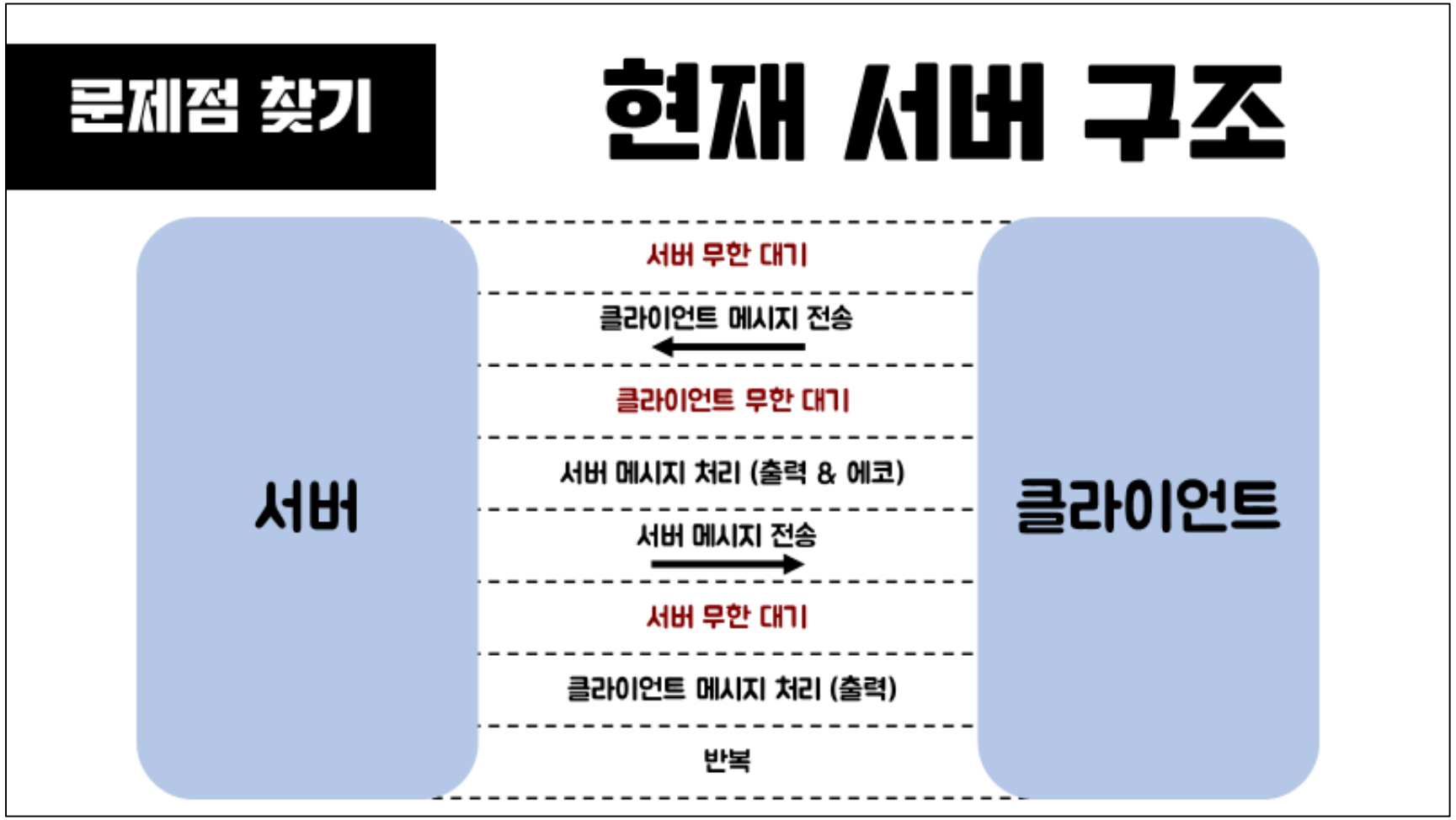
야매로 서버 개발자 되는 법 5강

비동기 소켓 프로그래밍

문제 해결 방법

뭘 했더라


문제점 찾기
현재 서버 구조
서버
클라이언트
서버 무한 대기
클라이언트 메시지 전송
클라이언트 무한 대기
서버 메시지 처리 (출력 & 에코)
서버 메시지 전송
서버 무한 대기
클라이언트 메시지 처리 (출력)
반복

해결방법

멈추는 게 문제였으니...

해결방법

안 멈추게 하려면....

해결방법

비동기 함수를 쓰자!

해결방법

Socket.Accept()

Socket.Send(byte[])

Socket.Receive(byte[])

해결방법

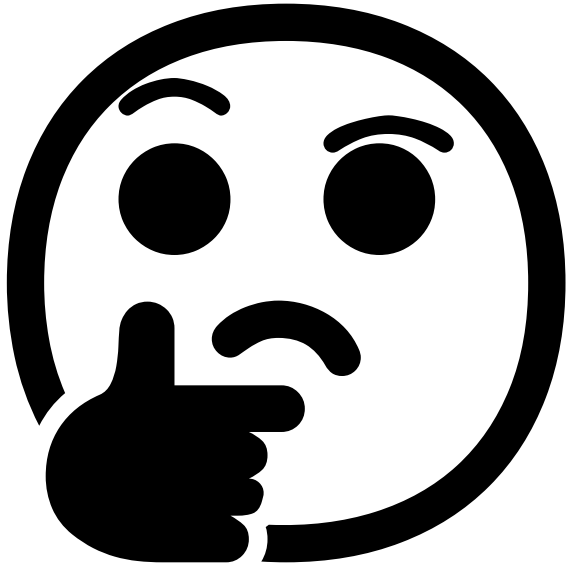
Socket.AcceptAsync(SocketAsyncEventArgs)

Socket.SendAsync(SocketAsyncEventArgs)

Socket.ReceiveAsync(SocketAsyncEventArgs)

SocketAsyncEventArgs

'SocketAsyncEventArgs'



SocketAsyncEventArgs

'SocketAsyncEventArgs'
비동기 통신에서 데이터 처리를 위한 클래스

SocketAsyncEventArgs

'SocketAsyncEventArgs'

그냥 데이터 셔틀

리스너와 커넥터

비동기 함수..?

잠깐!!!!

