flask-apispec

Release 0.7.0

Jul 01, 2018

Contents

Release v0.7.0. (*Changelog*)

flask-apispec is a lightweight tool for building REST APIs in Flask. **flask-apispec** uses webargs for request parsing, marshmallow for response formatting, and apispec to automatically generate Swagger markup. You can use **flask-apispec** with vanilla Flask or a fuller-featured framework like Flask-RESTful.

CHAPTER 1

Guide

1.1 Installation

1.1.1 From the PyPI

```
$ pip install -U flask-apispec
```

1.1.2 From Source

flask-apispec is developed on Github.

You can clone the public repo:

```
git clone https://github.com/jmcarp/flask-apispec.git
```

Once you have the source, you can install it into your site-packages with

```
$ pip install invoke
$ invoke install
$ python setup.py install
```

1.2 Usage

1.2.1 Decorators

Use the *use_kwargs* and *marshal_with* decorators on functions, methods, or classes to declare request parsing and response marshalling behavior, respectively.

```
import flask
from webargs import fields
from flask_apispec import use_kwargs, marshal_with

from .models import Pet
from .schemas import PetSchema

app = flask.Flask(__name__)

@app.route('/pets')
@use_kwargs({'species': fields.Str()})
@marshal_with(PetSchema(many=True))
def list_pets(**kwargs):
    return Pet.query.filter_by(**kwargs).all()
```

Decorators can also be applied to view classes, e.g. Flask's `MethodView` or flask-restful's `Resource`. For correct inheritance behavior, view classes should use the `ResourceMeta` meta-class; for convenience, **flask-apispec** provides `MethodResource`, which inherits from `MethodView` and uses the `ResourceMeta` and `MethodViewType` meta-classes.

```
from flask_apispec import MethodResource

@marshal_with(PetSchema)
class StoreResource(MethodResource):

    def get(self, pet_id):
        return Pet.query.filter_by(id=pet_id).one()

    @use_kwargs(PetSchema)
    def put(self, pet_id, **kwargs):
        pet = Pet.query.filter_by(id=pet_id).one()
        for key, value in kwargs.items():
            setattr(pet, key, value)
        session.add(pet)
        session.commit()
        return pet

    @marshal_with(None, code=204)
    def delete(self, pet_id):
        pet = Pet.query.filter_by(id=pet_id).one()
        session.delete(pet)
        session.commit()
        return None
```

1.2.2 Inheritance

Subclasses of view classes inherit both class and method decorators. Method decorators are inherited by method name. This makes it possible to add a new decorator in a subclass without repeating all existing decorators.

```
class PetResource(MethodResource):

    @use_kwargs({'species': fields.Str()})
    @marshal_with(PetSchema)
    def get(self, **kwargs):
        return Pet.query.filter_by(**kwargs).all()
```

(continues on next page)

(continued from previous page)

```
class PetResourceExtended(PetResource):

    @use_kwargs({'indoor': fields.Bool()})
    def get(self, **kwargs):
        return super(PetResourceExtended, self)(**kwargs)
```

To allow subclasses to flexibly override parent settings, **flask-apispec** also provides the `Ref` helper. Using `Ref` looks up variables by name on the associated class at runtime. In this example, all methods in the `PetResource` view class serialize their outputs with `PetSchema`.

```
from flask_apispec import Ref

@marshal_with(Ref('schema'))
class BaseResource(MethodResource):

    schema = None

class PetResource(BaseResource):

    schema = PetSchema

    def get(self, pet_id):
        return Pet.query.filter_by(id=pet_id).one()
```

1.2.3 Swagger documentation

flask-apispec automatically generates Swagger 2.0 documentation for view functions and classes using apispec.

```
from flask_apispec import FlaskApiSpec

docs = FlaskApiSpec(app)

docs.register(list_pets)

app.add_url_rule('/stores', view_func=StoreResource.as_view('Store'))
docs.register(StoreResource)
```

By default, **flask-apispec** serves Swagger JSON at /swagger and Swagger UI at /swagger-ui. To override either URL, set the `APISPEC_SWAGGER_URL` and `APISPEC_SWAGGER_UI_URL` variables on the Flask application config, respectively. To disable serving either resource, set the corresponding configuration variable to `None`.

To add Swagger markup that is not currently supported by apispec, use the *doc* decorator:

```
@doc(description='a pet store', tags=['pets'])
class PetResource(MethodResource):
    pass
```

1.3 API Reference

1.3.1 Annotations

flask_apispec.annotations.**doc**(*inherit=None*, ***kwargs*)

Annotate the decorated view function or class with the specified Swagger attributes.

Usage:

```
@doc(tags=['pet'], description='a pet store')
def get_pet(pet_id):
    return Pet.query.filter(Pet.id == pet_id).one()
```

Parameters **inherit** – Inherit Swagger documentation from parent classes

flask_apispec.annotations.**marshal_with**(*schema*, *code='default'*, *description=''*, *inherit=None*, *apply=None*)

Marshal the return value of the decorated view function using the specified schema.

