Django REST Framework - JSON API Documentation

Release 0.1.1

Kevin Brown

May 12, 2017

Contents

A parser and renderer for Django REST Framework that adds support for the JSON API specification.

Build status:

CHAPTER 1

Does this work?

This package is currently being actively developed, but is not widely used in production. If you find any problems when using this package, please create a bug report at the issue tracker so we can figure out how to fix it.

CHAPTER 2

How do I use this?

This is designed to be used as only a renderer and parser and does not provide any additional functionality that may be expected by JSON API.

Specific to a view(set)

```
from rest_framework import generics
from rest_framework_json_api.renderers import JsonApiRenderer


class ExampleView(generics.ListAPIView):
    renderer_classes = (JsonApiRenderer, )
```

The JSON API renderer is not limited to just list views and can be used on any of the generic views. It supports viewsets as well as non-generic views.

All views

The JSON API renderer can be used on all views by setting it as a default renderer.

```
# ...
REST_FRAMEWORK = {
    "DEFAULT_RENDERER_CLASSES": (
        "rest_framework_json_api.renderers.JsonApiRenderer",
        "rest_framework.renderers.BrowsableAPIRenderer",
        # Any other renderers
    ),
    "DEFAULT_PARSER_CLASSES": (
        "rest_framework_json_api.parsers.JsonApiParser",
        "rest_framework.parsers.FormParser",
        "rest_framework.parsers.MultiPartParser",
```

```
        # Any other parsers
    ),
}
#...
```

This may break the API root view of the Default Router, so you may want to instead apply it to your viewsets.

CHAPTER 3

What does this support?

The JSON API renderer supports all features of hyperlinked serializers and will normalize attributes such as the url field to match the JSON API specification.

Introspected resource types

JSON API uses resource types to determine what relations exist and how to better side-load resources automatically. It is recommended that resource types match the URL structure of the API and use a plural form. The resource type is determined from the model, and is the plural form of the verbose model name.

If a verbose name cannot be determined, the generic key`data` will be used for the resource type.

Hyperlinked relations

JSON API will detect hyperlinked relations and set up the url templates to match the destinations and attribute names automatically.

Nested serializers

JSON API will render nested serializers to match the compound document specification. This will theoretically support any depth of nested serializers, but only a single level is tested and supported.

Pagination

JSON API does not explicitly call out pagination within the specification, but instead leaves it flexible for the developer to implement. The JSON API renderer supports the default pagination provided by Django REST Framework by

adding it to the top level "meta" element. This can be overriden by using a modified render, or a paginator that relies on a header, such as the Link header based paginator.

CHAPTER 4

What this will not easily support

Due to limitations within the JSON API specification, as well as a need to handle the most common easy cases, this JSON API renderer will not work with all views. When designing views that work well with the JSON API specification, there are a few needs that you should keep in mind.

Anything not related to rendering or parsing

This package is only designed to be used as a renderer and parser and does not provide support for parts of the JSON API specification that are not unique to the JSON API specification. This includes features such as custom filtering of results and pagination that does not use the response body. Features such as side-loading of data using query parameters are also not supported.

CHAPTER 5

Isn't JSON API being actively developed?

Yes it is, and we will try to keep this package as close to the running specification as possible. This means that things may break during version changes, and until JSON API becomes stable we cannot guarantee backwards compatibility. Once JSON API stabilizes, a deprecation process will be established to match the policies of the JSON API specification.

CHAPTER 6

Recommended packages

This parser/renderer combination is only meant to be used as one of many packages that can be grouped together to create an API that supports the JSON API specification.

Pagination

The Link header based paginator will work with the renderer provided by this package.

JSON Patch

JSON API recommends using JSON Patch for *PATCH* requests, and allowing partial updates through the *PUT* HTTP method. JSON Patch support is available for Django REST Framework through a **'third party package <https://github.com/kevin-brown/drf-json-patch'__** and should be compatible.

CHAPTER 7

Contents:

Installation

At the command line either via easy_install or pip:

```
$ easy_install drf-json-api
$ pip install drf-json-api
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv drf-json-api
$ pip install drf-json-api
```

Usage

To use Django REST Framework - JSON API in a project:

```
import drf-json-api
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/kevin-brown/drf-json-api/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Django REST Framework - JSON API could always use more documentation, whether as part of the official Django REST Framework - JSON API docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kevin-brown/drf-json-api/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *drf-json-api* for local development.

1. Fork the *drf-json-api* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/drf-json-api.git
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/kevin-brown/drf-json-api under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests/test_detail.py
```

Credits

Development Lead

- Kevin Brown <kevin@kevinbrown.in>

Contributors

- John Whitlock <john@factorialfive.com>

History

0.1.0 (Unreleased)

- First release on PyPI.

CHAPTER 8

Feedback

If you have any suggestions or questions about **Django REST Framework - JSON API** feel free to email me at kevin@kevinbrown.in.

If you encounter any errors or problems with **Django REST Framework - JSON API**, please let me know! Open an Issue at the GitHub https://github.com/kevin-brown/drf-json-api main repository.