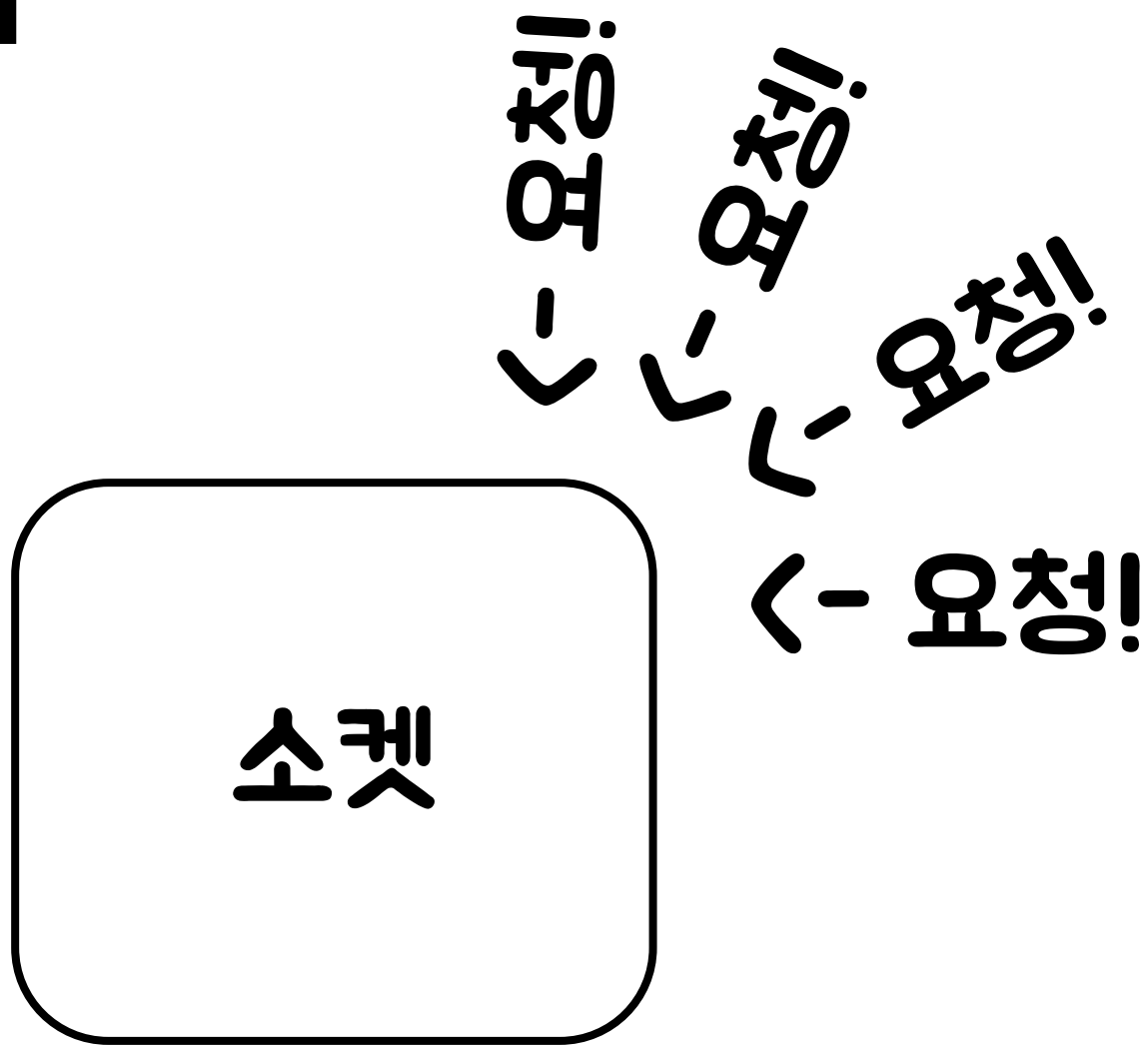
비동기 함수..?

비동기 함수는 위험해!

비동기 함수..?

어쩌라고;

응애 나 소켓



응애 나 소켓

요청

요청

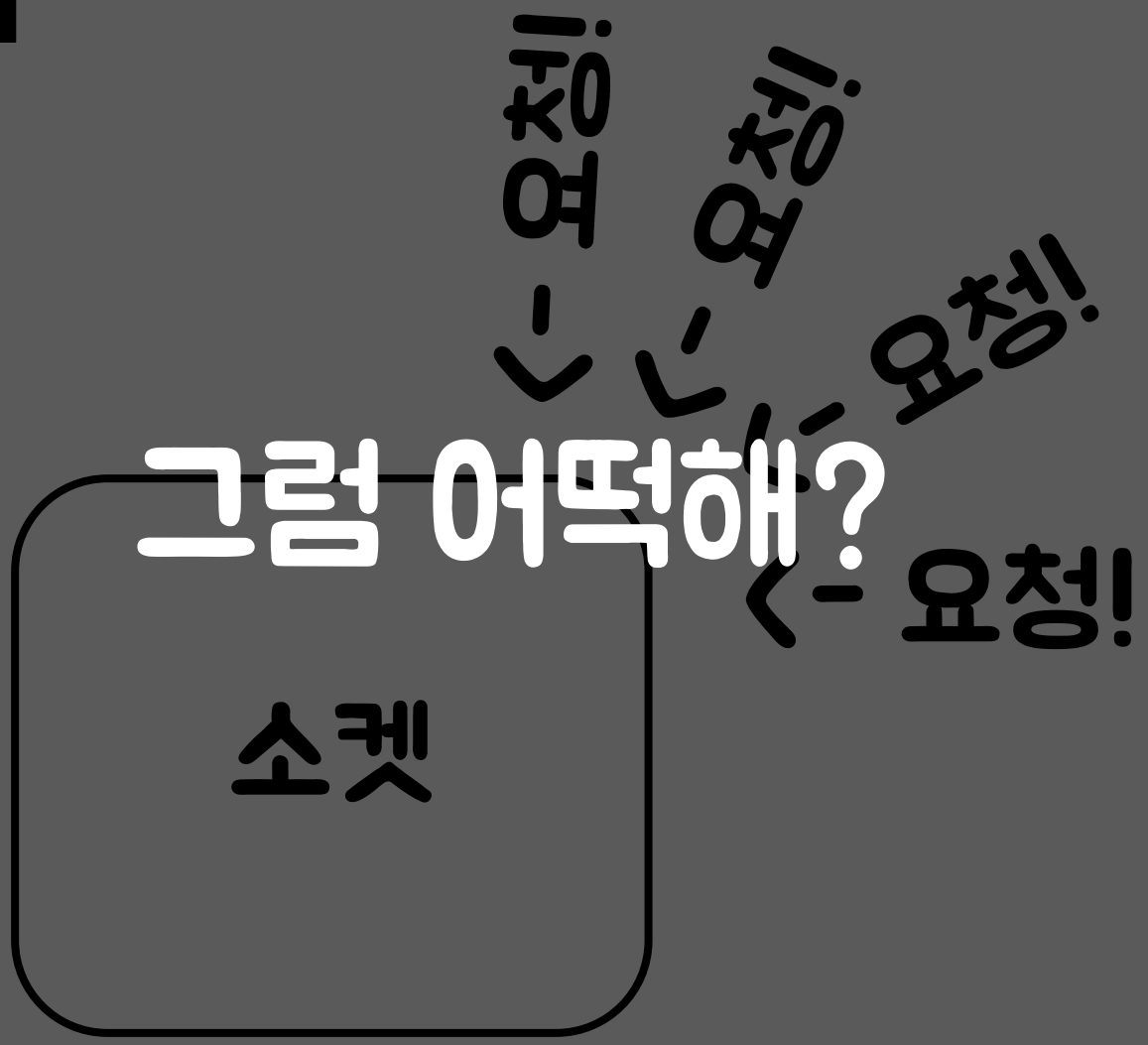
요청!

