WordPress API Workshop

How to create a custom translated REST API endpoint and make cool stuff

Christina Garofalo @ WordCamp Canada 2024 | July 12, 2024

Bonjour/Hi!

I’m self -taught

I’ve always had a knack for figuring things out. I have no formal training in computer sciences, but lots of hands on experience.

I’ve worked with WordPress for about 10 years

I've been with a Montreal-based agency for the past 7 years, working alongside an awesome collaborative team.

I’m on the WordPress Documentation Team

I joined in person at WCEU Athens 2023. Anyone can contribute. No programming knowledge required!

Christina Garofalo
Senior WordPress Developer

Today's Agenda

01 Intro

02 What is a REST API?

03 Out-of-the-Box API Endpoints

04 Let's Build a Custom Endpoint!

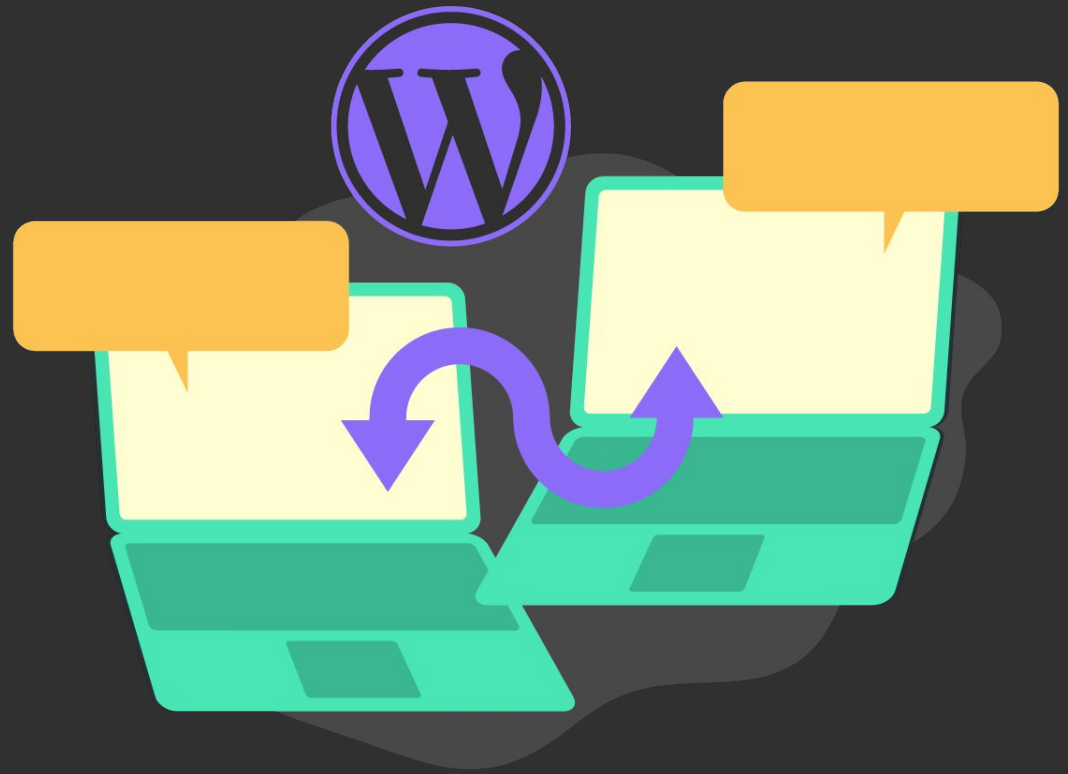
05 Security

06 Q & A

01
Intro

The WordPress API

- WordPress introduced the REST API in **version 4.7 (2016)** and it became the foundation for the WordPress block editor.
- The REST API sends and receives data in **JSON format**, making it easy to integrate into web applications.
- With the REST API, it is possible to do things such as build an entirely new admin experience, a completely new interactive front end experience, or create entirely **new applications with your WordPress content.**
- The REST API is one of many WordPress APIs. They are listed here: **https://codex.wordpress.org/WordPress_APIs**



