kết quả là một giá trị.
  - **Rejected**:
    - Khi promise **bị lỗi** thì sẽ ở trạng thái này.
    - Kết quả là một object lỗi.



03

Fetch API

03. Fetch API

- **Phương thức Fetch** dùng để **gọi lên** trên **server** thông qua một API để **lấy dữ liệu** từ trên server trả về.
- **Api** là **một url** để cho phép bên **Front-end** có thể **giao tiếp** được với bên **Back-end**.
- Cú pháp:

```
fetch('http://example.com/movies.json')
  .then((response) => {
    return response.json();
  })
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });
```

- Trong đó:
  - **fetch()** dùng để **gửi yêu cầu** lên server thông qua api.
  - **then()** được thực thi khi **có phản hồi** từ server trả về.
  - **catch()** được thực thi khi **không có phản** hồi từ máy chủ.

04

Async/Await

04. Async/Await

- **Async/Await** là một tính năng của JavaScript giúp chúng ta **làm việc với các hàm bất đồng bộ** theo cách dễ hiểu hơn.
- Nó được xây dựng trên Promise.
- **Async**: Khai báo một hàm bất đồng bộ.
  - Tự động biến đổi một hàm thông thường thành một Promise.
  - Từ khóa Async được đặt trước 1 hàm.
- **Await**: Tạm dừng việc thực hiện các hàm async.
  - Khi được đặt trước một Promise, nó sẽ đợi cho đến khi Promise kết thúc và trả về kết quả.
  - Await chỉ có thể được sử dụng bên trong các hàm async.

05

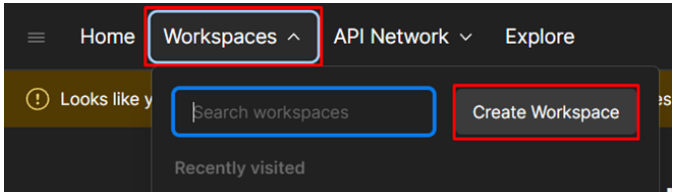
JSON server và Postman

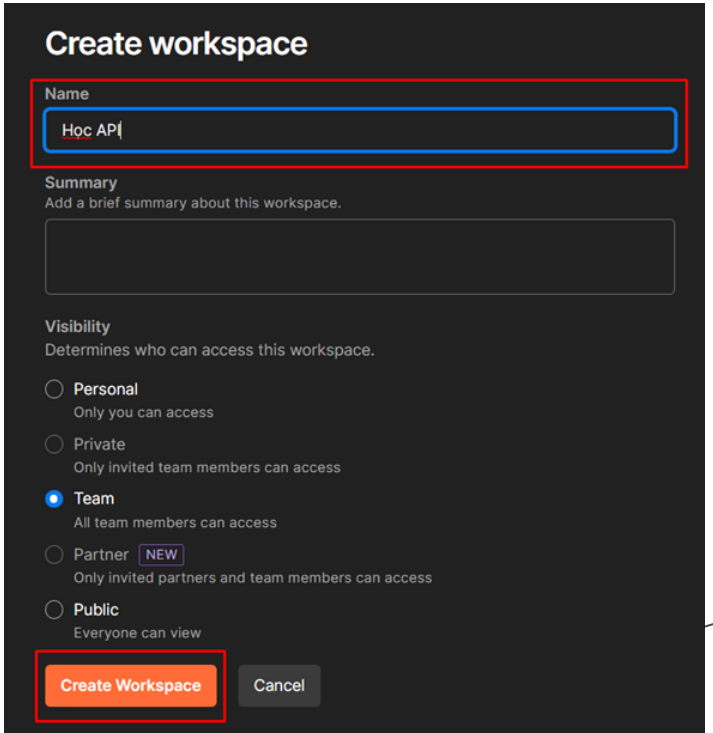
5.1. JSON server

- Khái niệm: **Json server** là một server để **fake API** và **trả chuỗi JSON**.
- Yêu cầu: **Cài đặt nodejs** vào máy tính, gõ **npm -v** để xem phiên bản.
- Hướng dẫn cấu hình:
  - **Bước 1**: Gõ lệnh **npm init** để khởi tạo file package.json.
  - **Bước 2**: Gõ lệnh **npm i json-server** để cài.
  - **Bước 3**: Tạo 1 file **database.json**.
  - **Bước 4**: Thêm vào mục script trong package.json dòng lệnh: **"start": "json-server --watch database.json"**.
  - **Bước 5**: Gõ lệnh **npm start** để chạy.