제발 그만해!!

요청!

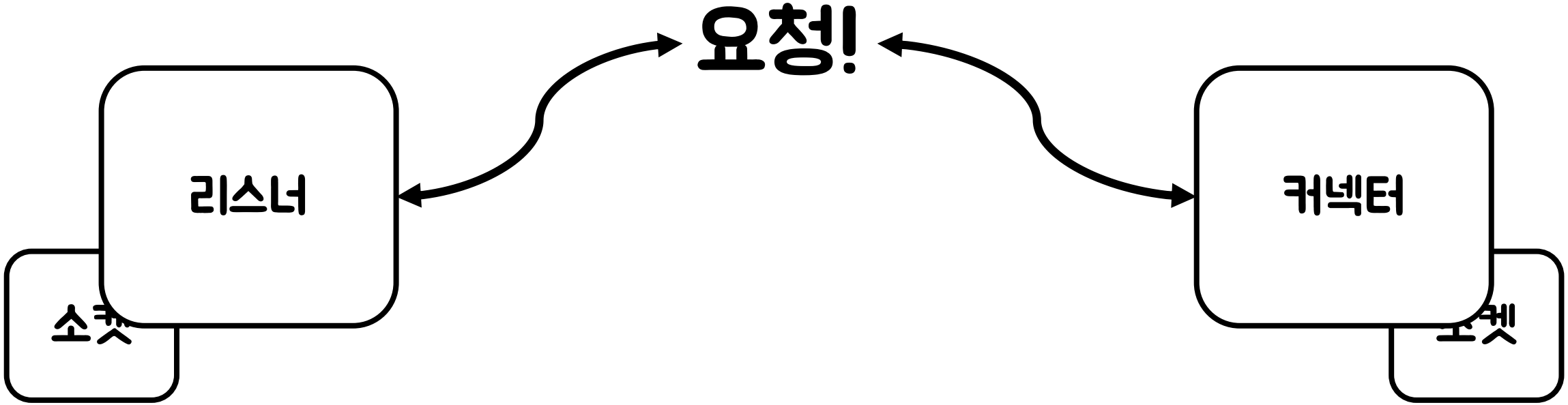
소켓

응애 나 소켓



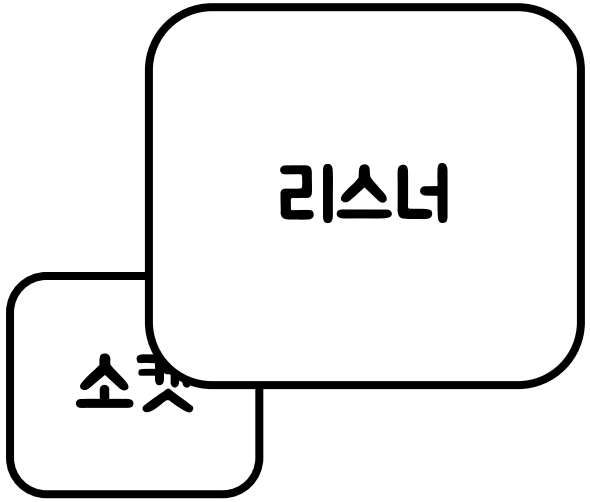
리스너 & 커넥터

내가 해줄게!