02

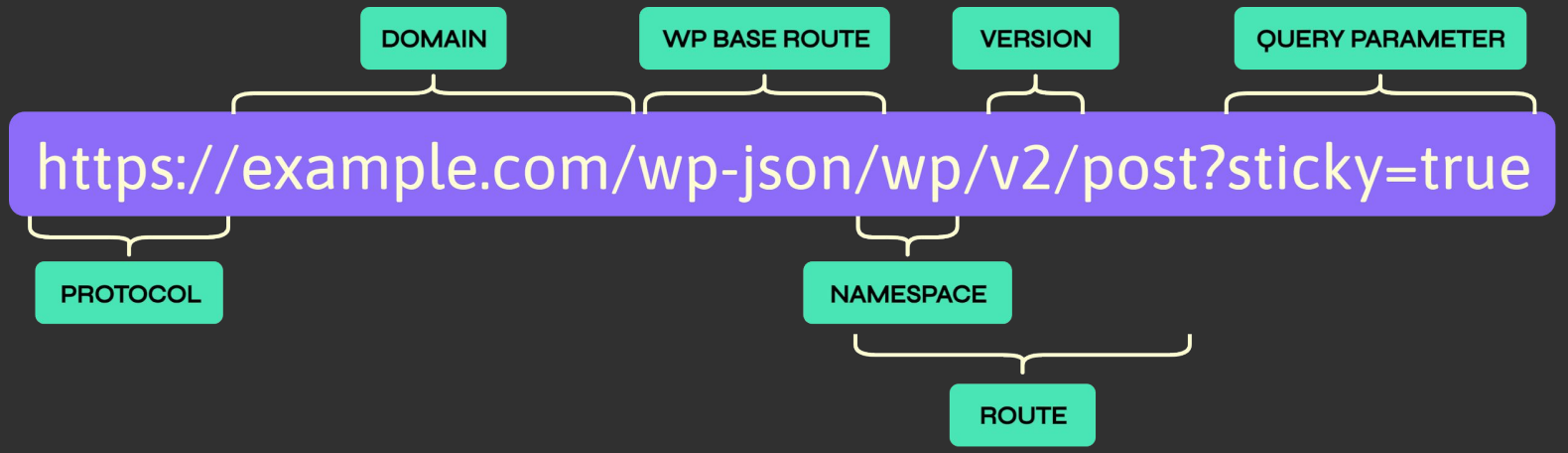
What is a REST API?

REpresentational State Transfer

A software architecture that imposes conditions of how an API should work.

Application Programming Interface

A standardized structure.



REST APIs have these methods

What can it do?

HTTP Headers

Additional information

03
Out-of-the-box API endpoints

WordPress API Endpoints

There is extensive documentation available about the endpoints and all of the parameters found in the REST API handbook.
https://developer.wordpress.org/rest-api/

All WordPress installations have standard endpoints baked in that can be used to manipulate:

