marshmallow-utils Documentation

Release 0.5.8

CERN

Nov 18, 2022
## CONTENTS

1  User’s Guide  
   1.1  Installation ............................................................... 3  
   1.2  Usage ................................................................. 3  

2  API Reference  
   2.1  API Docs ............................................................. 5  

3  Additional Notes  
   3.1  Contributing .......................................................... 13  
   3.2  Changes ............................................................. 15  
   3.3  License ............................................................... 17  
   3.4  Authors ............................................................... 17  

Python Module Index  

Index  


Extras and utilities for Marshmallow

Further documentation is available on https://marshmallow-utils.readthedocs.io/
This part of the documentation will show you how to get started in using Marshmallow-Utils.

1.1 Installation

Marshmallow-Utils is on PyPI so all you need is:

```bash
$ pip install marshmallow-utils
```

1.2 Usage

Extras and utilities for Marshmallow.

Currently, this library contains a couple of extra fields that helps with sanitizing data as shown in the following example:

```python
>>> from marshmallow_utils import fields
>>> from marshmallow import Schema
>>> class MySchema(Schema):
...     trim = fields.TrimmedString()
...     html = fields.SanitizedHTML()
...     text = fields.SanitizedUnicode()
...     isodate = fields.ISODateString()
...
>>> data = MySchema().load({
...     'trim': 'whitespace',
...     'html': '<script>evil()</script>',
...     'text': 'PDF copy/paste\u200b\u000b\u001b\u0018',
...     'isodate': '1999-10-27',
... })
>>> data['trim']
'whitespace'
>>> data['html']
'evil()'
>>> data['text']
'PDF copy/paste'
>>> data['isodate']
'1999-10-27'
```

Fields:
- **SanitizedUnicode**: Integrates the *ftfy* for fixing broken unicode text.
- **SanitizedHTML**: Integrates the *bleach* for HTML sanitization.
- **ISODateString**: Integrates the *arrow* for date parsing.
If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

2.1.1 Fields

Marshmallow fields.

```python
class marshmallow_utils.fields.BabelGettextDictField(locale, default_locale, **kwargs)
```

Translation string field (dump only).

This field dumps a translation string as output, by looking up the translation in the dictionary provided as input (the message catalog).

The lookup is performed via babel’s locale negotiation (e.g. en_US will also match en).

Basically the fields takes a data object like this:

```json
{'title': {'en': 'Text', 'da': 'Tekst'}}
```

and dumps this (in case the locale is english):

```json
{'title': 'Text'}
```

Initialize the field.

**Parameters**

- `locale` – The locale to lookup, or a function returning the locale.
- `default_locale` – The default locale in case the locale is not found. Can be a callable that returns the default locale.

```python
default_error_messages = {'invalid': 'Not a valid dictionary.', 'missing_locale': 'Translation not found for '}
```

Default error messages.

**property default_locale**

Get the default locale to be used.

**property locale**

Get the locale to be used.
class marshmallow_utils.fields.EDTFDateString(**kwargs)
    Extended Date/Time Format Level 0 date string field.
    A string field which an using the EDTF Validator.
    Constructor.

class marshmallow_utils.fields.FormatDate(format='medium', locale='en_US_POSIX', parse=True, **kwargs)
    Format a date object.
    Constructor.
    Parameters
    • format – The format to use (either short, medium, long or full).
    • locale – The current locale or a callable returning the current locale.

    format_value(value)
    Format an EDTF date.

class marshmallow_utils.fields.FormatDatetime(tzinfo=None, **kwargs)
    Format a datetime object.
    Constructor.
    format_value(value)
    Format an EDTF date.

    property tzinfo
    Get the timezone to use.

class marshmallow_utils.fields.FormatEDTF(format='medium', locale='en_US_POSIX', parse=True, **kwargs)
    Format an EDTF-formatted string.
    Constructor.
    Parameters
    • format – The format to use (either short, medium, long or full).
    • locale – The current locale or a callable returning the current locale.

    format_value(value)
    Format an EDTF date.

class marshmallow_utils.fields.FormatTime(tzinfo=None, **kwargs)
    Format a time object.
    Constructor.
    format_value(value)
    Format an EDTF date.