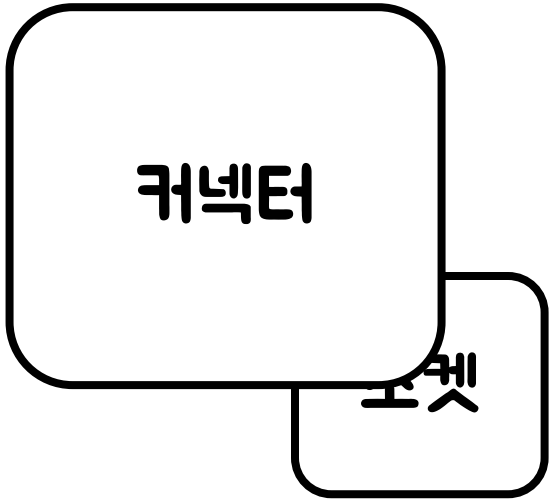


리스너 & 커넥터

서버 대행자

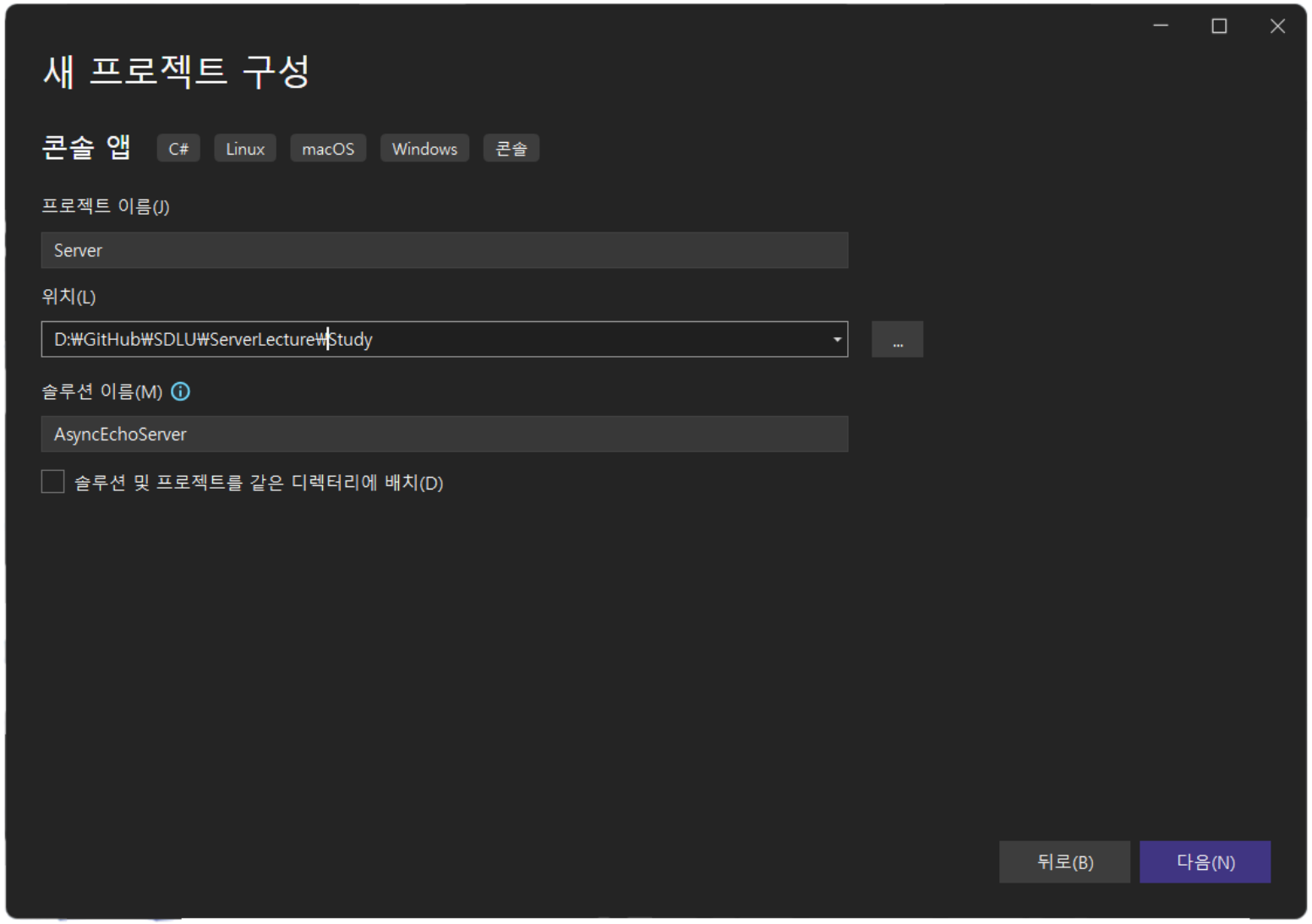


클라이언트 대행자



프로젝트 생성

프로젝트 생성


새 프로젝트 구성
콘솔 앱
C#
Linux
macOS
Windows
콘솔
프로젝트 이름(J)
Server
위치(L)
D:₩GitHub₩SDLU₩ServerLecture₩Study
...
솔루션 이름(M)
AsyncEchoServer
솔루션 및 프로젝트를 같은 디렉터리에 배치(D)
뒤로(B)
다음(N)

서론

Listener

```
1   public class Listener
2   {
3       private Socket listenSocket = null;
4       private List<Socket> clientSockets = new List<Socket>();
5
6       public Listener(IPEndPoint endPoint, int backlog)
7       {
8           listenSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
9           listenSocket.Bind(endPoint);
10          listenSocket.Listen(backlog);
11      }
12  }
```

Accept

```
public void StartAccept()
{
    SocketAsyncEventArgs args = new SocketAsyncEventArgs();
    args.Completed += OnAcceptCompleted;

    Accept(args);
}

private void OnAcceptCompleted(object sender, SocketAsyncEventArgs args)
{
    if (args.SocketError == SocketError.Success)
    {
        Socket clientSocket = args.AcceptSocket;
        StartReceive(clientSocket);

        lock (clientSocketsLocker)
            clientSockets.Add(clientSocket);

        IPEndPoint clientSocketEndPoint = clientSocket.RemoteEndPoint as IPEndPoint;
        Console.WriteLine($"클라이언트가 접속하였습니다. [{clientSocketEndPoint.Address}]");
    }
    else
        Console.WriteLine(args.SocketError);

    Accept(args);
}

private void Accept(SocketAsyncEventArgs args)
{
    args.AcceptSocket = null;

    bool pending = listenSocket.AcceptAsync(args);
    if (pending == false)
        OnAcceptCompleted(null, args);
}
```

Receive

```
private void StartReceive(Socket socket)
{
    SocketAsyncEventArgs receiveArgs = new SocketAsyncEventArgs();
    receiveArgs.Completed += OnReceiveCompleted;
    receiveArgs.UserToken = socket;

    Receive(socket, receiveArgs);
}

private void Receive(Socket socket, SocketAsyncEventArgs args)
{
    ArraySegment<byte> buffer = new ArraySegment<byte>(new byte[1024], 0, 1024);
    args.SetBuffer(buffer.Array, buffer.Offset, buffer.Count);

    bool pending = socket.ReceiveAsync(args);
    if (pending == false)
        OnReceiveCompleted(null, args);
}

private void OnReceiveCompleted(object sender, SocketAsyncEventArgs args)
{
    Socket socket = args.UserToken as Socket;
    if (args.SocketError == SocketError.Success && args.BytesTransferred > 0)
    {
        string receivedMessage = Encoding.UTF8.GetString(args.Buffer, 0, args.BytesTransferred);
        lock (handlerLocker)
            onMessageReceivedEvent?.Invoke(socket, receivedMessage);

        Receive(socket, args);
    }
    else
        Kick(socket);
}
```

Kick

```
1   public void Kick(Socket socket)
2   {
3       try
4       {
5           IPEndPoint clientSocketEndPoint = socket.RemoteEndPoint as IPEndPoint;
6
7           socket.Shutdown(SocketShutdown.Both);
8           socket.Close();
9
10          lock (clientSocketsLocker)
11              clientSockets.Remove(socket);
12
13          Console.WriteLine($"클라이언트가 접속 해제하였습니다. [{clientSocketEndPoint.Address}]");
14      }
15      catch { }
16  }
```

Listener

```
1  private Socket listenSocket = null;
2  private List<Socket> clientSockets = new List<Socket>();
3
4  private Action<Socket, string> onMessageReceivedEvent;
5
6  private object clientSocketsLocker = new object();
7  private object handlerLocker = new object();
8
9  public Listener(IPEndPoint endPoint, int backlog, Action<Socket, string> onMessageReceived)
10 {
11     listenSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
12     listenSocket.Bind(endPoint);
13     listenSocket.Listen(backlog);
14
15     onMessageReceivedEvent += onMessageReceived;
16 }
```

Broadcast

```
1  public void Broadcast(string message)
2  {
3      byte[] sendBytes = Encoding.UTF8.GetBytes(message);
4      sendArgs.SetBuffer(sendBytes);
5
6      lock (clientSocketsLocker)
7          clientSockets.ForEach(socket => socket.SendAsync(sendArgs));
8  }
```

Listener

```
1   private Socket listenSocket = null;
2   private List<Socket> clientSockets = new List<Socket>();
3
4   private Action<Socket, string> onMessageReceivedEvent;
5   private SocketAsyncEventArgs sendArgs;
6
7   private object clientSocketsLocker = new object();
8   private object handlerLocker = new object();
9
10  public Listener(IPEndPoint endPoint, int backlog, Action<Socket, string> onMessageReceived)
11  {
12      listenSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
13      listenSocket.Bind(endPoint);
14      listenSocket.Listen(backlog);
15
16      onMessageReceivedEvent += onMessageReceived;
17      sendArgs = new SocketAsyncEventArgs();
18  }
```