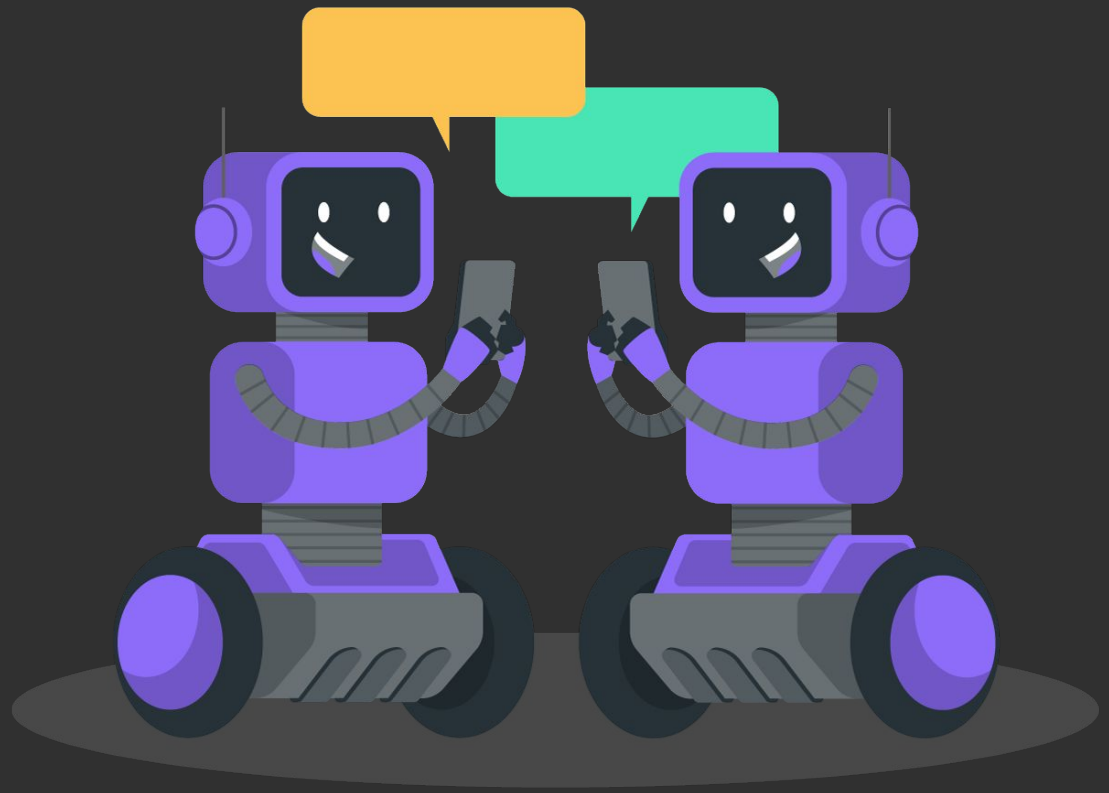
- Posts
- Media
- Users
- Tags
- Pages
- Comments
- Taxonomies
- Post Types
- Post Statuses
- General Blog Settings

No Humans Allowed!

APIs are not intended for humans to ‘read’.
APIs make it possible for systems to talk to each other.

Use **Firefox** or **Chrome with the JSONvue extension** to see structured JSON in your browser. Otherwise, it’s just a wall of text.

Other tools can be used to test endpoints, such as **Postman, Insomnia** or **Bruno**. These apps are more useful for testing endpoints with authentication, POST, PUT and DELETE methods.



Basic Endpoint

The most basic WordPress REST API endpoint is simply /wp-json.

This will return basic information about the WordPress installation.

https://wceh.local/wp-json/


```
{
name: "WordCamp Canada API Demo",
description: "A site demonstrating the REST API",
url: "http://wceh.local",
home: "http://wceh.local",
gmt_offset: -4,
timezone_string: "America/Toronto",
namespaces:
[
          "oembed/1.0",
          "wcmtl/v1",
          "wp/v2",
          "wp-site-health/v1",
          "wp-block-editor/v1"
],
authentication:
{
          application-passwords:
          {
                    endpoints:
                    {
                              authorization:
                              "http://wceh.local/wp-admin
                              /authorize-application.php"
                    }
          }
},
```

An Example: Posts Endpoint

`/wp-json/wp/v2/posts`
This endpoint will return a list of all posts.

It can also take parameters (aka arguments):
`/wp-json/wp/v2/posts?sticky=true`

https://wceh.local/wp-json/wp/v2/posts
https://wceh.local/wp-json/wp/v2/posts?sticky=true


```
[
    {
        id: 1,
        date: "2024-06-24T18:00:33",
        date_gmt: "2024-06-24T18:00:33",
        guid:
        {
            rendered: "http://wceh.local/?p=1"
        },
        modified: "2024-06-24T18:00:33",
        modified_gmt: "2024-06-24T18:00:33",
        slug: "hello-world",
        status: "publish",
        type: "post",
        link: "http://wceh.local/hello-world/",
        title:
        {
            rendered: "Hello world!"
        },
        content:
        {
            rendered: " <p>Welcome to WordPress. This
            is your first post. Edit or delete it, then
            start writing!</p> ",
            protected: false
        },
        excerpt:
        {
            rendered: "<p>Welcome to WordPress. This is
            your first post. Edit or delete it, then
            start writing!</p> ",
            protected: false
```

An Example: Users Endpoint

`/wp-json/wp/v2/users`

The users endpoint, which is a public endpoint, will return a list of **all users subscribed to your site, regardless of role**. It does not return the roles associated with the users, but it does return **usernames and IDs**, which can be used to figure out which account might be an admin and could be an attack vector for the ne'er-do-wells out there looking to gain access to your site. (More on security later).