class marshmallow_utils.fields.Function(serialze: None | Callable[[Any], Any] | Callable[[Any, dict], Any] = None, deserialize: None | Callable[[Any], Any] | Callable[[Any, dict], Any] = None, **kwargs)
    Enhanced marshmallow Function field.
The main difference between the original marshmallow.fields.Function is for the deserialize function, which can now also point to a three-argument function, with the third argument being the original data that was passed to Schema.load. The following example better demonstrates how this works:

```python
def serialize_foo(obj, context):
    return {'serialize_args': {'obj': obj, 'context': context}}

def deserialize_foo(value, context, data):
    return {'deserialize_args': {
        'value': value, 'context': context, 'data': data}}

class FooSchema(marshmallow.Schema):
    foo = Function(serialize_foo, deserialize_foo)

FooSchema().dump({'foo': 42})
{'foo': {
    'serialize_args': {
        'obj': {'foo': 42},
        'context': {}  # no context was passed
    }
}}

FooSchema().load({'foo': 42}).data
{'foo': {
    'deserialize_args': {
        'value': 42,
        'context': {},  # no context was passed
        'data': {'foo': 42},
    }
}}
```

class marshmallow_utils.fields.GenFunction(
    serialize: None | Callable[[Any], Any] | Callable[[Any, dict], Any] = None,
    deserialize: None | Callable[[Any], Any] | Callable[[Any, dict], Any] = None, **kwargs)

Function field which is always deserialized.

class marshmallow_utils.fields.GenMethod(
    serialize: str | None = None, deserialize: str | None = None, **kwargs)

Method field which is always deserialized.

class marshmallow_utils.fields.ISODateString(
    format: str | None = None, **kwargs)

ISO8601-formatted date string.
ISODateString serializes to a date string and if it can’t, the field is ignored (missing).
NOTE: It serializes None to None.

class marshmallow_utils.fields.ISOLangString(
    validate=<function validate_iso639_3>, *args, **kwargs)

ISO language string field.
ISO language string field initialization.

class marshmallow_utils.fields.IdentifierSet(
    cls_or_instance: Field | type, **kwargs)

Identifier list with deduplication.
It assumes the items of the list contain a `scheme` property.

```python
default_error_messages = {'multiple_values': 'Only one identifier per scheme is allowed.'}
```

Default error messages.

```
class marshmallow_utils.fields.Link(template=None, params=None, permission=None, when=<function always>, **kwargs)
```

A link field that knows how to generate a link from an object.

```
```

A links field that knows to look in the context for the schema.

```
class marshmallow_utils.fields.Method(serialze: str | None = None, deserialize: str | None = None, **kwargs)
```

Enhanced marshmallow Method field.

The main difference between the original marshmallow.fields.Method is for the `deserialize` method, which can now also point to a two-argument method, with the second argument being the original data that was passed to `Schema.load`. The following example better demonstrates how this works:

```python
class BarSchema(marshmallow.Schema):
    
    bar = Method('serialize_bar', 'deserialize_bar')

    # Exactly the same behavior as in `marshmallow.fields.Method``
    def serialize_bar(self, obj):
        return {'serialize_args': {'obj': obj}}

    def deserialize_bar(self, value, data):
        return {'deserialize_args': {'value': value, 'data': data}}

BarSchema().dump({'bar': 42})
{'bar': {
    'serialize_args': {
        'obj': {'bar': 42}
    }
}}

BarSchema().load({'bar': 42})
{'bar': {
    'deserialize_args': {
        'data': {'bar': 42},
    }
}}
```

(continues on next page)
Nested object attribute field.

```python
class marshmallow_utils.fields.NestedAttribute
```

String field which sanitizes HTML using the bleach library.

```python
class marshmallow_utils.fields.SanitizedHTML
```

You can override the defaults like this:

```python
class MySchema(Schema):
    html = fields.SanitizedHTML(tags=['a'], attrs={'a': ['href']})
```

**Parameters**

- **tags** – List of allowed tags.
- **attrs** – Dictionary of allowed attributes per tag.

Initialize field.

```python
class marshmallow_utils.fields.SanitizedUnicode
```

String field that sanitizes and fixes problematic unicode characters.
class marshmallow_utils.fields.StrippedHTML(*, load_default: typing.Any = <marshmallow.missing>,
missing: typing.Any = <marshmallow.missing>,
dump_default: typing.Any = <marshmallow.missing>,
default: typing.Any = <marshmallow.missing>,

String field which strips HTML entities.

