

Aiohttp

Сеансовый асинхронный http(s) клиент с автоматической поддержкой cookies. Пул подключений использует один сеанс.

Установка:

```
pip install aiohttp
```

Использование:

```
import asyncio
import aiohttp
from aiohttp import ClientSession
from util import async_timed

async def fetch_status(session: ClientSession, url: str) -> int:
    async with session.get(url) as result:
        return result.status

async def main():
    async with aiohttp.ClientSession() as session:
        url = 'https://www.example.com'
        status = await fetch_status(session, url)
        print(f'Состояние для {url} было равно {status}')
    asyncio.run(main())
```

По умолчанию в сеансе не более 100 подключений. Для увеличения можно создать экземпляр класса `TCPConnector`, входящего в состав `aiohttp`, указав максимальное число подключений, и передать его конструктору `ClientSession`. Подробнее в документации `aiohttp`.

Тайм-аут:

По умолчанию тайм-аут запроса 5 минут. Можно устанавливать на уровне сеанса или запроса.

```
import asyncio
import aiohttp
from aiohttp import ClientSession
```

```

async def fetch_status(session: ClientSession, url: str) -> int:
    ten_millis = aiohttp.ClientTimeout(total=.01)
    async with session.get(url, timeout=ten_millis) as result:
        return result.status

async def main():
    session_timeout = aiohttp.ClientTimeout(total=1, connect=.1)
    async with aiohttp.ClientSession(timeout=session_timeout) as session:
        await fetch_status(session, 'https://example.com')
    asyncio.run(main())

```

В этом случае полный тайм-аут 1 секунда, для установки соединения - 100мс. В функции `fetch_status` переопределяется в 10мс.

Множественные запросы

```

import asyncio
import aiohttp
from aiohttp import ClientSession
from chapter_04 import fetch_status

async def main():
    async with aiohttp.ClientSession() as session:
        urls = ['https://example.com' for _ in range(1000)]
        requests = [fetch_status(session, url) for url in urls]
        status_codes = await asyncio.gather(*requests)
        print(status_codes)
    asyncio.run(main())

```

Обработка ошибок

```

async def main():
    async with aiohttp.ClientSession() as session:
        urls = ['https://example.com', 'python://example.com']
        tasks = [fetch_status_code(session, url) for url in urls]
        results = await asyncio.gather(*tasks, return_exceptions=True)
        exceptions = [res for res in results if isinstance(res, Exception)]
        successful_results = [res for res in results if not isinstance(res, Exception)]
        print(f'Все результаты: {results}')
        print(f'Завершились успешно: {successful_results}')
        print(f'Завершились с исключением: {exceptions}')

```

Revision #3

Created 6 May 2025 01:25:16 by Admin

Updated 6 May 2025 02:34:56 by Admin