5.2. Postman

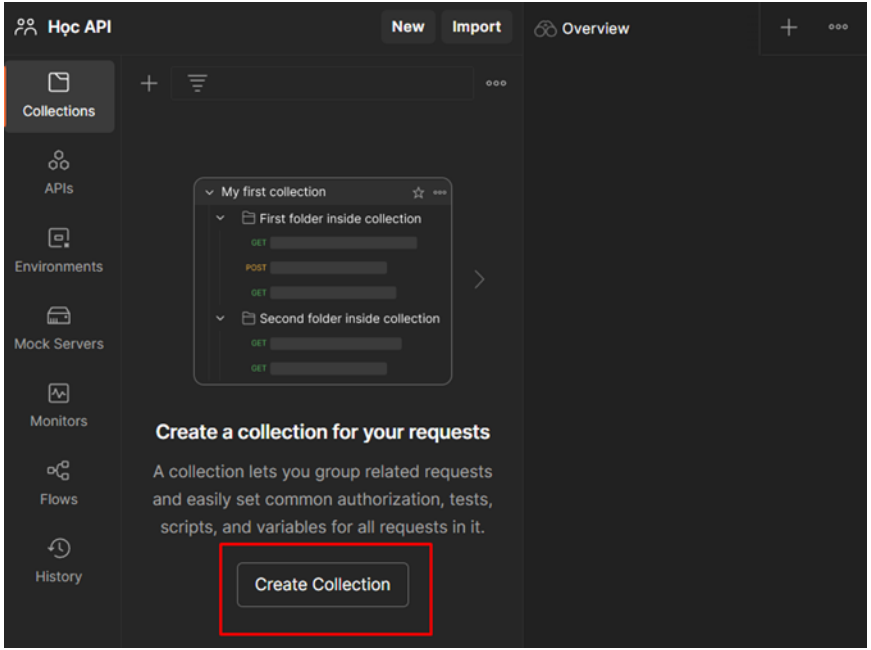
- Các bước cài đặt và sử dụng Postman:
  - **Bước 1**: Tải Postman và đăng ký tài khoản.
  - **Bước 2**: Chọn Create Workspace.

