Web-Socket

Мы уже рассматривали web-socket, выступая в роли клиента. Пришло время стать хостом.

К облегчению всех бэкенд-разработчиков, aiohttp уже имеет поддержку web-socket из коробки.

Как работать с веб-сокетами из aiohttp server

1. Нужно сделать View и Route — обычные View и Route, ничего особенного
2. Во View нужно создать web.WebSocketResponse. Это объект ответа, почти такой же, как web.Response, только с предустановленными статусом ответа (101) и специальными заголовками) (например Connection-Upgrade) и дополнительными методами
3. У этого экземляра *web.WebSocketResponse* нужно выполнить метод .prepare(). Как мы помним, метод .prepare() записывает данные ответа в сокет, то есть отправляет ответ клиенту. В момент выполнения этого шага ответ на апгрейд соединения уже послан пользователю, а мы все еще выполняем код View. С этого момента экземпляр *web.WebSocketResponse выполняет роль соединения*
4. Соединение прервется в момент, когда мы закроем его самостоятельно или закончим выполнение этого View, поэтому есть смысл бесконечно ожидать нового сообщения из этого соединения с помощью асинхронного цикла чтения
5. Когда нужно отправить какое-то сообщение, мы можем вызвать у этого соединения метод .push()
6. Когда нужно закрыть соединения, мы можем вызвать у него метод .close()

Пример из официальной документации aiohttp отлично демонстрирует этот алгоритм для одного соединения:

```
async def websocket_handler(request):

    ws = web.WebSocketResponse()
    await ws.prepare(request)

    async for msg in ws:
        if msg.type == aiohttp.WSMsgType.TEXT:
            if msg.data == 'close':
                await ws.close()
            else:
                await ws.send_str(msg.data + '/answer')
        elif msg.type == aiohttp.WSMsgType.ERROR:
            print('ws connection closed with exception %s' %
                  ws.exception())

    print('websocket connection closed')

    return ws
```

Если же у нас несколько открытых одновременно соединений, можно просто держать их где-то в памяти приложения. Например, держать их в виде словаря {user_id: ws_connection} и посылать сообщения по мере необходимости в нужные места.

Структура web-socket сообщения

WS-сообщение в aiohttp содержит два самых важных поля: type и data.

type — тип сообщения, бывает нескольких видов:

- *CONTINUATION* — технический тип, сигнализирующий о продолжении предыдущей информации в этом фрагменте
- *TEXT* — сообщение с текстовой полезной нагрузкой
- *BINARY* — сообщение с двоичной полезной нагрузкой
- *PING* — технический тип для проверки соединения
- *PONG* — технический тип для ответа на проверку соединения
- *CLOSE* — технический тип, говорящий о закрытии соединения

Также существуют типы, используемые внутри aiohttp: CLOSING, CLOSED, ERROR, но вряд ли вам предоставится возможность их использовать.

data — полезная нагрузка сообщения, бывает бинарной и текстовой.

- Бинарная нагрузка — некоторый набор байт, которые можно как-то использовать. Передача данных в бинарном виде обычно гораздо быстрее (из-за их меньшего размера), чем передача текстовых. Не стоит думать, что бинарные данные передаются как-то по-другому, чем текстовые — оба типа преобразуются в нолики и единички. Но при передаче бинарных данных на стороне приемника эти данные не декодируются в символы, а при передаче текстовых декодируются. Для работы с бинарными данными создано множество форматов, таких же, как xml и json, но для бит и байт. Например, гугловский ProtoBuf
- Текстовая нагрузка — это текст. А в текст можно положить json-данные, что позволяет работать с веб-сокетами почти также, как с любым View

Обычно для четкого разграничения, какие поля несет сообщение и с какой целью, в нагрузку добавляют собственный тип сообщения. Например, если рассматривать загрузку файлов на сервер с сообщениями о прогрессе загрузки, можно выделить три типа сообщений:

- INITIAL — начальное сообщение при установке соединения. Нужно, чтобы подтвердить наличие соединения. Не содержит никакой дополнительной информации
- UPLOAD_PROGRESS — сообщение, отправляемое при изменении статуса загрузки. Несет поле value, которое указывает текущий прогресс загрузки
- UPLOAD_FINISH — сообщение, отправляемое при завершении загрузки. Не несет каких-либо дополнительных полей, но позволяет клиенту правильно отобразить завершение загрузки

Как использовать нагрузку проще всего

Использовать текстовые сообщения, в каждое вкладывать json-нагрузку в виде словаря, обязательно содержащего ключ *kind*. По этому ключу можно понять, какой смысл несет это сообщение, и в коде задать соответствия типа и полей в нем.