Main

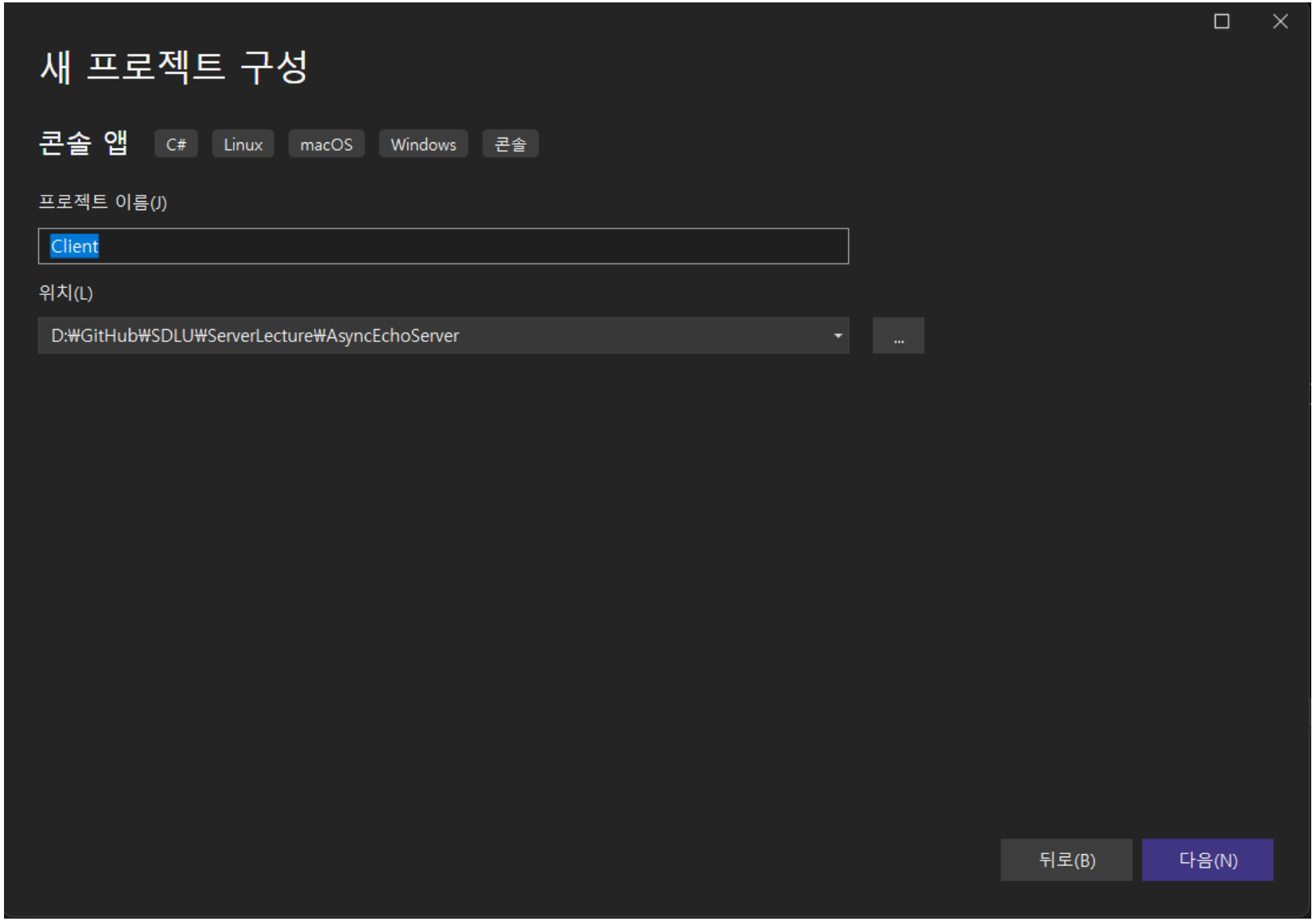
```
1   public class Program
2   {
3       private static Listener listener;
4
5       static void Main(string[] args)
6       {
7           string host = Dns.GetHostName();
8           IPHostEntry ipHost = Dns.GetHostEntry(host);
9           IPAddress ipAddress = ipHost.AddressList[1];
10          IPEndPoint endPoint = new IPEndPoint(ipAddress, 8081);
11
12          listener = new Listener(endPoint, 10, OnMessageReceived);
13          listener.StartAccept();
14
15          while (true)
16          {
17
18          }
19      }
20  }
```

OnMessageReceived

```
1   private static void OnMessageReceived(Socket socket, string msg)
2   {
3       IPEndPoint clientEndPoint = (socket.RemoteEndPoint as IPEndPoint);
4
5       string message = $"[{clientEndPoint.Address}] {msg}";
6       Console.WriteLine(message);
7       listener.Broadcast(message);
8
9       if (msg.IndexOf("exit") > -1)
10          listener.Kick(socket);
11  }
```

클라이언트

프로젝트 생성


새 프로젝트 구성
콘솔 앱
C#
Linux
macOS
Windows
콘솔
프로젝트 이름(J)
Client
위치(L)
D:₩GitHub₩SDLU₩ServerLecture₩AsyncEchoServer
...
뒤로(B)
다음(N)

Connector

```
1  public class Connector
2  {
3      private Socket serverSocket;
4
5      public Connector(IPEndPoint endPoint)
6      {
7          serverSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
8      }
9  }
```

Connect
Disconnect

```
1   public void Connect(IPEndPoint endPoint)
2   {
3       serverSocket.Connect(endPoint);
4
5       Console.WriteLine("서버와 연결되었습니다.");
6       StartReceive();
7   }
8
9   public void Disconnect()
10  {
11      serverSocket.Shutdown(SocketShutdown.Both);
12      serverSocket.Close();
13
14      Console.WriteLine("서버와 연결이 끊겼습니다.");
15  }
```

Receive

```
private void StartReceive()
{
    SocketAsyncEventArgs receiveArgs = new SocketAsyncEventArgs();
    receiveArgs.Completed += OnReceiveCompleted;

    Receive(receiveArgs);
}

private void Receive(SocketAsyncEventArgs args)
{
    ArraySegment<byte> buffer = new ArraySegment<byte>(new byte[1024], 0, 1024);
    args.SetBuffer(buffer.Array, buffer.Offset, buffer.Count);

    bool pending = serverSocket.ReceiveAsync(args);
    if (pending == false)
        OnReceiveCompleted(null, args);
}

private void OnReceiveCompleted(object sender, SocketAsyncEventArgs args)
{
    if (args.SocketError == SocketError.Success && args.BytesTransferred > 0)
    {
        string receivedMessage = Encoding.UTF8.GetString(args.Buffer, 0, args.BytesTransferred); // 받은 메세지 변환
        onMessageReceivedEvent?.Invoke(receivedMessage);

        Receive(args);
    }
    else
        Disconnect();
}
```

Connector

```
private Socket serverSocket;
private Action<string> onMessageReceivedEvent;

public Connector(IPEndPoint endPoint, Action<string> onMessageReceived)
{
    serverSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
    this.onMessageReceivedEvent = onMessageReceived;
}
```

