aiohttp-swaggerDocumentation

Release 1.0.0

Daniel Garcia - cr0hn

Oct 22, 2021

Contents

aiohttp-swagger: Swagger API Documentation builder for aiohttp server

Code	https://github.com/cr0hn/aiohttp-swagger
Issues	https://github.com/cr0hn/aiohttp-swagger/issues/
Python version	Python 3.4 and above
License	BSD
Author	Daniel Garcia (cr0hn) - @ggdaniel

CHAPTER 1

Index

1.1 Installation

1.1.1 Simple

Install `aiohttp-swagger` is so easy:

```
$ python3.5 -m pip install aiohttp-swagger
```

1.1.2 With extra performance

`aiohttp-swagger` also includes some optional dependencies to add extra performance but requires a bit different installation, because they (usually) depends of C extensions.

To install the tool with extra performance you must do:

```
$ python3.5 -m pip install 'aiohttp-swagger[performance]'
```

Note: Remember that `aiohttp-swagger` only runs in **Python 3.4** and above.

1.2 Quick start

Document an API is so simple:

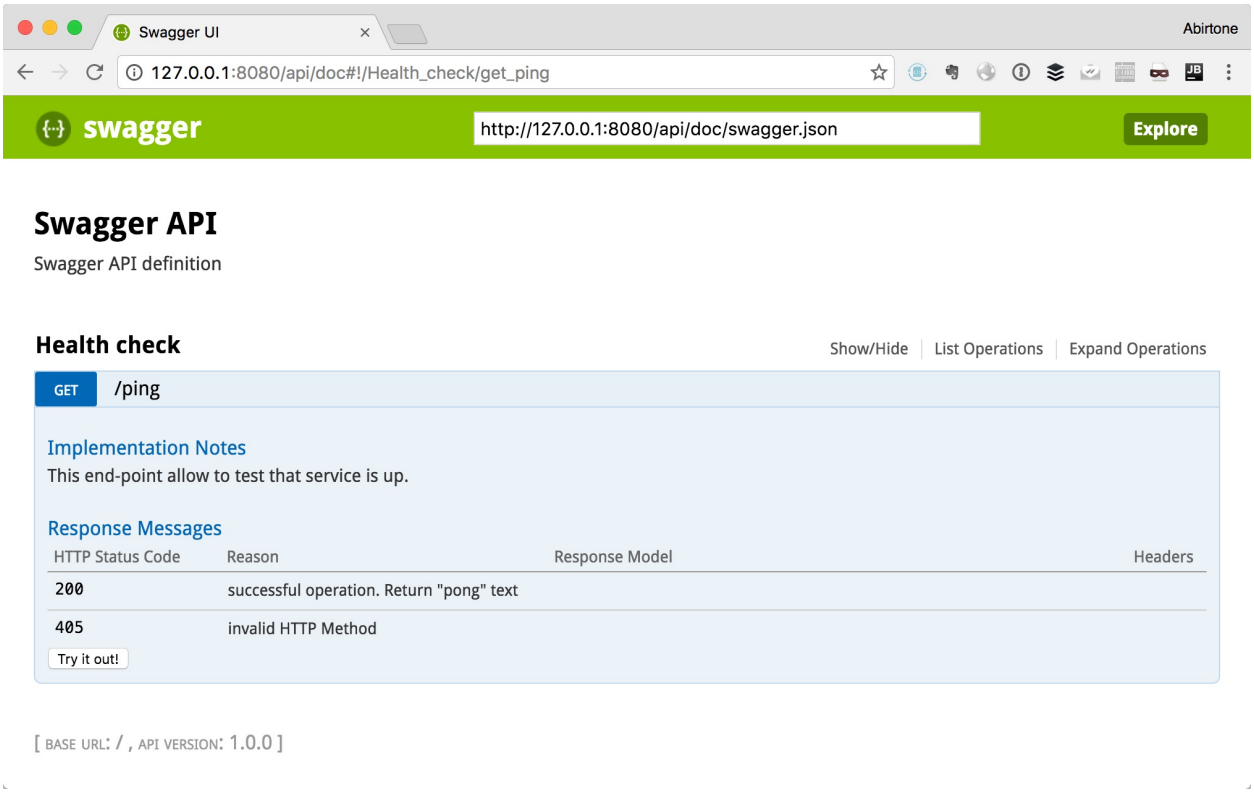
```
from aiohttp import web
from aiohttp_swagger import *
```

(continues on next page)

(continued from previous page)

```
async def ping(request):
    """
    ---
    description: This end-point allow to test that service is up.
    tags:
    - Health check
    produces:
    - text/plain
    responses:
        "200":
            description: successful operation. Return "pong" text
        "405":
            description: invalid HTTP Method
    """
    return web.Response(text="pong")


app = web.Application()
app.router.add_route('GET', "/ping", ping)

setup_swagger(app)

web.run_app(app, host="127.0.0.1")
```

It produces:



1.2.1 Where to access to API Doc

By default, API will be generated at URL: `yourdomain.com/api/doc`.

