
Title: Decoupled Drupal

Overview

Using Drupal JSON:API to create a Decoupled Drupal. Covers below topics.

1. Drupal modules and configuration.
2. Access and Authorisation.
3. Developer documentation and tools.
4. Security.
5. Exposed entities.
6. Caching.

Exposed entities API

By default all entities are exposed but can be restricted using JSON: API extras module under admin path /admin/config/services/jsonapi/resource_types

Also default API path that begins with /jsonapi can also be changed under admin path /admin/config/services/jsonapi/extras

API documentation based on openAPI specification: /admin/config/services/openapi/redoc/jsonapi

To begin we will enable below entities for GET and POST requests.

* Content Type
* Blocks
* Menu
* Users - Will be required for preview permission
* Environment Indicator

Types

Every resource in JSON:API must have a globally unique type property. The Drupal JSON:API implementation derives this type property from the entity type machine name and bundle machine name. For example, articles, pages, and users are given the types node--article, node--pages, and user--user, respectively.

URL Structure

A JSON:API url by default begins with/jsonapi/ and can be modified under admin path /admin/config/services/jsonapi/extras.

GET|POST

* /jsonapi/node/article will fetch all the article pages.
* /jsonapi/node/article?filter[drupal_internal__nid]=30 will fetch a page from content type article.

HTTP Methods

JSON:API specifies what HTTP Methods to accept. Those are: GET, POST, PATCH, and DELETE. Notably, PUT is not included.

* GET - Retrieve data, can be a collection of resources or an individual resource
* POST - Create a new resource
* PATCH - Update an existing resource
* DELETE - Remove an existing resource

Since we will be using only GET and POST, will not touch much into PATCH and DELETE.

Request headers

Make sure to use 'Content type' and 'Accept' headers when appropriate. See Client responsibility for more details.
https://jsonapi.org/format/#content-negotiation-clients

Accept: application/vnd.api+json
Content-Type: application/vnd.api+json

## Drupal Set Up

### Modules

* jsonapi-module is part of core but need to enable.

* Below other contibuted Modules that will assist with development.

#### JSON:API Explorer

* Provides options to filter and test API.
* API Explorer: /jsonapi/explorer/app

#### Schema

* JSON:API Schema provides JSON-Schema formatted schemas for JSON:API resources.
* Can be used as documentation by developers and testers.
* Path: /jsonapi/schema

#### jsonapi_hypermedia

* It adds support for rich, dynamic linking between your application's resources.
* Hypermedia will have id and operations.

Module : https://www.drupal.org/project/jsonapi_hypermedia

In our case jsonapi_hypermedia module will be useful for:

* Providing menu content and aliased URLs

* Providing "breadcrumbs".
* Providing metadata about linked resources.
* Providing custom labels for "call to action" buttons
.

Drupal Preview

* Editor able to preview the content before publishing
.
* Will require permission update and setting a URL to the external preview link. You can use tokens there, like [node:nid]

Frontend will require top call API to check user and user roles before allowing preview.

Drupal Module: https://www.drupal.org/project/dpl

JSON:API Boost
* Improves the performance of the JSON:API module by cache warming your resource types.
* Ability to warm cache manually and using drush.
* Ability to set enter the Frequency of refresh.
* Include cache warming to your post-deployment script
. This way your Drupal install will come online with warm caches protecting your site from stampedes, and providing a good performance improvement.

Other contributed modules available.

https://www.drupal.org/project/jsonapi_views

https://www.drupal.org/project/jsonapi_search_api

https://www.drupal.org/project/decoupled_menus (Not stable but can be reviewed)

https://www.drupal.org/sandbox/gabesullice/3132553 (In sandbox currently. This is most useful for improving the performance of polling requests to JSON:API resources because it eliminates the browser's need to download an unchanged response body).

There are other contributed modules in Ecosystem for JSON:API as well.
https://www.drupal.org/project/jsonapi/ecosystem

Allowing access to JSON:API (For frontend).

