Asyncio Streams

December 16, 2020

1 Streams

Los streams son async/await primitivos de alto nivel para trabajar con conexiones de red. Los streams permiten enviar y recibir datos sin utilizar callbacks o protocolos y transportes de bajo nivel.

Este es un ejemplo de un cliente:

[ ]:
```
import asyncio

async def tcp_echo_client(message):
    # open_connection establece una conexión de red, y retorna un par
    # (reader, writer)  StreamReader y StreamWriter
    reader, writer = await asyncio.open_connection(
        '127.0.0.1', 8888)

    print(f'Send: {message!r}')
    writer.write(message.encode())
    await writer.drain()

    data = await reader.read(100)
    print(f'Received: {data.decode()!r}')

    print('Close the connection')
    writer.close()
    await writer.wait_closed()

# asyncio.run(tcp_echo_client('Hello World!'))
```

[1]:
```
import asyncio

async def handle_echo(reader, writer):
    data = await reader.read(100)
    message = data.decode()
    addr = writer.get_extra_info('peername')

    print(f"Received {message!r} from {addr!r}")
    print(f"Send: {message!r}")
    writer.write(data)
    await writer.drain()

    print("Close the connection")
    writer.close()

async def main():
    server = await asyncio.start_server(
        handle_echo, '127.0.0.1', 8888)

    addr = server.sockets[0].getsockname()
    print(f'Serving on {addr}')

    async with server:
        await server.serve_forever()

# asyncio.run(main())
```

[3]:
```
import asyncio
import socket

async def wait_for_data():
    # Get a reference to the current event loop because
    # we want to access low-level APIs.
    loop = asyncio.get_running_loop()

    # Create a pair of connected sockets.
    rsock, wsock = socket.socketpair()

    # Register the open socket to wait for data.
    reader, writer = await asyncio.open_connection(sock=rsock)

    # Simulate the reception of data from the network
    loop.call_soon(wsock.send, 'abc'.encode())

    # Wait for data
    data = await reader.read(100)

    # Got data, we are done: close the socket
    print("Received:", data.decode())
    writer.close()
```

```python
    # Close the second socket
    wsock.close()

# asyncio.run(wait_for_data())
```