spectastic Documentation

Release 0.2.5

Jacob Straszynski

May 10, 2016

Contents

Contents:

CHAPTER 1

Spectastic

Request and response validation via Open API/Swagger schemas.

- Free software: Apache 2 License
- Documentation: https://spectastic.readthedocs.org.

1.1 Features

- Validation of Request-like objects against Open API/Swagger schemas.

1.2 TODO

- Response validation.
- Query parameter validation.
- collectionformat support that ties into werkzeug's datastructures.
- Authorization support not baked in.

CHAPTER 2

Installation

At the command line:

```
$ easy_install spectastic
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv spectastic
$ pip install spectastic
```

CHAPTER 3

Basic Usage

To use Spectastic in a project:

```
import spectastic
```

First, it's important to know that spectastic assumes that it's working with a valid Open API / Swagger schema to reduce it's dependency footprint. If you need to validate your schema, consider bravado-core.

Warning: You'll want to make sure you've specified `operation id's` for all paths. These aren't mandatory in an Open API specification, but are mandatory for spectastic.

3.1 Validating Incoming Requests

Spectastic offers two strategies for validation:

- Iterative API's that yield individual *`FieldError`* instances.
- *`ValidationErrors`* raising methods.

Most of spectastic's utility is contained within *`Operation`* instances. Lets make one using our test schema:

```
from spectastic.schema import Schema
from spectastic import Operation

schema = Spec(SPEC)
op = Operation.from_schema(schema, 'GetItem')
```

Once instantiated, you can validate an incoming *`BasicRequest`*:

```
op.validate_request(request)
```

Note: *`BasicRequest`* is probably not what you're using in your app. If you're using **flask**, check out *`convert_request()`* to make the conversion.

If there are validation errors, we'll raise a *`ValidationErrors`* exception, which contains an `errors` property consisting of a list of *`FieldError`*:

```
from spectastic.request import BasicRequest

request = BasicRequest(
```

```
    {"hello": "world"},
    {"Authorization": "basic beefbabaabbabeef"},
    {"query": "created:yesterday"},
    "/things/are/great",
)

try:
  op.validate_request(request)
except ValidationErrors as e:
  print e.errors[0].field
```

Note: Though it works out of the box, strictly speaking the request doesn't need to be a werkzeug Request. See *BasicRequest* for an example.

Spectastic is `MultiDict` and `Headers` aware. These data structures facilitate query parameters / headers that occur multiple times in a request e.g. a query such as "http://example.com?search=foo&search=bar".

3.2 Flask Integration

Spectastic's *flask_utils* module has some additional tools to automatically validate incoming requests for a given route against your schema:

```
from spectastic.contrib.flask_utils import validate_route

...

@validate_route(schema, 'GetItems')
@app.route('/items/')
def get_items(*args, **kwargs):
  return 'Success'
```

The *validate_route()* function has a few bonuses. The first argument may be a *Schema* instance or a callable that returns a Schema. An optional `responder` callable receives any *ValidationErrors* that may have occured and returns an appropriate flask-compatible response. You can use this to customize your error output.

The *default_responder()* simply outputs the general structure shown below along with a 400 status code:

```
{
  "errors": [
    {
      "msg": "Required path parameter is missing",
      "location": "path",
      "field": "query",
    },
    {
      ...
    }
  ]
}
```

CHAPTER 4

Spectastic API

4.1 Submodules

4.2 Errors

exception spectastic.errors.**FieldError**(*msg*, *location*, *field*)
Bases: exceptions.Exception

Parameters

- **msg** – The associated exception message.
- **location** – The location of the field e.g header, body, query.
- **field** – The name of the field.

VALID_LOCATIONS = ['header', 'body', 'query', 'path']

exception spectastic.errors.**ValidationErrors**(*errors=None*)
Bases: exceptions.Exception

4.3 Operation

exception spectastic.operation.**InvalidPath**
Bases: exceptions.Exception

Raised when a path does not match an operation's route.

class spectastic.operation.**Operation**(*schema*, *route*, *method*, *local_schema*)
Bases: object

Parameters

- **schema** (*dict*) – The overall schema for this API.
- **local_schema** (*dict*) – The schema specific to this operation.
- **route** (*string*) – The route for this operation.
- **method** (*string*) – The http method for this operation.

body_schema()
Returns a tuple of consisting of the body parameter and it's corresponding body schema.

extract_path_args(*path*)
Matches a request path against this operation's route, returning an intermediate dictionary of strings for each path parameter. The resulting dictionary is not validated nor are it's types coerced.

Return dict A dictionary of path arguments, with keys corresponding to the case sensitive name specified in the swagger schema.

static **from_schema**(*schema*, *operation_id*)

header_schema(*header_name*)
Returns the schema for a header parameter of a given name. The parameter name is case-insensitive.

