**I. Giới thiệu về REST API trong WordPress**

**1. REST API là gì?**

WordPress REST API là một phương thức giao tiếp giữa các ứng dụng và WordPress thông qua HTTP protocol. Nó cho phép các nhà phát triển tương tác với WordPress từ bên ngoài và thực hiện các hoạt động như đăng bài, chỉnh sửa nội dung, cập nhật thông tin người dùng và nhiều hơn nữa.

REST API cho phép các nhà phát triển sử dụng các ngôn ngữ lập trình khác nhau, như PHP, Python, Ruby, Java, C#, JavaScript, để tương tác với WordPress thông qua các yêu cầu HTTP. Điều này làm cho việc phát triển ứng dụng và tích hợp WordPress với các ứng dụng khác trở nên dễ dàng hơn bao giờ hết.

Các lệnh của REST API được thiết kế để truy cập vào các tài nguyên khác nhau của WordPress, chẳng hạn như danh sách bài viết, danh sách người dùng, danh sách danh mục, và nhiều hơn nữa. Các lệnh được định dạng theo chuẩn RESTful API và có thể được thực thi bằng cách gửi các yêu cầu HTTP như GET, POST, PUT và DELETE đến WordPress.

Với WordPress REST API, các nhà phát triển có thể tạo ra các ứng dụng di động, ứng dụng máy tính để bàn, các trang web tĩnh và động, và các sản phẩm khác, đồng thời tăng tính đa dạng và linh hoạt cho ứng dụng của mình.

**API là gì?**

API là viết tắt của cụm từ "Application Programming Interface", được dùng để mô tả các giao diện (interfaces) và quy tắc mà các ứng dụng phần mềm có thể sử dụng để tương tác với nhau. Một API cung cấp cho các nhà phát triển phương tiện để tương tác với một hệ thống hoặc một ứng dụng khác mà không cần phải biết chi tiết cài đặt bên trong của nó.

Trong lập trình, một API thường đóng vai trò như một cầu nối giữa các ứng dụng khác nhau, cho phép chúng giao tiếp và trao đổi dữ liệu một cách dễ dàng và hiệu quả. API thường được thiết kế để cung cấp cho các nhà phát triển các chức năng, tính năng hoặc dịch vụ cụ thể mà họ có thể sử dụng trong ứng dụng của mình, mà không cần phải viết lại các chức năng đó từ đầu.

Có nhiều loại API khác nhau, bao gồm API web (như RESTful API), API desktop, API mobile và nhiều hơn nữa. Mỗi loại API được thiết kế để tương tác với các loại ứng dụng khác nhau và có thể có định dạng và quy tắc khác nhau.

**REST ​​là gì?**

REST là viết tắt của "Representational State Transfer", là một kiến trúc phần mềm được sử dụng trong việc thiết kế các API (Application Programming Interface). RESTful API được thiết kế để cung cấp các giao diện lập trình ứng dụng cho phép các ứng dụng khác nhau tương tác và trao đổi dữ liệu với nhau thông qua HTTP protocol.

RESTful API là một kiểu của Web API, trong đó các yêu cầu HTTP được sử dụng để thực hiện các hoạt động cơ bản như đọc (GET), tạo (POST), cập nhật (PUT) hoặc xóa (DELETE) các tài nguyên trên server. RESTful API được thiết kế để tương thích với các tiêu chuẩn web như HTML, JSON hoặc XML, giúp dữ liệu có thể được truyền tải một cách đáng tin cậy và hiệu quả.

Các ứng dụng sử dụng RESTful API thường được thiết kế để truyền tải dữ liệu theo các luồng không đồng bộ và có thể hoạt động trên nhiều nền tảng khác nhau, bao gồm cả trên desktop, web, mobile và các thiết bị IoT. RESTful API đang trở thành một tiêu chuẩn trong việc xây dựng các ứng dụng phức tạp, đặc biệt là các ứng dụng mạng xã hội, các trang web thương mại điện tử và các ứng dụng di động.

**2. Tại sao WordPress hỗ trợ REST API?**

Việc tích hợp REST API vào WordPress nhằm mở rộng khả năng của nền tảng này, giúp các nhà phát triển dễ dàng kết nối WordPress với các trang web và ứng dụng khác. Điều này tạo điều kiện cho việc xây dựng các ứng dụng di động, dịch vụ bên ngoài và nhiều ứng dụng khác, tăng tính linh hoạt và mở rộng của WordPress.

**3. Lịch sử phát triển và các phiên bản của REST API trong WordPress**

REST API được tích hợp vào lõi của WordPress từ phiên bản 4.7, ra mắt vào tháng 12 năm 2016. Trước đó, REST API tồn tại dưới dạng một plugin riêng biệt. Việc tích hợp này đánh dấu một bước tiến quan trọng, cho phép các nhà phát triển tương tác với dữ liệu WordPress một cách linh hoạt và hiệu quả hơn.

