bravado

Release

May 08, 2017

Contents

Bravado is a python client library for Swagger 2.0 services.

More information on Swagger can be found on the Swagger website

It aims to be a complete replacement to swagger codegen.

Features include:

- Dynamically generated client - no code generation needed!
- Synchronous and Asynchronous http clients out of the box.
- Strict validations to verify that your Swagger Schema is v2.0 compatible.
- HTTP request and response validation against your Swagger Schema.
- Swagger models as Python types (no need to deal with JSON).
- REPL friendly navigation of your Swagger schema with docstrings for Resources, Operations and Models.
- Ingestion of your Swagger schema via http or a local file path.

Contents:

CHAPTER 1

Quickstart

Usage

Install the latest stable version from PyPi:

```
$ pip install --upgrade bravado
```

Your first Hello World! (or Hello Pet)

Here is a simple example to try from a REPL (like IPython):

```
from bravado.client import SwaggerClient

client = SwaggerClient.from_url("http://petstore.swagger.io/v2/swagger.json")
pet = client.pet.getPetById(petId=42).result()
```

If you were lucky, and pet Id with 42 was present, you will get back a result. It will be a dynamically created instance of `bravado.model.Pet` with attributes `category`, etc. You can even try `pet.category.id` or `pet.tags[0]`.

Sample Response:

```
Pet(category=Category(id=0L, name=u''), status=u'', name=u'', tags=[Tag(id=0L, name=u'
↪')], photoUrls=[u''], id=2)
```

If you got a `404`, try some other petId.

Lets try a POST call

Here we will demonstrate how `bravado` hides all the `JSON` handling from the user, and makes the code more Pythonic.

```
Pet = client.get_model('Pet')
Category = client.get_model('Category')
pet = Pet(id=42, name="tommy", category=Category(id=24))
client.pet.addPet(body=pet).result()
```

Time to get Twisted! (Asynchronous client)

`bravado` provides an out of the box asynchronous http client with an optional timeout parameter.

Your first Hello World! (or Hello Pet) above can be rewritten to use the asynchronous Fido client like so:

```
from bravado.client import SwaggerClient
from bravado.fido_client import FidoClient

client = SwaggerClient.from_url(
    'http://petstore.swagger.io/v2/swagger.json',
    FidoClient()
)

result = client.pet.getPetById(petId=42).result(timeout=4)
```

Note: `timeout` parameter here is the timeout (in seconds) the call will block waiting for the complete response. The default timeout is to wait indefinitely.

Note: To use Fido client you should install bravado with fido extra via `pip install bravado[fido]`.

This is too fancy for me! I want a simple dict response!

`bravado` has taken care of that as well. Configure the client to not use models.

```
from bravado.client import SwaggerClient
from bravado.fido_client import FidoClient

client = SwaggerClient.from_url(
    'http://petstore.swagger.io/v2/swagger.json',
    config={'use_models': False}
)

result = client.pet.getPetById(petId=42).result(timeout=4)
```

`result` will look something like:

```
{
    'category': {
```

```
        'id': 0L,
        'name': u''
    },
    'id': 2,
    'name': u'',
    'photoUrls': [u''],
    'status': u'',
    'tags': [
        {'id': 0L, 'name': u''}
    ]
}
```

CHAPTER 2

Configuration

Client Configuration

You can configure certain behaviours when creating a `SwaggerClient`.

bravado and bravado-core use the same config dict. The full documentation for bravado-core config keys is available too.

```
from bravado.client import SwaggerClient, SwaggerFormat

my_super_duper_format = SwaggerFormat(...)

config = {
    # === bravado config ===

    # Determines what is returned by the service call.
    'also_return_response': False,

    # === bravado-core config ====

    #  validate incoming responses
    'validate_responses': True,

    # validate outgoing requests
    'validate_requests': True,

    # validate the swagger spec
    'validate_swagger_spec': True,

    # Use models (Python classes) instead of dicts for #/definitions/{models}
    'use_models': True,

    # List of user-defined formats
    'formats': [my_super_duper_format],
```

```
}

client = SwaggerClient.from_url(..., config=config)
```

Config key	Type	Default	Description
also_return_response	boolean	False	Determines what is returned by the service call. Specifically, the return value of `HttpFuture.result()`. When `False`, the swagger result is returned. When `True`, the tuple `(swagger result, http response)` is returned. See *Getting access to the HTTP response*.

