aiohttp JSON API Documentation

Release 0.37.0

Vladimir Bolshakov

Sep 10, 2018

Contents

1 JSON API implementation for aiohttp **3**
- 1.1 Introduction . . . 3
- 1.2 Requirements . . . 3
- 1.3 Todo . . . 4
- 1.4 Credits . . . 4

2 Installation **5**
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5
- 2.3 Default setup of resources, routes and handlers . . . 6

3 Usage **7**

4 Contributing **9**
- 4.1 Types of Contributions . . . 9
- 4.2 Get Started! . . . 10
- 4.3 Pull Request Guidelines . . . 11
- 4.4 Tips . . . 11

5 Credits **13**
- 5.1 Development Lead . . . 13
- 5.2 Author of core idea . . . 13
- 5.3 Contributors . . . 13

6 History **15**
- 6.1 0.37.0 (2018-03-03) . . . 15
- 6.2 0.36.1 (2018-03-03) . . . 15
- 6.3 0.36.0 (2018-03-02) . . . 15
- 6.4 0.35.2 (2017-12-13) . . . 15
- 6.5 0.35.1 (2017-12-12) . . . 15
- 6.6 0.35.0 (2017-12-11) . . . 16
- 6.7 0.33.1 (2017-12-06) . . . 16
- 6.8 0.33.0 (2017-12-06) . . . 16
- 6.9 0.32.0 (2017-11-21) . . . 16
- 6.10 0.31.0 (2017-11-14) . . . 17
- 6.11 0.30.0 (2017-11-13) . . . 17
- 6.12 0.29.2 (2017-11-02) . . . 17

6.13 0.29.1 (2017-11-02) . . . 17
6.14 0.29.0 (2017-11-02) . . . 17
6.15 0.28.3 (2017-11-02) . . . 17
6.16 0.28.2 (2017-11-01) . . . 17
6.17 0.28.1 (2017-10-31) . . . 17
6.18 0.28.0 (2017-10-31) . . . 18
6.19 0.27.1 (2017-10-31) . . . 18
6.20 0.27.0 (2017-10-31) . . . 18
6.21 0.26.0 (2017-10-30) . . . 18
6.22 0.25.0 (2017-10-18) . . . 18
6.23 0.24.1 (2017-10-12) . . . 18
6.24 0.24.0 (2017-10-04) . . . 19
6.25 0.23.0 (2017-10-03) . . . 19
6.26 0.22.2 (2017-09-27) . . . 19
6.27 0.22.1 (2017-09-25) . . . 19
6.28 0.22.0 (2017-09-22) . . . 19
6.29 0.21.2 (2017-09-22) . . . 19
6.30 0.21.1 (2017-09-19) . . . 19
6.31 0.21.0 (2017-09-19) . . . 19
6.32 0.20.2 (2017-08-30) . . . 20
6.33 0.20.1 (2017-08-15) . . . 20
6.34 0.20.0 (2017-08-14) . . . 20
6.35 0.19.1 (2017-08-10) . . . 20
6.36 0.19.0 (2017-08-10) . . . 20
6.37 0.18.1 (2017-08-09) . . . 20
6.38 0.18.0 (2017-08-09) . . . 20
6.39 0.17.1 (2017-07-31) . . . 21
6.40 0.17.0 (2017-07-28) . . . 21
6.41 0.16.2 (2017-07-24) . . . 21
6.42 0.16.1 (2017-07-24) . . . 21
6.43 0.16.0 (2017-07-22) . . . 21
6.44 0.15.2 (2017-07-21) . . . 21
6.45 0.15.1 (2017-07-19) . . . 21
6.46 0.15.0 (2017-07-18) . . . 22
6.47 0.14.0 (2017-07-13) . . . 22
6.48 0.13.0 (2017-07-12) . . . 22
6.49 0.12.0 (2017-07-12) . . . 22
6.50 0.11.1 (2017-07-11) . . . 22
6.51 0.11.0 (2017-07-11) . . . 22
6.52 0.10.0 (2017-07-10) . . . 23
6.53 0.9.3 (2017-07-06) . . . 23
6.54 0.9.2 (2017-07-06) . . . 23
6.55 0.9.1 (2017-07-06) . . . 23
6.56 0.9.0 (2017-07-06) . . . 23
6.57 0.8.2 (2017-07-05) . . . 24
6.58 0.8.1 (2017-07-05) . . . 24
6.59 0.8.0 (2017-07-05) . . . 24
6.60 0.7.2 (2017-07-04) . . . 24
6.61 0.7.1 (2017-07-04) . . . 24
6.62 0.7.0 (2017-07-03) . . . 24
6.63 0.6.2 (2017-06-29) . . . 24
6.64 0.6.1 (2017-06-29) . . . 24
6.65 0.6.0 (2017-06-29) . . . 25
6.66 0.5.5 (2017-06-15) . . . 25

6.67 0.5.4 (2017-06-15) . . . 25
6.68 0.5.3 (2017-06-14) . . . 25
6.69 0.5.0 (2017-06-14) . . . 25
6.70 0.4.0 (2017-06-13) . . . 25
6.71 0.3.0 (2017-06-13) . . . 25
6.72 0.2.0 (2017-05-26) . . . 25
6.73 0.1.1 (2017-05-26) . . . 25

7 API Reference **27**
7.1 aiohttp_json_api package . . . 27
7.2 aiohttp_json_api.abc package . . . 53
7.3 aiohttp_json_api.fields package . . . 59

8 Indices and tables **75**

Python Module Index **77**

Contents:

CHAPTER 1

JSON API implementation for aiohttp

1.1 Introduction

This project heavily inspired by py-jsonapi (author is Benedikt Schmitt). Some parts of this project is improved and refactored dev-schema branch of **py-jsonapi**. At beginning of aiohttp-json-api this branch was a great, but not finished implementation of JSON API with *schema controllers*. Also, py-jsonapi is not asynchronous and use inside self-implemented Request/Response classes.

Some of base entities of py-jsonapi was replaced with aiohttp server's objects, some of it was divided into new separate entities (e.g. *JSONAPIContext* or *Registry*).

- Free software: MIT license
- Documentation: https://aiohttp-json-api.readthedocs.io

1.2 Requirements

- **Python 3.6** or newer
- aiohttp
- inflection
- multidict
- jsonpointer
- dateutil
- trafaret
- python-mimeparse

1.3 Todo

- Tutorials
- Improve documentation
- Tests
- Features description
- Customizable payload keys inflection (default is *dasherize* <-> *underscore*)
- Support for JSON API extensions (bulk creation etc.)
- Polymorphic relationships

1.4 Credits

This package was created with Cookiecutter and the cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install aiohttp JSON API, run this command in your terminal:

```
$ pip install aiohttp_json_api
```

This is the preferred method to install aiohttp JSON API, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for aiohttp JSON API can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vovanbo/aiohttp_json_api
```

Or download the tarball:

```
$ curl  -OL https://github.com/vovanbo/aiohttp_json_api/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.3 Default setup of resources, routes and handlers

Resource name	Method	Route	Handler
jsonapi.collection	GET	`/{type}`	*get_collection()*
jsonapi.collection	POST	`/{type}`	*post_resource()*
jsonapi.resource	GET	`/{type}/{id}`	*get_resource()*
jsonapi.resource	PATCH	`/{type}/{id}`	*patch_resource()*
jsonapi.resource	DELETE	`/{type}/{id}`	*delete_resource()*
jsonapi.relationships	GET	`/{type}/{id}/relationships/{relation}`	*get_relationship()*
jsonapi.relationships	POST	`/{type}/{id}/relationships/{relation}`	*post_relationship()*
jsonapi.relationships	PATCH	`/{type}/{id}/relationships/{relation}`	*patch_relationship()*
jsonapi.relationships	DELETE	`/{type}/{id}/relationships/{relation}`	*delete_relationship()*
jsonapi.related	GET	`/{type}/{id}/{relation}`	*get_related()*

CHAPTER 3

Usage

Todo: Tutorials will be added soon.

At this moment, the best way to examine features of this application is looking at simple example. Models of this example related to entities from official JSON API specification (e.g. here in "Compound Documents" section).

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/vovanbo/aiohttp_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.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vovanbo/aiohttp_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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *aiohttp_json_api* for local development.

1. Fork the *aiohttp_json_api* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/aiohttp_json_api.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv aiohttp_json_api
$ cd aiohttp_json_api/
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 aiohttp_json_api tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