**II. Cấu trúc và cách hoạt động của REST API trong WordPress**

**1. Routes and Endpoints**

**Endpoint:** là các hàm có sẵn thông qua API. Chúng thực hiện các chức năng cụ thể như truy xuất chỉ mục API, cập nhật một bài viết hoặc xóa một bình luận. Mỗi endpoint nhận một số tham số và trả về dữ liệu cho client.​

**Route:** là "tên" được sử dụng trong URL để truy cập các endpoint. Một route có thể liên kết với nhiều endpoint, và endpoint nào được sử dụng sẽ phụ thuộc vào phương thức HTTP.​

Ví dụ, với URL http://example.com/wp-json/wp/v2/posts/123:​

"Route" là wp/v2/posts/123 – Route không bao gồm wp-json vì wp-json là đường dẫn cơ sở cho chính API.​

Route này có 3 endpoint:

**GET**: kích hoạt phương thức get\_item, trả về dữ liệu bài viết cho client.​

**PUT:** kích hoạt phương thức update\_item, nhận dữ liệu để cập nhật và trả về dữ liệu bài viết đã được cập nhật.​

**DELETE:** kích hoạt phương thức delete\_item, trả về dữ liệu bài viết đã bị xóa cho client.​

**Tạo Endpoint**

Nếu nhà phát triển muốn tạo một endpoint trả về cụm từ **"Hello World, this is the WordPress REST API"** khi nhận được một yêu cầu **GET**, trước tiên cần đăng ký route cho endpoint đó.

Để đăng ký route, nhà phát triển nên sử dụng hàm register\_rest\_route(). Hàm này cần được gọi trong hook hành động rest\_api\_init.

register\_rest\_route() sẽ xử lý toàn bộ quá trình ánh xạ route tới endpoint.

Hãy thử tạo một route để trả về **"Hello World, this is the WordPress REST API"**.

/\*\*

\* This is our callback function that embeds our phrase in a WP\_REST\_Response

\*/

function prefix\_get\_endpoint\_phrase() {

// rest\_ensure\_response() wraps the data we want to return into a WP\_REST\_Response, and ensures it will be properly returned.

return rest\_ensure\_response( 'Hello World, this is the WordPress REST API' );

}

/\*\*

\* This function is where we register our routes for our example endpoint.

\*/

function prefix\_register\_example\_routes() {

// register\_rest\_route() handles more arguments but we are going to stick to the basics for now.

register\_rest\_route( 'hello-world/v1', '/phrase', array(

// By using this constant we ensure that when the WP\_REST\_Server changes our readable endpoints will work as intended.

'methods' => WP\_REST\_Server::READABLE,

// Here we register our callback. The callback is fired when this endpoint is matched by the WP\_REST\_Server class.

'callback' => 'prefix\_get\_endpoint\_phrase',

) );

}

add\_action( 'rest\_api\_init', 'prefix\_register\_example\_routes' );

Đối số đầu tiên được truyền vào register\_rest\_route()là không gian tên, cung cấp cho chúng ta một cách để nhóm các tuyến đường của mình. Đối số thứ hai được truyền vào là đường dẫn tài nguyên hoặc cơ sở tài nguyên. Đối với ví dụ của chúng ta, tài nguyên mà chúng ta đang truy xuất là cụm từ "Hello World, this is the WordPress REST API". Đối số thứ ba là một mảng các tùy chọn. Chúng ta chỉ định phương thức nào mà điểm cuối có thể sử dụng và lệnh gọi lại nào sẽ xảy ra khi điểm cuối được khớp (có thể thực hiện nhiều việc hơn nhưng đây là những điều cơ bản).

Đối số thứ ba cũng cho phép chúng ta cung cấp lệnh gọi lại quyền, có thể hạn chế quyền truy cập vào điểm cuối chỉ cho một số người dùng nhất định. Đối số thứ ba cũng cung cấp cách đăng ký đối số cho điểm cuối để các yêu cầu có thể sửa đổi phản hồi của điểm cuối của chúng ta. Chúng ta sẽ tìm hiểu các khái niệm đó trong phần điểm cuối của hướng dẫn này.

Khi chúng ta vào `https://ourawesomesite.com/wp-json/hello-world/v1/phrase`, giờ chúng ta có thể thấy REST API chào đón chúng ta một cách tử tế. Hãy cùng xem xét các tuyến đường sâu hơn một chút.

**Routes (Tuyến đường)**

Trong REST API, **routes** (tuyến đường) được biểu diễn bằng các **URI**. Route chính là phần được gắn vào cuối https://ourawesomesite.com/wp-json.

Route chỉ mục của API là /, do đó khi truy cập https://ourawesomesite.com/wp-json/, ta sẽ nhận được toàn bộ thông tin về API có sẵn. Tất cả các route khác nên được xây dựng dựa trên route này. Phần wp-json có thể được thay đổi, nhưng thông thường nên giữ nguyên để tránh gây nhầm lẫn.

Đảm bảo route là duy nhất

Các route cần phải có tính **duy nhất** để tránh xung đột. Ví dụ, nếu muốn tạo một route cho **sách (books)**, ta có thể sử dụng /books. Route này sẽ có địa chỉ https://ourawesomesite.com/wp-json/books.