* Update config
* Update CORS
* Authentication
* Whitelist IP (Optional)

Security/ Updating Resource Type

JSON:API Extras or Using Custom Code (Optional)
https://www.drupal.org/docs/core-modules-and-themes/core-modules/jsonapi-module/customizing-resources

Some of functionality for updating Resource type is supplied in JSON:API Extras. This module tries to use the core functionality where possible but supplies an interface and some extra features like:

* Altering the basepath to the API.
* Add enhancers to fields.
* Disable resources by default.

https://www.drupal.org/project/views_json_source

Things to Consider

* Routing - Drupal or Frontend
* Front-end catch-all route - Refer Link below.
* Menus - The menus can also be built as defined by Drupal's menu system. This allows editors and administrators to manage menu items via standard Drupal admin tools. JSON:API Menu Items allows custom menu entities to be returned via JSON:API.
* Requests - Refer Link below.

* Flexible image styles - With the help of the Consumer Image Styles module, you can include several image styles in the JSON:API response along with any image entity. This allows your front-end to decide which image style to use based on factors like the window size.
* Sending multiple requests - Use Promise.all in the front-end when requests are independent of each other . Otherwise, consider using the Subrequests module to send a single request containing interdependent subrequests.

https://evolvingweb.com/blog/building-decoupled-drupal-part-2

Remember

* JSON:API resource types can now be configured programmatically.
* Expose only what you use (Use
* Use Read-only mode if need is just to read only.
* Security through obscurity: secret base path : The base path for JSON:API is /jsonapi by default. This can be changed to something like /hidden/b69dhj027ooae/jsonapi, which is one way to reduce the effectiveness of automated attacks. Create sites/example.com/services.yml if it doesn't exist already and add this:
```
    parameters:
      jsonapi.base_path: /hidden/b69dhj027ooae/jsonapi
```
* Limit access to all JSON:API routes with an extra permission.
* Enable the HTTP Basic Authentication module, set the permission for the API user (and role) and set the encoded username and password to the 'Authorization' request header.
* Developer Documentation resources are openAPI doc (/admin/config/services/openapi/redoc/jsonapi), API Scheme (/jsonapi/schema) and API Explorer (/jsonapi/explorer/app).
* Option to disable fields exposed through API.

Drupal Docs

* Drupal JSON:API module doc index page - https://www.drupal.org/docs/core-modules-and-themes/core-modules/jsonapi-module

* Drupal's JSON:API security recommendations - https://www.drupal.org/docs/core-modules-and-themes/core-modules/jsonapi-module/security-considerations

Questions

Does Drupal manage routing?

In Drupal-managed routing, Drupal handles the logic and definitions for paths and menus, and content is retrieved using its Drupal-defined path.
Even if Drupal does not manage the routing we can still use the Drupal to add the path by editor and expose it through jsonAPI.

How much editing capability editors will have?

* Assigning a block to a region work?

* Create new block and assigned to a region. For example a simple content block.

* Disable a block.

* Allowed HTML? See below example.

    1. Headings (H1 through H6)
    2. Bold, italic, and underline buttons
    3. Basic code formatting
    4. Blockquotes
    5. Hyperlinks
    6. Ordered and unordered lists
    7. Embeds of media or other content

* Ability to update sitemap

* Ability to update robots.txt
* Purge/Invalidate cache.
* Maintenance page.
* Side wide alerts at the top of page.(May be Future).
* Adding Forms (Future).

Challenges Reported

String translation Issue

Module strings_i18n_json_export allows you to export your Drupal strings to a JSON i18n that can be read by another front-end framework. Like Vue.js for example.
https://www.drupal.org/project/strings_i18n_json_export

Configurations

For any configuration exchange Drupal configs have to be exported using module config_export_json.
https://www.drupal.org/project/config_export_json