Flask-Marshmallow

Release 1.2.1

unknown

Apr 15, 2024

CONTENTS

changelog // github // pypi // issues

CHAPTER

ONE

FLASK + MARSHMALLOW FOR BEAUTIFUL APIS

Flask-Marshmallow is a thin integration layer for Flask (a Python web framework) and marshmallow (an object serialization/deserialization library) that adds additional features to marshmallow, including URL and Hyperlinks fields for HATEOAS-ready APIs. It also (optionally) integrates with Flask-SQLAlchemy.

1.1 Get it now

```
pip install flask-marshmallow
```

Create your app.

```
from flask import Flask
from flask_marshmallow import Marshmallow

app = Flask(__name__)
ma = Marshmallow(app)
```

Write your models.

```
from your_orm import Model, Column, Integer, String, DateTime


class User(Model):
    email = Column(String)
    password = Column(String)
    date_created = Column(DateTime, auto_now_add=True)
```

Define your output format with marshmallow.

```
class UserSchema(ma.Schema):
    class Meta:
        # Fields to expose
        fields = ("email", "date_created", "_links")

    # Smart hyperlinking
    _links = ma.Hyperlinks(
        {
            "self": ma.URLFor("user_detail", values=dict(id="<id>")),
            "collection": ma.URLFor("users"),
        }
```

(continues on next page)

(continued from previous page)

```
    )


user_schema = UserSchema()
users_schema = UserSchema(many=True)
```

Output the data in your views.

```
@app.route("/api/users/")
def users():
    all_users = User.all()
    return users_schema.dump(all_users)


@app.route("/api/users/<id>")
def user_detail(id):
    user = User.get(id)
    return user_schema.dump(user)


# {
#     "email": "fred@queen.com",
#     "date_created": "Fri, 25 Apr 2014 06:02:56 -0000",
#     "_links": {
#         "self": "/api/users/42",
#         "collection": "/api/users/"
#     }
# }
```

1.2 Optional Flask-SQLAlchemy Integration

Flask-Marshmallow includes useful extras for integrating with Flask-SQLAlchemy and marshmallow-sqlalchemy.

To enable SQLAlchemy integration, make sure that both Flask-SQLAlchemy and marshmallow-sqlalchemy are installed.

```
pip install -U flask-sqlalchemy marshmallow-sqlalchemy
```

Next, initialize the `SQLAlchemy` and *`Marshmallow`* extensions, in that order.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_marshmallow import Marshmallow

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:////tmp/test.db"

# Order matters: Initialize SQLAlchemy before Marshmallow
db = SQLAlchemy(app)
ma = Marshmallow(app)
```

Note on initialization order

Flask-SQLAlchemy **must** be initialized before Flask-Marshmallow.

Declare your models like normal.

```
class Author(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(255))


class Book(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(255))
    author_id = db.Column(db.Integer, db.ForeignKey("author.id"))
    author = db.relationship("Author", backref="books")
```

Generate marshmallow Schemas from your models using *SQLAlchemySchema* or *SQLAlchemyAutoSchema*.

```
class AuthorSchema(ma.SQLAlchemySchema):
    class Meta:
        model = Author

    id = ma.auto_field()
    name = ma.auto_field()
    books = ma.auto_field()


class BookSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Book
        include_fk = True
```

You can now use your schema to dump and load your ORM objects.

```
db.create_all()
author_schema = AuthorSchema()
book_schema = BookSchema()
author = Author(name="Chuck Paluhniuk")
book = Book(title="Fight Club", author=author)
db.session.add(author)
db.session.add(book)
db.session.commit()
author_schema.dump(author)
# {'id': 1, 'name': 'Chuck Paluhniuk', 'books': [1]}
```

SQLAlchemySchema is nearly identical in API to `marshmallow_sqlalchemy.SQLAlchemySchema` with the following exceptions:

- By default, *SQLAlchemySchema* uses the scoped session created by Flask-SQLAlchemy.
- *SQLAlchemySchema* subclasses *flask_marshmallow.Schema*, so it includes the *jsonify* method.

Note: By default, Flask's `jsonify` method sorts the list of keys and returns consistent results to ensure that external HTTP caches aren't trashed. As a side effect, this will override ordered=True in the SQLAlchemySchema's `class`

`Meta` (if you set it). To disable this, set `JSON_SORT_KEYS=False` in your Flask app config. In production it's recommended to let `jsonify` sort the keys and not set `ordered=True` in your *SQLAlchemySchema* in order to minimize generation time and maximize cacheability of the results.

You can also use *ma.HyperlinkRelated* fields if you want relationships to be represented by hyperlinks rather than primary keys.