4.3 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 3.6 and later. Check https://travis-ci.org/vovanbo/aiohttp_json_api/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_aiohttp_json_api
```

CHAPTER 5

Credits

5.1 Development Lead

- Vladimir Bolshakov

5.2 Author of core idea

- Benedikt Schmitt

5.3 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.37.0 (2018-03-03)

- Drop Python 3.5 support
- Update documentation

6.2 0.36.1 (2018-03-03)

- Fix bug with content negotiation
- Add cherry-picked and improved helpers from python-mimeparse

6.3 0.36.0 (2018-03-02)

- Improve content negotiation (fix #185) with python-mimeparse
- Update requirements

6.4 0.35.2 (2017-12-13)

- Fix bug with wrong query parameters of links in compound documents

6.5 0.35.1 (2017-12-12)

- Fix `trafaret` requirement to 1.0.2 (included `rfc3339.Date`)

6.6 0.35.0 (2017-12-11)

BREAKING CHANGES!

- Schema is separated into Schema (marshaller) and Controller
- Request context instantiated in handlers and was renamed to `JSONAPIContext`
- Change signature of setup JSON API method in application (now we should pass a mapping between schemas and controllers)
- New abstract base class for Controller
- Schema and Controller must be initialized with only one parameter — `JSONAPIContext`
- Passing a context to almost each method of Schema no more required (context is accessible from Schema or Controller instance directly)
- Remove decorator for JSON API handlers (content negotiation moved to middleware)
- Refactored fields and schema modules
- Improved fetching of compound documents
- Examples are updated to conform with all changes in this release

6.7 0.33.1 (2017-12-06)

- Fix bug with no Accept header in request

6.8 0.33.0 (2017-12-06)

- Improve content type negotiation
- Improve documentation
- Add field for Date
- Add example based on fantasy database
- Introduce JSON API integration test suite (not done yet!)
- Improve collections helpers
- No more links normalization by default
- Move meta object at top level of result document

6.9 0.32.0 (2017-11-21)

- Constants, enums and structs refactoring *(backward incompatible)*
- Add useful typings
- Documentation fixes
- Extend development requirements

6.10 0.31.0 (2017-11-14)

- Improve performance of URL resolving again. At this time with usage of standard Python urllib
- Upgrade requirements

6.11 0.30.0 (2017-11-13)

- Improve performance of URL resolving in Link field (with cachetools)
- Upgrade requirements

6.12 0.29.2 (2017-11-02)

- Documentation improvements

6.13 0.29.1 (2017-11-02)

- Update README
- Upgrade requirements

6.14 0.29.0 (2017-11-02)

- Simple example of usage is added
- Fix bug in handler of relationship query

6.15 0.28.3 (2017-11-02)

- Fix bug with wrong read-only field error
- Don't require setup ID field in Schema to update a resource

6.16 0.28.2 (2017-11-01)

- Add multidict to requirements and to README

6.17 0.28.1 (2017-10-31)

- Fix small bug with wrong empty sorting in RequestContext

6.18 0.28.0 (2017-10-31)

- Add support for customizable inflection of fields from query string.
- Convert static methods of RequestContext to class methods
- Update docs of RequestContext methods

6.19 0.27.1 (2017-10-31)

- Fix packaging of ABCs and compats

6.20 0.27.0 (2017-10-31)

- Abstract base classes refactoring (separate Field and Schema ABCs)
- Fix bug with compound documents in case models has no property "resource_id"
- Remove buggy helper to check subclass of any instance
- Decompose setup application method to increase readability
- Properly error raised in jsonapi_handler decorator
- Use one field instead of two to check what type of relation have the field

6.21 0.26.0 (2017-10-30)

- Properly use abstract base classes. Inherit SchemaABC from ABC.
- Rename resource validation methods *(backward incompatible)*

6.22 0.25.0 (2017-10-18)

- Add Tuple field
- Fix bug with List field items enumeration
- Fix wrong conversion of Decimal field to string on deserialization
- Add yarl (>=0.13) to requirements

6.23 0.24.1 (2017-10-12)

- Add support for length range specifying for List field

6.24 0.24.0 (2017-10-04)

- Convert document render utility to async coroutine *(backward incompatible)*
- Rename Error class property "json" to "as_dict" to clarify

6.25 0.23.0 (2017-10-03)

- Use MultiDict for request context filters and FilterRule tuple *(backward incompatible)*
- Debug info on request context creation

6.26 0.22.2 (2017-09-27)

- Add support for nullable List field

6.27 0.22.1 (2017-09-25)

- Fix bug with wrong exit from compound documents fetch utility ("return" instead of "break")

6.28 0.22.0 (2017-09-22)

- Remove recursive fetching of compound documents. Replace it with simple loop.

6.29 0.21.2 (2017-09-22)

- Fix bug with fetching compound documents when query parameter "include" contains the same relation twice and more.

6.30 0.21.1 (2017-09-19)

- Fix bug with non-underscored relation name in relationship handlers

6.31 0.21.0 (2017-09-19)

- Add support for field names conversion passed to "include" request context
- Update development requirements

6.32 0.20.2 (2017-08-30)

- Avoid assertion in Registry ensure identifier method
- Make Schema getter of object id static
- Avoid to filter out empty fields of rendered documents (less memory and faster)
- Get id field of schema more safely in URI resource ID validator

6.33 0.20.1 (2017-08-15)

- Add support for load only fields (like a Marshmallow)

6.34 0.20.0 (2017-08-14)

- Asynchronous validators support
- Routes namespace can be customized
- Relative links support

6.35 0.19.1 (2017-08-10)

- Improve serialization result default keys creation

6.36 0.19.0 (2017-08-10)

- Refactor schema serializer to fix bug with no resource link in result
- Clean-up validation of expected ID in pre-validaiton of resource
- Use status property of ErrorList in error middleware to return HTTP status
- Remove default getter from Link field, because it doesn't used anymore

6.37 0.18.1 (2017-08-09)

- Migrate to trafaret >= 0.11.0
- Fix requirement of trafaret to version greater than 0.11.0

6.38 0.18.0 (2017-08-09)

- Properly handle missing values in deserialization and validation

6.39 0.17.1 (2017-07-31)

- Add support for validation of Relationships ID field

6.40 0.17.0 (2017-07-28)

- Normalize resource_id parameter usage in all mutation methods.
- Add fetch_resource schema coroutine to receive resource instance by ID.
- Add separate method for mapping deserialized data to schema.
- Context is required parameter for deserialization schema method.
- Move docs to ABC schema.
- Properly handle allow_none parameter for UUID field

6.41 0.16.2 (2017-07-24)

- Fix arguments passed to validators

6.42 0.16.1 (2017-07-24)

- Pass context to value setter in update methods

6.43 0.16.0 (2017-07-22)

- Strict member names and type checking to conform JSON API requirements (routes and schema level). See also: http://jsonapi.org/format/#document-member-names
- Strict check for overrides of handlers
- Improve debug logging

6.44 0.15.2 (2017-07-21)

- Initialize default relationships links in meta-class, to avoid bug with empty names of relationships fields

6.45 0.15.1 (2017-07-19)

- Rename resource ID parameter of query_resource schema' method.

6.46 0.15.0 (2017-07-18)

- Pagination is initialized from request by default. Remove separate class method of BasePagination to initialize pagination instance
- Improve value validation error for absent fields
- Improve validation error of string field with choices

6.47 0.14.0 (2017-07-13)

- Customisable JSON API handlers support
- DRY in handlers
- Move context builder from middleware to jsonapi_handler decorator
- Request context receive optional resource_type now

6.48 0.13.0 (2017-07-12)

- Revert back to asynchronous setters, because it's used in update relationships and it might want to query DB, for example

6.49 0.12.0 (2017-07-12)

- Base Registry class from UserDict, so Registry is a dict with ensure_identifier method.
- More strict typing checks on setup.

6.50 0.11.1 (2017-07-11)

- Fix bug with mutation not cloned resource in method for delete relationship
- Require JSON API content type on delete relationships

6.51 0.11.0 (2017-07-11)

- Method for update return original and updated resource as result. Updated resource is created via deepcopy. It will be useful to determine returned HTTP status
- Fix bug with enumeration (choices) in String field
- Fix bug with context event setup for OPTIONS, HEAD and another request methods not used in JSON API

6.52 0.10.0 (2017-07-10)

- Mass refactoring of schema, fields, validation and decorators
- Generic approach to setup Schema decorators is used (inspired by Marshmallow)
- Fields are used only for encode/decode now (with pre/post validation). Additional validators for fields must be created on schema level
- Custom JSON encoder with support JSONPointer serialization
- Remove boltons from requirements
- No more remap input data dictionary with key names to underscores conversion.
- Add helpers “first” and “make_sentinel” (cherry-picked from boltons)
- Fix enumeration (choices) support in String field

6.53 0.9.3 (2017-07-06)

- Setup content-type validation on mutation API methods (application/vnd.api+json is required)
- Properly get and encode relationships fields
- Update docs and typing for ensure_identifier Registry’s method

6.54 0.9.2 (2017-07-06)

- Fix bugs related to Python 3.5
- Generation of documentation on RTD is fixed

6.55 0.9.1 (2017-07-06)

- Python 3.5 compatibility changes

6.56 0.9.0 (2017-07-06)

- Handle aiohttp-json-api exceptions and errors in middleware. If exceptions is not related to JSON API errors, then exception is reraised
- Huge refactoring of RequestContext
- No more use of boltons cachedproperties, instead use parsing static methods related to each request context’ entity
- Update docs for RequestContext methods
- Add typings to RequestContext

6.57 0.8.2 (2017-07-05)

- Properly handle error with wrong relation name (raise HTTP 400)

6.58 0.8.1 (2017-07-05)

- Fix bdist_wheel python tag to support Python 3.5

6.59 0.8.0 (2017-07-05)

- Python 3.5 support (avoid usage of Python 3.6 format strings)
- Registry is plain object now
- Custom Registry support (*registry_class* parameter in `aiohttp_json_api.setup_jsonapi` method)
- Log debugging information at start about registered resources, methods and routes
- Use OrderedDict inside SchemaMeta

6.60 0.7.2 (2017-07-04)

- Fix bug with JSONPointer when part passed via __truediv__ is integer
- Validate relationship object before adding relationship in ToMany field

6.61 0.7.1 (2017-07-04)

- Fix bugs with validation of resource identifier in relationships fields
- Add typings for base fields

6.62 0.7.0 (2017-07-03)

- Setup of JSON API must be imported from package directly
- Fix bugs with decode fields and allow None values

6.63 0.6.2 (2017-06-29)

- Update HISTORY

6.64 0.6.1 (2017-06-29)

- Fix bug with Enum choices of String field

6.65 0.6.0 (2017-06-29)

- Return resource in update method of Schema class. This will be helpful in inherit classes of Schemas.

6.66 0.5.5 (2017-06-15)

- Setup auto-deploy to PyPI in Travis CI

6.67 0.5.4 (2017-06-15)

- Initial release on PyPI

6.68 0.5.3 (2017-06-14)

- Improve documentation

6.69 0.5.0 (2017-06-14)

- Don't use attrs package anymore
- Refactor requirements (move it into *setup.py*)

6.70 0.4.0 (2017-06-13)

- Schema imports refactoring (e.g. don't use `aiohttp_json_api.schema.schema.Schema` anymore)

6.71 0.3.0 (2017-06-13)

- Upgrade requirements

6.72 0.2.0 (2017-05-26)

- Fix setup.py
- Add test for Decimal trafaret field

6.73 0.1.1 (2017-05-26)

- Dirty initial version

CHAPTER 7

API Reference

Contents:

7.1 aiohttp_json_api package

7.1.1 Submodules

Common constants, enumerations and structures.

aiohttp_json_api.common.**ALLOWED_MEMBER_NAME_REGEX = re.compile('^[a-zA-Z0-9]([a-zA-Z0-9\\-_**
Compiled regexp of rule

aiohttp_json_api.common.**ALLOWED_MEMBER_NAME_RULE = '[a-zA-Z0-9]([a-zA-Z0-9\\-_]+[a-zA-Z0-9**
Regular expression rule for check allowed fields and types names

class aiohttp_json_api.common.**Event**
Bases: enum.Flag

Request event enumeration.

ALWAYS = 15

DELETE = 8

GET = 1

NEVER = 16

PATCH = 4

POST = 2

class aiohttp_json_api.common.**FilterRule**(*name*, *value*)
Bases: tuple

Filter rule

name
Alias for field number 0

value
Alias for field number 1

aiohttp_json_api.common.**JSONAPI = 'jsonapi'**
Key of JSON API stuff in aiohttp.web.Application

aiohttp_json_api.common.**JSONAPI_CONTENT_TYPE = 'application/vnd.api+json'**
JSON API Content-Type by specification

class aiohttp_json_api.common.**Relation**
Bases: enum.Enum

Types of relations enumeration.

TO_MANY = 2

TO_ONE = 1

class aiohttp_json_api.common.**ResourceID**(*type*, *id*)
Bases: tuple

JSON API resource identifier

id
Alias for field number 1

type
Alias for field number 0

class aiohttp_json_api.common.**SortDirection**
Bases: enum.Enum

Sorting direction enumeration.

ASC = '+'

DESC = '-'

class aiohttp_json_api.common.**Step**
Bases: enum.Enum

Marshalling step enumeration.

AFTER_DESERIALIZATION = 2

AFTER_SERIALIZATION = 4

BEFORE_DESERIALIZATION = 1

BEFORE_SERIALIZATION = 3

aiohttp_json_api.common.**logger = <Logger aiohttp-json-api (WARNING)>**
Logger instance

Request context.

class aiohttp_json_api.context.**JSONAPIContext**(*request*, *resource_type=None*)
Bases: object

JSON API request context.

FIELDS_RE = re.compile('fields\\[(?P<name>\\w[-\\w_]*)\\]')

FILTER_KEY = re.compile('filter\\[(?P<field>\\w[-\\w_]*)\\]')

FILTER_VALUE = re.compile('(?P<name>[a-z]+):(?P<value>.*)')

app

controller

classmethod convert_field_name(*field_name*)

event

fields

filters

get_filter(*field*, *name*, *default=None*)
Get filter from request context by name and field.

If the filter *name* has been applied on the *field*, the *filter* is returned and *default* otherwise.

Parameters

- **field** (*str*) – Name of field
- **name** (*str*) – Name of filter
- **default** (*Any*) – A fallback rule value for the filter.

Return type Any

get_order(*field*, *default=<SortDirection.ASC: '+'>*)
Get sorting order of field from request context.

Checks if a sort criterion (+ or −) for the *field* exists and returns it.

Parameters

- **Tuple[str, ..]] field** (*Union[str,*) –
- **default** (SortDirection) – Returned, if no criterion is set by the request.

Return type *SortDirection*

has_filter(*field*, *name*)
Check current context for existing filters of field.

Returns true, if the filter *name* has been applied at least once on the *field*.

Parameters

- **field** (*str*) – Name of field
- **name** (*str*) – Name of filter

Return type bool

include

inflect()
Make an underscored, lowercase form from the expression in the string.

Example:

```
>>> underscore("DeviceType")
"device_type"
```

As a rule of thumb you can think of underscore() as the inverse of camelize(), though there are cases where that does not hold:

```
>>> camelize(underscore("IOError"))
"IoError"
```

pagination

classmethod parse_request_fields(*request*)

Parse sparse fields from request query string.

Used for determine fields, which should be included in the response (sparse fieldset).

```
>>> from aiohttp_json_api.context import JSONAPIContext
>>> from aiohttp.test_utils import make_mocked_request
>>> request = make_mocked_request('GET', '/api/User?fields[User]=email,name&
↪fields[Post]=comments')
>>> JSONAPIContext.parse_request_fields(request)
OrderedDict([('User', ('email', 'name')), ('Post', ('comments',))])
```

Seealso http://jsonapi.org/format/#fetching-sparse-fieldsets

Return type Mutablemapping[str, Tuple[str, ...]]

classmethod parse_request_filters(*request*)

Parse filters from request query string.

Hint: Please note, that the *filter* strategy is not defined by the JSON API specification and depends on the implementation. If you want to use another filter strategy, feel free to **override** this method.

Returns a MultiDict with field names as keys and rules as values. Rule value is JSON deserialized from query string.

Filters can be applied using the query string.

```
>>> from aiohttp_json_api.context import JSONAPIContext
>>> from aiohttp.test_utils import make_mocked_request

>>> request = make_mocked_request('GET', '/api/User/?filter[name]=endswith:
↪"Simpson"')
>>> JSONAPIContext.parse_request_filters(request)
<MultiDict('name': FilterRule(name='endswith', value='Simpson'))>

>>> request = make_mocked_request('GET', '/api/User/?filter[name]=endswith:
↪"Simpson"&filter[name]=in:["Some","Names"]')
>>> JSONAPIContext.parse_request_filters(request)
<MultiDict('name': FilterRule(name='endswith', value='Simpson'), 'name':␣
↪FilterRule(name='in', value=['Some', 'Names']))>

>>> request = make_mocked_request('GET', '/api/User/?filter[name]=in:["Homer␣
↪Simpson", "Darth Vader"]')
>>> JSONAPIContext.parse_request_filters(request)
<MultiDict('name': FilterRule(name='in', value=['Homer Simpson', 'Darth Vader
↪']))>

>>> request = make_mocked_request('GET', '/api/User/?filter[some-
↪field]=startswith:"lisa"&filter[another-field]=lt:20')
>>> JSONAPIContext.parse_request_filters(request)
<MultiDict('some_field': FilterRule(name='startswith', value='lisa'),
↪'another_field': FilterRule(name='lt', value=20))>
```

The general syntax is:

```
"?filter[field]=name:rule"
```

where *rule* is a JSON value.

Raises

- ***HTTPBadRequest*** – If the rule of a filter is not a JSON object.
- ***HTTPBadRequest*** – If a filter name contains invalid characters.

Return type Mutablemapping[str, *FilterRule*]

classmethod parse_request_includes(*request*)

Parse compound documents parameters from request query string.

Returns the names of the relationships, which should be included into the response.

```
>>> from aiohttp_json_api.context import JSONAPIContext
>>> from aiohttp.test_utils import make_mocked_request
>>> request = make_mocked_request('GET', '/api/Post?include=author,comments.
↪author,some-field.nested')
>>> JSONAPIContext.parse_request_includes(request)
(('author',), ('comments', 'author'), ('some_field', 'nested'))
```

Seealso http://jsonapi.org/format/#fetching-includes

Return type Tuple[Tuple[str, ...], ...]

classmethod parse_request_sorting(*request*)

Parse sorting parameters of fields from request query string.

Returns a mapping with tuples as keys, and values with SortDirection, describing how the output should be sorted.