Tuy nhiên, đây không phải là một **cách làm tốt**, vì có thể xảy ra xung đột với các route khác trong API. Nếu một plugin khác cũng muốn đăng ký một route /books, sẽ xảy ra vấn đề do hai route bị trùng lặp và chỉ có một route có thể hoạt động.

Hàm register\_rest\_route() có tham số thứ tư là một giá trị boolean để xác định liệu route mới có **ghi đè** lên một route đã tồn tại hay không. Tuy nhiên, việc ghi đè không thực sự giải quyết được vấn đề, vì trong nhiều trường hợp, cả hai route đều quan trọng và phục vụ mục đích khác nhau.

Giải pháp: Sử dụng **namespace**

Để tránh xung đột, **namespace** được sử dụng để tổ chức các route theo từng nhóm riêng biệt. Ví dụ, thay vì sử dụng /books, chúng ta có thể đặt route là /myplugin/v1/books. Điều này giúp đảm bảo rằng các route không bị trùng lặp với các plugin hoặc API khác.

**2. Các phương thức HTTP được hỗ trợ (GET, POST, PUT, DELETE).**

Các phương thức HTTP (HTTP methods) đôi khi còn được gọi là động từ HTTP (HTTP verbs). Chúng là các cách khác nhau để giao tiếp qua giao thức HTTP. Các phương thức chính được sử dụng trong WordPress REST API bao gồm:

**GET**: Được sử dụng để lấy dữ liệu từ API.

**POST**: Được sử dụng để tạo mới tài nguyên (ví dụ: người dùng, bài viết, danh mục).

**PUT:** Được sử dụng để cập nhật tài nguyên.

**DELETE**: Được sử dụng để xóa tài nguyên.

**OPTIONS**: Được sử dụng để cung cấp thông tin về tài nguyên.

Xử lý trường hợp phương thức HTTP không được hỗ trợ

Không phải mọi client (máy khách) đều hỗ trợ đầy đủ các phương thức này, vì chúng được giới thiệu trong HTTP 1.1. Trong một số trường hợp, firewall có thể chặn các yêu cầu DELETE hoặc PUT, gây khó khăn khi thao tác dữ liệu.

Để khắc phục vấn đề này, WordPress REST API hỗ trợ cách thay thế bằng:

Tham số \_method trong URL.

Header X-HTTP-Method-Override trong yêu cầu.

Ví dụ: Nếu không thể gửi yêu cầu DELETE, nhà phát triển có thể sử dụng yêu cầu POST như sau:

POST https://ourawesomesite.com/wp-json/my-shop/v1/products/1?\_method=DELETE

Yêu cầu trên sẽ xóa sản phẩm số 1, ngay cả khi máy khách không hỗ trợ phương thức DELETE.

**3. Authentication và Authorization trong REST API.**

**Authentication**

***Xác thực bằng Cookie trong WordPress REST API***

**Xác thực bằng Cookie** là phương thức xác thực tiêu chuẩn được tích hợp sẵn trong WordPress. Khi người dùng đăng nhập vào bảng điều khiển WordPress, hệ thống sẽ tự động thiết lập cookie cần thiết, giúp các nhà phát triển plugin và theme chỉ cần kiểm tra trạng thái đăng nhập của người dùng.

Tuy nhiên, WordPress REST API sử dụng một cơ chế đặc biệt gọi là **nonce** để ngăn chặn các cuộc tấn công **CSRF (Cross-Site Request Forgery)**. Cơ chế này ngăn chặn các trang web bên ngoài thực hiện hành động trái phép thay mặt người dùng mà không có sự đồng ý rõ ràng. Vì vậy, khi làm việc với REST API, cần xử lý nonce một cách đặc biệt.

**Đối với nhà phát triển sử dụng API JavaScript có sẵn của WordPress**, nonce sẽ được xử lý tự động. Đây là cách được khuyến nghị để sử dụng API trong plugin và theme. Các mô hình dữ liệu tùy chỉnh có thể mở rộng từ wp.api.models.Base để đảm bảo nonce được gửi đúng cách trong các yêu cầu tùy chỉnh.

**Đối với các nhà phát triển gửi yêu cầu Ajax thủ công**, nonce phải được truyền trong mỗi yêu cầu. REST API sử dụng nonce với hành động được đặt thành wp\_rest. Nonce có thể được truyền qua tham số dữ liệu \_wpnonce(trong dữ liệu POST hoặc chuỗi truy vấn của yêu cầu GET), hoặc thông qua tiêu đề X-WP-Nonce.

Nếu không có nonce hợp lệ, API sẽ đặt người dùng hiện tại thành 0, biến yêu cầu thành yêu cầu không xác thực, ngay cả khi người dùng đã đăng nhập vào WordPress.