```
class BookSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Book

    author = ma.HyperlinkRelated("author_detail")
```

```
with app.test_request_context():
    print(book_schema.dump(book))
# {'id': 1, 'title': 'Fight Club', 'author': '/authors/1'}
```

The first argument to the *HyperlinkRelated* constructor is the name of the view used to generate the URL, just as you would pass it to the `url_for` function. If your models and views use the `id` attribute as a primary key, you're done; otherwise, you must specify the name of the attribute used as the primary key.

To represent a one-to-many relationship, wrap the *HyperlinkRelated* instance in a `marshmallow.fields.List` field, like this:

```
class AuthorSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Author

    books = ma.List(ma.HyperlinkRelated("book_detail"))
```

```
with app.test_request_context():
    print(author_schema.dump(author))
# {'id': 1, 'name': 'Chuck Paluhniuk', 'books': ['/books/1']}
```

CHAPTER

TWO

API

2.1 flask_marshmallow

Integrates the marshmallow serialization/deserialization library with your Flask application.

class flask_marshmallow.**Marshmallow**(*app: Flask | None = None*)

Wrapper class that integrates Marshmallow with a Flask application.

To use it, instantiate with an application:

```
from flask import Flask

app = Flask(__name__)
ma = Marshmallow(app)
```

The object provides access to the *Schema* class, all fields in marshmallow.fields, as well as the Flask-specific fields in *flask_marshmallow.fields*.

You can declare schema like so:

```
class BookSchema(ma.Schema):
    class Meta:
        fields = ("id", "title", "author", "links")

    author = ma.Nested(AuthorSchema)

    links = ma.Hyperlinks(
        {
            "self": ma.URLFor("book_detail", values=dict(id="<id>")),
            "collection": ma.URLFor("book_list"),
        }
    )
```

In order to integrate with Flask-SQLAlchemy, this extension must be initialized *after* flask_sqlalchemy.SQLAlchemy.

```
db = SQLAlchemy(app)
ma = Marshmallow(app)
```

This gives you access to ma.SQLAlchemySchema and ma.SQLAlchemyAutoSchema, which generate marshmallow Schema classes based on the passed in model or table.

```
class AuthorSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Author
```

Parameters
app (*Flask*) – The Flask application object.

init_app(*app: Flask*)

Initializes the application with the extension.

Parameters
app (*Flask*) – The Flask application object.

class flask_marshmallow.**Schema**(*, *only: Sequence[str] | AbstractSet[str] | None = None*, *exclude: Sequence[str] | AbstractSet[str] = ()*, *many: bool = False*, *context: dict | None = None*, *load_only: Sequence[str] | AbstractSet[str] = ()*, *dump_only: Sequence[str] | AbstractSet[str] = ()*, *partial: bool | Sequence[str] | AbstractSet[str] | None = None*, *unknown: str | None = None*)

Base serializer with which to define custom serializers.

See marshmallow.Schema for more details about the *Schema* API.

jsonify(*obj: Any*, *many: bool | None = None*, **args*, ***kwargs*) → Response

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether `obj` should be serialized as an instance or as a collection. If None, defaults to the value of the `many` attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to `flask.jsonify`.

Changed in version 0.6.0: Takes the same arguments as `marshmallow.Schema.dump`. Additional keyword arguments are passed to `flask.jsonify`.

Changed in version 0.6.3: The `many` argument for this method defaults to the value of the `many` attribute on the Schema. Previously, the `many` argument of this method defaulted to False, regardless of the value of `Schema.many`.

flask_marshmallow.**pprint**(*obj*, **args*, ***kwargs*) → None

Pretty-printing function that can pretty-print OrderedDicts like regular dictionaries. Useful for printing the output of `marshmallow.Schema.dump()`.

Deprecated since version 3.7.0: marshmallow.pprint will be removed in marshmallow 4.

2.2 flask_marshmallow.fields

Custom, Flask-specific fields.

See the `marshmallow.fields` module for the list of all fields available from the marshmallow library.

class flask_marshmallow.fields.**AbsoluteURLFor**(*endpoint: str*, *values: Dict[str, Any] | None = None*, ***kwargs*)

Field that outputs the absolute URL for an endpoint.

flask_marshmallow.fields.**AbsoluteUrlFor**

alias of *AbsoluteURLFor*

class flask_marshmallow.fields.**Config**(*key: str*, ***kwargs*)

A field for Flask configuration values.

Examples:

