OASTools

Release 0.0.2

Sep 17, 2019

Contents

CHAPTER 1

Contents

1.1 About

1.1.1 License

The software is released under the MIT license

```
The MIT License (MIT)

Copyright (c) 2019 Bilal Shaikh

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.
```

1.1.2 Acknowledgements

This package was created by coallating and adapting a variety of existing tools. The tools and their licences are listed here:

1.2 Installation

1.2.1 Prerequisites

- Python
- Pip

1.2.2 Latest revision from GitHub

Run the following command to install the latest version from GitHub:

```
pip install git+https://github.com/BilalShaikh42/oastools.git#egg=oastools[all]
```

1.2.3 Latest release From PyPI

Run the following command to install the latest release from PyPI:

```
pip install oastools[all]
```

1.3 Overview

1.4 oastools

1.4.1 oastools package

Subpackages

oastools.parse package

Submodules

oastools.parse.parser module

parser

Author Bilal Shaikh < bilalshaikh42@gmail.com >

Date 2019-08-29

Copyright 2019, Bilal Shaikh

License MIT

Adapted from Swagger_Parser (MIT Licence) to work with openapi 3 https://github.com/Trax-air/swagger-parser/blob/master/swagger_parser/swagger_parser.py

class oastools.parse.parser.**OASParser**(*oas_spec*, *use_example=True*)

Bases: object

build_definitions_example()

Parse all definitions in the swagger specification.

build_one_definition_example(*def_name*)

Build the example for the given definition. :param def_name: Name of the definition.

Returns True if the example has been created, False if an error occured.

static check_type(*value*, *type_def*)

Check if the value is in the type given in type_def. :param value: the var to test. :param type_def: string representing the type in swagger.

Returns True if the type is correct, False otherwise.

static get_definition_name_from_ref(*ref*)
Get the definition name of the given $ref value(Swagger value). :param ref: ref value (ex: "#/definitions/CustomDefinition")

Returns The definition name corresponding to the ref.

get_dict_definition(*dict*, *get_list=False*)
Get the definition name of the given dict. :param dict: dict to test. :param get_list: if set to true, return a list of definition that match the body.

if False, only return the first.

Returns The definition name or None if the dict does not match any definition. If get_list is True, return a list of definition_name.

get_example_from_prop_spec(*prop_spec*, *from_allof=False*)
Return an example value from a property specification. :param prop_spec: the specification of the property. :param from_allof: whether these properties are part of an

allOf section

Returns An example value

get_path_spec(*path*, *action=None*)
Get the specification matching with the given path. :param path: path we want the specification. :param action: get the specification for the given action.

Returns A tuple with the base name of the path and the specification. Or (None, None) if no specification is found.

get_paths_data()
Get data for each paths in the swagger specification. Get also the list of operationId.

get_request_data(*path*, *action*, *body=None*)
Get the default data and status code of the given path + action request. :param path: path of the request. :param action: action of the request(get, post, delete...) :param body: body sent, used to sent it back for post request.

Returns A tuple with the default response data and status code In case of default status_code, use 0

get_response_example(*resp_spec*)
Get a response example from a response spec.

get_send_request_correct_body(*path*, *action*)
Get an example body which is correct to send to the given path with the given action. :param path: path of the request :param action: action of the request (get, post, put, delete)

Returns A dict representing a correct body for the request or None if no body is required.

validate_additional_properties(*valid_response*, *response*)

Validates additional properties. In additional properties, we only need to compare the values of the dict, not the keys

Parameters

- **valid_response** – An example response (for example generated in _get_example_from_properties(self, spec)) Type is DICT
- **response** – The actual dict coming from the response Type is DICT

Returns A boolean - whether the actual response validates against the given example

validate_definition(*definition_name*, *dict_to_test*, *definition=None*)
Validate the given dict according to the given definition. :param definition_name: name of the the definition. :param dict_to_test: dict to test.

Returns True if the given dict match the definition, False otherwise.

validate_request(*path*, *action*, *body=None*, *query=None*)

Check if the given request is valid. Validates the body and the query # Rules to validate the BODY: # Let's limit this to mime types that either contain 'text' or 'json' # 1. if body is None, there must not be any required parameters in # the given schema # 2. if the mime type contains 'json', body must not be '', but can # be {} # 3. if the mime type contains 'text', body can be any string # 4. if no mime type ('consumes') is given.. DISALLOW # 5. if the body is empty ('' or {}), there must not be any required parameters # 6. if there is something in the body, it must adhere to the given schema # -> will call the validate body function

Parameters

- **path** – path of the request.
- **action** – action of the request(get, post, delete...).
- **body** – body of the request.
- **query** – dict with the query parameters.

Returns True if the request is valid, False otherwise.

class oastools.parse.parser.**OpenApiParser**(*path=None*)
Bases: object

Module contents

oastools.resolve package

Submodules

oastools.resolve.resolver module

resolver

Author Bilal Shaikh < bilalshaikh42@gmail.com >

Date 2019-08-30

Copyright 2019, Bilal Shaikh

License MIT

class oastools.resolve.resolver.**Resolver**(*openapi_spec*, *path=None*)
Bases: object

oastools.resolve.resolver.**main**()

oastools.resolve.resolver.**resolve**(*path*, *rootpath*)

oastools.resolve.resolver.**traverse**(*spec*, *rootpath*, *callback=<function resolve>*)

Module contents

oastools.utils package

Submodules

oastools.utils.errors module

exception oastools.utils.errors.**Error**(*basemessage='Unspecified Error'*, *message=''*)
Bases: Exception

exception oastools.utils.errors.**NotImplementedError**(*message=''*)
Bases: *oastools.utils.errors.Error*

An error to indicate that the requested operation has not yet been implemented

exception oastools.utils.errors.**ParseError**(*basemessage='ParserError'*, *message=''*)
Bases: *oastools.utils.errors.Error*

oastools.utils.files module

class oastools.utils.files.**FileType**
Bases: enum.Enum

An enumeration.

JSON = 1

UNKWN = 0

YAML = 2

oastools.utils.files.**parse_file**(*path*)

Module contents

oastools.validate package

Module contents

Submodules

oastools.core module

oastools

Author Name <email>

Date 2019-8-28

Copyright 2019, Bilal Shaikh

License MIT

class oastools.core.**ExampleClass**(*arg_1*, *arg_2*, *kwarg_1=None*, *kwarg_2=None*)

Bases: object

Descipton of ExampleClass

attr_1

description of attr_1

Type type of attr_1

attr_2

description of attr_2

Type type of attr_2

...

method_1(*arg_1*, *arg_2*, *kwarg_1=None*, *kwarg_2=None*)

Description of method_1

Parameters

- **arg_1** (type of arg_1) – description of arg_1
- **arg_2** (type of arg_2) – description of arg_2
- **kwarg_1** (type of kwarg_1, optional) – description of kwarg_1
- **kwarg_2** (type of kwarg_2, optional) – description of kwarg_2
- .. –

Returns description of return value

Return type type of return value

Raises *:obj:'type of raised exception(s)'* – description of raised exceptions

Module contents

Python Module Index

o

Index

V

Y