**Lưu ý:** Trước đây, nhiều hệ thống không hỗ trợ tốt các yêu cầu DELETE. Ví dụ, PHP không chuyển đổi nội dung của yêu cầu DELETE thành một biến toàn cục ($\_POST). Vì vậy, phương pháp đáng tin cậy nhất là gửi nonce qua tiêu đề yêu cầu.

Do phương thức này dựa vào cookie của WordPress, nên nó chỉ có thể áp dụng khi sử dụng REST API trong WordPress và người dùng hiện tại đã đăng nhập. Ngoài ra, người dùng phải có quyền phù hợp để thực hiện hành động tương ứng.

**Ví dụ về sử dụng nonce trong WordPress REST API**

**Tạo nonce trong JavaScript**

WordPress tạo nonce thông qua đoạn mã PHP sau:

<?php

wp\_localize\_script( 'wp-api', 'wpApiSettings', array(

'root' => esc\_url\_raw( rest\_url() ),

'nonce' => wp\_create\_nonce( 'wp\_rest' )

) );

**Gửi nonce trong yêu cầu API**

Nonce này sau đó được sử dụng trong mô hình JavaScript gốc của WordPress:

options.beforeSend = function(xhr) {

xhr.setRequestHeader('X-WP-Nonce', wpApiSettings.nonce);

if (beforeSend) {

return beforeSend.apply(this, arguments);

}

};

**Sử dụng jQuery AJAX để chỉnh sửa tiêu đề bài viết**

Ví dụ sau đây minh họa cách thay đổi tiêu đề của một bài viết bằng jQuery AJAX:

$.ajax( {

url: wpApiSettings.root + 'wp/v2/posts/1',

method: 'POST',

beforeSend: function ( xhr ) {

xhr.setRequestHeader( 'X-WP-Nonce', wpApiSettings.nonce );

},

data:{

'title' : 'Hello Moon'

}

} ).done( function ( response ) {

console.log( response );

} );

***Xác thực cơ bản bằng Mật khẩu Ứng dụng (Application Passwords)***

Từ WordPress 5.6, hệ thống hỗ trợ **Application Passwords**, giúp xác thực bằng mật khẩu ứng dụng thay vì sử dụng nonce. Người dùng có thể tạo Application Passwords trong trang chỉnh sửa tài khoản người dùng (wp-admin -> Users -> Edit User).

Khi sử dụng Application Passwords, thông tin đăng nhập có thể được gửi qua HTTP Basic Authentication (RFC 7617). Ví dụ sử dụng cURL để lấy danh sách người dùng với quyền chỉnh sửa (context=edit):

curl --user "USERNAME:PASSWORD" https://HOSTNAME/wp-json/wp/v2/users?context=edit

**Plugin hỗ trợ xác thực trong WordPress**

Ngoài các phương thức xác thực có sẵn, bạn có thể cài đặt các plugin để hỗ trợ các phương thức xác thực khác, chẳng hạn như:

***OAuth 1.0a Server*** – Sử dụng giao thức OAuth 1.0a để xác thực an toàn.

***JSON Web Tokens (JWT)*** – Sử dụng token JWT để xác thực không trạng thái.

Các plugin này giúp hỗ trợ xác thực từ xa, đặc biệt hữu ích khi kết nối WordPress với các ứng dụng bên ngoài.

**Authorization (Ủy quyền)**

**Ủy quyền** là quá trình xác định quyền hạn của người dùng sau khi đã được xác thực, trả lời câu hỏi "Bạn được phép làm gì?". Trong WordPress, quyền hạn của người dùng được quản lý thông qua **roles** (vai trò) và **capabilities** (khả năng). Mỗi vai trò có một tập hợp các khả năng xác định những hành động mà người dùng có thể thực hiện.

Ví dụ:

Người dùng với vai trò **Administrator** có toàn quyền trên hệ thống.​

Người dùng với vai trò **Editor** có thể quản lý và xuất bản bài viết, nhưng không thể thay đổi cài đặt hệ thống.​

Người dùng với vai trò **Author** chỉ có thể viết và quản lý bài viết của riêng họ.​

Khi một yêu cầu được gửi đến WordPress REST API, hệ thống sẽ kiểm tra thông tin xác thực để xác định danh tính người dùng, sau đó kiểm tra quyền hạn để xác định xem người dùng có được phép thực hiện hành động đó hay không.​

Bảo mật API

Để bảo vệ API khỏi các truy cập trái phép, bạn có thể sử dụng các plugin bảo mật như **REST API Authentication and Security**. Plugin này cung cấp nhiều phương thức xác thực như API Key, JWT, Basic Auth và OAuth 2.0, giúp bảo vệ các endpoint của REST API khỏi truy cập công khai.