Per-request Configuration

Configuration can also be applied on a per-request basis by passing in `_request_options` to the service call.

```
client = SwaggerClient.from_url(...)
request_options = { ... }
client.pet.getPetById(petId=42, _request_options=request_options).result()
```

Config key	Type	Default	Description
connect_timeout	float	N/A	TCP connect timeout in seconds. This is passed along to the http_client when making a service call.
headers	dict	N/A	Dict of http headers to to send with the outgoing request.
response_callbacks	list of callables	[]	List of callables that are invoked after the incoming response has been validated and unmarshalled but before being returned to the calling client. This is useful for client decorators that would like to hook into the post-receive event. The callables are executed in the order they appear in the list. Two parameters are passed to each callable: - `incoming_response` of type `bravado_core.response.IncomingResponse` - `operation` of type `bravado_core.operation.Operation`
timeout	float	N/A	TCP idle timeout in seconds. This is passed along to the http_client when making a service call.

CHAPTER 3

Advanced Usage

Validations

`bravado` validates the schema against the Swagger 2.0 Spec. Validations are also done on the requests and the responses.

Validation example:

```
pet = Pet(id="I should be integer :(", name="tommy")
client.pet.addPet(body=pet).result()
```

will result in an error like so:

```
TypeError: id's value: 'I should be integer :(' should be in types (<type 'long'>,
↪<type 'int'>)
```

Note: If you'd like to disable validation of outgoing requests, you can set `validate_requests` to `False` in the `config` passed to `SwaggerClient.from_url(...)`.

The same holds true for incoming responses with the `validate_responses` config option.

Adding Request Headers

`bravado` allows you to pass request headers along with any request.

```
Pet = client.get_model('Pet')
Category = client.get_model('Category')
pet = Pet(id=42, name="tommy", category=Category(id=24))
swagger_client.pet.addPet(
    body=pet,
```

```
    _request_options={"headers": {"foo": "bar"}},
).result()
```

Docstrings

`bravado` provides docstrings to operations and models to quickly get the parameter and response types. Due to an implementation limitation, an operation's docstring looks like a class docstring instead of a function docstring. However, the most useful information about parameters and return type is present in the `Docstring` section.

Note: The `help` built-in does not work as expected for docstrings. Use the `?` method instead.

```
>> petstore.pet.getPetById?

Type:       CallableOperation
String Form:<bravado.client.CallableOperation object at 0x241b5d0>
File:       /some/dir/bravado/bravado/client.py
Definition: c.pet.getPetById(self, **op_kwargs)
Docstring:
[GET] Find pet by ID

Returns a single pet

:param petId: ID of pet to return
:type petId: integer
:returns: 200: successful operation
:rtype: object
:returns: 400: Invalid ID supplied
:returns: 404: Pet not found
Constructor Docstring::type operation: :class:`bravado_core.operation.Operation`
Call def:   c.pet.getPetById(self, **op_kwargs)
Call docstring:
Invoke the actual HTTP request and return a future that encapsulates
the HTTP response.

:rtype: :class:`bravado.http_future.HTTPFuture`
```

Docstrings for models can be retrieved as expected:

```
>> pet_model = petstore.get_model('Pet')
>> pet_model?

Type:       type
String Form:<class 'bravado_core.model.Pet'>
File:       /some/dir/bravado_core/model.py
Docstring:
Attributes:

category: Category
id: integer
name: string
photoUrls: list of string
status: string - pet status in the store
tags: list of Tag
```

```
Constructor information:
 Definition:pet_type(self, **kwargs)
```

Default Values

`bravado` uses the default values from the spec if the value is not provided in the request.