```
>>> from aiohttp_json_api.context import JSONAPIContext
>>> from aiohttp.test_utils import make_mocked_request
>>> request = make_mocked_request('GET', '/api/Post?sort=name,-age,+comments.
↪count')
>>> JSONAPIContext.parse_request_sorting(request)
OrderedDict([(('name',), <SortDirection.ASC: '+'>), (('age',), <SortDirection.
↪DESC: '-'>), (('comments', 'count'), <SortDirection.ASC: '+'>)])
```

Seealso http://jsonapi.org/format/#fetching-sorting

Return type Mutablemapping[Tuple[str, ...], *SortDirection*]

registry

request

resource_type

schema

Return type Optional[*SchemaABC*]

sorting

class aiohttp_json_api.controller.**DefaultController**(*context*)

Bases: *aiohttp_json_api.abc.contoller.ControllerABC*

add_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-to-many-relationships

Adds the members specified in the JSON API relationship object *data* to the relationship, unless the relationships already exist.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by `fetch_resource()` in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.

static default_add(*field*, *resource*, *data*, *sp*, ***kwargs*)

static default_include(*field*, *resources*, ***kwargs*)

static default_query(*field*, *resource*, ***kwargs*)

static default_remove(*field*, *resource*, *data*, *sp*, ***kwargs*)

fetch_compound_documents(*field*, *resources*, ***kwargs*)

See also:

http://jsonapi.org/format/#fetching-includes

Fetches the related resources. The default method uses the controller's *default_include()*. **Can be overridden.**

Parameters

- **field** (FieldABC) – Relationship field.
- **resources** – A list of resources.
- **context** (JSONAPIContext) – Request context instance.
- **rest_path** (*list*) – The name of the relationships of the returned relatives, which will also be included.

Return type list

Returns A list with the related resources. The list is empty or has exactly one element in the case of *to-one* relationships. If *to-many* relationships are paginated, the relatives from the first page should be returned.

query_relatives(*field*, *resource*, ***kwargs*)

Controller for the *related* endpoint of the relationship with then name *relation_name*.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by `fetch_resource()` in handler.

remove_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-to-many-relationships

Deletes the members specified in the JSON API relationship object *data* from the relationship.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by `fetch_resource()` in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance.

update_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-relationships

Updates the relationship with the JSON API name *relation_name*.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by `fetch_resource()` in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.

update_resource(*resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating

Updates an existing *resource*. **You should overridde this method** in order to save the changes in the database.

The default implementation uses the `BaseField` descriptors to update the resource.

Parameters

- **resource_id** – The id of the resource
- **data** (*dict*) – The JSON API resource object with the update information
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance

JSON encoder extension.

class aiohttp_json_api.encoder.**JSONEncoder**(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Bases: `json.encoder.JSONEncoder`

Overloaded JSON encoder with JSONPointer support.

default(*o*)

Add JSONPointer serializing support to default json.dumps.

Errors.

exception aiohttp_json_api.errors.**Error**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: Exception

Base class for all exceptions thrown by the API.

All subclasses of this exception are catches and converted into a response. All other exceptions will be replaced by an HTTPInternalServerError exception.

Seealso http://jsonapi.org/format/#errors

as_dict
Represent instance of Error as dictionary.

status = 500

exception aiohttp_json_api.errors.**ErrorList**(*errors=None*)

Bases: Exception

Exception contains list of errors.

Can be used to store a list of exceptions, which occur during the execution of a request.

Seealso http://jsonapi.org/format/#error-objects

Seealso http://jsonapi.org/examples/#error-objects-multiple-errors

append(*error*)
Append the *Error* error to the error list.

Parameters **error** (Error) – JSON API Error instance

extend(*errors*)
Append errors to the list.

Parameters **errors** – *ErrorList* or a sequence of *Error*.

json
Create the JSON API error object.

Seealso http://jsonapi.org/format/#error-objects

status
Return the most specific HTTP status code for all errors.

For single error in list returns its status. For many errors returns maximal status code.

exception aiohttp_json_api.errors.**HTTPBadRequest**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 400 Bad Request.

The request could not be fulfilled due to the incorrect syntax of the request.

status = 400

exception aiohttp_json_api.errors.**HTTPUnauthorized**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 401 Unauthorized.

The requester is not authorized to access the resource. This is similar to 403 but is used in cases where authentication is expected but has failed or has not been provided.

status = 401

exception aiohttp_json_api.errors.**HTTPForbidden**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 403 Forbidden.

The request was formatted correctly but the server is refusing to supply the requested resource. Unlike 401, authenticating will not make a difference in the server's response.

status = 403

exception aiohttp_json_api.errors.**HTTPNotFound**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 404 Not found.

The resource could not be found. This is often used as a catch-all for all invalid URIs requested of the server.

status = 404

exception aiohttp_json_api.errors.**HTTPMethodNotAllowed**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 405 Method not allowed.

The resource was requested using a method that is not allowed. For example, requesting a resource via a POST method when the resource only supports the GET method.

status = 405

exception aiohttp_json_api.errors.**HTTPNotAcceptable**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 406 Not acceptable.

The resource is valid, but cannot be provided in a format specified in the Accept headers in the request.

status = 406

exception aiohttp_json_api.errors.**HTTPConflict**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 409 Conflict.

The request cannot be completed due to a conflict in the request parameters.

status = 409

exception aiohttp_json_api.errors.**HTTPGone**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 410 Gone.

The resource is no longer available at the requested URI and no redirection will be given.

status = 410

exception aiohttp_json_api.errors.**HTTPPreConditionFailed**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 412 Precondition failed.

The server does not meet one of the preconditions specified by the client.

status = 412

exception aiohttp_json_api.errors.**HTTPUnsupportedMediaType**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 415 Unsupported media type.

The client provided data with a media type that the server does not support.

status = 415

exception aiohttp_json_api.errors.**HTTPUnprocessableEntity**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 422 Unprocessable entity.

The request was formatted correctly but cannot be processed in its current form. Often used when the specified parameters fail validation errors.

WebDAV - RFC 4918

status = 422

exception aiohttp_json_api.errors.**HTTPLocked**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 423 Locked.

The requested resource was found but has been locked and will not be returned.

WebDAV - RFC 4918

status = 423

exception aiohttp_json_api.errors.**HTTPFailedDependency**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 424 Failed dependency.

The request failed due to a failure of a previous request.

WebDAV - RFC 4918

status = 424

exception aiohttp_json_api.errors.**HTTPTooManyRequests**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 429 Too many requests.

The user has sent too many requests in a given amount of time ("rate limiting").

Additional HTTP Status Codes - RFC 6585

status = 429

exception aiohttp_json_api.errors.**HTTPInternalServerError**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 500 Internal server error.

A generic status for an error in the server itself.

status = 500

exception aiohttp_json_api.errors.**HTTPNotImplemented**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 501 Not implemented.

The server cannot respond to the request. This usually implies that the server could possibly support the request in the future — otherwise a 4xx status may be more appropriate.

status = 501

exception aiohttp_json_api.errors.**HTTPBadGateway**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 502 Bad gateway.

The server is acting as a proxy and did not receive an acceptable response from the upstream server.

status = 502

exception aiohttp_json_api.errors.**HTTPServiceUnavailable**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 503 Service unavailable.

The server is down and is not accepting requests.

status = 503

exception aiohttp_json_api.errors.**HTTPGatewayTimeout**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 504 Gateway timeout.

The server is acting as a proxy and did not receive a response from the upstream server.

status = 504

exception aiohttp_json_api.errors.**HTTPVariantAlsoNegotiates**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 506 Variant also negotiates.

Transparent content negotiation for the request results in a circular reference.

status = 506

exception aiohttp_json_api.errors.**HTTPInsufficientStorage**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 507 Insufficient storage.

The user or server does not have sufficient storage quota to fulfill the request.

WebDAV - RFC 4918

status = 507

exception aiohttp_json_api.errors.**HTTPNotExtended**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.Error*

HTTP 510 Not extended.

Further extensions to the request are necessary for it to be fulfilled.

status = 510

exception aiohttp_json_api.errors.**ValidationError**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.HTTPBadRequest*

JSON API validation error. HTTP 400 Bad request.

Raised, if the structure of a json document in a request body is invalid.

Please note, that this does not include semantic errors, like an unknown typename.

This type of exception is used often in the `jsonapi.validator` and `jsonapi.validation` modules.

Seealso http://jsonapi.org/format/#document-structure

exception aiohttp_json_api.errors.**InvalidType**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.ValidationError*

JSON API invalid type error. HTTP 400 Bad request.

Raised if an input value (part of a JSON API document) has the wrong type.

This type of exception is often raised during decoding.

Seealso http://jsonapi.org/format/#document-structure

exception aiohttp_json_api.errors.**InvalidValue**(*, *id_=None*, *about=''*, *code=None*, *title=None*, *detail=''*, *source_parameter=None*, *source_pointer=None*, *meta=None*)

Bases: *aiohttp_json_api.errors.ValidationError*

JSON API invalid value error. HTTP 400 Bad request.

Raised if an input value (part of a JSON API document) has an invalid value.

Seealso http://jsonapi.org/format/#document-structure

exception aiohttp_json_api.errors.**UnresolvableIncludePath**(*path*, ***kwargs*)

Bases: *aiohttp_json_api.errors.HTTPBadRequest*

JSON API unresolvable include path error. HTTP 400 Bad request.

Raised if an include path does not exist. The include path is part of the `include` query argument. (An include path is invalid, if a relationship mentioned in it is not defined on a resource).

Seealso http://jsonapi.org/format/#fetching-includes

exception aiohttp_json_api.errors.**UnsortableField**(*type*, *field*, ***kwargs*)

Bases: *aiohttp_json_api.errors.HTTPBadRequest*

JSON API unsortable field. HTTP 400 Bad request.

If a field is used as sort key, but sorting is not supported on this field.

Seealso http://jsonapi.org/format/#fetching-sorting

exception aiohttp_json_api.errors.**UnsortableField**(*type*, *field*, ***kwargs*)

Bases: *aiohttp_json_api.errors.HTTPBadRequest*

JSON API unsortable field. HTTP 400 Bad request.

If a field is used as sort key, but sorting is not supported on this field.

Seealso http://jsonapi.org/format/#fetching-sorting

exception aiohttp_json_api.errors.**ResourceNotFound**(*type*, *id*, ***kwargs*)

Bases: *aiohttp_json_api.errors.HTTPNotFound*

JSON API resource not found error. HTTP 404 Not found.

Raised, if a resource does not exist.

Handlers.

aiohttp_json_api.handlers.**get_collection**(*request*)

Fetch resources collection, render JSON API document and return response.

Uses the `query_collection()` method of the schema to query the resources in the collection.

Seealso http://jsonapi.org/format/#fetching

aiohttp_json_api.handlers.**post_resource**(*request*)

Create resource, render JSON API document and return response.

Uses the `create_resource()` method of the schema to create a new resource.

Seealso http://jsonapi.org/format/#crud-creating

aiohttp_json_api.handlers.**get_resource**(*request*)
Get single resource, render JSON API document and return response.

Uses the `query_resource()` method of the schema to query the requested resource.

Seealso http://jsonapi.org/format/#fetching-resources

aiohttp_json_api.handlers.**patch_resource**(*request*)
Update resource (via PATCH), render JSON API document and return response.

Uses the `update_resource()` method of the schema to update a resource.

Seealso http://jsonapi.org/format/#crud-updating

aiohttp_json_api.handlers.**delete_resource**(*request*)
Remove resource.

Uses the `delete_resource()` method of the schema to delete a resource.

Seealso http://jsonapi.org/format/#crud-deleting

aiohttp_json_api.handlers.**get_relationship**(*request*)
Get relationships of resource.

Parameters **request** (`Request`) – Request instance

Returns Response

aiohttp_json_api.handlers.**post_relationship**(*request*)
Create relationships of resource.

Uses the `add_relationship()` method of the schemato add new relationships.

Seealso http://jsonapi.org/format/#crud-updating-relationships

aiohttp_json_api.handlers.**patch_relationship**(*request*)
Update relationships of resource.

Uses the `update_relationship()` method of the schema to update the relationship.

Seealso http://jsonapi.org/format/#crud-updating-relationships

aiohttp_json_api.handlers.**delete_relationship**(*request*)
Remove relationships of resource.

Uses the `delete_relationship()` method of the schema to update the relationship.

Seealso http://jsonapi.org/format/#crud-updating-relationships

aiohttp_json_api.handlers.**get_related**(*request*)
Get related resources.

Uses the `query_relative()` method of the schema to query the related resource.

Seealso http://jsonapi.org/format/#fetching

Helpers.

aiohttp_json_api.helpers.**best_match**(*supported*, *header*)
Return mime-type with the highest quality ('q') from list of candidates. Takes a list of supported mime-types and finds the best match for all the media-ranges listed in header. The value of header must be a string that conforms to the format of the HTTP Accept: header. The value of 'supported' is a list of mime-types. The list of supported mime-types should be sorted in order of increasing desirability, in case of a situation where there is a tie.

Cherry-picked from python-mimeparse and improved.

```
>>> best_match(['application/xbel+xml', 'text/xml'],
               'text/*;q=0.5,*/*; q=0.1')
('text/xml', ('text', '*', {'q': '0.5'}))
```

Return type Tuple[str, Optional[Tuple[str, str, Dict[str, str]]]]

aiohttp_json_api.helpers.**ensure_collection**(*value*, *exclude=()*)
Ensure value is collection.

aiohttp_json_api.helpers.**first**(*iterable*, *default=None*, *key=None*)
Return first element of *iterable*.

Return first element of *iterable* that evaluates to `True`, else return `None` or optional *default*.