You can modify the URI adding the parameter `swagger_url` in `setup_swagger`.

You can specify UI version (Version 2 and 3 are supported) by adding the parameter `ui_version` in `ui_version`.

```
from aiohttp import web
from aiohttp_swagger import *

async def ping(request):
    """
    ---
    description: This end-point allow to test that service is up.
    tags:
    - Health check
    produces:
    - text/plain
    responses:
        "200":
            description: successful operation. Return "pong" text
        "405":
            description: invalid HTTP Method
    """
    return web.Response(text="pong")


app = web.Application()
app.router.add_route('GET', "/ping", ping)

setup_swagger(app, swagger_url="/api/v1/doc", ui_version=2)  # <-- NEW Doc URI

web.run_app(app, host="127.0.0.1")
```

1.3 Customizing Doc description and more

You can change this valued for Swagger doc:

1. API **Base URL**: Modify global prefix of your API.
2. **Description**: Long description of your API
3. API **Version**: Version of your API
4. **Title**: Title for your API
5. **Contact**: Contact info.
6. **Template Path**:Path to custom swagger template file.

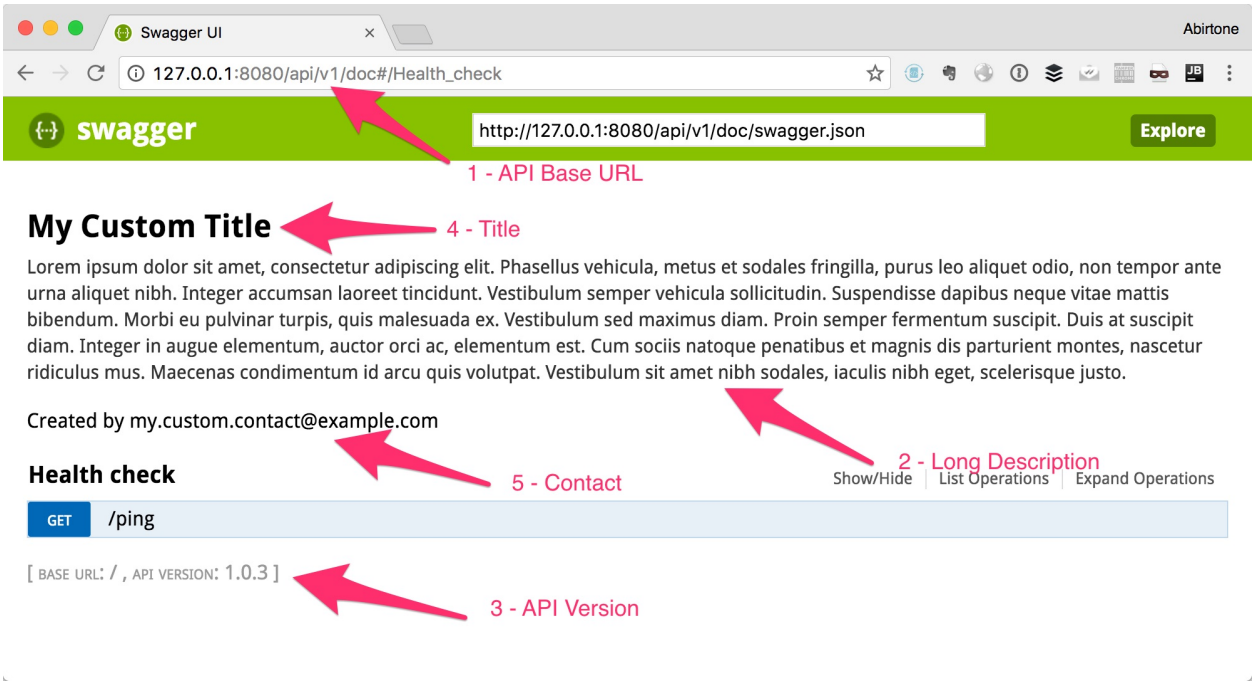
```
from aiohttp import web
from aiohttp_swagger import *

async def ping(request):
    """
    ---
    description: This end-point allow to test that service is up.
```

(continues on next page)