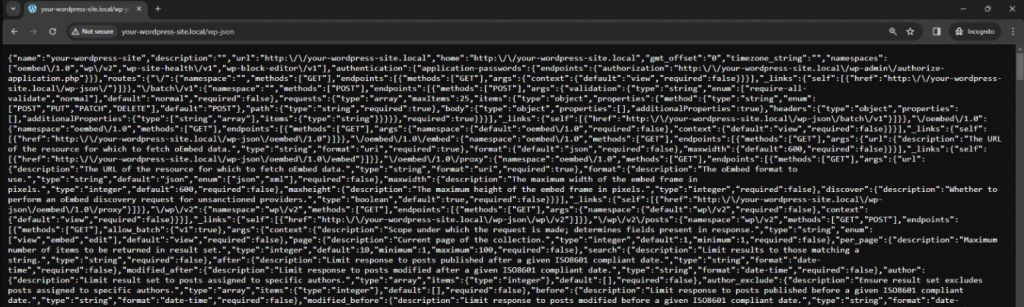
**III. Hướng dẫn sử dụng REST API trong WordPress**

**1. Cách kích hoạt và sử dụng REST API trong WordPress**

Chức năng REST API được kích hoạt mặc định trong WordPress. Sau khi mua gói hosting WordPress và cài đặt CMS, bạn có thể truy cập liên kết sau để kiểm tra xem REST API có đang hoạt động không:

http://domain.tld/wp-json/

*(Thay thế domain.tld bằng địa chỉ trang web của bạn.)*



Nếu bạn thấy danh sách các endpoint hiển thị, nghĩa là REST API đã được kích hoạt. Lưu ý rằng tính năng này chỉ khả dụng trên WordPress phiên bản 4.7 trở lên.

**Cài đặt môi trường kiểm thử**

**Cài đặt công cụ cURL**  
Trên dòng lệnh (Command Prompt hoặc Terminal) của máy tính, cài đặt **cURL** để gửi và nhận yêu cầu từ máy chủ WordPress REST API phục vụ mục đích kiểm thử.

**Cài đặt plugin Basic Auth cho WordPress REST API**  
Cài đặt plugin **WordPress REST API Basic Auth handler** để thiết lập phương thức xác thực, giúp truy cập các dữ liệu riêng tư yêu cầu quyền quản trị.

**Kết nối đến WordPress qua SSH bằng WP-CLI**

Mở **WP-CLI**.

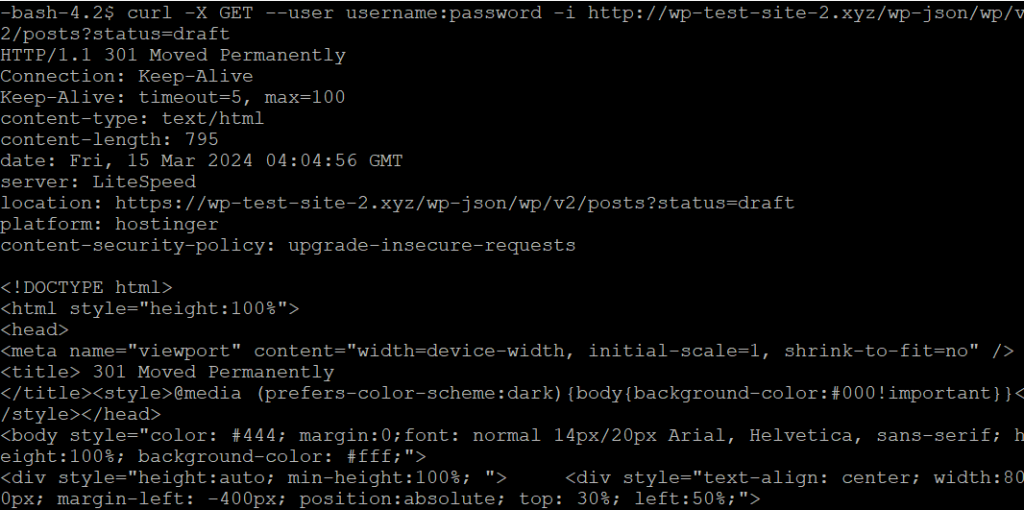
Kết nối SSH đến trang web của bạn.

Nếu sử dụng Hostinger, bạn có thể tìm thông tin đăng nhập SSH trong **hPanel → Quản lý Website → Nâng cao → Truy cập SSH**.

Sau khi thiết lập, hãy thử truy xuất dữ liệu giới hạn để kiểm tra xem xác thực REST API có hoạt động hay không.

Ví dụ: Sử dụng lệnh sau trong Terminal để lấy danh sách bài viết chưa được xuất bản (status=draft):

curl -X GET --user username:password -i http://domain.tld/wp-json/wp/v2/posts?status=draft



**Cách sử dụng WordPress REST API**

Bây giờ, hãy xem cách sử dụng WordPress REST API. Chúng tôi đã đề cập đến một số ví dụ ở trên, nhưng trong phần này, chúng tôi sẽ chỉ cho bạn chính xác cách truy cập và tìm nạp dữ liệu.

*Bước 1:* Truy cập REST API

Nếu bạn muốn tìm nạp dữ liệu từ một trang web WordPress trực tiếp, bạn có thể truy cập REST API trực tiếp từ trình duyệt của mình. Tất cả những gì bạn phải làm là nhập địa chỉ sau vào thanh tìm kiếm (thay thế bằng tên miền và phần mở rộng của riêng bạn):

mysite.com/wp-json/wp/v2

Điều này sẽ hiển si tệp JSON của trang web WordPress của bạn.