```
from flask import Flask

app = Flask(__name__)
app.config["API_TITLE"] = "Pet API"


class FooSchema(Schema):
    user = String()
    title = Config("API_TITLE")
```

This field should only be used in an output schema. A `ValueError` will be raised if the config key is not found in the app config.

Parameters

key (*str*) – The key of the configuration value.

class flask_marshmallow.fields.**File**(**args*, ***kwargs*)

A binary file field for uploaded files.

Examples:

```
class ImageSchema(Schema):
    image = File(required=True)
```

default_error_messages = {'invalid': 'Not a valid file.'}

Default error messages for various kinds of errors. The keys in this dictionary are passed to `Field.make_error`. The values are error messages passed to `marshmallow.exceptions.ValidationError`.

deserialize(*value: Any*, *attr: str | None = None*, *data: Mapping[str, Any] | None = None*, ***kwargs*)

Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in `data` to deserialize.
- **data** – The raw input data passed to `Schema.load`.
- **kwargs** – Field-specific keyword arguments.

Raises

ValidationError – If an invalid value is passed or if a required value is missing.

class flask_marshmallow.fields.**Hyperlinks**(*schema: Dict[str,* URLFor | *str]*, ***kwargs*)

Field that outputs a dictionary of hyperlinks, given a dictionary schema with *URLFor* objects as values.

Example:

```
_links = Hyperlinks(
    {
```

(continues on next page)

(continued from previous page)

```
        "self": URLFor("author", values=dict(id="<id>")),
        "collection": URLFor("author_list"),
    }
)
```

URLFor objects can be nested within the dictionary.

```
_links = Hyperlinks(
    {
        "self": {
            "href": URLFor("book", values=dict(id="<id>")),
            "title": "book detail",
        }
    }
)
```

Parameters

schema (*dict*) – A dict that maps names to *URLFor* fields.

class flask_marshmallow.fields.**URLFor**(*endpoint: str*, *values: Dict[str, Any] | None = None*, ***kwargs*)

Field that outputs the URL for an endpoint. Acts identically to Flask's `url_for` function, except that arguments can be pulled from the object to be serialized, and `**values` should be passed to the `values` parameter.

Usage:

```
url = URLFor("author_get", values=dict(id="<id>"))
https_url = URLFor(
    "author_get",
    values=dict(id="<id>", _scheme="https", _external=True),
)
```

Parameters

- **endpoint** (*str*) – Flask endpoint name.
- **values** (*dict*) – Same keyword arguments as Flask's url_for, except string arguments enclosed in < > will be interpreted as attributes to pull from the object.
- **kwargs** – keyword arguments to pass to marshmallow field (e.g. `required`).

flask_marshmallow.fields.**UrlFor**

alias of *URLFor*

2.3 flask_marshmallow.validate

Custom validation classes for various types of data.

class flask_marshmallow.validate.**FileSize**(*min: str | None = None*, *max: str | None = None*, *min_inclusive: bool = True*, *max_inclusive: bool = True*, *error: str | None = None*)

Validator which succeeds if the file passed to it is within the specified size range. If `min` is not specified, or is specified as `None`, no lower bound exists. If `max` is not specified, or is specified as `None`, no upper bound exists.

The inclusivity of the bounds (if they exist) is configurable. If `min_inclusive` is not specified, or is specified as `True`, then the `min` bound is included in the range. If `max_inclusive` is not specified, or is specified as `True`, then the `max` bound is included in the range.

Example:

```
class ImageSchema(Schema):
    image = File(required=True, validate=FileSize(min="1 MiB", max="2 MiB"))
```

Parameters

- **min** – The minimum size (lower bound). If not provided, minimum size will not be checked.
- **max** – The maximum size (upper bound). If not provided, maximum size will not be checked.
- **min_inclusive** – Whether the `min` bound is included in the range.
- **max_inclusive** – Whether the `max` bound is included in the range.
- **error** – Error message to raise in case of a validation error. Can be interpolated with `{input}`, `{min}` and `{max}`.

class flask_marshmallow.validate.**FileType**(*accept: Iterable[str]*, *error: str | None = None*)

Validator which succeeds if the uploaded file is allowed by a given list of extensions.

Example:

```
class ImageSchema(Schema):
    image = File(required=True, validate=FileType([".png"]))
```

Parameters

- **accept** – A sequence of allowed extensions.
- **error** – Error message to raise in case of a validation error. Can be interpolated with `{input}` and `{extensions}`.

2.4 flask_marshmallow.sqla

Integration with Flask-SQLAlchemy and marshmallow-sqlalchemy. Provides `SQLAlchemySchema` and `SQLAlchemyAutoSchema` classes that use the scoped session from Flask-SQLAlchemy.

class flask_marshmallow.sqla.**DummySession**

Placeholder session object.

class flask_marshmallow.sqla.**HyperlinkRelated**(*endpoint: str*, *url_key: str = 'id'*, *external: bool = False*, ***kwargs*)

Field that generates hyperlinks to indicate references between models, rather than primary keys.

Parameters

- **endpoint** (*str*) – Flask endpoint name for generated hyperlink.
- **url_key** (*str*) – The attribute containing the reference's primary key. Defaults to "id".
- **external** (*bool*) – Set to `True` if absolute URLs should be used, instead of relative URLs.

class flask_marshmallow.sqla.**SQLAlchemyAutoSchema**(**args*, ***kwargs*)

SQLAlchemyAutoSchema that automatically generates marshmallow fields from a SQLAlchemy model's or table's column. Uses the scoped session from Flask-SQLAlchemy by default.

See marshmallow_sqlalchemy.SQLAlchemyAutoSchema for more details on the *SQLAlchemyAutoSchema* API.

OPTIONS_CLASS

alias of *SQLAlchemyAutoSchemaOpts*

opts: SchemaOpts = <flask_marshmallow.sqla.SQLAlchemyAutoSchemaOpts object>

class flask_marshmallow.sqla.**SQLAlchemyAutoSchemaOpts**(*meta*, ***kwargs*)

class flask_marshmallow.sqla.**SQLAlchemySchema**(**args*, ***kwargs*)

SQLAlchemySchema that associates a schema with a model via the `model` class Meta option, which should be a `db.Model` class from flask_sqlalchemy. Uses the scoped session from Flask-SQLAlchemy by default.

See marshmallow_sqlalchemy.SQLAlchemySchema for more details on the *SQLAlchemySchema* API.

OPTIONS_CLASS

alias of *SQLAlchemySchemaOpts*

opts: SchemaOpts = <flask_marshmallow.sqla.SQLAlchemySchemaOpts object>

class flask_marshmallow.sqla.**SQLAlchemySchemaOpts**(*meta*, ***kwargs*)

CHAPTER

THREE

USEFUL LINKS

- Flask docs
- marshmallow docs

CHAPTER

FOUR

PROJECT INFO

4.1 License