The value is stripped using the bleach library. Any already escaped value is being unescaped before return.

class marshmallow_utils.fields.TZDateTime(timezone=datetime.timezone.utc, format='iso', **kwargs)

Datetime field which converts naive datetimes to TZ aware datetimes.

Defaults to setting the timezone to UTC, and using ISO format.

Initialize the field.

class marshmallow_utils.fields.TrimmedString(*, load_default: typing.Any = <marshmallow.missing>,
missing: typing.Any = <marshmallow.missing>,
dump_default: typing.Any = <marshmallow.missing>,
default: typing.Any = <marshmallow.missing>,

String field which strips whitespace at the ends of the string.

2.1.2 Schemas

Marshmallow schemas.

class marshmallow_utils.schemas.GeometryObjectSchema(*, only: types.StrSequenceOrSet | None = None, exclude: types.StrSequenceOrSet = (),
many: bool = False, context: dict | None = None, load_only: types.StrSequenceOrSet = (),
dump_only: types.StrSequenceOrSet = (),
partial: bool | types.StrSequenceOrSet = False, unknown: str | None = None)

A GeoJSON Geometry Object schema.

See https://tools.ietf.org/html/rfc7946#section-3.1

Only Point, MultiPoint and Polygon are supported.
get_obj_type(obj)
Finds the object type in the dictionary.
This function is needed because the schemas return the dict itself. Not a geojson object.

class marshmallow_utils.schemas.IdentifierSchema(allowed_schemes, identifier_required=True, **kwargs)
Identifier with automatic scheme detection.
Constructor.
Parameters
- allowed_schemes – a dictionary of allowed schemes. Each key must contain a validator function and a scheme label.
- identifier_required – True when the identifier value is required.

error_messages: Dict[str, str] = {'invalid_identifier': 'Invalid {scheme} identifier.', 'invalid_scheme': 'Invalid scheme.', 'required': 'Missing data for required field.', 'unknown_scheme': 'No valid scheme recognized for identifier.'}
Overrides for default schema-level error messages

load_scheme(data, **kwargs)
Loads the scheme of the identifier.

normalize_identifier(data, **kwargs)
Normalizes the identifier based on the scheme.

validate_identifier(data, **kwargs)
Validate the identifier format and scheme.

class marshmallow_utils.schemas.MultiPointSchema(*, only: types.StrSequenceOrSet | None = None, exclude: types.StrSequenceOrSet = (), many: bool = False, context: dict | None = None, load_only: types.StrSequenceOrSet = (), dump_only: types.StrSequenceOrSet = (), partial: bool | types.StrSequenceOrSet = False, unknown: str | None = None)
GeoJSON MultiPoint schema.
See https://tools.ietf.org/html/rfc7946#section-3.1.3

class marshmallow_utils.schemas.PointSchema(*, only: types.StrSequenceOrSet | None = None, exclude: types.StrSequenceOrSet = (), many: bool = False, context: dict | None = None, load_only: types.StrSequenceOrSet = (), dump_only: types.StrSequenceOrSet = (), partial: bool | types.StrSequenceOrSet = False, unknown: str | None = None)
GeoJSON Point schema.
See https://tools.ietf.org/html/rfc7946#section-3.1.2

class marshmallow_utils.schemas.PolygonSchema(*, only: types.StrSequenceOrSet | None = None, exclude: types.StrSequenceOrSet = (), many: bool = False, context: dict | None = None, load_only: types.StrSequenceOrSet = (), dump_only: types.StrSequenceOrSet = (), partial: bool | types.StrSequenceOrSet = False, unknown: str | None = None)
GeoJSON Polygon schema.

See https://tools.ietf.org/html/rfc7946#section-3.1.6
CHAPTER THREE

ADDITIONAL NOTES

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

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

3.1.1 Types of Contributions

Report Bugs


If you are reporting a bug, please include:

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

Fix Bugs

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

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.
Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/marshmallow-utils/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 :)

3.1.2 Get Started!

Ready to contribute? Here’s how to set up marshmallow-utils for local development.

1. Fork the inveniosoftware/marshmallow-utils repo on GitHub.
2. Clone your fork locally:

   $ git clone git@github.com:your_name_here/marshmallow-utils.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 marshmallow-utils
   $ cd marshmallow-utils/
   $ pip install -e .[all]

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 tests:

   $ ./run-tests.sh

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8
   as well as build the Sphinx documentation and run doctests.