```
>>> first([0, False, None, [], (), 42])
42
>>> first([0, False, None, [], ()]) is None
True
>>> first([0, False, None, [], ()], default='ohai')
'ohai'
>>> import re
>>> m = first(re.match(regex, 'abc') for regex in ['b.*', 'a(.*)'])
>>> m.group(1)
'bc'
```

The optional *key* argument specifies a one-argument predicate function like that used for *filter()*. The *key* argument, if supplied, should be in keyword form. For example, finding the first even number in an iterable:

```
>>> first([1, 1, 3, 4, 5], key=lambda x: x % 2 == 0)
4
```

Contributed by Hynek Schlawack, author of the original standalone module

aiohttp_json_api.helpers.**get_mime_type_params**(*mime_type*)

aiohttp_json_api.helpers.**get_processors**(*obj*, *tag*, *field*, *default=None*)

aiohttp_json_api.helpers.**get_router_resource**(*app*, *resource*)
Return route of JSON API application for resource.

aiohttp_json_api.helpers.**is_collection**(*obj*, *exclude=()*)
Return True if `obj` is a collection type.

aiohttp_json_api.helpers.**is_generator**(*obj*)
Return True if `obj` is a generator.

aiohttp_json_api.helpers.**is_indexable_but_not_string**(*obj*)
Return True if `obj` is indexable but isn't a string.

aiohttp_json_api.helpers.**is_iterable_but_not_string**(*obj*)
Return True if `obj` is an iterable object that isn't a string.

aiohttp_json_api.helpers.**make_sentinel**(*name='_MISSING'*, *var_name=None*)
Create sentinel instance.

Creates and returns a new **instance** of a new class, suitable for usage as a "sentinel", a kind of singleton often used to indicate a value is missing when `None` is a valid input.

```
>>> make_sentinel(var_name='_MISSING')
_MISSING
```

The most common use cases here in project are as default values for optional function arguments, partly because of its less-confusing appearance in automatically generated documentation. Sentinels also function well as placeholders in queues and linked lists.

Note: By design, additional calls to `make_sentinel` with the same values will not produce equivalent objects.

```
>>> make_sentinel('TEST') == make_sentinel('TEST')
False
>>> type(make_sentinel('TEST')) == type(make_sentinel('TEST'))
False
```

Parameters

- **name** (*str*) – Name of the Sentinel
- **var_name** (*str*) – Set this name to the name of the variable in its respective module enable pickleability.

aiohttp_json_api.helpers.**quality_and_fitness_parsed**(*mime_type*, *parsed_ranges*)

Find the best match for a mime-type amongst parsed media-ranges.

Find the best match for a given mime-type against a list of media_ranges that have already been parsed by parse_media_range(). Returns a tuple of the fitness value and the value of the 'q' quality parameter of the best match, or (-1, 0) if no match was found. Just as for quality_parsed(), 'parsed_ranges' must be a list of parsed media ranges.

Cherry-picked from python-mimeparse and improved.

Return type Tuple[Tuple[float, int], Optional[Tuple[str, str, Dict[str, str]]]]

Extended JSONPointer from python-json-pointer

class aiohttp_json_api.jsonpointer.**JSONPointer**(*pointer*)

Bases: jsonpointer.JsonPointer

Middleware.

aiohttp_json_api.middleware.**jsonapi_middleware**(*app*, *handler*)

Middleware for handling JSON API errors.

Pagination

This module contains helper for the pagination feature: http://jsonapi.org/format/#fetching-pagination

We have built-in support for:

- *limit*, *offset* based pagination (*LimitOffset*),
- *number*, *size* based pagination (*NumberSize*),
- and *cursor* based pagination (*Cursor*).

All helpers have a similar interface. Here is an example for the *NumberSize* pagination:

```
>>> from aiohttp.test_utils import make_mocked_request
>>> from aiohttp_json_api.pagination import NumberSize
>>> request = make_mocked_request('GET', 'http://example.org/api/Article/?sort=date_
↪added')
>>> p = NumberSize(request, total_resources=106)
>>> p.links()
{
    'self': 'http://example.org/api/Article/?sort=date_added&page%5Bnumber%5D=0&page
↪%5Bsize%5D=25',
    'first': 'http://example.org/api/Article/?sort=date_added&page%5Bnumber%5D=0&page
↪%5Bsize%5D=25',
    'last': 'http://example.org/api/Article/?sort=date_added&page%5Bnumber%5D=4&page
↪%5Bsize%5D=25',
    'next': 'http://example.org/api/Article/?sort=date_added&page%5Bnumber%5D=1&page
↪%5Bsize%5D=25'
}
>>> p.meta()
{'total-resources': 106, 'last-page': 4, 'page-number': 0, 'page-size': 25}
```

aiohttp_json_api.pagination.**DEFAULT_LIMIT = 25**
The default number of resources on a page.

class aiohttp_json_api.pagination.**PaginationABC**(*request*)
Bases: abc.ABC

Pagination abstract base class.

links()
Return pagination links.

Must be overridden.

A dictionary, which must be included in the top-level *links object*. It contains these keys:

- *self* The link to the current page
- *first* The link to the first page
- *last* The link to the last page
- *prev* The link to the previous page (only set, if a previous page exists)
- *next* The link to the next page (only set, if a next page exists)

Return type Mutablemapping[~KT, ~VT]

meta()
Return meta object of pagination.

Must be overridden.

A dictionary, which must be included in the top-level *meta object*.

Return type Mutablemapping[~KT, ~VT]

page_link(***kwargs*)
Return link to page.

Uses the uri and replaces the *page* query parameters with the values in *pagination*.

```
pager.page_link({"offset": 10, "limit": 5})
pager.page_link({"number": 10, "size": 5})
pager.page_link({"cursor": 1, "limit": 5})
# ...
```

Parameters **kwargs** – Additional parameters to query string

Return type str

Returns The URL to the page

url

Return type URL

class aiohttp_json_api.pagination.**LimitOffset**(*request*, *total_resources=0*)

Bases: *aiohttp_json_api.pagination.PaginationABC*

Implements a pagination based on *limit* and *offset* values.

```
/api/Article/?sort=date_added&page[limit]=5&page[offset]=10
```

Parameters

- **limit** (*int*) – The number of resources on a page.
- **offset** (*int*) – The offset, which leads to the current page.
- **total_resources** (*int*) – The total number of resources in the collection.

links()

Return pagination links.

Must be overridden.

A dictionary, which must be included in the top-level *links object*. It contains these keys:

- *self* The link to the current page
- *first* The link to the first page
- *last* The link to the last page
- *prev* The link to the previous page (only set, if a previous page exists)
- *next* The link to the next page (only set, if a next page exists)

Return type Mutablemapping[~KT, ~VT]

meta()

Return meta object of paginator.

- *total-resources* The total number of resources in the collection
- *page-limit* The number of resources on a page
- *page-offset* The offset of the current page

Return type Mutablemapping[~KT, ~VT]

class aiohttp_json_api.pagination.**NumberSize**(*request*, *total_resources*)

Bases: *aiohttp_json_api.pagination.PaginationABC*

Implements a pagination based on *number* and *size* values.

```
/api/Article/?sort=date_added&page[size]=5&page[number]=10
```

Parameters

- **request** (*Request*) –
- **number** (*int*) – The number of the current page.
- **size** (*int*) – The number of resources on a page.
- **total_resources** (*int*) – The total number of resources in the collection.

last_page

Return the number of the last page.

Return type int

limit

Return the limit, based on the page `size`.

Return type int

links()

Return pagination links.

Must be overridden.

A dictionary, which must be included in the top-level *links object*. It contains these keys:

- *self* The link to the current page
- *first* The link to the first page
- *last* The link to the last page
- *prev* The link to the previous page (only set, if a previous page exists)
- *next* The link to the next page (only set, if a next page exists)

Return type Mutablemapping[~KT, ~VT]

meta()

Return meta object of pagination.

- *total-resources* The total number of resources in the collection
- *last-page* The index of the last page
- *page-number* The number of the current page
- *page-size* The (maximum) number of resources on a page

Return type Mutablemapping[~KT, ~VT]

offset

Return the offset.

Offset based on the page `size` and `number`.

Return type int

class aiohttp_json_api.pagination.**Cursor**(*request*, *prev_cursor=None*, *next_cursor=None*, *cursor_regex=None*)

Bases: *aiohttp_json_api.pagination.PaginationABC*

Implements a (generic) approach for a cursor based pagination.

```
/api/Article/?sort=date_added&page[limit]=5&page[cursor]=19395939020
```

Parameters

- **request** (*Request*) –
- **prev_cursor** – The cursor to the previous page
- **next_cursor** – The cursor to the next page
- **cursor_regex** (*str*) – The cursor in the query string must match this regular expression. If it doesn't, an exception is raised.

FIRST = jsonapi:first
The cursor to the first page

LAST = jsonapi:last
The cursor to the last page

links(*prev_cursor=None*, *next_cursor=None*)
Return links object of paginator.

Parameters

- **prev_cursor** (*str*) – The cursor to the previous page.
- **next_cursor** (*str*) – The cursor to the next page.

Return type Mutablemapping[~KT, ~VT]

meta()
Return meta object of paginator.

- *page-limit* The number of resources per page

Return type Mutablemapping[~KT, ~VT]

Application registry.

class aiohttp_json_api.registry.**Registry**(***kwargs*)
Bases: collections.UserDict

JSON API application registry.

This is a dictionary created on JSON API application set up. It contains a mapping between types, resource classes and schemas.

data

ensure_identifier(*obj*, *asdict=False*)
Return the identifier object for the *resource*.

(*ResourceID*)

```
>>> registry.ensure_identifier({'type': 'something', 'id': 123})
ResourceID(type='something', id='123')
```

Parameters

- **obj** – A two tuple `(typename, id)`, a resource object or a resource document, which contains the *id* and *type* key `{"type": ..., "id": ...}`.
- **asdict** (*bool*) – Return ResourceID as dictionary if true

Return type `Union[ResourceID, Dict[str, str]]`

Base schema

This module contains the base schema which implements the encoding, decoding, validation and update operations based on `fields`.

class aiohttp_json_api.schema.**BaseSchema**(*context*)

Bases: *aiohttp_json_api.abc.schema.SchemaABC*

A schema defines how we can serialize a resource and patch it. It also allows to patch a resource. All in all, it defines a **controller** for a *type* in the JSON API.

If you want, you can implement your own request handlers and only use the schema for validation and serialization.

static default_getter(*field*, *resource*, ***kwargs*)

static default_setter(*field*, *resource*, *data*, *sp*, ***kwargs*)

deserialize_resource(*data*, *sp*, ***, *expected_id=None*, *validate=True*, *validation_steps=None*)

Decodes the JSON API resource object *data* and returns a dictionary which maps the key of a field to its decoded input data.

Parameters

- **data** – The received JSON API resource object
- **sp** (`JSONPointer`) – The JSON pointer to the source of *data*.
- **context** (`JSONAPIContext`) – Request context instance
- **expected_id** (*str*) – If passed, then ID of resource will be compared with this value. This is required in update methods
- **validate** (*bool*) – Is validation required?
- **validation_steps** (*tuple*) – Required validation steps

Return type OrderedDict

Returns An ordered dictionary which maps a fields key to a two tuple `(data, sp)` which contains the input data and the source pointer to it.

classmethod get_field(*key*)

Return type *FieldABC*

static get_object_id(*resource*)

Can be overridden.

Returns the id (string) of the resource. The default implementation looks for a property `resource.id`, an id method `resource.id()`, `resource.get_id()` or a key `resource["id"]`.

Parameters **resource** – A resource object

Return type str

Returns The string representation of ID of the *resource*

classmethod get_relationship_field(*relation_name*, *source_parameter=None*)

get_value(*field*, *resource*, ***kwargs*)

map_data_to_schema(*data*)

Return type Dict[~KT, ~VT]

opts = <aiohttp_json_api.abc.schema.SchemaOpts object>

post_validate_resource(*data*)
Validates the decoded *data* of JSON API resource object.

Parameters

- **data** (*OrderedDict*) – The *memo* object returned from *deserialize_resource()*.
- **context** (JSONAPIContext) – Request context instance

pre_validate_field(*field*, *data*, *sp*)
Validates the input data for a field, **before** it is deserialized. If the field has nested fields, the nested fields are validated first.

Parameters

- **field** (BaseField) –
- **data** – The input data for the field.
- **sp** (aiohttp_json_api.jsonpointer.JSONPointer) – The pointer to *data* in the original document. If *None*, there was no input data for this field.

pre_validate_resource(*data*, *sp*, *, *expected_id=None*)
Validates a JSON API resource object received from an API client:

```
schema.pre_validate_resource(
    data=request.json["data"], sp="/data"
)
```

Parameters

- **data** – The received JSON API resource object
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance
- **expected_id** (*str*) – If passed, then ID of resrouce will be compared with this value. This is required in update methods

serialize_relationship(*relation_name*, *resource*, *, *pagination=None*)

See also:

http://jsonapi.org/format/#document-resource-object-relationships

Creates the JSON API relationship object of the relationship *relation_name*.

Parameters

- **relation_name** (*str*) – The name of the relationship

- **resource** – A resource object
- **pagination** (PaginationABC) – Describes the pagination in case of a *to-many* relationship.

Return type dict

Returns The JSON API relationship object for the relationship *relation_name* of the *resource*

serialize_resource(*resource*, ***kwargs*)

See also:

http://jsonapi.org/format/#document-resource-objects

Parameters **resource** – A resource object

Return type MutableMapping

set_value(*field*, *resource*, *data*, *sp*, ***kwargs*)

Useful typing.

aiohttp_json_api.typings.**Callee = typing.Union[typing.Callable, typing.Coroutine]**
Type for callable or co-routine

aiohttp_json_api.typings.**RequestFields**
alias of typing.MutableMapping

aiohttp_json_api.typings.**RequestFilters**
alias of typing.MutableMapping

aiohttp_json_api.typings.**RequestIncludes**
alias of typing.Tuple

aiohttp_json_api.typings.**RequestSorting**
alias of typing.MutableMapping

aiohttp_json_api.typings.**ResourceIdentifier = typing.Union[aiohttp_json_api.common.Resource**
Type for Resource identifier

Utilities related to JSON API.

aiohttp_json_api.utils.**error_to_response**(*request*, *error*)
Convert an Error or ErrorList to JSON API response.

Parameters

- **request** (*Request*) – The web request instance.
- **ErrorList] error** (*typing.Union[Error,*) – The error, which is converted into a response.

Return type Response

aiohttp_json_api.utils.**get_compound_documents**(*resources*, *ctx*)
Get compound documents of resources.

See also:

http://jsonapi.org/format/#fetching-includes

Fetches the relationship paths *paths*.

Parameters

- **resources** – A list with the primary data (resources) of the compound response document.
- **ctx** – A web Request context

Returns A two tuple with a list of the included resources and a dictionary, which maps each resource (primary and included) to a set with the names of the included relationships.

aiohttp_json_api.utils.**jsonapi_response**(*data*, *, *status=200*, *reason=None*, *headers=None*, *dumps=None*)

Return JSON API response.

Parameters

- **data** – Rendered JSON API document
- **status** – HTTP status of JSON API response
- **reason** – Readable reason of error response
- **headers** – Headers
- **dumps** – Custom JSON dumps callable

Returns Response instance

aiohttp_json_api.utils.**render_document**(*data*, *included*, *ctx*, *, *pagination=None*, *links=None*)

Render JSON API document.

Parameters

- **data** – One or many resources
- **included** – Compound documents
- **ctx** – Request context
- **pagination** – Pagination instance
- **links** – Additional links

Return type Mutablemapping[~KT, ~VT]

Returns Rendered JSON API document

aiohttp_json_api.utils.**serialize_resource**(*resource*, *ctx*)

Serialize resource by schema.

Parameters

- **resource** – Resource instance
- **ctx** – Request context

Returns Serialized resource

aiohttp_json_api.utils.**validate_uri_resource_id**(*schema*, *resource_id*)

Validate resource ID from URI.

Parameters

- **schema** – Resource schema
- **resource_id** – Resource ID

7.1.2 Module contents

JSON API implementation for aiohttp.

aiohttp_json_api.**setup_app_registry**(*app*, *registry_class*, *config*)
Set up JSON API application registry.

aiohttp_json_api.**setup_custom_handlers**(*custom_handlers*)
Set up default and custom handlers for JSON API application.

aiohttp_json_api.**setup_jsonapi**(*app*, *config*, ***, *base_path='/api'*, *version='1.0'*, *meta=None*, *context_cls=None*, *registry_class=None*, *custom_handlers=None*, *log_errors=True*, *routes_namespace=None*)
Set up JSON API in aiohttp application.

This function will setup resources, handlers and middleware.

Parameters

- **app** (*Application*) – Application instance
- **controllers** (*Sequence[DefaultController]*) – List of controllers to register in JSON API
- **base_path** (*str*) – Prefix of JSON API routes paths
- **version** (*str*) – JSON API version (used in `jsonapi` key of document)
- **meta** (*dict*) – Meta information will added to response (`meta` key of document)
- **context_cls** – Override of JSONAPIContext class (must be subclass of *JSONAPIContext*)
- **registry_class** – Override of Registry class (must be subclass of *Registry*)
- **custom_handlers** – Sequence or mapping with overrides of default JSON API handlers.

  If your custom handlers named in conform with convention of this application, then pass it as sequence:

```
custom_handlers=(get_collection, patch_resource)
```

  If you have custom name of these handlers, then pass it as mapping:

```
custom_handlers={
    'get_collection': some_handler_for_get_collection,
    'patch_resource': another_handler_to_patch_resource
}
```

- **log_errors** (*bool*) – Log errors handled by *jsonapi_middleware()*
- **routes_namespace** (*str*) – Namespace of JSON API application routes

Returns aiohttp Application instance with configured JSON API

Return type Application

aiohttp_json_api.**setup_resources**(*app*, *base_path*, *handlers*, *routes_namespace*)
Set up JSON API application resources.

7.2 aiohttp_json_api.abc package

7.2.1 Submodules

class aiohttp_json_api.abc.contoller.**ControllerABC**(*context*)

Bases: abc.ABC

add_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-to-many-relationships

Adds the members specified in the JSON API relationship object *data* to the relationship, unless the relationships already exist.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by *fetch_resource()* in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.

create_resource(*data*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-creating

Creates a new resource instance and returns it. **You should override this method.**

The default implementation passes the attributes, (dereferenced) relationships and meta data from the JSON API resource object *data* to the constructor of the resource class. If the primary key is *writable* on creation and a member of *data*, it is also passed to the constructor.

Parameters **data** (*dict*) – The JSON API deserialized data by schema.

static default_add(*field*, *resource*, *data*, *sp*, ***kwargs*)

static default_include(*field*, *resources*, ***kwargs*)

static default_query(*field*, *resource*, ***kwargs*)

static default_remove(*field*, *resource*, *data*, *sp*, ***kwargs*)

delete_resource(*resource_id*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-deleting

Deletes the *resource*. **You must overridde this method.**

Parameters

- **resource_id** – The id of the resource or the resource instance
- **context** (JSONAPIContext) – Request context instance

fetch_compound_documents(*field*, *resources*, *, *rest_path=None*, ***kwargs*)

See also:

http://jsonapi.org/format/#fetching-includes

Fetches the related resources. The default method uses the controller's *default_include()*. **Can be overridden.**

Parameters

- **field** (FieldABC) – Relationship field.
- **resources** – A list of resources.
- **context** (JSONAPIContext) – Request context instance.
- **rest_path** (*list*) – The name of the relationships of the returned relatives, which will also be included.

Return type list

Returns A list with the related resources. The list is empty or has exactly one element in the case of *to-one* relationships. If *to-many* relationships are paginated, the relatives from the first page should be returned.

fetch_resource(*resource_id*, ***kwargs*)

query_collection(***kwargs*)

See also:

http://jsonapi.org/format/#fetching

Fetches a subset of the collection represented by this schema. **Must be overridden.**

Parameters **context** (JSONAPIContext) – Request context instance.

query_relatives(*field*, *resource*, ***kwargs*)

Controller for the *related* endpoint of the relationship with then name *relation_name*.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by *fetch_resource()* in handler.

query_resource(*resource_id*, ***kwargs*)

See also:

http://jsonapi.org/format/#fetching

Fetches the resource with the id *id_*. **Must be overridden.**

Parameters

- **resource_id** (*str*) – The id of the requested resource.
- **context** (JSONAPIContext) – A request context instance

Raises ***ResourceNotFound*** – If there is no resource with the given *id_*.

remove_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-to-many-relationships

Deletes the members specified in the JSON API relationship object *data* from the relationship.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by *fetch_resource()* in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance.

update_relationship(*field*, *resource*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating-relationships

Updates the relationship with the JSON API name *relation_name*.

Parameters

- **field** (FieldABC) – Relationship field.
- **resource** – Resource instance fetched by *fetch_resource()* in handler.
- **data** (*str*) – The JSON API relationship object with the update information.
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.

update_resource(*resource_id*, *data*, *sp*, ***kwargs*)

See also:

http://jsonapi.org/format/#crud-updating

Updates an existing *resource*. **You should overridde this method** in order to save the changes in the database.

The default implementation uses the BaseField descriptors to update the resource.

Parameters

- **resource_id** – The id of the resource
- **data** (*dict*) – The JSON API resource object with the update information
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance

class aiohttp_json_api.abc.contoller.**ControllerMeta**(*name*, *bases*, *attrs*)

Bases: abc.ABCMeta, *aiohttp_json_api.abc.processors.MetaProcessors*

Field abstract base class

class aiohttp_json_api.abc.field.**FieldABC**
Bases: abc.ABC

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

key

Return type str

mapped_key

Return type Optional[str]

name

Return type Optional[str]

post_validate(*schema*, *data*, *sp*)
Validates the decoded input *data* for this field. This method is called after *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The decoded input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

sp

Return type *JSONPointer*

class aiohttp_json_api.abc.processors.**MetaProcessors**
Bases: object

Schema abstract base classes

class aiohttp_json_api.abc.schema.**SchemaABC**(*context*)
Bases: abc.ABC

OPTIONS_CLASS
alias of *SchemaOpts*

class Options
Bases: object

static default_getter(*field*, *resource*, ***kwargs*)

static default_setter(*field*, *resource*, *data*, *sp*, ***kwargs*)

deserialize_resource(*data*, *sp*, ***, *expected_id=None*, *validate=True*, *validation_steps=None*)
Decodes the JSON API resource object *data* and returns a dictionary which maps the key of a field to its decoded input data.

Parameters

- **data** – The received JSON API resource object
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance
- **expected_id** (*str*) – If passed, then ID of resource will be compared with this value. This is required in update methods
- **validate** (*bool*) – Is validation required?
- **validation_steps** (*tuple*) – Required validation steps

Return type OrderedDict

Returns An ordered dictionary which maps a fields key to a two tuple `(data, sp)` which contains the input data and the source pointer to it.

classmethod get_field(*key*)

Return type *FieldABC*

classmethod get_relationship_field(*relation_name*, *source_parameter=None*)

get_value(*field*, *resource*, ***kwargs*)

opts = <aiohttp_json_api.abc.schema.SchemaOpts object>

post_validate_resource(*data*)
Validates the decoded *data* of JSON API resource object.

Parameters

- **data** (*OrderedDict*) – The *memo* object returned from *deserialize_resource()*.
- **context** (JSONAPIContext) – Request context instance

pre_validate_field(*field*, *data*, *sp*)
Validates the input data for a field, **before** it is deserialized. If the field has nested fields, the nested fields are validated first.

Parameters

- **field** (BaseField) –
- **data** – The input data for the field.
- **sp** (aiohttp_json_api.jsonpointer.JSONPointer) – The pointer to *data* in the original document. If *None*, there was no input data for this field.

pre_validate_resource(*data*, *sp*, ***, *expected_id=None*)
Validates a JSON API resource object received from an API client:

```
schema.pre_validate_resource(
    data=request.json["data"], sp="/data"
)
```

Parameters

- **data** – The received JSON API resource object
- **sp** (JSONPointer) – The JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – Request context instance
- **expected_id** (*str*) – If passed, then ID of resrouce will be compared with this value. This is required in update methods

serialize_relationship(*relation_name*, *resource*, *, *pagination=None*)

See also:

http://jsonapi.org/format/#document-resource-object-relationships

Creates the JSON API relationship object of the relationship *relation_name*.

Parameters

- **relation_name** (*str*) – The name of the relationship
- **resource** – A resource object
- **pagination** (PaginationABC) – Describes the pagination in case of a *to-many* relationship.

Return type dict

Returns The JSON API relationship object for the relationship *relation_name* of the *resource*

serialize_resource(*resource*, **kwargs*)

set_value(*field*, *resource*, *data*, *sp*, **kwargs*)

class aiohttp_json_api.abc.schema.**SchemaMeta**(*name*, *bases*, *attrs*)

Bases: abc.ABCMeta, *aiohttp_json_api.abc.processors.MetaProcessors*

class aiohttp_json_api.abc.schema.**SchemaOpts**(*options*)

Bases: object

class Meta options for the *SchemaABC*. Defines defaults.

aiohttp_json_api.abc.schema.**issubclass**(*subclass*, *baseclass*)

Just like the built-in *issubclass()*, this function checks whether *subclass* is inherited from *baseclass*. Unlike the built-in function, this issubclass will simply return False if either argument is not suitable (e.g., if *subclass* is not an instance of type), instead of raising TypeError.

Args: subclass (type): The target class to check. baseclass (type): The base class *subclass* will be checked against.

```
>>> class MyObject(object): pass
...
>>> issubclass(MyObject, object)   # always a fun fact
True
>>> issubclass('hi', 'friend')
False
```

7.2.2 Module contents

7.3 aiohttp_json_api.fields package

7.3.1 Submodules

Fields

Note: Always remember that you can model the JSON API completly with the fields in `base_fields`.

Index

- *String*
- *Integer*
- *Float*
- *Complex*
- *Decimal*
- *Fraction*
- *DateTime*
- *TimeDelta*
- *UUID*
- *Boolean*
- *URI*
- *Email*
- *Dict*
- *List*
- *Number*
- *Str*
- *Bool*

This module contains fields for several standard Python types and classes from the standard library.

class aiohttp_json_api.fields.attributes.**String**(*, *allow_blank=False*, *regex=None*, *choices=None*, *min_length=None*, *max_length=None*, ***kwargs*)

Bases: *aiohttp_json_api.fields.base.Attribute*

deserialize(*schema*, *data*, *sp*, ***kwargs*)

Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)

Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Integer**(*, *gte=None*, *lte=None*, *gt=None*, *lt=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Float**(*, *gte=None*, *lte=None*, *gt=None*, *lt=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Complex**(*, *meta=False*, *load_only=False*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Encodes a `complex` number as JSON object with a *real* and *imag* member:

```
{"real": 1.2, "imag": 42}
```

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Decimal**(*, *gte=None*, *lte=None*, *gt=None*, *lt=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Encodes and decodes a `decimal.Decimal` as a string.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Fraction**(*, *min=None*, *max=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Stores a `fractions.Fraction` in an object with a *numerator* and *denominator* member:

```
# 1.5
{"numerator": 2, "denominator": 3}
```

Parameters

- **min** (*float*) – The fraction must be greater or equal than this value.
- **max** (*float*) – The fraction must be less or equal than this value.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.

- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**DateTime**(*, *allow_blank=False*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Stores a datetime.datetime in ISO-8601 as recommended in http://jsonapi.org/recommendations/#date-and-time-fields.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**TimeDelta**(*, *min=None*, *max=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Stores a datetime.timedelta as total number of seconds.

Parameters

- **min** (*datetime.timedelta*) – The timedelta must be greater or equal than this value.
- **max** (*datetime.timedelta*) – The timedelta must be less or equal than this value.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**UUID**(*, *version=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Encodes and decodes a `uuid.UUID`.

Parameters **version** (*int*) – The required version of the UUID.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Boolean**(***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Ensures that the input is a `bool`.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before `deserialize()`.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**URI**(*, *meta=False*, *load_only=False*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Parses the URI with `rfc3986.urlparse()` and returns the result.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Email**(***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Checks if a string is syntactically correct Email address.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before `deserialize()`.

Parameters

- **schema** (`BaseSchema`) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (`JSONPointer`) – A JSON pointer to the source of *data*.
- **context** (`JSONAPIContext`) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**Dict**(*field*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Realises a dictionary which has only values of a special field:

```
todo = Dict(String(regex=".*[A-z0-9].*"))
```

Note: If you deal with dictionaries with values of different types, you can still use the more general `Attribute` field to model this data.

You are not forced to use a `Dict` field! It is only a helper.

Parameters **field** (`Attribute`) – All values of the dictionary are encoded and decoded using this field.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

class aiohttp_json_api.fields.attributes.**List**(*field*, *min_length=0*, *max_length=None*, ***kwargs*)
Bases: *aiohttp_json_api.fields.base.Attribute*

Note: If your list has items of different types, you can still use the more general `Attribute` field to model this data.

You are not forced to use a `List` field! It is only a helper.

Parameters **field** (`Attribute`) – All values of the list are encoded and decoded using this field.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

aiohttp_json_api.fields.attributes.**Number**
alias of *aiohttp_json_api.fields.attributes.Float*

aiohttp_json_api.fields.attributes.**Str**
alias of *aiohttp_json_api.fields.attributes.String*

aiohttp_json_api.fields.attributes.**Bool**
alias of *aiohttp_json_api.fields.attributes.Boolean*

Base fields

This module contains the definition for all basic fields. A field describes how data should be serialized to JSON and deserialized again and allows to define special methods for the different CRUD operations defined by the http://jsonapi.org specification.

You should only work with the following fields directly:

- *Link*

  For JSON API link objects (usually included in a JSON API links object).

  seealso http://jsonapi.org/format/#document-links

- *Attribute*

  Represent information about a resource object (but not a relationship or a link).

  seealso http://jsonapi.org/format/#document-resource-object-attributes

- ToOne, ToMany

  Represent relationships on a resource object.

  seealso http://jsonapi.org/format/#document-resource-object-relationships

---

Todo: Add support for nested fields (aka embedded documents).

---

class aiohttp_json_api.fields.base.**BaseField**(*, *name=None*, *mapped_key=None*, *allow_none=False*, *writable=<Event.ALWAYS: 15>*, *required=<Event.NEVER: 16>*)
Bases: *aiohttp_json_api.abc.field.FieldABC*

This class describes the base for all fields defined on a schema and knows how to serialize, deserialize and validate the field. A field is usually directly mapped to a property (*mapped_key*) on the resource object, but this mapping can be customized by implementing custom *getters* and *setters* (via decorators).

Hint: The inheritance of fields is currently implemented using the `deepcopy()` function from the standard library. This means, that in some rare cases, it is necessarily that you implement a custom `__deepcopy__()` method when you subclass *BaseField*.

Parameters

- **name** (*str*) – The name of the field in the JSON API document. If not explicitly given, it's the same as *key*.
- **mapped_key** (*str*) – The name of the associated property on the resource class. If not explicitly given, it's the same as *key*.
- **allow_none** (*bool*) – Allow to receive 'null' value
- **writable** (Event) – Can be any `Event` enumeration value and describes in which CRUD context the field is writable.
- **required** (Event) – Can be any `Event` enumeration value and describes in which CRUD context the field is required as input.

deserialize(*schema*, *data*, *sp*, ***kwargs*)
Deserialize the raw *data* from the JSON API input document and returns it.

key

Return type `str`

mapped_key

Return type `Optional[str]`

name

Return type `Optional[str]`

post_validate(*schema*, *data*, *sp*)
Validates the decoded input *data* for this field. This method is called after *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The decoded input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before *deserialize()*.

Parameters

- **schema** (BaseSchema) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (JSONPointer) – A JSON pointer to the source of *data*.
- **context** (JSONAPIContext) – A JSON API request context instance

serialize(*schema*, *data*, ***kwargs*)
Serialize the passed *data*.

sp

Return type *JSONPointer*

class aiohttp_json_api.fields.base.**Attribute**(*, *meta=False*, *load_only=False*, ***kwargs*)

Bases: *aiohttp_json_api.fields.base.BaseField*

See also:

http://jsonapi.org/format/#document-resource-object-attributes

An attribute is always part of the resource's JSON API attributes object, unless *meta* is set, in which case the attribute appears in the resource's meta object.

Per default, an attribute is mapped to a property on the resource object. You can customize this behaviour by implementing your own *getter* and *setter*:

```
class Article(BaseSchema):

    title = Attribute()
```

Does the same as:

```
class Article(BaseSchema):

    title = Attribute()

    @gets('title')
    def title(self, article):
        return article.title

    @sets('title')
    def title(self, article, new_title):
        article.title = new_title
        return None
```

Parameters

- **meta** (*bool*) – If true, the attribute is part of the resource's *meta* object.
- **load_only** (*bool*) – If *True* skip this field during serialization, otherwise its value will be present in the serialized data.
- ****kwargs** – The init arguments for the *BaseField*.

class aiohttp_json_api.fields.base.**Link**(*route*, *link_of*, *, *name=None*, *normalize=False*, *absolute=True*)

Bases: *aiohttp_json_api.fields.base.BaseField*

See also:

http://jsonapi.org/format/#document-links

```
class Article(BaseSchema):

    self = Link(route="some_route_name")

    author = ToOne()
    author_related = Link(
        route="another_route_name", link_of="author"
    )
```

In the http://jsonapi.org specification, a link is always part of a JSON API links object and is either a simple string or an object with the members *href* and *meta*.

A link is only readable and *not* mapped to a property on the resource object (You can however define a *getter*).

Parameters

- **route** (*str*) – A route name for the link
- **link_of** (*str*) – If given, the link is part of the links object of the field with the key *link_of* and appears otherwise in the resource object's links objects. E.g.: `link_of = "author"`.
- **normalize** (*bool*) – If true, the *serialize* method normalizes the link so that it is always an object.

serialize(*schema*, *data*, ***kwargs*)
Normalizes the links object if wished.

class aiohttp_json_api.fields.base.**Relationship**(*, *dereference=True*, *require_data=<Event.ALWAYS: 15>*, *foreign_types=None*, *links=None*, *id_field=None*, ***kwargs*)

Bases: *aiohttp_json_api.fields.base.BaseField*

See also:

http://jsonapi.org/format/#document-resource-object-relationships

Additionally to attributes and basic fields, we must know how to *include* the related resources in the case of relationships. This class defines the common interface of *to-one* and *to-many* relationships (links object, meta object, *self* link, *related* link, validation, . . .).

Relationships are always part of the resource's JSON API relationships object.

See also:

- `ToOne`
- `ToMany`

Parameters

- **require_data** (`Event`) – If true, the JSON API relationship object must contain the *data* member. Must be a `Event` instance.
- **dereference** (*bool*) – If true, the relationship linkage is dereferenced automatic when decoded. (Implicitly sets *require_data* to Event.ALWAYS)
- **foreign_types** (*Sequence[str]*) – A set with all foreign types. If given, this list is used to validate the input data. Leave it empty to allow all types.

add_link(*link*)
Adds a new link to the links object.

pre_validate(*schema*, *data*, *sp*)
Validates the raw JSON API input for this field. This method is called before `deserialize()`.

Parameters

- **schema** (`BaseSchema`) – The schema this field has been defined on.
- **data** – The raw input data
- **sp** (`JSONPointer`) – A JSON pointer to the source of *data*.

- **context** (JSONAPIContext) – A JSON API request context instance

relation = None

validate_relationship_object(*schema*, *data*, *sp*)
Asserts that *data* is a JSON API relationship object.

- *data* is a dictionary
- *data* must be not empty
- *data* may only have the members *data*, *links* or *meta*.
- *data* must contain a *data* member, if `dereference` or `require_data` is true.

validate_resource_identifier(*schema*, *data*, *sp*)

See also:

http://jsonapi.org/format/#document-resource-identifier-objects

Asserts that *data* is a JSON API resource identifier with the correct *type* value.

Schema decorators

This module contains some decorators, which can be used instead of the descriptors on the `BaseField` class.

Todo: Allow to define a *getter*, . . . , *includer* for multiple fields:

```
@includes("author", "comments")
def include_all(self, article, **kargs):
    return (article.author, article.comments)

@validates("a", "b")
def validate_a_and_b(self, a, spa, b, spb, **kargs):
    if a > b:
        raise InvalidValue("a must be less than b", source_pointer=spa)
    return None
```

Todo: Use convention over configuration:

```
@gets("author")
def get_author(self, article, **kargs):
    return article.author_id

# Should be the same as

def get_author(self, article, **kargs):
    return article.author_id
```

---

class aiohttp_json_api.fields.decorators.**Tag**
Bases: enum.Enum

An enumeration.

ADD = 'add'

GET = 'get'

INCLUDE = 'include'

QUERY = 'query'

REMOVE = 'remove'

SET = 'set'

VALIDATE = 'validate'

aiohttp_json_api.fields.decorators.**gets**(*field_key*)

Decorator for marking the getter of a field:

```
class Article(BaseSchema):

    title = String()

    @gets("title")
    def get_title(self, article):
        return article.get_title()
```

A field can have at most **one** getter.

Parameters **field_key** (*str*) – The key of the field.

aiohttp_json_api.fields.decorators.**sets**(*field_key*)

Decorator for marking the setter of a field:

```
class Article(BaseSchema):

    title = String()

    @sets("title")
    def update_title(self, article, title, sp):
        article.set_title(title)
        return None
```

A field can have at most **one** updater.

Parameters **field_key** (*str*) – The key of the field.

aiohttp_json_api.fields.decorators.**updates**(*field_key*)

Alias for *sets()*.

aiohttp_json_api.fields.decorators.**validates**(*field_key*, *step=<Step.AFTER_DESERIALIZATION: 2>*, *on=<Event.ALWAYS: 15>*)

Decorator for adding a validator:

```
class Article(BaseSchema):

    created_at = DateTime()

    @validates("created_at")
    def validate_created_at(self, data, sp, context):
        if created_at > datetime.utcnow():
            detail = "Must be in the past."
            raise InvalidValue(detail=detail, source_pointer=sp)
```

A field can have as many validators as you want. Note, that they are not necessarily called in the order of their definition.

Parameters

- **field_key** (*str*) – The key of the field.
- **step** (Step) – Must be any Step enumeration value (e.g. Step.BEFORE_DESERIALIZATION)
- **on** (Event) – Validator's Event

aiohttp_json_api.fields.decorators.**adds**(*field_key*)

Decorator for marking the adder of a relationship:

```
class Article(BaseSchema):

    comments = ToMany()

    @adds("comments")
    def add_comments(self, field, resource, data, sp,
                     context=None, **kwargs):
        for comment in comment:
            comment.article_id = article.id
```

A relationship can have at most **one** adder.

Parameters **field_key** (*str*) – The key of the relationship.

aiohttp_json_api.fields.decorators.**removes**(*field_key*)

Decorator for marking the remover of a relationship:

```
class Article(BaseSchema):

    comments = ToMany()

    @removes("comments")
    def remove_comments(self, field, resource, data, sp,
                        context=None, **kwargs):
        for comment in comment:
            comment.article_id = None
```

A relationship can have at most **one** remover.

Parameters **field_key** (*str*) – The key of the relationship.

aiohttp_json_api.fields.decorators.**includes**(*field_key*)

Decorator for marking the includer of a relationship:

```
class Article(BaseSchema):

    author = ToOne()

    @includes("author")
    def include_author(self, field, resources, context, **kwargs):
        return article.load_author()
```

A field can have at most **one** includer.

Hint: The includer should receive list of all resources related to request. This able to make one request for all related includes at each step of recursively fetched compound documents. Look at *get_compound_documents()* for more details about how it works.

Parameters **field_key** (*str*) – The name of the relationship.

aiohttp_json_api.fields.decorators.**queries**(*field_key*)

Decorator for marking the function used to query the resources in a relationship:

```
class Article(BaseSchema):

    comments = ToMany()