Raises KeyError If the header is not found.

header_schemas()
Returns a list of all header parameter schemas.

iter_request_body_errors(*request_body*)
Validates a request body against the schema, yielding a *FieldError* for each failure.

iter_request_errors(*request*)
Validates an entire request, yielding a *FieldError* for each failure.

Parameters **request** (BasicRequest) – A request conforming to the structure outlined in *BasicRequest*

iter_request_header_errors(*request_headers*)
Validates individual request headers against the schema, yielding a *FieldError* for each failure.

iter_request_path_errors(*request_path*)
Validates a request path against the schema, yielding a *FieldError* for each failure.

iter_request_query_errors(*query_params*)
Validates a request query against the schema, yielding a *FieldError* for each failure.

path_schema(*path_param_name*)
Returns the schema for a path parameter of a given name. The parameter name is case-insensitive.

Raises KeyError If the query is not found.

path_schemas()
Returns a list of all path parameter schemas.

query_schema(*query_param_name*)
Returns the schema for a query parameter of a given name. The parameter name is case-insensitive.

Raises KeyError If the query is not found.

query_schemas()
Returns a list of all query parameter schemas.

validate_request(*request*)
Validates all components of a request.

Parameters **request** – A request conforming to the structure outlined in *BasicRequest*

Raises *ValidationErrors* When validation fails.

Return bool True on success.

validate_request_body(*request_body*)
Validates a request body against the operation schema.

Raises *ValidationErrors* When validation fails.

Return bool True on success.

validate_request_headers(*headers*)
Validates headers against the operation.

Raises *ValidationErrors* When validation fails.

Return bool True on success.

validate_request_path(*path_arguments*)
Validates headers against the operation.

Raises *ValidationErrors* When validation fails.

Return bool True on success.

validate_request_query(*query_params*)
Validates the query parameters from a request.

Parameters **query_params** (*dict|werkzeug.datastructures.MultiDict*) – A dictionary like object of query parameters.

Raises *ValidationErrors* When validation fails.

Return bool True on success.

validator

exception spectastic.operation.**OperationNotFound**
Bases: exceptions.Exception

Raised when an operation cannot be found in a schema.

spectastic.operation.**coerce_param**(*value*, *schema*)
Coerces a named parameter within a path, query, or headers to the type specified in the appropriate schema. If the string is not properly formated, raise a FieldError.

Parameters

- **value** (*string*) – The stringified value
- **schema** (*dict*) – The schema dictionary for the parameter.

Raises *FieldError* When the primitive type is not coercible.

Returns The value, coerced according to the parameter definition for the field named `name` located in `location`.

spectastic.operation.**find_operation**(*schema*, *operation_id*)
Returns a tuple of the route, method and schema for an operation.

Raises *OperationNotFound* When the operation_id does not exist anywhere in the sec.

4.4 Schema

class spectastic.schema.**Schema**(*schema_dict*)
Bases: dict

Simple wrapper around Swagger / Open API schema's. At some 'semblance' of type safety. Mostly used to resolve all references with the provided schema to make spectastic's job easier.

spectastic.schema.**resolve_all**(*schema_dict*)
Recursively resolve all refs to appropriate references within the spec dictionary.

Parameters **schema_dict** (*dict*) – A raw, json-decoded schema.

spectastic.schema.**resolve_ref**(*schema_dict*, *ref*)
Resolves a local ref in the form of `#/definitions/dog` or `#/parameters/cat`.

Parameters **schema_dict** (*dict*) – A raw, json-decoded schema.

4.5 Request

class spectastic.request.**BasicRequest**(*body*, *headers*, *query*, *path*)
Bases: object

Demonstrates the most basic interface required by spectastic for proper request validation.

Parameters

- **headers** (*dict*) – A dictionary-like object of headers.
- **query** (*dict*) – A dictionary-like object of query parameters.
- **body** (*dict*) – Generally a json-encoded string, or JSON decoded dictionary.
- **path** (*string*) – The request path of the request.

4.6 Flask Utilities

spectastic.contrib.flask_utils.**convert_request**(*flask_request*)
Converts a flask `flask.Request` to a *BasicRequest*

Returns *BasicRequest*

spectastic.contrib.flask_utils.**default_responder**(*validation_errors*)
Generates a flask response from provided `validation_errors`.

Parameters **validation_errors** (ValidationErrors) – A instance containing an aggregate of all errors discovered during request parsing.

spectastic.contrib.flask_utils.**validate_route**(*schema*, *operation_id*, *responder=None*)

Parameters

- **schema** (Schema) – The schema to use for validation. May also be a callable that receives a flask request and returns a *Schema*.
- **operation_id** (*string*) – The operation id to validate. May also be a callable that receives a flask request and returns a *Schema*.

4.7 Module contents

CHAPTER 5

Credits

5.1 Development Lead

- Jacob Straszynski <jacob.straszynski@planet.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.2.2 (2016-03-24)

- Addressed an issue when validating objects with more than one required field.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

S

B

C

D

E

F

H

I

O

P

Q

R

S

V