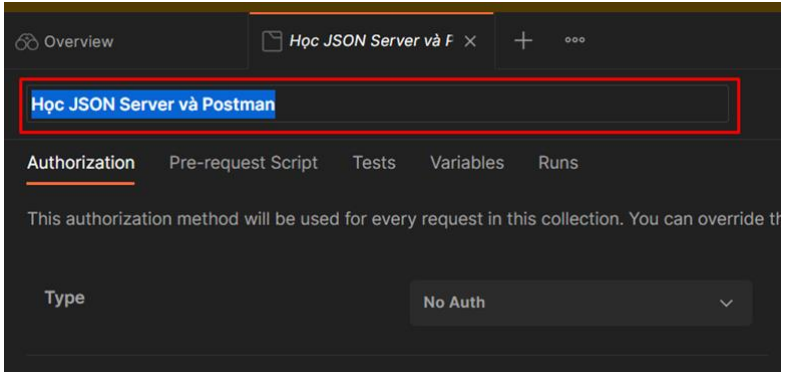




5.2. Postman

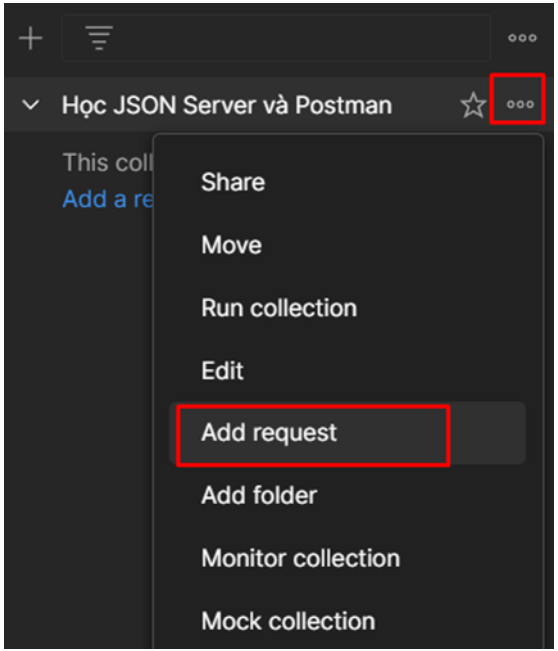
- Các bước cài đặt và sử dụng Postman:
  - **Bước 3**: Chọn Create Collection và đặt tên cho Collection.