Sau đó, bạn có thể thêm các phần tử vào URL này để truy cập dữ liệu cụ thể, như chúng tôi sẽ chỉ cho bạn ở bước tiếp theo.

*Bước 2:* Thực hiện yêu cầu đến API REST

Như chúng tôi đã đề cập trước đó, bạn có thể sử dụng các điểm cuối để truy cập dữ liệu cụ thể trên trang web của mình. Nếu bạn muốn truy xuất danh sách tất cả các bài đăng của mình, chỉ cần thêm điểm cuối **/ bài đăng** vào địa chỉ của bạn:

mysite.com/wp-json/wp/v2/posts

Nếu bạn muốn truy xuất một bài đăng cụ thể, bạn chỉ cần thêm ID của nó (bạn sẽ tìm thấy điều này trên trang **Bài đăng** trong bảng điều khiển WordPress của mình):

mysite.com/wp-json/wp/v2/posts/4567

Trong khi đó, nếu bạn muốn tìm nạp dữ liệu về người dùng của mình, bạn sẽ sử dụng yêu cầu sau:

mysite.com/wp-json/wp/v2/users

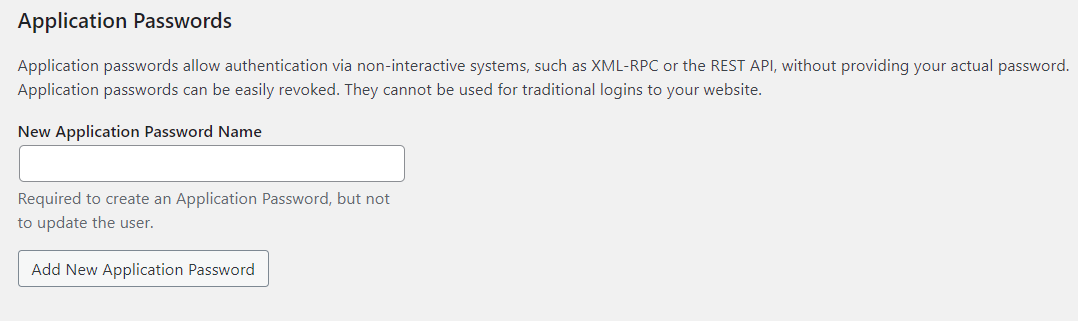
Đây chỉ là những ví dụ đơn giản, nhưng bạn có thể làm được rất nhiều điều. Ví dụ: bạn có thể tìm nạp các bài đăng có chứa các thuật ngữ cụ thể, thay đổi trạng thái của bài đăng từ "bản nháp" thành "xuất bản", v.v.

*Bước 3:* Xác thực yêu cầu của bạn

WordPress REST API cho phép bạn tìm nạp bất kỳ loại dữ liệu nào trên trang web của mình, nhưng không phải tất cả đều có sẵn công khai. Trong một số trường hợp, bạn sẽ cần xác thực yêu cầu của mình.

Để thực hiện các yêu cầu được xác thực đối với WordPress REST API, trước tiên bạn cần lấy khóa API. Để thực hiện việc này, hãy điều hướng đến **Người dùng → Hồ sơ** [trong bảng điều khiển WordPress của bạn](https://jetpack.com/blog/wordpress-admin-dashboard/).

Sau đó, cuộn xuống phần **Mật khẩu ứng dụng**. Tại đây, bạn sẽ cần nhập tên cho mật khẩu của mình (để tham khảo) và nhấp vào **Thêm mật khẩu ứng dụng mới**.



Điều này sẽ tạo khóa API cho bạn, bạn sẽ cần sử dụng khóa này trong các yêu cầu của mình. Ví dụ: nếu khóa API của bạn là “1234567890”, bạn có thể đưa nó vào tiêu đề **Ủy quyền** như thế này:

<https://mysite.com/wp-json/wp/v2/posts?Authorization=Bearer1234567890>

Hãy nhớ thay thế mã 1234567890 bằng khóa API bạn đã sao chép và xóa mọi khoảng trắng.

Bạn cũng có thể truy xuất danh sách các bài đăng được viết bởi một tác giả cụ thể, với ID người dùng của họ. Bạn có thể tìm ID của họ bằng cách vào phần **Người dùng** trong bảng điều khiển WordPress của bạn và nhấp vào tên tác giả. ID sẽ được hiển thị trong URL của trang tác giả của họ.

Giả sử tên của một tác giả là "Joe" và ID của họ là "123." Trong trường hợp này, bạn có thể sử dụng URL sau để truy xuất danh sách tất cả các bài đăng do Joe viết:

<https://mysite.com/wp-json/wp/v2/posts?author=123&Authorization=Bearer1234567890>

Nếu bạn không thể tìm thấy ID của người dùng, hồ sơ của họ có thể đã bị sửa đổi theo cách mà ID không còn hiển thị nữa. Trong trường hợp này, bạn có thể truy xuất danh sách tất cả các bài đăng do người dùng viết bằng tên đăng nhập hoặc địa chỉ email của họ thay vì ID.

