
Async Responses

Maciej Janiszewski

Apr 22, 2020

CONTENTS

1	Introduction	1
1.1	Installation	1
1.2	Usage	1
1.2.1	As an instance	1
1.2.2	As a context manager	1
1.2.3	With dict as handler	1
1.2.4	With exception as handler	2
1.2.5	With string as handler	2
1.2.6	With callable as handler	2
1.2.7	With a custom status code	3
1.2.8	With a custom response class	3
2	API	5
3	Changelog	7
3.1	1.0.0 (21.04.2020)	7
4	Indices and tables	9
	Python Module Index	11
	Index	13

INTRODUCTION

Async Responses is a library providing an easy way to mock aiohttp responses inspired by [aioresponses](#).

1.1 Installation

Library is available on PyPi, you can simply install it using pip:

```
$ pip install async-responses
```

1.2 Usage

1.2.1 As an instance

```
ar = AsyncResponses()  
ar.get(...)
```

1.2.2 As a context manager

```
with AsyncResponses() as ar:  
    ar.get(...)
```

1.2.3 With dict as handler

Passing dict as handler argument to async-responses would result in it being returned as a JSON payload.

```
import aiohttp  
from async_responses import AsyncResponses  
  
async def test_response():  
    ar = AsyncResponses()  
    payload = {'status': 'ok'}  
    ar.get('http://mock.url', '/v1/status', handler=payload)  
    async with aiohttp.ClientSession() as session:  
        response = await session.get('http://mock.url/v1/status')  
        assert await response.json() == payload
```

1.2.4 With exception as handler

Calling Async Responses with an exception as handler argument would result in it being raised.

```
import aiohttp
from async_responses import AsyncResponses
import pytest

async def test_response():
    ar = AsyncResponses()
    ar.get('http://mock.url', '/v1/status', handler=ZeroDivisionError)
    with pytest.raises(ZeroDivisionError):
        async with aiohttp.ClientSession() as session:
            await session.get('http://mock.url/v1/status')
```

1.2.5 With string as handler

```
import aiohttp
from async_responses import AsyncResponses

async def test_response():
    ar = AsyncResponses()
    payload = 'ok'
    ar.get('http://mock.url', '/v1/status', handler=payload)
    async with aiohttp.ClientSession() as session:
        response = await session.get('http://mock.url/v1/status')
        assert await response.text() == payload
```

1.2.6 With callable as handler

```
import aiohttp
from async_responses import AsyncResponses

async def test_response():
    def handler(data, **kwargs):
        return {'status': 'ok'}

    ar = AsyncResponses()
    ar.get('http://mock.url', '/v1/status', handler=handler)
    async with aiohttp.ClientSession() as session:
        response = await session.get('http://mock.url/v1/status')
        assert await response.json() == {'status': 'ok'}
```

1.2.7 With a custom status code

```
import aiohttp
from async_responses import AsyncResponses

async def test_response():
    payload = {'status': 'not good'}
    ar = AsyncResponses()
    ar.get('http://mock.url', '/v1/status', handler=payload, status=500)
    async with aiohttp.ClientSession() as session:
        response = await session.get('http://mock.url/v1/status')
        assert response.status == 500
        assert await response.json() == payload
```

1.2.8 With a custom response class

async-responses will make use of a response class passed as an argument to ClientSession, so you can use things like custom JSON serializer:

```
import aiohttp

async def test_response():
    class CustomResponse(aiohttp.ClientResponse):
        async def json(self, *args, **kwargs):
            return {'hello': 'world'}

    ar = AsyncResponses()
    ar.get('http://mock.url', '/v1/status', handler='')
    async with aiohttp.ClientSession(response_class=CustomResponse) as session:
        response = await session.get('http://mock.url/v1/status')
        assert await response.json() == {'hello': 'world'}
        assert isinstance(response, CustomResponse)
```


class `async_responses.Response` (*method, hostname, path, handler, status*)

class `async_responses.Call` (*method, url, args, kwargs*)

class `async_responses.AsyncResponses` (*, *mock_module=None, passthrough=[]*)
Async Responses context manager

Parameters

- **passthrough** – list of patterns of URLs which won't be mocked
- **mock_module** – mock module, defaults to `unittest.mock`

add (*method, hostname, path, handler, status=200*)

Mocks aiohttp response for the next request matching parameters.

Parameters

- **method** (`str`) – HTTP method, for example `get`
- **hostname** (`str`) – server hostname
- **path** (`str`) – path
- **handler** (`Union[Callable, dict, str, Exception]`) – response, can be either a dict, string, callable or an exception
- **status** (`int`) – status code, defaults to 200

property calls

List of calls

Return type `List[Call]`

get (*hostname, path, handler, status=200*)

Mocks GET request. Shorthand for `add('get', *args)`

Return type `None`

passthrough (*pattern*)

Adds passthrough. Requests to URLs which match the pattern won't be mocked.

Return type `None`

patch (*hostname, path, handler, status=200*)

Mocks PATCH request. Shorthand for `add('patch', *args)`

Return type `None`

post (*hostname, path, handler, status=200*)

Mocks POST request. Shorthand for `add('post', *args)`

Return type None

put (*hostname, path, handler, status=200*)

Mocks PUT request. Shorthand for `add('put', *args)`

Return type None

reset ()

Resets mock

Return type None

property responses

List of responses

Return type List[Response]

CHANGELOG

3.1 1.0.0 (21.04.2020)

- Initial release

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

`async_responses`, 5

A

`add()` (*async_responses.AsyncResponses* method), 5

`async_responses` (module), 5

`AsyncResponses` (class in *async_responses*), 5

C

`Call` (class in *async_responses*), 5

`calls()` (*async_responses.AsyncResponses* property),
5

G

`get()` (*async_responses.AsyncResponses* method), 5

P

`passthrough()` (*async_responses.AsyncResponses*
method), 5

`patch()` (*async_responses.AsyncResponses* method), 5

`post()` (*async_responses.AsyncResponses* method), 5

`put()` (*async_responses.AsyncResponses* method), 6

R

`reset()` (*async_responses.AsyncResponses* method), 6

`Response` (class in *async_responses*), 5

`responses()` (*async_responses.AsyncResponses* prop-
erty), 6