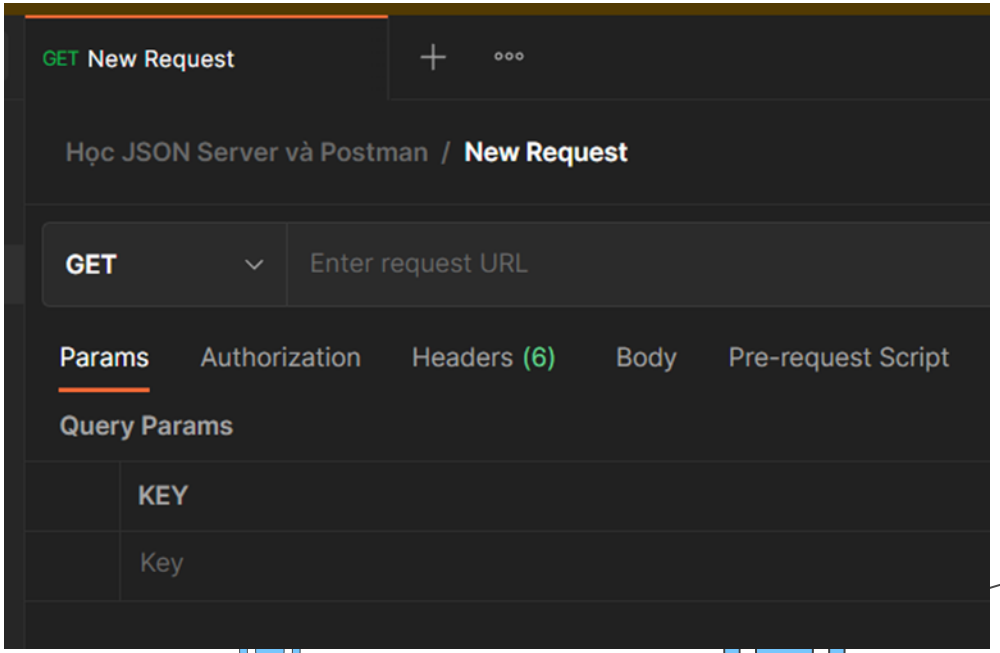




5.2. Postman

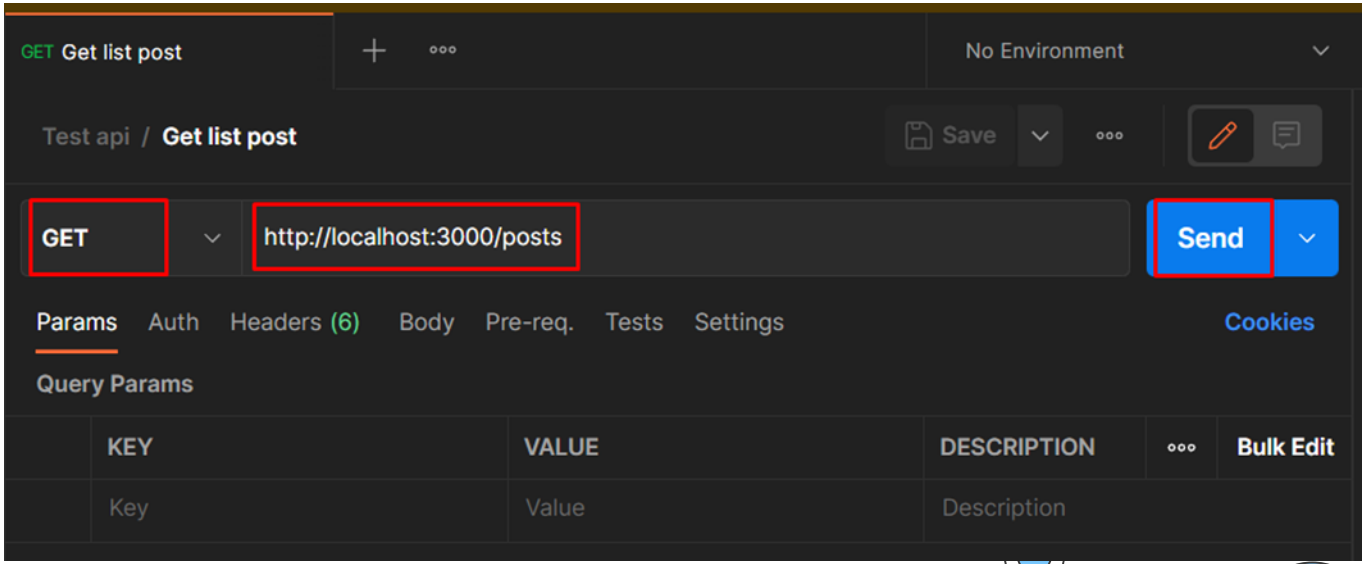
- Các bước cài đặt và sử dụng Postman:
  - **Bước 4**: Tạo mới một Request.





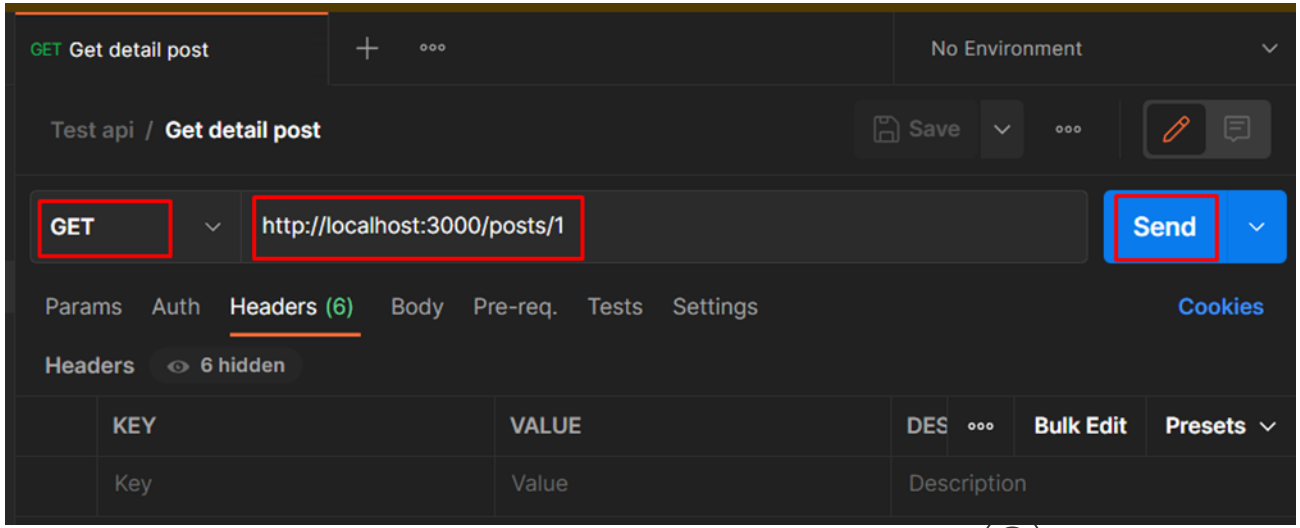
5.2. Postman

- Phương thức **GET**:
  - **Để lấy** một hoặc nhiều bản ghi.
  - Ví dụ 1: Lấy ra danh sách các bài viết.