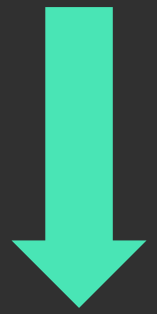

```
[
        {
                id: 1,
                name: "Christina",
                url: "http://wceh.local",
                description: "",
                link:
                "https://wceh.local/author/christina/",
                slug: "christina",
                avatar_urls:
                {
                        24:
                        "https://secure.gravatar.com/avatar/57563c471581211b8818c051015343c5?s=24&d=mm&r=g",
                        48:
                        "https://secure.gravatar.com/avatar/57563c471581211b8818c051015343c5?s=48&d=mm&r=g",
                        96:
                        "https://secure.gravatar.com/avatar/57563c471581211b8818c051015343c5?s=96&d=mm&r=g"
                },
```

04

Let's Build a Custom Endpoint!

Step-By-Step

You can include custom code in your theme or create a plugin. Whatever you choose, don't edit core WordPress files.



You can get a copy of the demo plugin here:
https://github.com/cgarofalo/wceh-api

1. Custom Post Type
2. Define the route
3. Create the callback function
4. Test your endpoint
5. Build your app and use your endpoint

1. Create a post type

Since WordPress already has built-in endpoints for the default post types, most likely, you'll need an endpoint for anything beyond that.

```php
// Add the custom Post type
add_action( 'init', 'wcehapi_custom_post_type' );

/**
* Register Custom Post Type
*
* @return void
*/
function wcehapi_custom_post_type() {
  register_post_type(
     'wcehapi_cat',
     [
        'labels'              => [
           'name'          => __( 'Cats', 'wcehapi' ),
           'singular_name' => __( 'Cat', 'wcehapi' ),
        ],
        'public'              => false,
        'show_ui'             => true,
        'has_archive'         => false,
        'hierarchical'        => false,
        'show_in_rest'        => false,
        'exclude_from_search' => true,
        'capability_type'     => 'post',
        'rewrite'             => [
           'slug' => 'cats',
        ],
        'supports'            => [
           'title',
           'editor',
           'thumbnail',
        ],
     ]
  );
}
```

2. Define a route

Next we'll need to define the route. The route is going to be the url that you'll need to hit to get your JSON data. It is structured like this: `/wp-json/your-namespace/v1/whatever-you-want`

As of WordPress 5.5, you must provide the permission_callback parameter. When defining your endpoints, the `permission_callback` parameter defines who has permission to access the endpoint. Setting this to `__return_true` will create a public endpoint that anyone can access. If the callback used here returns false, WordPress will return a `rest_forbidden` error.

```php
namespace WCEHAPI\plugin\API;

add_action( 'rest_api_init', __NAMESPACE__ . '\\register_routes' );


/**
* Register the routes for the plugin.
*
* @return void
*/
function register_routes() {
  $version   = '1';
  $namespace = 'wcmtl/v' . $version;

  // Register Routes

  // wp-json/wcmtl/v1/cats
  register_rest_route(
     $namespace,
     '/cats',
     [
        'methods'
        'permission_callback' => '__return_true',
     ]
  );
}
```