Send

```
1  public void Send(string message)
2  {
3      byte[] sendBytes = Encoding.UTF8.GetBytes(message);
4      sendArgs.SetBuffer(sendBytes);
5
6      serverSocket.SendAsync(sendArgs);
7  }
```

Connector

```
1   private Socket serverSocket;
2   private Action<string> onMessageReceivedEvent;
3
4   private SocketAsyncEventArgs sendArgs;
5
6   public Connector(IPEndPoint endPoint, Action<string> onMessageReceived)
7   {
8       serverSocket = new Socket(endPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
9
10      this.onMessageReceivedEvent = onMessageReceived;
11      sendArgs = new SocketAsyncEventArgs();
12  }
```

Main

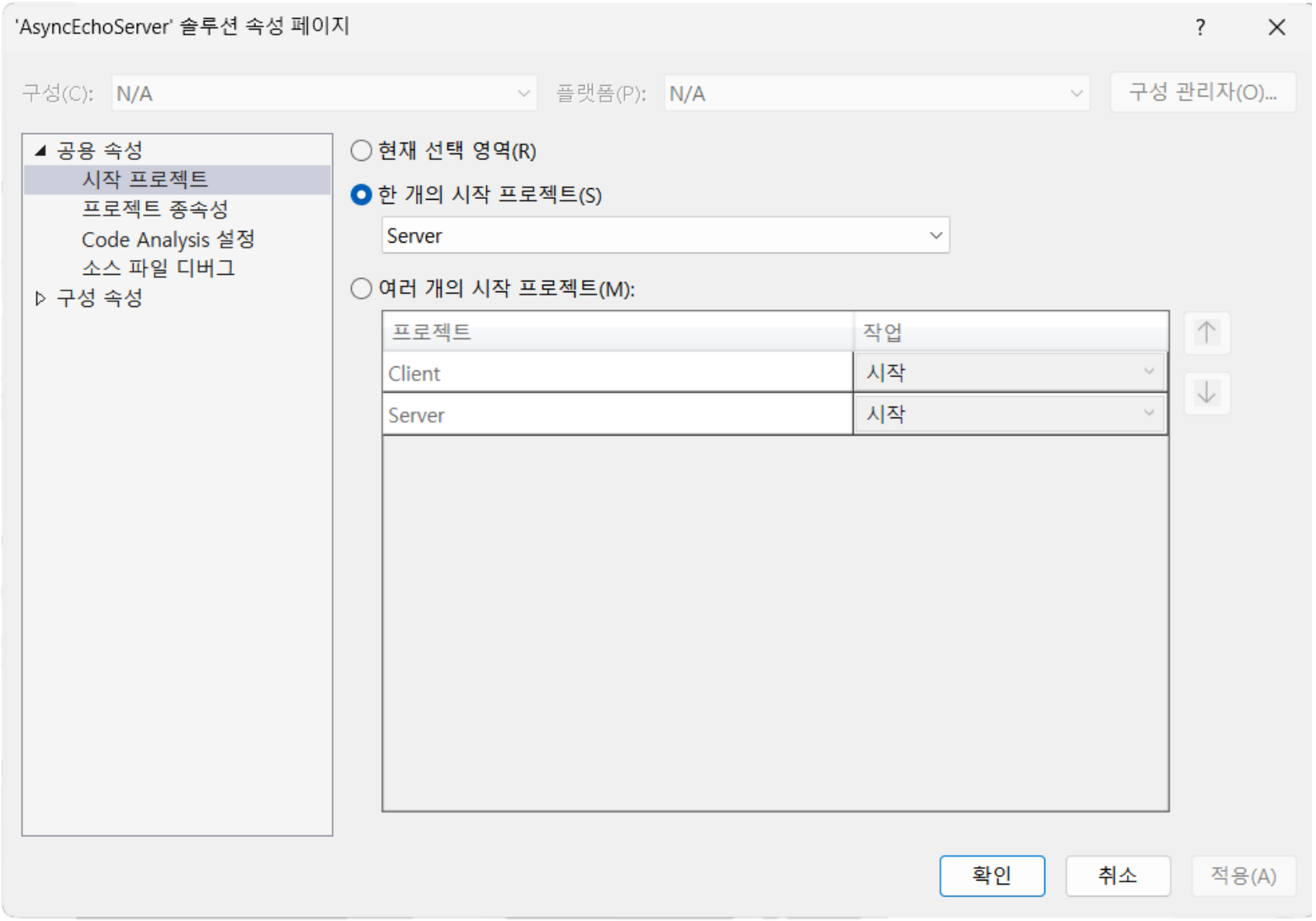
```
1   public class Program
2   {
3       private static Connector connector;
4
5       static void Main(string[] args)
6       {
7           IPAddress ipAddress = IPAddress.Parse("172.31.1.175");
8           IPEndPoint endPoint = new IPEndPoint(ipAddress, 8081);
9
10          connector = new Connector(endPoint, OnMessageReceived);
11          connector.Connect(endPoint);
12
13          while (true)
14          {
15              string message = Console.ReadLine();
16              connector.Send(message);
17
18              if (message.IndexOf("exit") > -1)
19              {
20                  connector.Disconnect();
21                  break;
22              }
23          }
24
25          Console.WriteLine("클라이언트 종료.");
26      }
27  }
```