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

   $ git add .
   $ git commit -s
   -m "component: title without verbs"
   -m "*: NEW Adds your new feature."
   -m "*: FIX Fixes an existing issue."
   -m "*: BETTER Improves and existing feature."
   -m "*: Changes something that should not be visible in release notes."
   $ git push origin name-of-your-bugfix-or-feature
7. Submit a pull request through the GitHub website.

### 3.1.3 Pull Request Guidelines

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

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.5 and 3.6. Check [https://travis-ci.org/inveniosoftware/marshmallow-utils/pull_requests](https://travis-ci.org/inveniosoftware/marshmallow-utils/pull_requests) and make sure that the tests pass for all supported Python versions.

### 3.2 Changes

Version 0.5.8 (released 2022-11-18)

- Add translations
- Add translations workflow

Version 0.5.7 (released 2022-07-29)

- Add validation message for unrecognised identifier schema

Version 0.5.6 (released 2022-05-13)

- Move from setup.py to setup.cfg for purely declarative package definitions.
- Fix bug with empty locale on babel gettext_from_dict.

Version 0.5.5 (released 2022-03-01)

- Updated German translations.

Version 0.5.4 (released 2022-01-28)

- i18n: integrate translation flow
- fields: gettext_from_dict fallback lng to en.
- dep: bumped sphinx 4.2.0

Version 0.5.3 (released 2021-08-03)

- Fix bug in IdentifierSchema when identifier was not required.

Version 0.5.2 (released 2021-08-02)

- Add group of errors for proper handling of lists.

Version 0.5.0 (released 2021-07-08)

- Refactored to make it easier to use sanitization functions outside of Marshmallow fields.

Version 0.4.0 (released 2021-04-16)

- Change IdentifierScheme API to allow schemas unknown by IDUtils.

Version 0.3.10 (released 2021-04-12)

- Fix TZDateTime field serialization of None values.

Version 0.3.9 (released 2021-04-12)
• Adds TZDateTime field.
Version 0.3.7 (released 2021-03-05)
  • Fix issue in SanitizedHTML to allow passing empty lists to remove all tags.
Version 0.3.6 (released 2021-03-04)
  • Serializes None ISODates by removing them.
Version 0.3.5 (released 2021-01-25)
  • Adds BabelGettextDictField for dumping translation strings from dicts.
Version 0.3.4 (released 2021-01-24)
  • Makes Link field a bit easier to use by allowing a string to be passed in addition to an URITemplate.
Version 0.3.3 (released 2021-01-20)
  • Adds support for automatic scheme detection on identifiers.
  • Adds support for identifier sets.
Version 0.3.2 (released 2020-11-10)
  • Adds support for GeoJSON geometry object validation.
Version 0.3.1 (released 2020-11-08)
  • Adds support for localization of date, time, datetime, EDTF with string parsing support.
Version 0.3.0 (released 2020-11-06)
  • Adds support for proper localization of EDTF dates.
  • Refactored EDTFDateString field.
Version 0.1.5 (released 2020-09-24)
  • Fix to expand querystring params correctly
Version 0.1.4 (released 2020-09-17)
  • Adds LinksSchema similar to LinksField
Version 0.1.3 (released 2020-09-16)
  • Moved additional utilities into the library.
Version 0.1.2 (released 2020-09-16)
  • Ported GenFunction and GenMethod
Version 0.1.1 (released 2020-09-11)
  • Minor fix to allow a lower ftfy package version than the latest.
Version 0.1.0 (released 2020-09-11)
  • Initial public release.
3.3 License

MIT License

Copyright (C) 2020 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

3.4 Authors

Extras and utilities for Marshmallow

- Lars Holm Nielsen <lars.holm.nielsen@cern.ch>
- Guillaume Viger <guillaume.viger@northwestern.edu>
PYTHON MODULE INDEX