(continued from previous page)

```
    tags:
    - Health check
    produces:
    - text/plain
    responses:
        "200":
            description: successful operation. Return "pong" text
        "405":
            description: invalid HTTP Method
    """
    return web.Response(text="pong")

app = web.Application()

app.router.add_route('GET', "/ping", ping)

long_description = """
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus vehicula, metus et␣
↪sodales fringilla, purus leo aliquet odio, non tempor ante urna aliquet nibh.␣
↪Integer accumsan laoreet tincidunt. Vestibulum semper vehicula sollicitudin.␣
↪Suspendisse dapibus neque vitae mattis bibendum. Morbi eu pulvinar turpis, quis␣
↪malesuada ex. Vestibulum sed maximus diam. Proin semper fermentum suscipit. Duis at␣
↪suscipit diam. Integer in augue elementum, auctor orci ac, elementum est. Cum␣
↪sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.␣
↪Maecenas condimentum id arcu quis volutpat. Vestibulum sit amet nibh sodales,␣
↪iaculis nibh eget, scelerisque justo.

Nunc eget mauris lectus. Proin sit amet volutpat risus. Aliquam auctor nunc sit amet␣
↪feugiat tempus. Maecenas nec ex dolor. Nam fermentum, mauris ut suscipit varius,␣
↪odio purus luctus mauris, pretium interdum felis sem vel est. Proin a turpis vitae␣
↪nunc volutpat tristique ac in erat. Pellentesque consequat rhoncus libero, ac␣
↪sollicitudin odio tempus a. Sed vestibulum leo erat, ut auctor turpis mollis id. Ut␣
↪nec nunc ex. Maecenas eu turpis in nibh placerat ullamcorper ac nec dui. Integer ac␣
↪lacus neque. Donec dictum tellus lacus, a vulputate justo venenatis at. Morbi␣
↪malesuada tellus quis orci aliquet, at vulputate lacus imperdiet. Nulla eu diam␣
↪quis orci aliquam vulputate ac imperdiet elit. Quisque varius mollis dolor in␣
↪interdum.
"""

setup_swagger(app,
              description=long_description,
              title="My Custom Title",
              api_version="1.0.3",
              contact="my.custom.contact@example.com"
              template_path="my/custom/path/to/swagger.yaml")

web.run_app(app, host="127.0.0.1")
```

It produces:



Follows default swagger Jinja2 template which is used in the library:

```
swagger: "2.0"
info:
  description: |
    {{ description }}
  version: "{{ version }}"
  title: {{ title }}
  {% if contact %}
  contact:
    name: {{ contact }}
  {% endif %}
basePath: {{ base_path }}
schemes:
  - http
  - https
{% if definitions %}
definitions:
{{ definitions|nesteddict2yaml }}
{% endif %}
{% if security_definitions %}
securityDefinitions:
{{ security_definitions|nesteddict2yaml }}
{% endif %}
paths:
```

1.3.1 Adding Swagger from external file

Per End-Point level

We can add the Swagger doc from an external YAML file at end-point level. You only need to decorate the end-point function handler:

```python
from aiohttp import web
from aiohttp_swagger import *


@swagger_path("example_swagger_partial.yaml")  # <-- Load Swagger info from external file
async def example_2(request):
    """
    Example 3 handler description. This description is only for Sphinx.
    """
    return web.Response(text="Example")


async def example_3(request):
    """
    Description end-point
    """
    return web.Response(text="Example")

app = web.Application()

app.router.add_route('GET', "/example1", example_1)
app.router.add_route('GET', "/example2", example_2)

setup_swagger(app)

web.run_app(app, host="127.0.0.1")
```

External file must have this format:

```yaml
tags:
- user
summary: Create user
description: This can only be done by the logged in user.
operationId: examples.api.api.createUser
produces:
- application/json
parameters:
- in: body
  name: body
  description: Created user object
  required: false
  schema:
    type: object
    properties:
      id:
        type: integer
        format: int64
      username:
        type:
          - "string"
          - "null"
      firstName:
        type: string
      lastName:
        type: string
      email:
```

(continues on next page)

(continued from previous page)

```
        type: string
      password:
        type: string
      phone:
        type: string
      userStatus:
        type: integer
        format: int32
        description: User Status
responses:
  "201":
    description: successful operation
```

Note: Pay attention that file doesn't contain information about HTTP Method or End-Point name. This information will be added automatically

Global Swagger YAML

`aiohttp-swagger` also allow to build an external YAML Swagger file and load it before:

```
from aiohttp import web
from aiohttp_swagger import *

async def ping(request):
    """
    This is my usually Sphinx doc

    >>> import json
    >>> ping(None)

    :param request: Context injected by aiohttp framework
    :type request: RequestHandler
    """
    return web.Response(text="pong")

app = web.Application()

app.router.add_route('GET', "/ping", ping)

setup_swagger(app, swagger_from_file="example_swagger.yaml")  # <-- Loaded Swagger␣
↪from external YAML file

web.run_app(app, host="127.0.0.1")
```