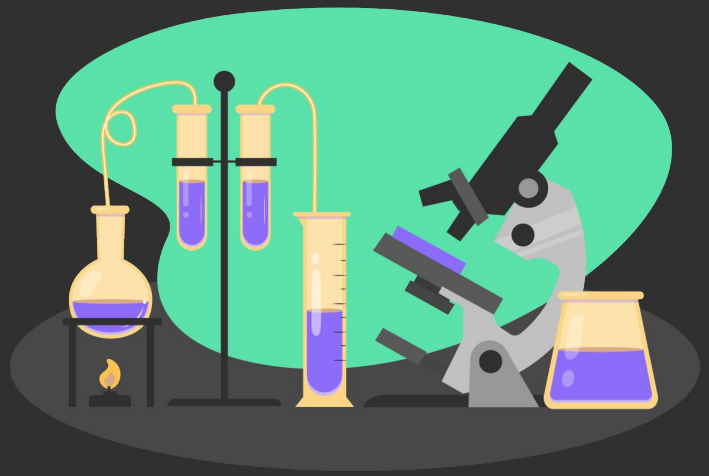
3. Create Callback Function

Create the callback function. This is what builds and returns the JSON. This is where the magic happens. You'll need to always return your data with the `rest_ensure_response()` function. This function will JSON encode the array of data that you pass it, and make sure that a response code is sent, ideally 200.

```php
/**
* Callback for the cats endpoint
*
* @return \WP_Error|\WP_HTTP_Response|\WP_REST_Response
*/
function get_cats() {

// WP_Query or logic goes here

$data = [
     'cats' => [
        [
            'name'    => 'Pekoe',
            'colour'  => 'orange',
            'breed'   => 'domestic shorthair',
            'pattern' => 'tabby',
        ],
        [
            'name'    => 'Milo',
            'colour'  => 'grey',
            'breed'   => 'domestic shorthair',
            'pattern' => 'solid',
        ],
        [
            'name'    => 'Poppy',
            'colour'  => 'cream',
            'breed'   => 'siamese',
            'pattern' => 'pointed',
        ],
    ],
  ];

  return rest_ensure_response( $data );
}
```

4. Test Your Endpoint

Visit your endpoint! Do cool stuff with it, the sky's the limit. We make React apps to add animation and interactivity to our frontend builds.