    @queries("comments")
    def query_comments(self, article_id, **kargs):
        pass
```

A field can have at most **one** query method.

Todo: Add an example.

Parameters **field_key** (*str*) – The name of the relationship.

Relationships

class aiohttp_json_api.fields.relationships.**ToOne**(*, *dereference=True*, *require_data=<Event.ALWAYS: 15>*, *foreign_types=None*, *links=None*, *id_field=None*, ***kwargs*)

Bases: *aiohttp_json_api.fields.base.Relationship*

See also:

- http://jsonapi.org/format/#document-resource-object-relationships
- http://jsonapi.org/format/#document-resource-object-linkage

Describes how to serialize, deserialize and update a *to-one* relationship.

relation = 1

serialize(*schema*, *data*, ***kwargs*)

Composes the final relationships object.

Return type Mutablemapping[~KT, ~VT]

validate_relationship_object(*schema*, *data*, *sp*)

Checks additionaly to `Relationship.validate_relationship_object()` that the *data* member is a valid resource linkage.

class aiohttp_json_api.fields.relationships.**ToMany**(*, *pagination=None*, ***kwargs*)

Bases: *aiohttp_json_api.fields.base.Relationship*

See also:

- http://jsonapi.org/format/#document-resource-object-relationships
- http://jsonapi.org/format/#document-resource-object-linkage

Describes how to serialize, deserialize and update a *to-many* relationship. Additionally to *to-one* relationships, *to-many* relationships must also support adding and removing relatives.

Parameters **pagination** (aiohttp_json_api.pagination.PaginationABC) – The pagination helper *class* used to paginate the *to-many* relationship.

relation = 2

serialize(*schema*, *data*, *links=None*, *pagination=None*, ***kwargs*)
Composes the final JSON API relationships object.

Parameters **pagination** (PaginationABC) – If not *None*, the links and meta members of the pagination helper are added to the final JSON API relationship object.

Return type Mutablemapping[~KT, ~VT]

validate_relationship_object(*schema*, *data*, *sp*)
Checks additionaly to Relationship.validate_relationship_object() that the *data* member is a list of resource identifier objects.

Additional trafaret's fields

class aiohttp_json_api.fields.trafarets.**DecimalTrafaret**(*places=None*, *rounding=None*, *allow_nan=False*, ***kwargs*)

Bases: trafaret.base.Float

check_and_return(*data*)

convertable = (<class 'str'>, <class 'bytes'>, <class 'numbers.Real'>, <class 'int'>)

value_type
alias of decimal.Decimal

7.3.2 Module contents

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

aiohttp_json_api, 52
aiohttp_json_api.abc, 59
aiohttp_json_api.abc.contoller, 53
aiohttp_json_api.abc.field, 55
aiohttp_json_api.abc.processors, 56
aiohttp_json_api.abc.schema, 56
aiohttp_json_api.common, 27
aiohttp_json_api.context, 28
aiohttp_json_api.controller, 31
aiohttp_json_api.encoder, 33
aiohttp_json_api.errors, 34
aiohttp_json_api.fields, 73
aiohttp_json_api.fields.attributes, 59
aiohttp_json_api.fields.base, 65
aiohttp_json_api.fields.decorators, 69
aiohttp_json_api.fields.relationships, 72
aiohttp_json_api.fields.trafarets, 73
aiohttp_json_api.handlers, 40
aiohttp_json_api.helpers, 41
aiohttp_json_api.jsonpointer, 43
aiohttp_json_api.middleware, 43
aiohttp_json_api.pagination, 43
aiohttp_json_api.registry, 47
aiohttp_json_api.schema, 48
aiohttp_json_api.typings, 50
aiohttp_json_api.utils, 50

Index

A

ADD (aiohttp_json_api.fields.decorators.Tag attribute), 69
add_link() (aiohttp_json_api.fields.base.Relationship method), 68
add_relationship() (aiohttp_json_api.abc.contoller.ControllerABC method), 53
add_relationship() (aiohttp_json_api.controller.DefaultController method), 31
adds() (in module aiohttp_json_api.fields.decorators), 71
AFTER_DESERIALIZATION (aiohttp_json_api.common.Step attribute), 28
AFTER_SERIALIZATION (aiohttp_json_api.common.Step attribute), 28
aiohttp_json_api (module), 52
aiohttp_json_api.abc (module), 59
aiohttp_json_api.abc.contoller (module), 53
aiohttp_json_api.abc.field (module), 55
aiohttp_json_api.abc.processors (module), 56
aiohttp_json_api.abc.schema (module), 56
aiohttp_json_api.common (module), 27
aiohttp_json_api.context (module), 28
aiohttp_json_api.controller (module), 31
aiohttp_json_api.encoder (module), 33
aiohttp_json_api.errors (module), 34
aiohttp_json_api.fields (module), 73
aiohttp_json_api.fields.attributes (module), 59
aiohttp_json_api.fields.base (module), 65
aiohttp_json_api.fields.decorators (module), 69
aiohttp_json_api.fields.relationships (module), 72
aiohttp_json_api.fields.trafarets (module), 73
aiohttp_json_api.handlers (module), 40
aiohttp_json_api.helpers (module), 41
aiohttp_json_api.jsonpointer (module), 43
aiohttp_json_api.middleware (module), 43
aiohttp_json_api.pagination (module), 43
aiohttp_json_api.registry (module), 47
aiohttp_json_api.schema (module), 48
aiohttp_json_api.typings (module), 50
aiohttp_json_api.utils (module), 50
ALLOWED_MEMBER_NAME_REGEX (in module aiohttp_json_api.common), 27
ALLOWED_MEMBER_NAME_RULE (in module aiohttp_json_api.common), 27
ALWAYS (aiohttp_json_api.common.Event attribute), 27
app (aiohttp_json_api.context.JSONAPIContext attribute), 29
append() (aiohttp_json_api.errors.ErrorList method), 34
as_dict (aiohttp_json_api.errors.Error attribute), 34
ASC (aiohttp_json_api.common.SortDirection attribute), 28
Attribute (class in aiohttp_json_api.fields.base), 67

B

BaseField (class in aiohttp_json_api.fields.base), 65
BaseSchema (class in aiohttp_json_api.schema), 48
BEFORE_DESERIALIZATION (aiohttp_json_api.common.Step attribute), 28
BEFORE_SERIALIZATION (aiohttp_json_api.common.Step attribute), 28
best_match() (in module aiohttp_json_api.helpers), 41
Bool (in module aiohttp_json_api.fields.attributes), 65
Boolean (class in aiohttp_json_api.fields.attributes), 63

C

Callee (in module aiohttp_json_api.typings), 50
check_and_return() (aiohttp_json_api.fields.trafarets.DecimalTrafaret method), 73
Complex (class in aiohttp_json_api.fields.attributes), 60
controller (aiohttp_json_api.context.JSONAPIContext attribute), 29
ControllerABC (class in aiohttp_json_api.abc.contoller), 53
ControllerMeta (class in aiohttp_json_api.abc.contoller), 55

convert_field_name() (aiohttp_json_api.context.JSONAPIContext class method), 29
convertable (aiohttp_json_api.fields.trafarets.DecimalTrafaret attribute), 73
create_resource() (aiohttp_json_api.abc.contoller.ControllerABC method), 53
Cursor (class in aiohttp_json_api.pagination), 47

D

data (aiohttp_json_api.registry.Registry attribute), 47
DateTime (class in aiohttp_json_api.fields.attributes), 62
Decimal (class in aiohttp_json_api.fields.attributes), 61
DecimalTrafaret (class in aiohttp_json_api.fields.trafarets), 73
default() (aiohttp_json_api.encoder.JSONEncoder method), 33
default_add() (aiohttp_json_api.abc.contoller.ControllerABC static method), 53
default_add() (aiohttp_json_api.controller.DefaultController static method), 32
default_getter() (aiohttp_json_api.abc.schema.SchemaABC static method), 56
default_getter() (aiohttp_json_api.schema.BaseSchema static method), 48
default_include() (aiohttp_json_api.abc.contoller.ControllerABC static method), 53
default_include() (aiohttp_json_api.controller.DefaultController static method), 32
DEFAULT_LIMIT (in module aiohttp_json_api.pagination), 44
default_query() (aiohttp_json_api.abc.contoller.ControllerABC static method), 53
default_query() (aiohttp_json_api.controller.DefaultController static method), 32
default_remove() (aiohttp_json_api.abc.contoller.ControllerABC static method), 53
default_remove() (aiohttp_json_api.controller.DefaultController static method), 32
default_setter() (aiohttp_json_api.abc.schema.SchemaABC static method), 57
default_setter() (aiohttp_json_api.schema.BaseSchema static method), 48
DefaultController (class in aiohttp_json_api.controller), 31
DELETE (aiohttp_json_api.common.Event attribute), 27
delete_relationship() (in module aiohttp_json_api.handlers), 41
delete_resource() (aiohttp_json_api.abc.contoller.ControllerABC method), 53
delete_resource() (in module aiohttp_json_api.handlers), 41
DESC (aiohttp_json_api.common.SortDirection attribute), 28
deserialize() (aiohttp_json_api.abc.field.FieldABC method), 56
deserialize() (aiohttp_json_api.fields.attributes.Complex method), 60
deserialize() (aiohttp_json_api.fields.attributes.DateTime method), 62
deserialize() (aiohttp_json_api.fields.attributes.Decimal method), 61
deserialize() (aiohttp_json_api.fields.attributes.Dict method), 64
deserialize() (aiohttp_json_api.fields.attributes.Float method), 60
deserialize() (aiohttp_json_api.fields.attributes.Fraction method), 61
deserialize() (aiohttp_json_api.fields.attributes.Integer method), 60
deserialize() (aiohttp_json_api.fields.attributes.List method), 64
deserialize() (aiohttp_json_api.fields.attributes.String method), 59
deserialize() (aiohttp_json_api.fields.attributes.TimeDelta method), 62
deserialize() (aiohttp_json_api.fields.attributes.URI method), 63
deserialize() (aiohttp_json_api.fields.attributes.UUID method), 63
deserialize() (aiohttp_json_api.fields.base.BaseField method), 66
deserialize_resource() (aiohttp_json_api.abc.schema.SchemaABC method), 57
deserialize_resource() (aiohttp_json_api.schema.BaseSchema method), 48
Dict (class in aiohttp_json_api.fields.attributes), 64

E

Email (class in aiohttp_json_api.fields.attributes), 64
ensure_collection() (in module aiohttp_json_api.helpers), 42
ensure_identifier() (aiohttp_json_api.registry.Registry method), 47
Error, 34
error_to_response() (in module aiohttp_json_api.utils), 50
ErrorList, 34
event (aiohttp_json_api.context.JSONAPIContext attribute), 29
Event (class in aiohttp_json_api.common), 27
extend() (aiohttp_json_api.errors.ErrorList method), 34

F

fetch_compound_documents() (aiohttp_json_api.abc.contoller.ControllerABC

method), 53
fetch_compound_documents() (aiohttp_json_api.controller.DefaultController method), 32
fetch_resource() (aiohttp_json_api.abc.contoller.ControllerABC method), 54
FieldABC (class in aiohttp_json_api.abc.field), 56
fields (aiohttp_json_api.context.JSONAPIContext attribute), 29
FIELDS_RE (aiohttp_json_api.context.JSONAPIContext attribute), 28
FILTER_KEY (aiohttp_json_api.context.JSONAPIContext attribute), 28
FILTER_VALUE (aiohttp_json_api.context.JSONAPIContext attribute), 28
FilterRule (class in aiohttp_json_api.common), 27
filters (aiohttp_json_api.context.JSONAPIContext attribute), 29
FIRST (aiohttp_json_api.pagination.Cursor attribute), 47
first() (in module aiohttp_json_api.helpers), 42
Float (class in aiohttp_json_api.fields.attributes), 60
Fraction (class in aiohttp_json_api.fields.attributes), 61

G

GET (aiohttp_json_api.common.Event attribute), 27
GET (aiohttp_json_api.fields.decorators.Tag attribute), 69
get_collection() (in module aiohttp_json_api.handlers), 40
get_compound_documents() (in module aiohttp_json_api.utils), 50
get_field() (aiohttp_json_api.abc.schema.SchemaABC class method), 57
get_field() (aiohttp_json_api.schema.BaseSchema class method), 48
get_filter() (aiohttp_json_api.context.JSONAPIContext method), 29
get_mime_type_params() (in module aiohttp_json_api.helpers), 42
get_object_id() (aiohttp_json_api.schema.BaseSchema static method), 48
get_order() (aiohttp_json_api.context.JSONAPIContext method), 29
get_processors() (in module aiohttp_json_api.helpers), 42
get_related() (in module aiohttp_json_api.handlers), 41
get_relationship() (in module aiohttp_json_api.handlers), 41
get_relationship_field() (aiohttp_json_api.abc.schema.SchemaABC class method), 57
get_relationship_field() (aiohttp_json_api.schema.BaseSchema class method), 49
get_resource() (in module aiohttp_json_api.handlers), 40
get_router_resource() (in module aiohttp_json_api.helpers), 42
get_value() (aiohttp_json_api.abc.schema.SchemaABC method), 57
get_value() (aiohttp_json_api.schema.BaseSchema method), 49
gets() (in module aiohttp_json_api.fields.decorators), 70

H

has_filter() (aiohttp_json_api.context.JSONAPIContext method), 29
HTTPBadGateway, 38
HTTPBadRequest, 34
HTTPConflict, 36
HTTPFailedDependency, 37
HTTPForbidden, 35
HTTPGatewayTimeout, 38
HTTPGone, 36
HTTPInsufficientStorage, 39
HTTPInternalServerError, 37
HTTPLocked, 37
HTTPMethodNotAllowed, 35
HTTPNotAcceptable, 35
HTTPNotExtended, 39
HTTPNotFound, 35
HTTPNotImplemented, 38
HTTPPreConditionFailed, 36
HTTPServiceUnavailable, 38
HTTPTooManyRequests, 37
HTTPUnauthorized, 34
HTTPUnprocessableEntity, 36
HTTPUnsupportedMediaType, 36
HTTPVariantAlsoNegotiates, 38

I

id (aiohttp_json_api.common.ResourceID attribute), 28
include (aiohttp_json_api.context.JSONAPIContext attribute), 29
INCLUDE (aiohttp_json_api.fields.decorators.Tag attribute), 70
includes() (in module aiohttp_json_api.fields.decorators), 71
inflect() (aiohttp_json_api.context.JSONAPIContext method), 29
Integer (class in aiohttp_json_api.fields.attributes), 60
InvalidType, 39
InvalidValue, 40
is_collection() (in module aiohttp_json_api.helpers), 42