Để thực hiện việc này, bạn sẽ phải sử dụng tham số "slug" thay vì tham số "author" trong yêu cầu của mình:

<https://mysite.com/wp-json/wp/v2/posts?slug=joe&Authorization=Bearer1234567890>

Tham số "slug" cho phép bạn chỉ định tên đăng nhập hoặc địa chỉ email của người dùng. WordPress REST API sẽ trả về danh sách tất cả các bài đăng do người dùng viết.

**2. Truy vấn dữ liệu từ WordPress thông qua REST API (ví dụ: lấy danh sách bài viết, thông tin người dùng).**

​Để truy vấn dữ liệu từ WordPress thông qua REST API, bạn có thể sử dụng các endpoint có sẵn để lấy thông tin như danh sách bài viết hoặc thông tin người dùng. Dưới đây là hướng dẫn chi tiết:​

**1. Lấy danh sách bài viết:** Để lấy danh sách các bài viết, bạn có thể gửi yêu cầu GET tới endpoint /wp-json/wp/v2/posts/. Ví dụ:​

bash

GET http://yourdomain.com/wp-json/wp/v2/posts/

Yêu cầu này sẽ trả về một danh sách các bài viết dưới định dạng JSON, bao gồm các thông tin như ID, tiêu đề, nội dung, ngày đăng, v.v. Bạn có thể sử dụng các tham số truy vấn để lọc, sắp xếp hoặc phân trang kết quả. Ví dụ:​

**Lấy các bài viết được sắp xếp theo ngày đăng giảm dần:**

GET http://yourdomain.com/wp-json/wp/v2/posts?orderby=date&order=desc

**Lấy 10 bài viết từ trang thứ 2:**

GET http://yourdomain.com/wp-json/wp/v2/posts?page=2&per\_page=10

**2. Lấy thông tin người dùng:** Để truy xuất thông tin người dùng, bạn có thể sử dụng endpoint /wp-json/wp/v2/users/. Ví dụ:​

**Lấy danh sách tất cả người dùng:**

GET http://yourdomain.com/wp-json/wp/v2/users/

**Lấy thông tin của một người dùng cụ thể dựa trên ID:**

GET http://yourdomain.com/wp-json/wp/v2/users/123

Trong đó, 123 là ID của người dùng mà bạn muốn truy xuất thông tin. Lưu ý rằng việc truy cập thông tin người dùng có thể yêu cầu quyền xác thực, tùy thuộc vào cài đặt bảo mật của trang web. ​[plugins.com.vn](https://plugins.com.vn/wordpress-rest-api-la-gi/)[vietnix.vn](https://vietnix.vn/wordpress-rest-api-la-gi/)

**Lưu ý:**

Đảm bảo rằng REST API đã được kích hoạt trên trang WordPress của bạn. Bạn có thể kiểm tra bằng cách truy cập http://yourdomain.com/wp-json/ và xem liệu có danh sách các endpoint được hiển thị hay không. ​

Khi truy cập các endpoint yêu cầu quyền xác thực (như thông tin người dùng), bạn cần sử dụng các phương thức xác thực phù hợp, chẳng hạn như Cookie Authentication hoặc Basic Authentication. ​

Bằng cách sử dụng các endpoint trên, bạn có thể dễ dàng truy vấn và thao tác với dữ liệu trên trang WordPress thông qua REST API.​

**3. Thêm, sửa, xóa dữ liệu bằng REST API.**

​Để thực hiện các thao tác thêm, sửa và xóa dữ liệu trong WordPress thông qua REST API, bạn có thể sử dụng các phương thức HTTP tương ứng như **POST**, **PUT** (hoặc **PATCH**) và **DELETE.** Dưới đây là hướng dẫn chi tiết cho từng thao tác:​

**1. Thêm dữ liệu (Create) bằng phương thức POST:** Để thêm một bài viết mới, bạn sẽ gửi một yêu cầu **POST** đến endpoint /wp-json/wp/v2/posts/. Yêu cầu này cần bao gồm các thông tin như tiêu đề, nội dung và trạng thái của bài viết. Ví dụ, sử dụng cURL:​

bash

curl -X POST https://yourdomain.com/wp-json/wp/v2/posts/ \

-H "Content-Type: application/json" \

-H "Authorization: Basic base64\_encode(username:password)" \

-d '{

"title": "Tiêu đề bài viết mới",

"content": "Nội dung bài viết mới",

"status": "publish"

}'

Trong đó, base64\_encode(username:password) là chuỗi mã hóa Base64 của tên người dùng và mật khẩu, được sử dụng cho xác thực Basic Authentication. Bạn có thể sử dụng các phương pháp xác thực khác như Cookie Authentication hoặc OAuth tùy theo cấu hình của trang web.​