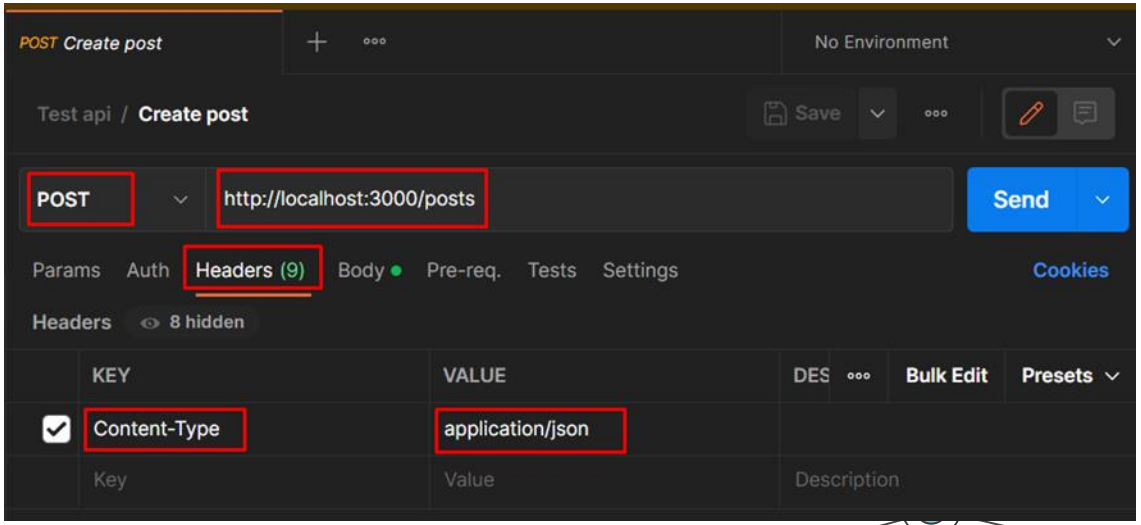
5.2. Postman

- Phương thức **GET**:
  - **Để lấy** một hoặc nhiều bản ghi.
  - Ví dụ 2: Lấy ra một bài viết.



