Implementing Async HTTP with AioHTTP

Dan Tofan
PhD

@dan_tofan | https://programmingwithdan.com

AioHTTP Use Cases

Web applications

REST APIs

Real-time applications

Web scrapers

Overview of AioHTTP

Async by design

Powerful features

High performance

Key Features

Integrated HTTP server and client

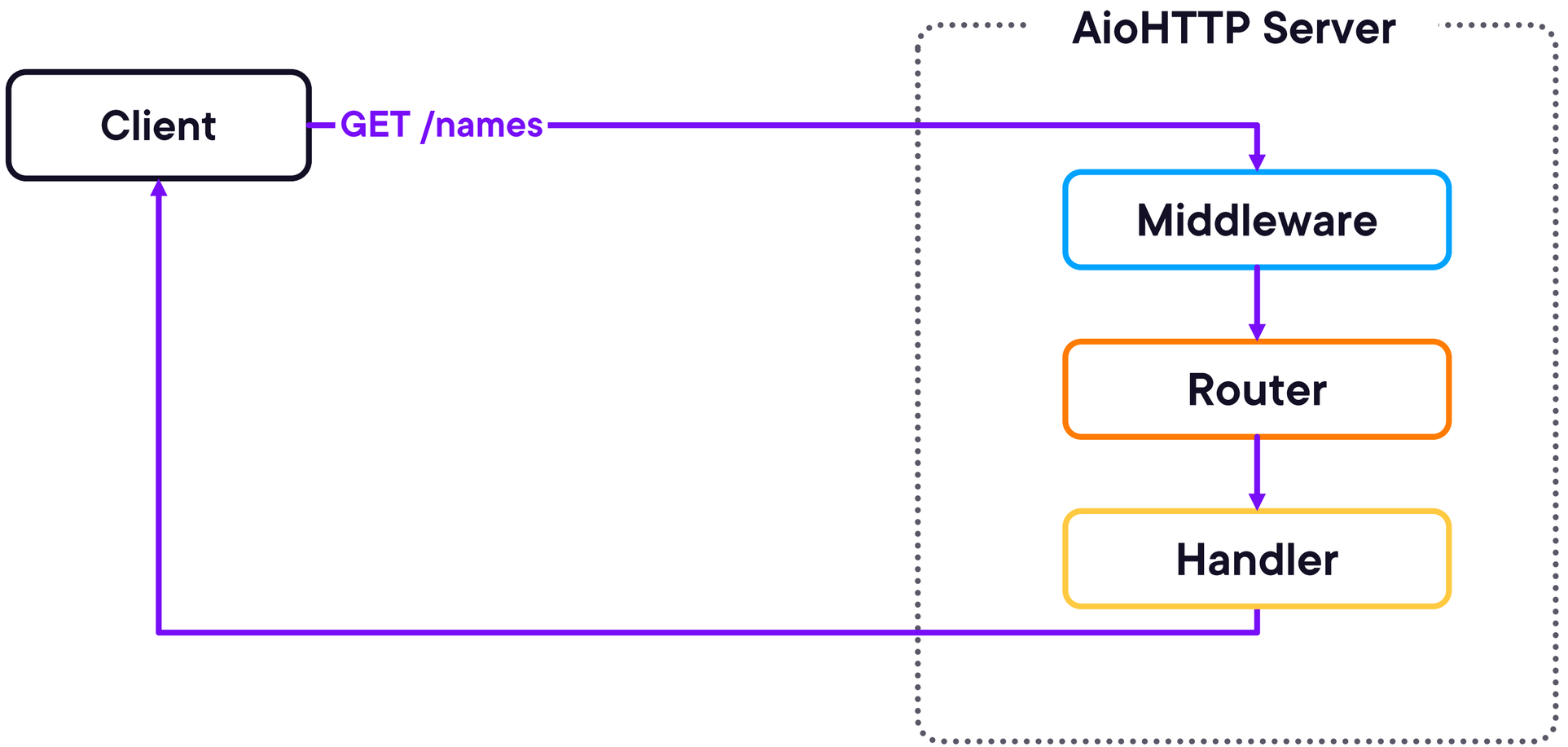
WebSocket support

AioHTTP Components

HTTP server

HTTP client

AioHTTP Request Flow



```
# pip install aiohttp

from aiohttp import web


async def get_names(request):
  return web.Response(text='hi all')


app = web.Application()

app.router.add_get('/names', get_names)

web.run_app(app)
```

- Install AioHTTP
- Import the web module of AioHTTP
- Coroutine with request handler
- HTTP response is 'hi all'
- Main container for server
- Register GET route and link to handler
- Start server

Demo: Building an API with AioHTTP

Read all names:	GET	/names
Read name by id:	GET	/names/{id}
Add new name:	POST	/names

Simple HTTP Request Using the Requests Library

```
# pip install requests

import requests



response = requests.get(URL)

data = response.json()

print(data)
```

```
import aiohttp
import asyncio


URL = "http://localhost"



async def fetch_url():
 async with aiohttp.ClientSession() as s:
    async with s.get(URL) as r:
      data = await r.json()
      print(data)



asyncio.run(fetch_url())
```

- Import prerequisites
- Assuming local URL
- Define new coroutine
- Create new client session
- Make a GET request
- Parse JSON response
- Print result
- Start the event loop

Demo: Fetching Data Concurrently

HTTP Clients

AioHTTP client	vs.	Requests client
Asynchronous (non-blocking)		Synchronous (blocking)
Requires async knowledge		Very intuitive
High performance		Low performance
API clients of fast, scalable apps		Scripts, basic apps