**2. Sửa dữ liệu (Update) bằng phương thức PUT hoặc PATCH:** Để cập nhật một bài viết hiện có, bạn sẽ gửi một yêu cầu PUT hoặc PATCH đến endpoint /wp-json/wp/v2/posts/{id}/, trong đó {id} là ID của bài viết cần cập nhật. Ví dụ, để cập nhật tiêu đề của bài viết có ID là 10:​

bash

curl -X PUT https://yourdomain.com/wp-json/wp/v2/posts/10 \

-H "Content-Type: application/json" \

-H "Authorization: Basic base64\_encode(username:password)" \

-d '{

"title": "Tiêu đề mới"

}'

**3. Xóa dữ liệu (Delete) bằng phương thức DELETE:** Để xóa một bài viết, bạn sẽ gửi một yêu cầu DELETE đến endpoint /wp-json/wp/v2/posts/{id}/. Ví dụ, để xóa bài viết có ID là 10:​

bash

curl -X DELETE https://yourdomain.com/wp-json/wp/v2/posts/10 \

-H "Authorization: Basic base64\_encode(username:password)"

**Lưu ý quan trọng:**

Các thao tác thêm, sửa và xóa dữ liệu yêu cầu quyền truy cập phù hợp. Đảm bảo rằng bạn sử dụng phương thức xác thực thích hợp và tài khoản có đủ quyền hạn để thực hiện các thao tác này.​

Để bảo mật, nên sử dụng HTTPS cho tất cả các yêu cầu REST API để mã hóa thông tin truyền tải.​

Trước khi thực hiện các thao tác trên môi trường thực tế, nên thử nghiệm trên môi trường phát triển để đảm bảo an toàn và tránh ảnh hưởng đến dữ liệu thực.​

Để biết thêm chi tiết về cách sử dụng WordPress REST API, bạn có thể tham khảo tài liệu chính thức của WordPress hoặc các hướng dẫn chi tiết khác.​

**IV. Ứng dụng của REST API trong phát triển web**

​REST API (Representational State Transfer Application Programming Interface) là một giao diện cho phép các ứng dụng giao tiếp với nhau thông qua các giao thức HTTP. Trong phát triển web, REST API đóng vai trò quan trọng trong việc:​

**1. Phát triển giao diện frontend riêng biệt cho WordPress:** Sử dụng WordPress REST API, các nhà phát triển có thể xây dựng giao diện người dùng (frontend) độc lập, tách biệt hoàn toàn khỏi hệ thống quản lý nội dung (backend) của WordPress. Điều này cho phép sử dụng các framework JavaScript hiện đại như React, Vue hoặc Angular để tạo ra trải nghiệm người dùng phong phú và tương tác hơn. Bằng cách gọi các endpoint của REST API, frontend có thể lấy dữ liệu bài viết, trang, thông tin người dùng và hiển thị chúng một cách linh hoạt.​

**2. Kết nối WordPress với các ứng dụng bên ngoài:** REST API cho phép WordPress tương tác với các ứng dụng và dịch vụ bên ngoài, mở rộng khả năng tích hợp và trao đổi dữ liệu. Ví dụ, bạn có thể tích hợp WordPress với ứng dụng di động, hệ thống CRM, hoặc các dịch vụ bên thứ ba khác để đồng bộ hóa dữ liệu, tự động hóa quy trình và cung cấp trải nghiệm liền mạch cho người dùng.​

**3. Các công cụ hỗ trợ làm việc với REST API:** Để làm việc hiệu quả với REST API, có nhiều công cụ hỗ trợ giúp kiểm thử và quản lý các yêu cầu HTTP:

**Postman:** Là một công cụ phổ biến cho phép gửi các yêu cầu HTTP và xem phản hồi một cách trực quan. Postman hỗ trợ cả REST, GraphQL và SOAP APIs, cung cấp môi trường kiểm thử mạnh mẽ cho các nhà phát triển.

**cURL:** Là một công cụ dòng lệnh mạnh mẽ để thực hiện các yêu cầu HTTP. cURL thường được sử dụng trong các tập lệnh hoặc khi cần kiểm thử nhanh các endpoint của API.

**Fetch API:** Là một API JavaScript tích hợp sẵn trong các trình duyệt hiện đại, cho phép thực hiện các yêu cầu HTTP một cách dễ dàng trong mã nguồn frontend. Fetch API hỗ trợ làm việc với REST API trực tiếp từ ứng dụng web.​

**Axios:** Là một thư viện JavaScript dựa trên Promise, giúp thực hiện các yêu cầu HTTP một cách dễ dàng và linh hoạt. Axios hỗ trợ cả môi trường trình duyệt và Node.js, cung cấp các tính năng như hủy yêu cầu, chặn yêu cầu/phản hồi và tự động chuyển đổi dữ liệu JSON.​

Việc sử dụng các công cụ này giúp quá trình phát triển, kiểm thử và gỡ lỗi REST API trở nên hiệu quả và thuận tiện hơn, đảm bảo ứng dụng hoạt động ổn định và đáp ứng yêu cầu người dùng.​