Websockets

Python 3.x 3.5–3.6

Examples

Simple Echo with aiohttp

aiohttp provides asynchronous websockets.

Python 3.x ≥3.5

```
import asyncio
from aiohttp import ClientSession

with ClientSession() as session:
    async def hello_world():

        websocket = await session.ws_connect("wss://echo.websocket.org")

        websocket.send_str("Hello, world!")

        print("Received:", (await websocket.receive()).data)

        await websocket.close()

    loop = asyncio.get_event_loop()
    loop.run_until_complete(hello_world())
```

Wrapper Class with aiohttp

aiohttp.ClientSession may be used as a parent for a custom WebSocket class.

Python 3.x ≥3.5

```
import asyncio
from aiohttp import ClientSession

class EchoWebSocket(ClientSession):

    URL = "wss://echo.websocket.org"

    def __init__(self):
        super().__init__()
        self.websocket = None

    async def connect(self):
        """Connect to the WebSocket."""
        self.websocket = await self.ws_connect(self.URL)

    async def send(self, message):
        """Send a message to the WebSocket."""
        assert self.websocket is not None, "You must connect first!
        self.websocket.send_str(message)
        print("Sent:", message)

    async def receive(self):
        """Receive one message from the WebSocket."""
        assert self.websocket is not None, "You must connect first!
        return (await self.websocket.receive()).data

    async def read(self):
        """Read messages from the WebSocket."""
        assert self.websocket is not None, "You must connect first!

        while self.websocket.receive():
            message = await self.receive()
            print("Received:", message)
            if message == "Echo 9!":
                break
```

Using Autobahn as a Websocket Factory

The Autobahn package can be used for Python web socket server factories.

Python Autobahn package documentation

To install, typically one would simply use the terminal command

(For Linux):

```
sudo pip install autobahn
```

(For Windows):

```
python -m pip install autobahn
```

Then, a simple echo server can be created in a Python script:

```
from autobahn.asyncio.websocket import WebSocketServerProtocol
class MyServerProtocol(WebSocketServerProtocol):
    '''When creating server protocol, the
    user defined class inheriting the
    WebSocketServerProtocol needs to override
    the onMessage, onConnect, et-c events for
    user specified functionality, these events
    define your server's protocol, in essence'''
    def onMessage(self,payload,isBinary):
        '''The onMessage routine is called
        when the server receives a message.
        It has the required arguments payload
        and the bool isBinary. The payload is the
        actual contents of the "message" and isBinary
        is simply a flag to let the user know that
        the payload contains binary data. I typically
        elsewise assume that the payload is a string.
        In this example, the payload is returned to sender verbatim.'''
        self.sendMessage(payload,isBinary)
if__name__=='__main__':
    try:
        importasyncio
    except ImportError:
        '''Trollius = 0.3 was renamed'''
        import trollius as asyncio
    from autobahn.asyncio.websocketimportWebSocketServerFactory
    factory=WebSocketServerFactory()
    '''Initialize the websocket factory, and set the protocol to the
    above defined protocol(the class that inherits from
    autobahn.asyncio.websocket.WebSocketServerProtocol)'''
    factory.protocol=MyServerProtocol
    '''This above line can be thought of as "binding" the methods
    onConnect, onMessage, et-c that were described in the MyServerProtocol class
    to the server, setting the servers functionality, ie, protocol'''
    loop=asyncio.get_event_loop()
    coro=loop.create_server(factory,'127.0.0.1',9000)
    server=loop.run_until_complete(coro)
```

In this example, a server is being created on the localhost (127.0.0.1) on port 9000. This is the listening IP and port. This is important information, as using this, you could identify your computer's LAN address and port forward from your modem, though whatever routers you have to the computer. Then, using google to investigate your WAN IP, you could design your website to send WebSocket messages to your WAN IP, on port 9000 (in this example).

It is important that you port forward from your modem back, meaning that if you have routers daisy chained to the modem, enter into the modem's configuration settings, port forward from the modem to the connected router, and so forth until the final router your computer is connected to is having the information being received on modem port 9000 (in this example) forwarded to it.

Syntax

Parameters

Remarks