OnMessageReceived

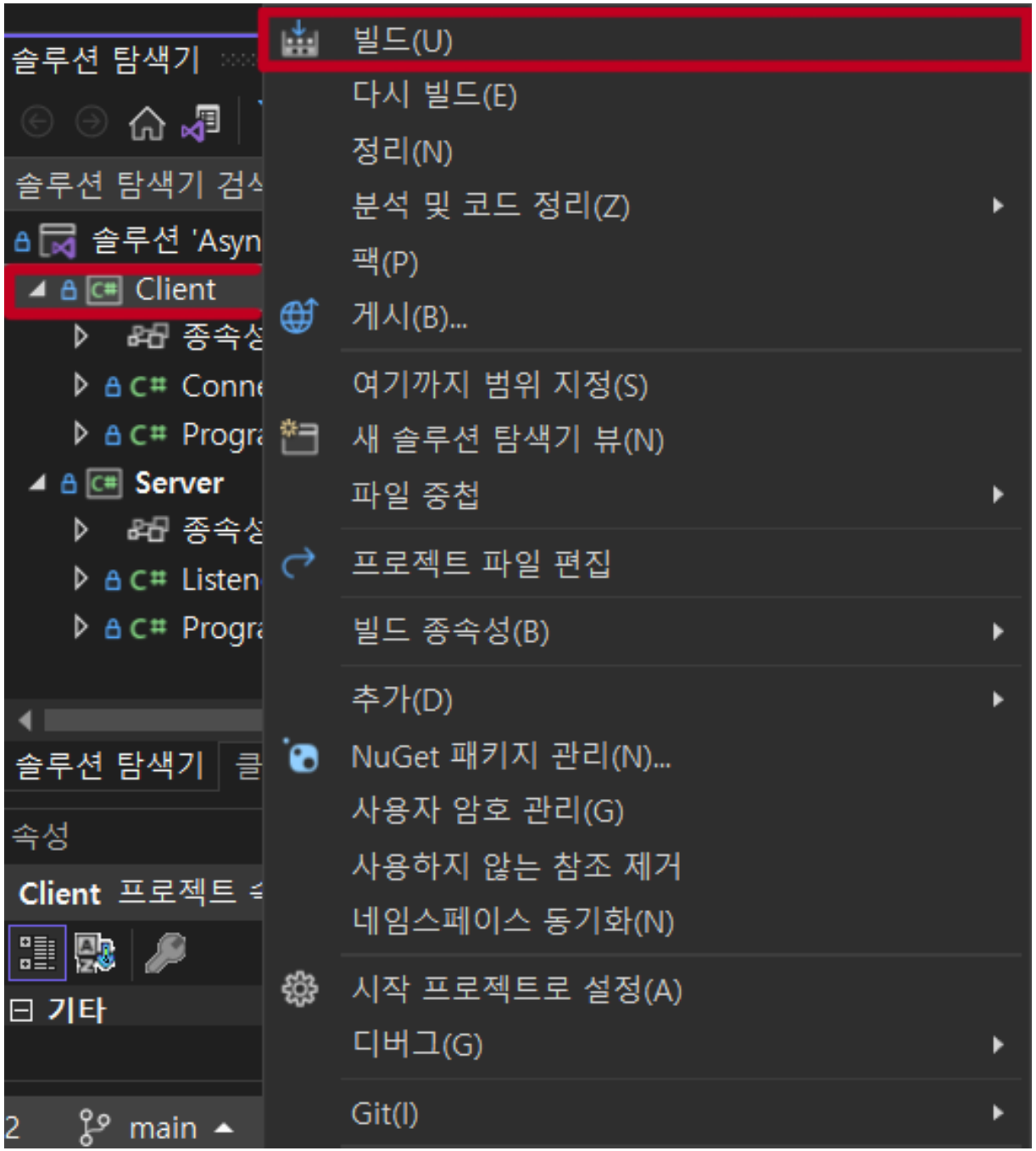
```
1  private static void OnMessageReceived(string msg)
2  {
3      Console.WriteLine(msg);
4  }
```

디버깅

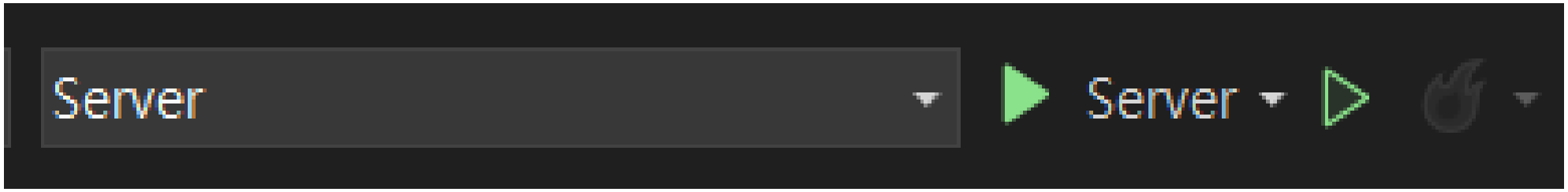
시작 프로그램


'AsyncEchoServer' 솔루션 속성 페이지
구성(C): N/A
플랫폼(P): N/A
구성 관리자(O)...
공용 속성
시작 프로젝트
프로젝트 종속성
Code Analysis 설정
소스 파일 디버그
구성 속성
현재 선택 영역(R)
한 개의 시작 프로젝트(S)
Server
여러 개의 시작 프로젝트(M):
프로젝트
작업
Client
시작
Server
시작
확인
취소
적용(A)