In the Pet Store example, operation `findPetsByStatus` has a `default` of `available`. That means, `bravado` will plug that value in if no value is provided for the parameter.

```
client.pet.findPetByStatus()
```

Loading swagger.json by file path

`bravado` also accepts `swagger.json` from a file path. Like so:

```
client = SwaggerClient.from_url('file:///some/path/swagger.json')
```

Alternatively, you can also use the `load_file` helper method.

```
from bravado.swagger_model import load_file

client = SwaggerClient.from_spec(load_file('/path/to/swagger.json'))
```

Getting access to the HTTP response

The default behavior for a service call is to return the swagger result like so:

```
pet = petstore.pet.getPetById(petId=42).result()
print pet.name
```

However, there are times when it is necessary to have access to the actual HTTP response so that the HTTP headers or HTTP status code can be used. This is easily done via configuration to return a `(swagger result, http response)` tuple from the service call.

```
petstore = Swagger.from_url(..., config={'also_return_response': True})
pet, http_response = petstore.pet.getPetById(petId=42).result()
assert isinstance(http_response, bravado_core.response.IncomingResponse)
print http_response.headers
print http_response.status_code
print pet.name
```

CHAPTER 4

Changelog

8.4.0 (2016-09-27)

- Remove support for Python 2.6, fixing a build failure.
- Switch from Python 3.4 to Python 3.5 for tests.

8.3.0 (2016-06-03)

- Bravado using Fido 3.2.0 python 3 ready

8.2.0 (2016-04-29)

- Bravado compliant to Fido 3.0.0
- Dropped use of concurrent futures in favor of crochet EventualResult
- Workaround for bypassing a unicode bug in python *requests* < 2.8.1

8.1.2 (2016-04-18)

- Don't unnecessarily constrain the version of twisted when not using python 2.6

8.1.1 (2016-04-13)

- Removed logic to build multipart forms. Using python 'requests' instead to build the entire http request.

8.1.0 (2016-04-04)

- Support for YAML Swagger specs - PR #198
- Remove pytest-mock dependency from requirements-dev.txt. No longer used and it was breaking the build.
- Requires bravado-core >= 4.2.2
- Fix unit test for default values getting sent in the request

8.0.1 (2015-12-02)

- Require twisted < 15.5.0 since Python 2.6 support was dropped

8.0.0 (2015-11-25)

- Support for recursive $refs
- Support for remote $refs e.g. Swagger 2.0 specs that span multiple json files
- Requires bravado-core 4.0.0 which is not backwards compatible (See its CHANGELOG)
- Transitively requires swagger-spec-validator 2.0.2 which is not backwards compatible (See its CHANGELOG)

7.0.0 (2015-10-23)

- Support per-request response_callbacks to enable `SwaggerClient` decorators to instrument an `IncomingResponse` post-receive. This is a non-backwards compatible change iff you have implemented a custom `HttpClient`. Consult the changes in signature to `HttpClient.request()` and `HttpFuture`'s constructor.
- Config option `also_return_response` is supported on a per-request basis.

6.1.1 (2015-10-19)

- Fix `IncomingResponse` subclasses to provide access to the http headers.
- Requires bravado-core >= 3.1.0

6.1.0 (2015-10-19)

- Clients can now access the HTTP response from a service call to access things like headers and status code. See Advanced Usage

6.0.0 (2015-10-12)

- User-defined formats are no longer global. The registration mechanism has changed and is now done via configuration. See Configuration

5.0.0 (2015-08-27)

- Update ResourceDecorator to return an operation as a CallableOperation instead of a function wrapper (for the docstring). This allows further decoration of the ResourceDecorator.

4.0.0 (2015-08-10)

- Consistent bravado.exception.HTTPError now thrown from both Fido and Requests http clients.
- HTTPError refactored to contain an optional detailed message and Swagger response result.

3.0.0 (2015-08-03)

- Support passing in connect_timeout and timeout via _request_options to the Fido and Requests clients
- Timeout in HTTPFuture now defaults to None (wait indefinitely) instead of 5s. You should make sure any calls to http_future.result(..) without a timeout are updated accordingly.

2.1.0 (2015-07-20)

- Add warning for deprecated operations

2.0.0 (2015-07-13)

- Assume responsibility for http invocation (used to be in bravado-core)

1.1.0 (2015-07-06)

- Made bravado compatible with Py34

1.0.0 (2015-06-26)

- Fixed petstore demo link
- Pick up bug fixes from bravado-core 1.1.0

1.0.0-rc2 (2015-06-01)

- Renamed ResponseLike to IncomingResponse to match bravado-core

1.0.0-rc1 (2015-05-13)

- Initial version - large refactoring/rewrite of swagger-py 0.7.5 to support Swagger 2.0

CHAPTER 5

Indices and tables

- genindex
- modindex