
asyncws Documentation

Release 0.1

Dave P

Jun 02, 2017

Contents

1	Indices and tables	3
	Python Module Index	5

asyncws is a library for developing websocket applications in Python 3. It implements [RFC 6455](#), passes the [Auto-bahn Testsuite](#) and supports SSL/TSL out of the box.

Based on [PEP 3156](#) and coroutines it makes it easy to write highly concurrent websocket based applications.

Echo server example:

```
import asyncio
import asyncws

@asyncio.coroutine
def echo(websocket):
    while True:
        frame = yield from websocket.recv()
        if frame is None:
            break
        yield from websocket.send(frame)

server = asyncws.start_server(echo, '127.0.0.1', 8000)
asyncio.get_event_loop().run_until_complete(server)
asyncio.get_event_loop().run_forever()
```

Corresponding echo client example:

```
import asyncio
import asyncws

@asyncio.coroutine
def echo():
    websocket = yield from asyncws.connect('ws://localhost:8000')
    while True:
        yield from websocket.send('hello')
        echo = yield from websocket.recv()
        if echo is None:
            break
        print(echo)

asyncio.get_event_loop().run_until_complete(echo())
asyncio.get_event_loop().close()
```

API Documentation

class `asyncws.Websocket` (*reader, writer*)

Class that wraps the websocket protocol.

Parameters

- **writer** – Access to `get_extra_info()`. See [StreamWriter](#).
- **request** – HTTP request that arrives at the server during handshaking. See [BaseHTTPRequestHandler](#). Set to `None` if it's a client websocket.
- **response** – HTTP response that arrives at the client after handshaking is complete. See [HTTPResponse](#). Set to `None` if it's a server websocket.

close (*status=1000, reason=''*)

Start the close handshake by sending a close frame to the websocket endpoint. Once the endpoint responds with a corresponding close the underlying transport is closed.

To force close the websocket without going through the close handshake call `self.writer.close()` which will immediately tear down the underlying transport.

Parameters

- **status** – See [Status Codes](#).
- **reason** – Why the websocket is being closed.

Raises Exception – When there is an error sending data to the endpoint.

recv()

Receive websocket frame from endpoint.

This coroutine will block until a complete frame is ready.

Returns Websocket text or data frame on success. Returns `None` if the connection is closed or there is an error.

send(data, flush=False)

Send a data frame to websocket endpoint.

Parameters

- **data** – If data is of type `str` then the data is sent as a text frame. If data is of type `byte` then the data is sent as a binary frame.
- **flush** – When set to `True` then the send buffer is flushed immediately.

Raises Exception – When there is an error sending data to the endpoint only if flush is set to `True`.

asyncws.connect(wsurl, **kws)

Connect to a websocket server. Connect will automatically carry out a websocket handshake.

Parameters

- **wsurl** – Websocket uri. See [RFC6455 URIs](#).
- **kws** – See [open_connection](#).

Returns `Websocket` object on success.

Raises Exception – When there is an error during connection or handshake.

asyncws.start_server(func, host=None, port=None, **kws)

Start a websocket server, with a callback for each client connected.

Parameters

- **func** – Called with a `Websocket` parameter when a client connects and handshake is successful.
- **kws** – See [start_server](#)

Returns

The return value is the same as [start_server](#)

CHAPTER 1

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`asyncws`, [1](#)

A

[asyncws \(module\)](#), 1

C

[close\(\) \(asyncws.Websocket method\)](#), 1

[connect\(\) \(in module asyncws\)](#), 2

R

[recv\(\) \(asyncws.Websocket method\)](#), 2

S

[send\(\) \(asyncws.Websocket method\)](#), 2

[start_server\(\) \(in module asyncws\)](#), 2

W

[Websocket \(class in asyncws\)](#), 1