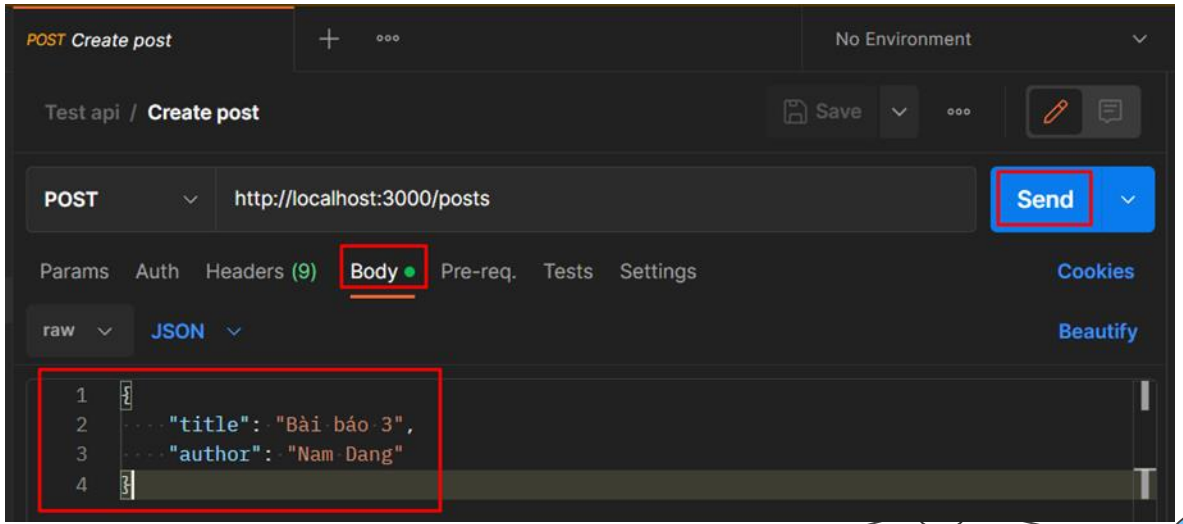
5.2. Postman

- Phương thức **POST**:
  - Để **tạo mới** một bản ghi.
  - Ví dụ: Tạo mới một bài viết. (Trường hợp data gửi lên dạng json thì headers phải thêm Content-Type là application/json).



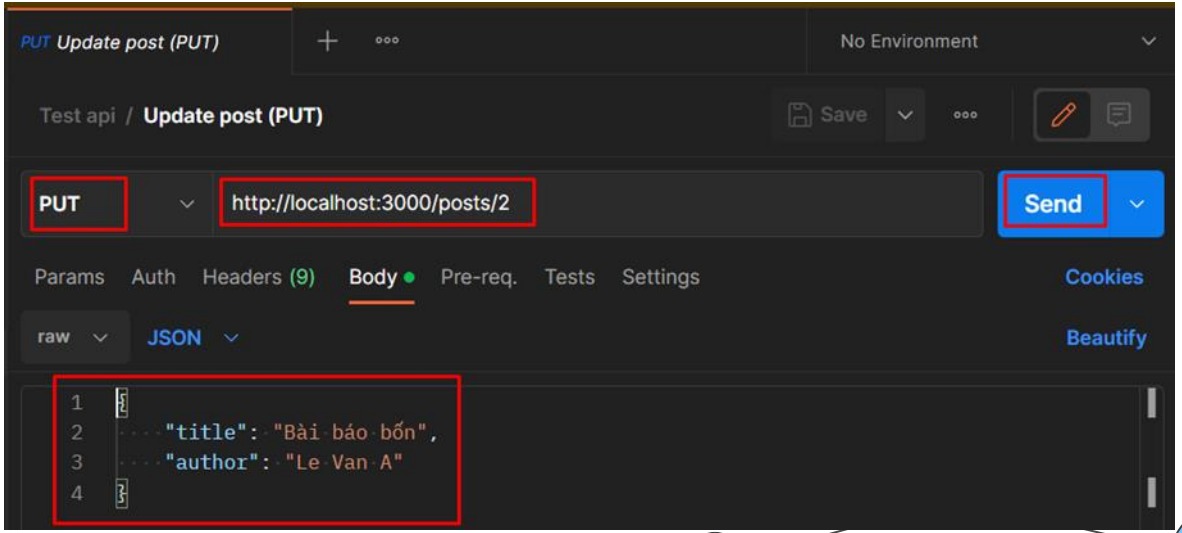
5.2. Postman

- Phương thức **POST**:
  - Để **tạo mới** một bản ghi.
  - Ví dụ: Tạo mới một bài viết. (Trường hợp data gửi lên dạng json thì headers phải thêm Content-Type là application/json).



