Middleware

Промежуточный слой бизнес-логики, который может выполняться и до и после того, как запрос дойдет до View.

С помощью middleware можно вынести общую логику, которая должна работать сразу в нескольких View. Самым частым примером является авторизационная middleware, которая проверяет предоставленные клиентом данные авторизации и снабжает объект запроса авторизованным или анонимным пользователем.

Упрощенно это выглядит так: сначала запрос проходит все middleware по порядку их добавления, потом обрабатывается во View и возвращается обратно через все middleware в обратном порядке.

Пример. middleware для обработки ошибок после обработки запроса
Сначала выполняются все стоящие middleware за этой, а потом view. Если возникла ошибка, она отлавливается на этом этапе и преобразуется в унифицированный вид:

```
@web.middleware
async def error_middleware(request, handler):
    try:
        return await handler(request)
    except web.HTTPException as ex:
        return json_response(status=ex.status, text_status=ex.text, data={})
    except Exception as e:
        return json_response(status=500, text_status=str(e), data={})
```

Middleware всегда принимает на вход объект запроса и следующий за этой middleware обработчик запроса: следующую middleware, или, если эта последняя в цепочке, сам View.

Подробнее о middleware на примере aiohttp вы можете прочитать здесь.

Добавлять middleware в приложение можно двумя способами.

1. При создании:

```
from aiohttp import web

web.Application(middlewares=[error_middleware])
```

2. Динамически (этот способ мне нравится больше):

```
from aiohttp import web

app = web.Application()
app.middlewares.append(error_middleware)
```

Примеры

1. работа с middlewares