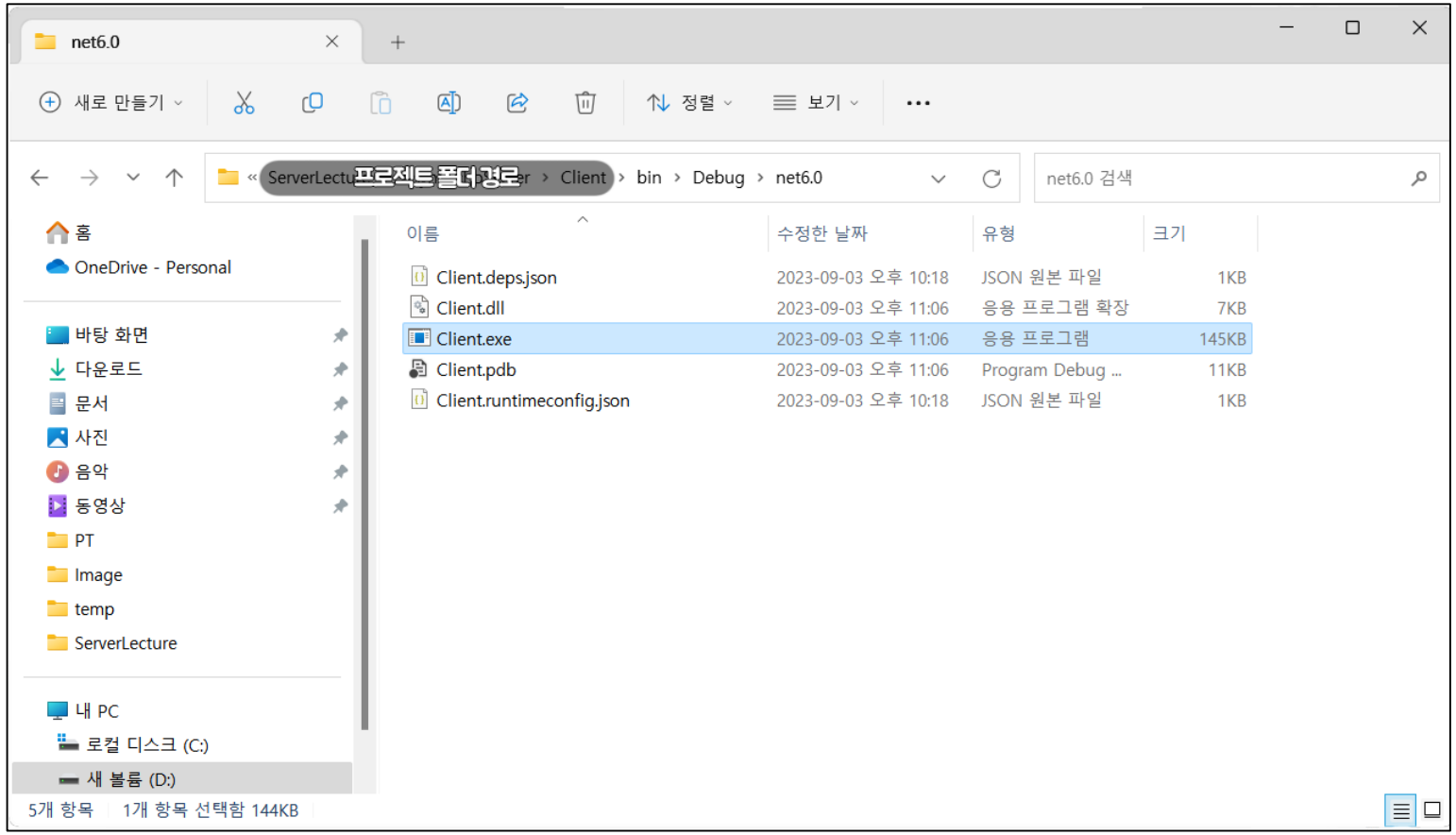
빌드


솔루션 탐색기
솔루션 탐색기 검
솔루션 'Asyn
Client
종속성
Conne
Progra
Server
종속성
Listen
Progra
솔루션 탐색기
속성
Client 프로젝트
기타
main
빌드(U)
다시 빌드(E)
정리(N)
분석 및 코드 정리(Z)
팩(P)
게시(B)...
여기까지 범위 지정(S)
새 솔루션 탐색기 뷰(N)
파일 중첩
프로젝트 파일 편집
빌드 종속성(B)
추가(D)
NuGet 패키지 관리(N)...
사용자 암호 관리(G)
사용하지 않는 참조 제거
네임스페이스 동기화(N)
시작 프로젝트로 설정(A)
디버그(G)
Git(I)

실행

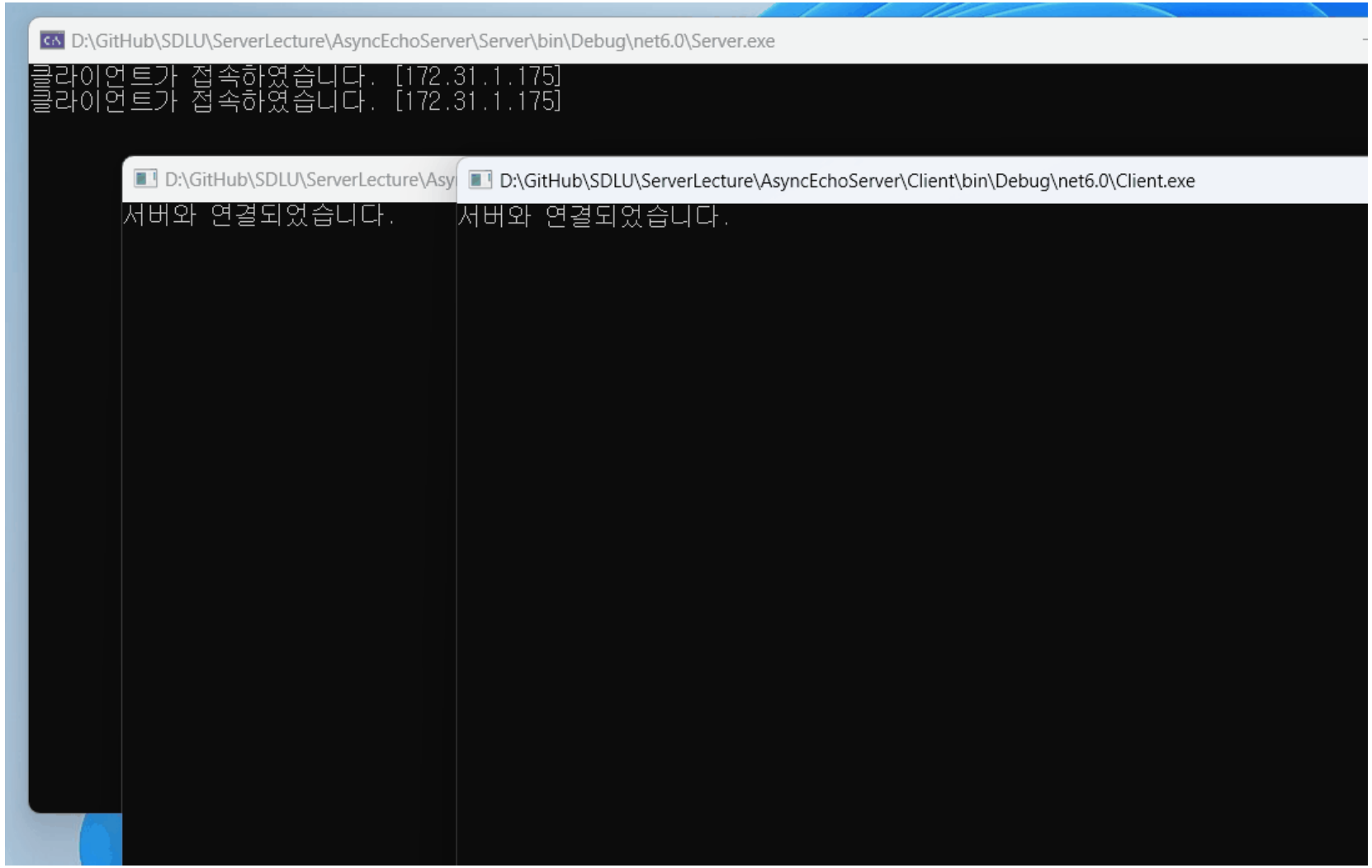


실행


net6.0
새로 만들기
정렬
보기
ServerLectu
프로젝트 폴더 경로
r › Client › bin › Debug › net6.0
net6.0 검색
홈
OneDrive - Personal
바탕 화면
다운로드
문서
사진
음악
동영상
PT
Image
temp
ServerLecture
내 PC
로컬 디스크 (C:)
새 볼륨 (D:)
이름
수정한 날짜
유형
크기
Client.deps.json 2023-09-03 오후 10:18 JSON 원본 파일 1KB
Client.dll 2023-09-03 오후 11:06 응용 프로그램 확장 7KB
Client.exe 2023-09-03 오후 11:06 응용 프로그램 145KB
Client.pdb 2023-09-03 오후 11:06 Program Debug ... 11KB
Client.runtimeconfig.json 2023-09-03 오후 10:18 JSON 원본 파일 1KB
5개 항목
1개 항목 선택함 144KB

X 2

디버깅


D:\GitHub\SDLU\ServerLecture\AsyncEchoServer\Server\bin\Debug\net6.0\Server.exe
클라이언트가 접속하였습니다. [172.31.1.175]
클라이언트가 접속하였습니다. [172.31.1.175]
D:\GitHub\SDLU\ServerLecture\Asy
서버와 연결되었습니다.
D:\GitHub\SDLU\ServerLecture\AsyncEchoServer\Client\bin\Debug\net6.0\Client.exe
서버와 연결되었습니다.