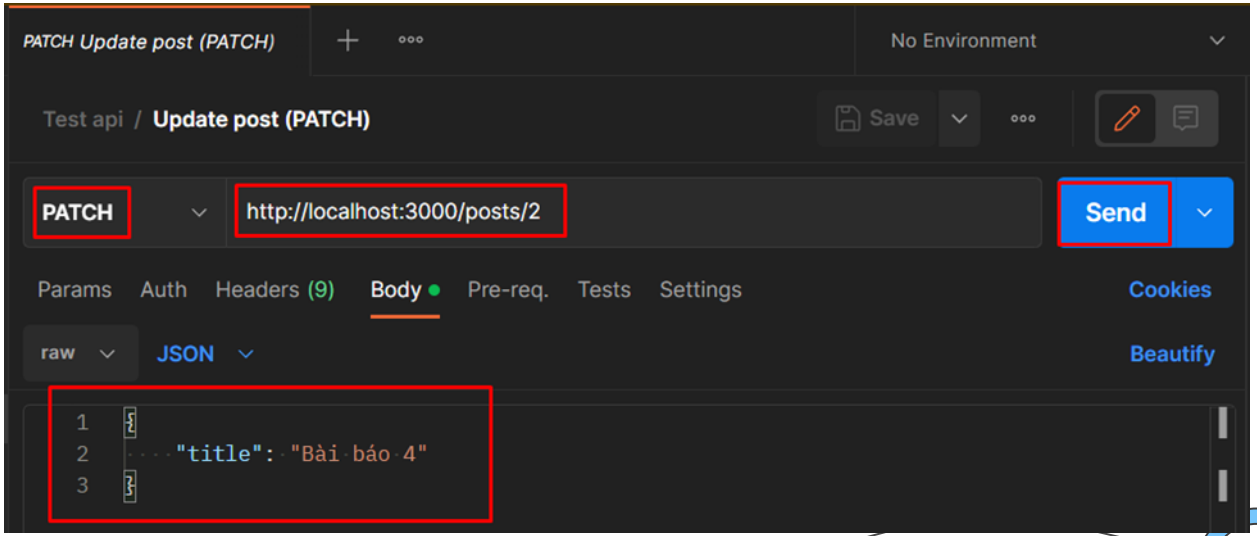
5.2. Postman

- Phương thức **PUT**:
  - Để **cập nhật** một bản ghi.
  - Nhưng phải gửi lên **đủ các cặp key/value** (kể cả những cặp key/value không cần cập nhật).
  - Ví dụ: Cập nhật một bài viết.



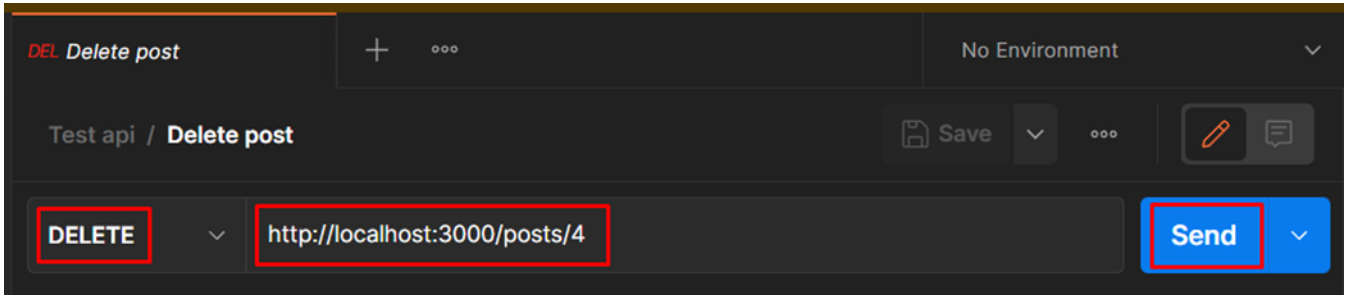
5.2. Postman

- Phương thức **PATCH**:
  - Để **cập nhật** một bản ghi.
  - **Chỉ cần** gửi lên các **cặp key/value cần cập nhật**.
  - Ví dụ: Cập nhật một bài viết.



5.2. Postman

- Phương thức **DELETE**:
  - Để **xóa** một bản ghi.
  - Ví dụ: Xóa một bài viết.



Bài tập

- **Link bài tập**: https://dacavn.notion.site/B-i-t-p-b-i-12-Javascript-n-ng-cao-Ti-t-2-ae7a858b29754d57b160e108af595a01?pvs=4