is_generator() (in module aiohttp_json_api.helpers), 42
is_indexable_but_not_string() (in module aiohttp_json_api.helpers), 42
is_iterable_but_not_string() (in module aiohttp_json_api.helpers), 42
issubclass() (in module aiohttp_json_api.abc.schema), 58

J

json (aiohttp_json_api.errors.ErrorList attribute), 34
JSONAPI (in module aiohttp_json_api.common), 28
JSONAPI_CONTENT_TYPE (in module aiohttp_json_api.common), 28
jsonapi_middleware() (in module aiohttp_json_api.middleware), 43
jsonapi_response() (in module aiohttp_json_api.utils), 51
JSONAPIContext (class in aiohttp_json_api.context), 28
JSONEncoder (class in aiohttp_json_api.encoder), 33
JSONPointer (class in aiohttp_json_api.jsonpointer), 43

K

key (aiohttp_json_api.abc.field.FieldABC attribute), 56
key (aiohttp_json_api.fields.base.BaseField attribute), 66

L

LAST (aiohttp_json_api.pagination.Cursor attribute), 47
last_page (aiohttp_json_api.pagination.NumberSize attribute), 46
limit (aiohttp_json_api.pagination.NumberSize attribute), 46
LimitOffset (class in aiohttp_json_api.pagination), 45
Link (class in aiohttp_json_api.fields.base), 67
links() (aiohttp_json_api.pagination.Cursor method), 47
links() (aiohttp_json_api.pagination.LimitOffset method), 45
links() (aiohttp_json_api.pagination.NumberSize method), 46
links() (aiohttp_json_api.pagination.PaginationABC method), 44
List (class in aiohttp_json_api.fields.attributes), 64
logger (in module aiohttp_json_api.common), 28

M

make_sentinel() (in module aiohttp_json_api.helpers), 42
map_data_to_schema() (aiohttp_json_api.schema.BaseSchema method), 49
mapped_key (aiohttp_json_api.abc.field.FieldABC attribute), 56
mapped_key (aiohttp_json_api.fields.base.BaseField attribute), 66
meta() (aiohttp_json_api.pagination.Cursor method), 47
meta() (aiohttp_json_api.pagination.LimitOffset method), 45
meta() (aiohttp_json_api.pagination.NumberSize method), 46
meta() (aiohttp_json_api.pagination.PaginationABC method), 44
MetaProcessors (class in aiohttp_json_api.abc.processors), 56

N

name (aiohttp_json_api.abc.field.FieldABC attribute), 56
name (aiohttp_json_api.common.FilterRule attribute), 27
name (aiohttp_json_api.fields.base.BaseField attribute), 66
NEVER (aiohttp_json_api.common.Event attribute), 27
Number (in module aiohttp_json_api.fields.attributes), 65
NumberSize (class in aiohttp_json_api.pagination), 45

O

offset (aiohttp_json_api.pagination.NumberSize attribute), 46
OPTIONS_CLASS (aiohttp_json_api.abc.schema.SchemaABC attribute), 56
opts (aiohttp_json_api.abc.schema.SchemaABC attribute), 57
opts (aiohttp_json_api.schema.BaseSchema attribute), 49

P

page_link() (aiohttp_json_api.pagination.PaginationABC method), 44
pagination (aiohttp_json_api.context.JSONAPIContext attribute), 30
PaginationABC (class in aiohttp_json_api.pagination), 44
parse_request_fields() (aiohttp_json_api.context.JSONAPIContext class method), 30
parse_request_filters() (aiohttp_json_api.context.JSONAPIContext class method), 30
parse_request_includes() (aiohttp_json_api.context.JSONAPIContext class method), 31
parse_request_sorting() (aiohttp_json_api.context.JSONAPIContext class method), 31
PATCH (aiohttp_json_api.common.Event attribute), 27
patch_relationship() (in module aiohttp_json_api.handlers), 41
patch_resource() (in module aiohttp_json_api.handlers), 41
POST (aiohttp_json_api.common.Event attribute), 27
post_relationship() (in module aiohttp_json_api.handlers), 41
post_resource() (in module aiohttp_json_api.handlers), 40
post_validate() (aiohttp_json_api.abc.field.FieldABC method), 56
post_validate() (aiohttp_json_api.fields.base.BaseField method), 66
post_validate_resource() (aiohttp_json_api.abc.schema.SchemaABC method), 57

post_validate_resource() (aiohttp_json_api.schema.BaseSchema method), 49
pre_validate() (aiohttp_json_api.abc.field.FieldABC method), 56
pre_validate() (aiohttp_json_api.fields.attributes.Boolean method), 63
pre_validate() (aiohttp_json_api.fields.attributes.Complex method), 60
pre_validate() (aiohttp_json_api.fields.attributes.DateTime method), 62
pre_validate() (aiohttp_json_api.fields.attributes.Decimal method), 61
pre_validate() (aiohttp_json_api.fields.attributes.Email method), 64
pre_validate() (aiohttp_json_api.fields.attributes.Float method), 60
pre_validate() (aiohttp_json_api.fields.attributes.Fraction method), 61
pre_validate() (aiohttp_json_api.fields.attributes.Integer method), 60
pre_validate() (aiohttp_json_api.fields.attributes.List method), 64
pre_validate() (aiohttp_json_api.fields.attributes.String method), 59
pre_validate() (aiohttp_json_api.fields.attributes.TimeDelta method), 62
pre_validate() (aiohttp_json_api.fields.attributes.URI method), 63
pre_validate() (aiohttp_json_api.fields.attributes.UUID method), 63
pre_validate() (aiohttp_json_api.fields.base.BaseField method), 66
pre_validate() (aiohttp_json_api.fields.base.Relationship method), 68
pre_validate_field() (aiohttp_json_api.abc.schema.SchemaABC method), 57
pre_validate_field() (aiohttp_json_api.schema.BaseSchema method), 49
pre_validate_resource() (aiohttp_json_api.abc.schema.SchemaABC method), 57
pre_validate_resource() (aiohttp_json_api.schema.BaseSchema method), 49

Q

quality_and_fitness_parsed() (in module aiohttp_json_api.helpers), 43
queries() (in module aiohttp_json_api.fields.decorators), 72
QUERY (aiohttp_json_api.fields.decorators.Tag attribute), 70
query_collection() (aiohttp_json_api.abc.contoller.ControllerABC method), 54
query_relatives() (aiohttp_json_api.abc.contoller.ControllerABC method), 54
query_relatives() (aiohttp_json_api.controller.DefaultController method), 32
query_resource() (aiohttp_json_api.abc.contoller.ControllerABC method), 54

R

registry (aiohttp_json_api.context.JSONAPIContext attribute), 31
Registry (class in aiohttp_json_api.registry), 47
relation (aiohttp_json_api.fields.base.Relationship attribute), 69
relation (aiohttp_json_api.fields.relationships.ToMany attribute), 73
relation (aiohttp_json_api.fields.relationships.ToOne attribute), 72
Relation (class in aiohttp_json_api.common), 28
Relationship (class in aiohttp_json_api.fields.base), 68
REMOVE (aiohttp_json_api.fields.decorators.Tag attribute), 70
remove_relationship() (aiohttp_json_api.abc.contoller.ControllerABC method), 54
remove_relationship() (aiohttp_json_api.controller.DefaultController method), 32
removes() (in module aiohttp_json_api.fields.decorators), 71
render_document() (in module aiohttp_json_api.utils), 51
request (aiohttp_json_api.context.JSONAPIContext attribute), 31
RequestFields (in module aiohttp_json_api.typings), 50
RequestFilters (in module aiohttp_json_api.typings), 50
RequestIncludes (in module aiohttp_json_api.typings), 50
RequestSorting (in module aiohttp_json_api.typings), 50
resource_type (aiohttp_json_api.context.JSONAPIContext attribute), 31
ResourceID (class in aiohttp_json_api.common), 28
ResourceIdentifier (in module aiohttp_json_api.typings), 50
ResourceNotFound, 40

S

schema (aiohttp_json_api.context.JSONAPIContext attribute), 31
SchemaABC (class in aiohttp_json_api.abc.schema), 56
SchemaABC.Options (class in aiohttp_json_api.abc.schema), 56

SchemaMeta (class in aiohttp_json_api.abc.schema), 58
SchemaOpts (class in aiohttp_json_api.abc.schema), 58
serialize() (aiohttp_json_api.abc.field.FieldABC method), 56
serialize() (aiohttp_json_api.fields.attributes.Boolean method), 63
serialize() (aiohttp_json_api.fields.attributes.Complex method), 61
serialize() (aiohttp_json_api.fields.attributes.DateTime method), 62
serialize() (aiohttp_json_api.fields.attributes.Decimal method), 61
serialize() (aiohttp_json_api.fields.attributes.Dict method), 64
serialize() (aiohttp_json_api.fields.attributes.Email method), 64
serialize() (aiohttp_json_api.fields.attributes.Float method), 60
serialize() (aiohttp_json_api.fields.attributes.Fraction method), 62
serialize() (aiohttp_json_api.fields.attributes.Integer method), 60
serialize() (aiohttp_json_api.fields.attributes.List method), 65
serialize() (aiohttp_json_api.fields.attributes.String method), 60
serialize() (aiohttp_json_api.fields.attributes.TimeDelta method), 62
serialize() (aiohttp_json_api.fields.attributes.URI method), 63
serialize() (aiohttp_json_api.fields.attributes.UUID method), 63
serialize() (aiohttp_json_api.fields.base.BaseField method), 66
serialize() (aiohttp_json_api.fields.base.Link method), 68
serialize() (aiohttp_json_api.fields.relationships.ToMany method), 73
serialize() (aiohttp_json_api.fields.relationships.ToOne method), 72
serialize_relationship() (aiohttp_json_api.abc.schema.SchemaABC method), 58
serialize_relationship() (aiohttp_json_api.schema.BaseSchema method), 49
serialize_resource() (aiohttp_json_api.abc.schema.SchemaABC method), 58
serialize_resource() (aiohttp_json_api.schema.BaseSchema method), 50
serialize_resource() (in module aiohttp_json_api.utils), 51
SET (aiohttp_json_api.fields.decorators.Tag attribute), 70
set_value() (aiohttp_json_api.abc.schema.SchemaABC method), 58
set_value() (aiohttp_json_api.schema.BaseSchema method), 50
sets() (in module aiohttp_json_api.fields.decorators), 70
setup_app_registry() (in module aiohttp_json_api), 52
setup_custom_handlers() (in module aiohttp_json_api), 52
setup_jsonapi() (in module aiohttp_json_api), 52
setup_resources() (in module aiohttp_json_api), 52
SortDirection (class in aiohttp_json_api.common), 28
sorting (aiohttp_json_api.context.JSONAPIContext attribute), 31
sp (aiohttp_json_api.abc.field.FieldABC attribute), 56
sp (aiohttp_json_api.fields.base.BaseField attribute), 66
status (aiohttp_json_api.errors.Error attribute), 34
status (aiohttp_json_api.errors.ErrorList attribute), 34
status (aiohttp_json_api.errors.HTTPBadGateway attribute), 38
status (aiohttp_json_api.errors.HTTPBadRequest attribute), 34
status (aiohttp_json_api.errors.HTTPConflict attribute), 36
status (aiohttp_json_api.errors.HTTPFailedDependency attribute), 37
status (aiohttp_json_api.errors.HTTPForbidden attribute), 35
status (aiohttp_json_api.errors.HTTPGatewayTimeout attribute), 38
status (aiohttp_json_api.errors.HTTPGone attribute), 36
status (aiohttp_json_api.errors.HTTPInsufficientStorage attribute), 39
status (aiohttp_json_api.errors.HTTPInternalServerError attribute), 37
status (aiohttp_json_api.errors.HTTPLocked attribute), 37
status (aiohttp_json_api.errors.HTTPMethodNotAllowed attribute), 35
status (aiohttp_json_api.errors.HTTPNotAcceptable attribute), 36
status (aiohttp_json_api.errors.HTTPNotExtended attribute), 39
status (aiohttp_json_api.errors.HTTPNotFound attribute), 35
status (aiohttp_json_api.errors.HTTPNotImplemented attribute), 38
status (aiohttp_json_api.errors.HTTPPreConditionFailed attribute), 36
status (aiohttp_json_api.errors.HTTPServiceUnavailable attribute), 38
status (aiohttp_json_api.errors.HTTPTooManyRequests attribute), 37
status (aiohttp_json_api.errors.HTTPUnauthorized attribute), 35

status (aiohttp_json_api.errors.HTTPUnprocessableEntity attribute), 37
status (aiohttp_json_api.errors.HTTPUnsupportedMediaType attribute), 36
status (aiohttp_json_api.errors.HTTPVariantAlsoNegotiates attribute), 39
Step (class in aiohttp_json_api.common), 28
Str (in module aiohttp_json_api.fields.attributes), 65
String (class in aiohttp_json_api.fields.attributes), 59

T

Tag (class in aiohttp_json_api.fields.decorators), 69
TimeDelta (class in aiohttp_json_api.fields.attributes), 62
TO_MANY (aiohttp_json_api.common.Relation attribute), 28
TO_ONE (aiohttp_json_api.common.Relation attribute), 28
ToMany (class in aiohttp_json_api.fields.relationships), 72
ToOne (class in aiohttp_json_api.fields.relationships), 72
type (aiohttp_json_api.common.ResourceID attribute), 28

U

UnresolvableIncludePath, 40
UnsortableField, 40
update_relationship() (aiohttp_json_api.abc.contoller.ControllerABC method), 55
update_relationship() (aiohttp_json_api.controller.DefaultController method), 33
update_resource() (aiohttp_json_api.abc.contoller.ControllerABC method), 55
update_resource() (aiohttp_json_api.controller.DefaultController method), 33
updates() (in module aiohttp_json_api.fields.decorators), 70
URI (class in aiohttp_json_api.fields.attributes), 63
url (aiohttp_json_api.pagination.PaginationABC attribute), 45
UUID (class in aiohttp_json_api.fields.attributes), 62

V

VALIDATE (aiohttp_json_api.fields.decorators.Tag attribute), 70
validate_relationship_object() (aiohttp_json_api.fields.base.Relationship method), 69
validate_relationship_object() (aiohttp_json_api.fields.relationships.ToMany method), 73
validate_relationship_object() (aiohttp_json_api.fields.relationships.ToOne method), 72
validate_resource_identifier() (aiohttp_json_api.fields.base.Relationship method), 69
validate_uri_resource_id() (in module aiohttp_json_api.utils), 51
validates() (in module aiohttp_json_api.fields.decorators), 70
ValidationError, 39
value (aiohttp_json_api.common.FilterRule attribute), 28
value_type (aiohttp_json_api.fields.trafarets.DecimalTrafaret attribute), 73