```
Copyright Steven Loria and contributors

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.
```

4.2 Changelog

4.2.1 1.2.1 (2024-03-18)

Bug fixes:

- Fix `File` field when it receives an empty value (#301, apiflask/apiflask#551). Thanks @uncle-lv.
- Fix `validate.FileSize` to handle `SpooledTemporaryFile` (#300). Thanks again @uncle-lv.

4.2.2 1.2.0 (2024-02-05)

Features:

- Performance improvement to `validate.FileSize` (#293). Thanks @uncle-lv.

4.2.3 1.1.0 (2024-01-16)

Features:

- Add type coverage (#290).

4.2.4 1.0.0 (2024-01-16)

Features:

- Add field `fields.File`, `validate.FileType`, and `validate.FileSize` for deserializing uploaded files (#280, #282). Thanks @uncle-lv for the PR
- Add field `Config` for serializing Flask configuration values (#280, #281). Thanks @greyli for the PR.

Support:

- Support marshmallow-sqlalchemy>=0.29.0.
- Test against Python 3.12.
- Drop support for Python 3.7.

Other changes:

- *Backwards-incompatible*: Remove `flask_marshmallow.__version__` and `flask_marshmallow.__version_info__` attributes (#284). Use feature detection or `importlib.metadata.version("flask-marshmallow")` instead.

4.2.5 0.15.0 (2023-04-05)

- Changes to supported software versions.
  - python3.6 or later and marshmallow>=3.0.0 are now required
  - Add support for python3.11
  - For `sqlalchemy` integration, marshmallow-sqlalchemy>=0.28.2 and flask-sqlalchemy>=3.0.0 are now required
- *Backwards-incompatible*: `URLFor` and `AbsoluteURLFor` now do not accept parameters for `flask.url_for` as top-level parameters. They must always be passed in the `values` dictionary, as explained in the v0.14.0 changelog.

Bug fixes:

- Address distutils deprecation warning in Python 3.10 (#242). Thanks @GabrielLins64 for the PR.

4.2.6 0.14.0 (2020-09-27)

- Add `values` argument to `URLFor` and `AbsoluteURLFor` for passing values to `flask.url_for`. This prevents unrelated parameters from getting passed (#52, #67). Thanks @AlrasheedA for the PR.

Deprecation:

- Passing params to `flask.url_for` via `URLFor`'s and `AbsoluteURLFor`'s constructor params is deprecated. Pass `values` instead.

```
# flask-marshmallow<0.14.0


class UserSchema(ma.Schema):
    _links = ma.Hyperlinks(
        {
            "self": ma.URLFor("user_detail", id="<id>"),
        }
    )


# flask-marshmallow>=0.14.0


class UserSchema(ma.Schema):
    _links = ma.Hyperlinks(
        {
            "self": ma.URLFor("user_detail", values=dict(id="<id>")),
        }
    )
```

4.2.7 0.13.0 (2020-06-07)

Bug fixes:

- Fix compatibility with marshmallow-sqlalchemy<0.22.0 (#189). Thanks @PatrickRic for reporting.

Other changes:

- Remove unused `flask_marshmallow.sqla.SchemaOpts`.

4.2.8 0.12.0 (2020-04-26)

- *Breaking change*: `ma.ModelSchema` and `ma.TableSchema` are removed, since these are deprecated upstream.

> **Warning:** It is highly recommended that you use the newer `ma.SQLAlchemySchema` and `ma.SQLAlchemyAutoSchema` classes instead of `ModelSchema` and `TableSchema`. See the release notes for marshmallow-sqlalchemy 0.22.0 for instructions on how to migrate.

If you need to use `ModelSchema` and `TableSchema` for the time being, you'll need to import these directly from `marshmallow_sqlalchemy`.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_marshmallow import Marshmallow

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:////tmp/test.db"

db = SQLAlchemy(app)
ma = Marshmallow(app)

# flask-marshmallow<0.12.0


class AuthorSchema(ma.ModelSchema):
    class Meta:
        model = Author


# flask-marshmallow>=0.12.0 (recommended)


class AuthorSchema(ma.SQLAlchemyAutoSchema):
    class Meta:
        model = Author
        load_instance = True


# flask-marshmallow>=0.12.0 (not recommended)

from marshmallow_sqlalchemy import ModelSchema


class AuthorSchema(ModelSchema):
    class Meta:
        model = Author
        sql_session = db.session
```

Bug fixes:

- Fix binding Flask-SQLAlchemy's scoped session to `ma.SQLAlchemySchema` and `ma.SQLAlchemyAutoSchema`. (#180). Thanks @fnalonso for reporting.

4.2.9 0.11.0 (2020-02-09)

Features:

- Add support for `SQLAlchemySchema`, `SQLAlchemyAutoSchema`, and `auto_field` from marshmallow-sqlalchemy>=0.22.0 (#166).

Bug fixes:

- Properly restrict marshmallow-sqlalchemy version based on Python version (#158).

Other changes:

- Test against Python 3.8.

4.2.10 0.10.1 (2019-05-05)

Bug fixes:

- marshmallow 3.0.0rc6 compatibility (#134).

4.2.11 0.10.0 (2019-03-09)

Features:

- Add `ma.TableSchema` (#124).
- SQLAlchemy requirements can be installed with `pip install 'flask-marshmallow[sqlalchemy]'`.