Usage:

```
class PetSchema(Schema):
    class Meta:
        fields = ('name', 'category')

@marshal_with(PetSchema)
def get_pet(pet_id):
    return Pet.query.filter(Pet.id == pet_id).one()
```

Parameters

- **schema** – Schema class or instance, or None
- **code** – Optional HTTP response code
- **description** – Optional response description
- **inherit** – Inherit schemas from parent classes
- **apply** – Marshal response with specified schema

flask_apispec.annotations.**use_kwargs**(*args*, *locations=None*, *inherit=None*, *apply=None*, ***kwargs*)

Inject keyword arguments from the specified webargs arguments into the decorated view function.

Usage:

```
from marshmallow import fields

@use_kwargs({'name': fields.Str(), 'category': fields.Str()})
def get_pets(**kwargs):
    return Pet.query.filter_by(**kwargs).all()
```

Parameters

- **args** – Mapping of argument names to Field objects, Schema, or a callable which accepts a request and returns a Schema
- **locations** – Default request locations to parse

- **inherit** – Inherit args from parent classes
- **apply** – Parse request with specified args

flask_apispec.annotations.**wrap_with**(*wrapper_cls*)
Use a custom `Wrapper` to apply annotations to the decorated function.

Parameters **wrapper_cls** – Custom `Wrapper` subclass

1.3.2 Extension

class flask_apispec.extension.**FlaskApiSpec**(*app=None*)
Flask-apispec extension.

Usage:

```
app = Flask(__name__)
app.config.update({
    'APISPEC_SPEC': APISpec(
        title='pets',
        version='v1',
        plugins=[MarshmallowPlugin()],
    ),
    'APISPEC_SWAGGER_URL': '/swagger/',
})
docs = FlaskApiSpec(app)

@app.route('/pet/<pet_id>')
def get_pet(pet_id):
    return Pet.query.filter(Pet.id == pet_id).one()

docs.register(get_pet)
```

Parameters

- **app** (*Flask*) – App associated with API documentation
- **spec** (*APISpec*) – apispec specification associated with API documentation

register(*target*, *endpoint=None*, *blueprint=None*, *resource_class_args=None*, *resource_class_kwargs=None*)
Register a view.

Parameters

- **target** – view function or view class.
- **endpoint** – (optional) endpoint name.
- **blueprint** – (optional) blueprint name.
- **resource_class_args** (*tuple*) – (optional) args to be forwarded to the view class constructor.
- **resource_class_kwargs** (*dict*) – (optional) kwargs to be forwarded to the view class constructor.

CHAPTER 2

Project info

2.1 Changelog

2.1.1 0.7.0 (2018-07-01)

Features:

- Supports apispec>=0.39.0 (#105). Older apispec versions are no longer supported.
- Upgrade swagger-ui to version 3.17.2 (#76). Thanks @paxnovem.

2.1.2 0.6.1 (2018-06-25)

Bug fixes:

- Fix resolution of path parameters (#92). Thanks @DStape for the fix.

2.1.3 0.6.0 (2018-03-11)

Features:

- Support marshmallow 3 beta. Thanks @tonycpsu for the PR.

2.1.4 0.5.0 (2018-03-04)

Features:

- Allow a schema factory to be passed to `use_args` and `use_kwargs` (#79). Thanks @decaz for the PR.

2.1.5 0.4.2 (2017-10-23)

Bug fixes:

- Fix wrapping of data parsed by schema with `many=True` (#64). Thanks @decaz for the catch and patch.

2.1.6 0.4.1 (2017-10-08)

Bug fixes:

- Include static assets for swagger-ui in distribution (#28, #57). Thanks @ArthurPBressan for reporting.

2.1.7 0.4.0 (2017-06-18)

Features:

- Add `resource_class_args` and `resource_class_kwargs` to `FlaskApiSpec.register` for passing constructor arguments to `MethodResource` classes. Thanks @elatomo.
- Add `FlaskApiSpec.init_app` method to support app factories (#21). Thanks @lafrech for the suggestion and thanks @dases for the PR.
- Defer registering views until `init_app` is called. Thanks @kageurufu for the PR.
- Add support for documenting headers and query params (#32). Thanks @rodjjo.
- Upon calling `FlaskApiSpec(app)`, register rules which have already been registered on `app` (#48). Thanks @henryfjordan for the fix.

Bug fixes:

- Return an empty list of parameters for undecorated views (#48). Thanks @henryfjordan for the fix.

Other changes:

- Test against Python 3.6. Drop support for Python 3.3.
- Support apispec>=0.17.0. Thanks @rth for fixing support for 0.20.0.

2.1.8 0.3.2 (2015-12-06)

- Fix Swagger-UI favicons. Thanks @benbeadle.

2.1.9 0.3.1 (2015-11-12)

- Update Swagger-UI assets. Thanks @evocateur.

2.1.10 0.3.0 (2015-11-11)

- Bundle templates and static files with install. Thanks @bmorgan21.
- Use readthedocs for documentation.

2.1.11 0.2.0 (2015-11-03)

- Add `FlaskApiSpec` Flask extension.
- Serve Swagger and Swagger-UI automatically.
- Reorganize file structure.

2.1.12 0.1.3 (2015-11-01)

- Rename to flask-apispec.
- Update to latest version of apispec.

2.1.13 0.1.2

- Update to latest version of webargs.

2.1.14 0.1.1

- Restrict inheritance to HTTP verbs.

2.1.15 0.1.0

- First release.

2.2 License

Python Module Index

f

Index