

---

# **aiohttp Documentation**

*Release 2.3.5-*

**aiohttp contributors**

**Nov 29, 2017**



---

## Contents

---

<b>1</b>	<b>Key Features</b>	<b>3</b>
<b>2</b>	<b>Library Installation</b>	<b>5</b>
<b>3</b>	<b>Getting Started</b>	<b>7</b>
<b>4</b>	<b>Tutorial</b>	<b>9</b>
<b>5</b>	<b>Source code</b>	<b>11</b>
<b>6</b>	<b>Dependencies</b>	<b>13</b>
<b>7</b>	<b>Communication channels</b>	<b>15</b>
<b>8</b>	<b>Contributing</b>	<b>17</b>
<b>9</b>	<b>Authors and License</b>	<b>19</b>
<b>10</b>	<b>Policy for Backward Incompatible Changes</b>	<b>21</b>
<b>11</b>	<b>Table Of Contents</b>	<b>23</b>



HTTP client/server for asyncio and Python.



# CHAPTER 1

---

## Key Features

---

- Supports both aiohttp-client and HTTP Server.
- Supports both Server WebSockets and Client WebSockets out-of-the-box.
- Web-server has aiohttp-web-middlewares, aiohttp-web-signals and pluggable routing.





## CHAPTER 2

---

### Library Installation

---

```
$ pip install aiohttp
```

You may want to install *optional* cchardet library as faster replacement for chardet:

```
$ pip install cchardet
```

For speeding up DNS resolving by client API you may install aiodns as well. This option is highly recommended:

```
$ pip install aiodns
```



## CHAPTER 3

---

### Getting Started

---

Client example:

```
import aiohttp
import asyncio
import asyncio_timeout

async def fetch(session, url):
    with asyncio_timeout.timeout(10):
        async with session.get(url) as response:
            return await response.text()

async def main():
    async with aiohttp.ClientSession() as session:
        html = await fetch(session, 'http://python.org')
        print(html)

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

Server example:

```
from aiohttp import web

async def handle(request):
    name = request.match_info.get('name', "Anonymous")
    text = "Hello, " + name
    return web.Response(text=text)

app = web.Application()
app.router.add_get('/', handle)
app.router.add_get('/{name}', handle)

web.run_app(app)
```

**Note:** Throughout this documentation, examples utilize the *async/await* syntax introduced by [PEP 492](#) that is only valid for Python 3.5+.

If you are using Python 3.4, please replace `await` with `yield from` and `async def` with a `@coroutine` decorator. For example, this:

```
async def coro(...):
    ret = await f()
```

should be replaced by:

```
@asyncio.coroutine
def coro(...):
    ret = yield from f()
```

## CHAPTER 4

---

### Tutorial

---

Polls tutorial



## CHAPTER 5

---

Source code

---

The project is hosted on [GitHub](#)

Please feel free to file an issue on the [bug tracker](#) if you have found a bug or have some suggestion in order to improve the library.

The library uses [Travis](#) for Continuous Integration.





- Python 3.4.2+
- *chardet*
- *multidict*
- *async\_timeout*
- *yaml*
- *Optional* cchardet as faster replacement for chardet.

Install it explicitly via:

```
$ pip install cchardet
```

- *Optional* aiodns for fast DNS resolving. The library is highly recommended.

```
$ pip install aiodns
```



## CHAPTER 7

---

### Communication channels

---

*aio-libs* google group: <https://groups.google.com/forum/#!forum/aio-libs>

Feel free to post your questions and ideas here.

*gitter chat* <https://gitter.im/aio-libs/Lobby>

We support [Stack Overflow](#). Please add *aiohttp* tag to your question there.



## CHAPTER 8

---

### Contributing

---

Please read the instructions for contributors before making a Pull Request.



## CHAPTER 9

---

### Authors and License

---

The `aihttp` package is written mostly by Nikolay Kim and Andrew Svetlov.

It's *Apache 2* licensed and freely available.

Feel free to improve this package and send a pull request to [GitHub](#).





---

## Policy for Backward Incompatible Changes

---

*aiohttp* keeps backward compatibility.

After deprecating some *Public API* (method, class, function argument, etc.) the library guaranties the usage of *deprecated API* is still allowed at least for a year and half after publishing new release with deprecation.

All deprecations are reflected in documentation and raises `DeprecationWarning`.

Sometimes we are forced to break the own rule for sake of very strong reason. Most likely the reason is a critical bug which cannot be solved without major API change, but we are working hard for keeping these changes as rare as possible.



# CHAPTER 11

---

## Table Of Contents

---

To see the full table of contents open the link.



P

Python Enhancement Proposals

PEP 492, 8