Data Definitions

`aiohttp-swagger` allow to specify data models and to reuse it later when documenting API. Following example shows how to define nested object and reuse it when writing swagger doc.

```
async def users_with_data_def(request):
    """
    ---
```

(continues on next page)

(continued from previous page)

```
    description: This endpoint returns user which is defined though data definition␣
↪during initialization.
    tags:
    - Users
    produces:
    - application/json
    responses:
        "200":
            description: Successful operation, returns User object nested permisiion␣
↪list
            schema:
              $ref: '#/definitions/User'
    """
    users = fetch_users()
    return web.Response(json.dumps(users))

app = web.Application()

app.router.add_route('GET', "/users", users_with_data_def)

setup_swagger(app, definitions={
    "User": {
      "type": "object",
      "properties": {
        "username": {
          "type": "string",
          "description": "User's username name",
          "default": "John"
        },
        "permissions": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/Permission"
          }
        }
      }
    },
    "Permission": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string",
          "description": "Permission name"
        }
      }
    }
})

web.run_app(app, host="127.0.0.1")
```

Nested applications

`aiohttp-swagger` is compatible with aiohttp Nested applications feature. In this case *api_base_url* argument of *setup_swagger* function should be the same as *prefix* argument of *add_subapp* method:

```
from aiohttp import web
from aiohttp_swagger import *

async def ping(request):
    return web.Response(text="pong")

sub_app = web.Application()

sub_app.router.add_route('GET', "/ping", ping)

setup_swagger(sub_app,
              swagger_from_file="example_swagger.yaml",
              api_base_url='/sub_app_prefix')

app = web.Application()

app.add_subapp(prefix='/sub_app_prefix', subapp=sub_app)

web.run_app(app, host="127.0.0.1")
```

Swagger validation

`aiohttp-swagger` allows to perform online swagger validation. By default this feature is turned off *(swagger_validator_url=")*:

```
setup_swagger(app,
              api_base_url='/sub_app_prefix',
              swagger_validator_url='//online.swagger.io/validator'
              )
```

1.4 Frequently Asked Questions

1.4.1 Where start the Swagger documentation in my function doc?

`aiohttp-swagger` try to find the string `---`. When it find this string pattern, the next text until the end of function are considered Swagger doc.

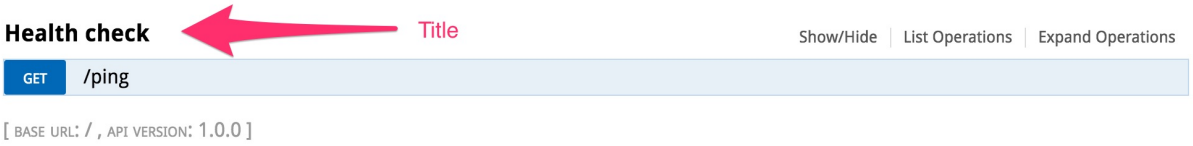
1.4.2 Can I Combine the Swagger documentation with my usually Sphinx doc?

Sure! Your Sphinx doc must be first of the `---` limiter.

```
async def ping(request):
    """
    This is my usually Sphinx doc

    >>> import json
    >>> ping(None)

    :param request: Context injected by aiohttp framework
    :type request: RequestHandler

    ---
    description: This end-point allow to test that service is up.
    tags:
    - Health check
    produces:
    - text/plain
    responses:
        "200":
            description: successful operation. Return "pong" text
        "405":
            description: invalid HTTP Method
    """
```

1.4.3 How can I group a list of End-Point?

End-Point will be grouped by their title. The end-point with the same title will be grouped automatically:



1.4.4 How can I change the Title of a group of End-Points?

Swagger has a tag that uses to build the titles. The tag name is `tags`. The format is:

```
tags:  # <-- TAG USED FOR THE TITLE
- Health check
description: This end-point allow to test that service is up.
produces:
- text/plain
responses:
    "200":
        description: successful operation. Return "pong" text
    "405":
        description: invalid HTTP Method
```

1.4.5 What happens if I use YAML file and function documentation for build Swagger doc at the same time?

If two method are provided, `aiohttp-swagger` will use the YAML over the function doc.

CHAPTER 2

Examples

If you need more examples, there are some in this folder: aiohttp_swagger/examples.

CHAPTER 3

What's new?

You can read entire list in CHANGELOG file.

CHAPTER 4

Contributing

You want to collaborate on this project? Nice! Your contribution is very welcome :)

You can send me a GitHub Pull Request with your proposal.