Bug fixes:

- `URLFor`, `AbsoluteURLFor`, and `HyperlinkRelated` serialize to `None` if a passed attribute value is `None` (#18, #68, #72). Thanks @RobinRamuel, @ocervell, and @feigner for reporting.

Support:

- Test against Python 3.7.
- Drop support for Python 3.4. Only Python 2.7 and >=3.5 are supported.

4.2.12 0.9.0 (2018-04-29)

- Add support for marshmallow 3 beta. Thanks @SBillion for the PR.
- Drop support for Python 3.3. Only Python 2.7 and >=3.4 are supported.
- Updated documentation to fix example `ma.URLFor` target.

4.2.13 0.8.0 (2017-05-28)

- Fix compatibility with marshmallow>=3.0.

Support:

- *Backwards-incompatible*: Drop support for marshmallow<=2.0.0.
- Test against Python 3.6.

4.2.14 0.7.0 (2016-06-28)

- `many` argument to `Schema.jsonify` defaults to value of the `Schema` instance's `many` attribute (#42). Thanks @singingwolfboy.
- Attach `HyperlinkRelated` to `Marshmallow` instances. Thanks @singingwolfboy for reporting.

Support:

- Upgrade to invoke>=0.13.0.
- Updated documentation to reference `HyperlinkRelated` instead of `HyperlinkModelSchema`. Thanks @singingwolfboy.

- Updated documentation links to readthedocs.io subdomain. Thanks @adamchainz.

4.2.15 0.6.2 (2015-09-16)

- Fix compatibility with marshmallow>=2.0.0rc2.

Support:

- Tested against Python 3.5.

4.2.16 0.6.1 (2015-09-06)

- Fix compatibility with marshmallow-sqlalchemy>=0.4.0 (#25). Thanks @svenstaro for reporting.

Support:

- Include docs in release tarballs.

4.2.17 0.6.0 (2015-05-02)

Features:

- Add Flask-SQLAlchemy/marshmallow-sqlalchemy support via the `ModelSchema` and `HyperlinkModelSchema` classes.
- `Schema.jsonify` now takes the same arguments as `marshmallow.Schema.dump`. Additional keyword arguments are passed to `flask.jsonify`.
- `Hyperlinks` field supports serializing a list of hyperlinks (#11). Thanks @royrusso for the suggestion.

Deprecation/Removal:

- Remove support for `MARSHMALLOW_DATEFORMAT` and `MARSHMALLOW_STRICT` config options.

Other changes:

- Drop support for marshmallow<1.2.0.

4.2.18 0.5.1 (2015-04-27)

- Fix compatibility with marshmallow>=2.0.0.

4.2.19 0.5.0 (2015-03-29)

- *Backwards-incompatible*: Remove `flask_marshmallow.SchemaOpts` class and remove support for `MARSHMALLOW_DATEFORMAT` and `MARSHMALLOW_STRICT` (#8). Prevents a `RuntimeError` when instantiating a `Schema` outside of a request context.

4.2.20 0.4.0 (2014-12-22)

- *Backwards-incompatible*: Rename `URL` and `AbsoluteURL` to `URLFor` and `AbsoluteURLFor`, respectively, to prevent overriding marshmallow's `URL` field (#6). Thanks @svenstaro for the suggestion.
- Fix bug that raised an error when deserializing `Hyperlinks` and `URL` fields (#9). Thanks @raj-kesavan for reporting.

Deprecation:

- `Schema.jsonify` is deprecated. Use `flask.jsonify` on the result of `Schema.dump` instead.
- The `MARSHMALLOW_DATEFORMAT` and `MARSHMALLOW_STRICT` config values are deprecated. Use a base `Schema` class instead (#8).

4.2.21 0.3.0 (2014-10-19)

- Supports marshmallow >= 1.0.0-a.

4.2.22 0.2.0 (2014-05-12)

- Implementation as a proper class-based Flask extension.
- Serializer and fields classes are available from the `Marshmallow` object.

4.2.23 0.1.0 (2014-04-25)

- First release.
- `Hyperlinks`, `URL`, and `AbsoluteURL` fields implemented.
- `Serializer#jsonify` implemented.

PYTHON MODULE INDEX

f

INDEX