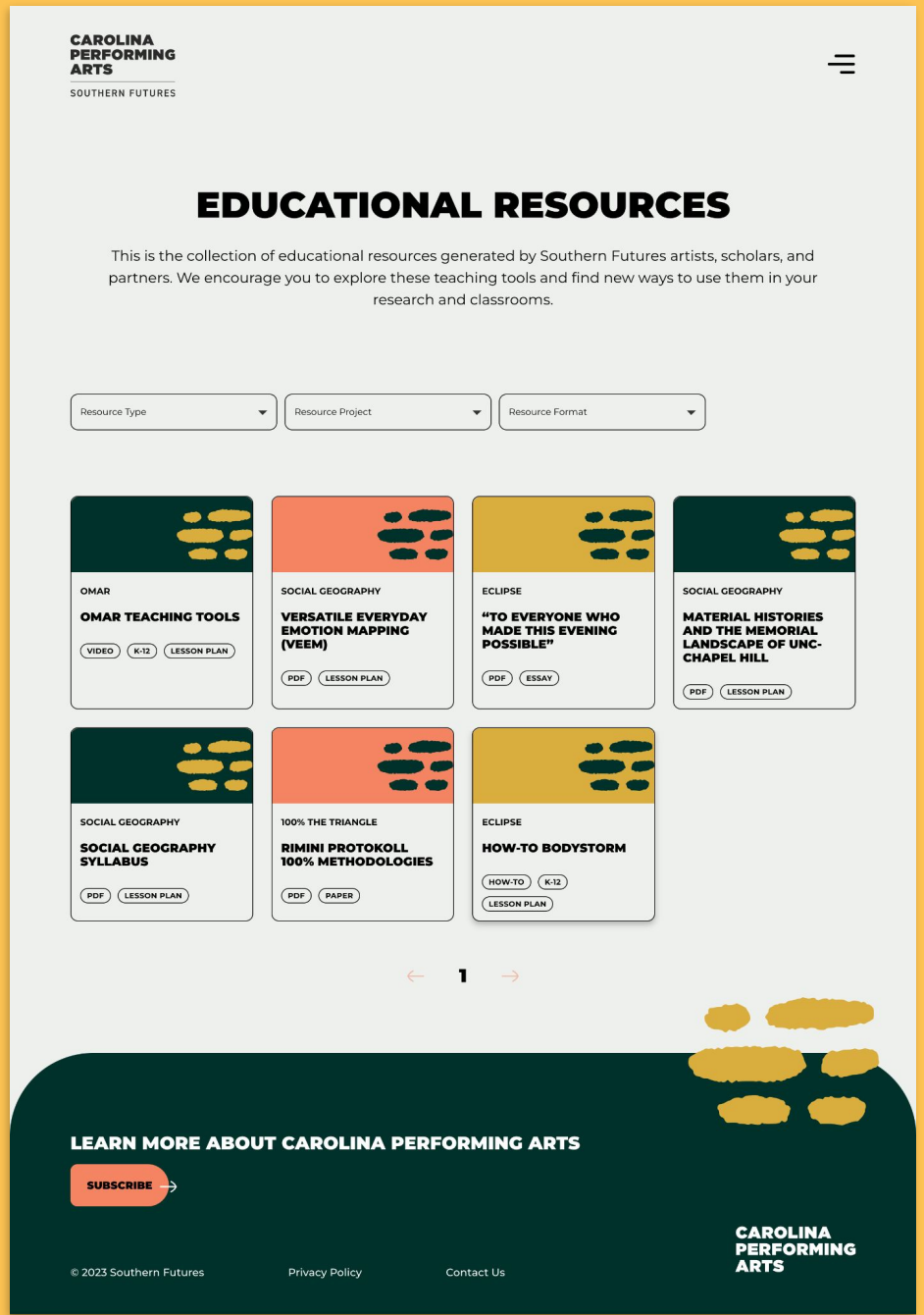



```
{
        cats:
        [
                {
                        name: "Pekoe",
                        colour: "orange",
                        breed: "domestic shorthair",
                        pattern: "tabby"
                },
                {
                        name: "Milo",
                        colour: "grey",
                        breed: "domestic shorthair",
                        pattern: "solid"
                },
                {
                        name: "Poppy",
                        colour: "cream",
                        breed: "siamese",
                        pattern: "pointed"
                }
        ]
}
```

5. Build your app and use your endpoint!

Real world example:
Carolina Performing Arts: Southern Futures

The filterable education resources on this page use two custom API endpoints to power the React app (Resources + Filters). The result is a lightning fast filtering system without needing to refresh the page.



05
Security

Security

For your first endpoints, create a **public GET endpoint**. You won't need to worry about security, as the information provided by your custom endpoint should not be sensitive and will be read-only.

You can create **POST, PUT and DELETE endpoints**, but be sure to **secure them**. You wouldn't want someone or an outside system accessing these endpoints and modifying your database.

There is a way to lock down your endpoints so that you can only access them if you are logged in as an administrator or have specific permissions (Or whatever condition is specified in the `permissions_callback` for that particular endpoint).

Lock It Down

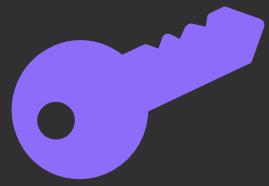
If a user needs to be authenticated to interact with an endpoint, they must generate an application password and pass along the credentials. **Application Passwords** were added in WordPress 5.6 and allow for a way to authenticate without exposing the user's primary password. They can be revoked and regenerated at any time. **They are not dissimilar to an API Key.**

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

There are two ways to authenticate natively in WordPress:

1. **Cookies (nonces)**
2. **Application Passwords**

Authentication plugins can add other forms of authentication, like OAuth & JSON web tokens, should you require them.



06
Q & A

THANKS

Do you have any questions?

christina.garofalo@plank.co

https://dev.to/cgarofalo

https://profiles.wordpress.org/cold-iron-chef/

https://www.linkedin.com/in/cgarofalo

https://www.reddit.com/user/ColdIronChef/