m
marshmallow_utils, 3
marshmallow_utils.fields, 5
marshmallow_utils.schemas, 10
<table>
<thead>
<tr>
<th>B</th>
<th>BabelGettextDictField</th>
<th>(class in marshmallow_utils.fields), 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>default_error_messages</td>
<td>(marshmallow_utils.fields.BabelGettextDictField attribute), 5</td>
</tr>
<tr>
<td></td>
<td>default_error_messages</td>
<td>(marshmallow_utils.fields.IdentifierSet attribute), 8</td>
</tr>
<tr>
<td></td>
<td>default_locale</td>
<td>(marshmallow_utils.fields.BabelGettextDictField property), 5</td>
</tr>
<tr>
<td>E</td>
<td>EDTFDateString</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td></td>
<td>error_messages</td>
<td>(marshmallow_utils.schemas.IdentifierSchema attribute), 11</td>
</tr>
<tr>
<td>F</td>
<td>format_value()</td>
<td>(marshmallow_utils.fields.FormatDate method), 6</td>
</tr>
<tr>
<td></td>
<td>format_value()</td>
<td>(marshmallow_utils.fields.FormatDatetime method), 6</td>
</tr>
<tr>
<td></td>
<td>format_value()</td>
<td>(marshmallow_utils.fields.FormatEDTF method), 6</td>
</tr>
<tr>
<td></td>
<td>format_value()</td>
<td>(marshmallow_utils.fields.FormatTime method), 6</td>
</tr>
<tr>
<td></td>
<td>FormatDate</td>
<td>(class in marshmallow_utils.fields), 6</td>
</tr>
<tr>
<td></td>
<td>FormatDatetime</td>
<td>(class in marshmallow_utils.fields), 6</td>
</tr>
<tr>
<td></td>
<td>FormatEDTF</td>
<td>(class in marshmallow_utils.fields), 6</td>
</tr>
<tr>
<td></td>
<td>FormatTime</td>
<td>(class in marshmallow_utils.fields), 6</td>
</tr>
<tr>
<td></td>
<td>Function</td>
<td>(class in marshmallow_utils.fields), 6</td>
</tr>
<tr>
<td>G</td>
<td>GenFunction</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td></td>
<td>GenMethod</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td></td>
<td>GeometryObjectSchema</td>
<td>(class in marshmallow_utils.schemas), 10</td>
</tr>
<tr>
<td>I</td>
<td>IdentifierSchema</td>
<td>(class in marshmallow_utils.schemas), 11</td>
</tr>
<tr>
<td></td>
<td>IdentifierSet</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td></td>
<td>ISODateString</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td></td>
<td>ISOLangString</td>
<td>(class in marshmallow_utils.fields), 7</td>
</tr>
<tr>
<td>L</td>
<td>Link</td>
<td>(class in marshmallow_utils.fields), 8</td>
</tr>
<tr>
<td></td>
<td>Links</td>
<td>(class in marshmallow_utils.fields), 8</td>
</tr>
<tr>
<td></td>
<td>load_scheme()</td>
<td>(marshmallow_utils.schemas.IdentifierSchema method), 11</td>
</tr>
<tr>
<td></td>
<td>locale</td>
<td>(marshmallow_utils.fields.BabelGettextDictField property), 5</td>
</tr>
<tr>
<td>M</td>
<td>marshmallow_utils module, 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>marshmallow_utils.fields module, 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>marshmallow_utils.schemas module, 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method</td>
<td>(class in marshmallow_utils.fields), 8</td>
</tr>
<tr>
<td></td>
<td>module</td>
<td>marshmallow_utils, 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>marshmallow_utils.fields, 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>marshmallow_utils.schemas, 10</td>
</tr>
<tr>
<td></td>
<td>MultiPointSchema</td>
<td>(class in marshmallow_utils.schemas), 11</td>
</tr>
<tr>
<td>N</td>
<td>NestedAttribute</td>
<td>(class in marshmallow_utils.fields), 9</td>
</tr>
<tr>
<td></td>
<td>normalize_identifier()</td>
<td>(marshmallow_utils.schemas.IdentifierSchema method), 11</td>
</tr>
</tbody>
</table>
PointSchema (class in marshmallow_utils.schemas), 11
PolygonSchema (class in marshmallow_utils.schemas), 11

SanitizedHTML (class in marshmallow_utils.fields), 9
SanitizedUnicode (class in marshmallow_utils.fields), 9
StrippedHTML (class in marshmallow_utils.fields), 9

TrimmedString (class in marshmallow_utils.fields), 10
TZDateTime (class in marshmallow_utils.fields), 10
tzinfo (marshmallow_utils.fields.FormatDateTime property), 6

validate_identifier() (marshmallow_utils.schemas.IdentifierSchema method), 11