APM

> Asynchronnous Programming Model
> 비동기 프로그래밍 모델

▼ Index

1. Server
2. Client

1. Server

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace Server
{
    class Connection
    {
        // 메시지는 개행으로 구분한다.
        private static char CR = (char) 0x0D;
        private static char LF = (char) 0x0A;

        private Socket socket;
        private IPEndPoint info;

        private byte[] buffer = new byte[1024];
        private StringBuilder sb = new StringBuilder();

        public Connection(Socket socket)
        {
            this.socket = socket;
            this.info   = (IPEndPoint)socket.RemoteEndPoint;
            // buffer로 메시지를 받고 Receive함수로 메시지가 올 때까지 대기 한다.
            this.socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);

            Console.WriteLine($"Clinet:(From:{info.Address.ToString()}:{info.Port}, Connection Time : {DateTime.Now.ToString("yyyy-MM-
            Send("Welcome server!\r\n>");
        }

        private void Receive(IAsyncResult result)
        {
            if (socket.Connected)
            {
                // EndReceive를 호출하여 데이터 사이즈를 받는다.
                // EndReceive는 대기를 끝내는 것이다.
                int size = socket.EndReceive(result);

                sb.Append(Encoding.Unicode.GetString(buffer, 0, size).Trim('\0'));
                // 메시지의 끝이 이스케이프 \r\n의 형태이면 서버에 표시한다.
                if (sb.Length >= 2 && sb[sb.Length - 2] == CR && sb[sb.Length - 1] == LF)
                {
                    // 개행은 없애고..
                    sb.Length = sb.Length - 2;
                    string msg = sb.ToString();
                    Console.WriteLine(msg);
                    Send($"Echo - {msg}\r\n>");
                    if ("exit".Equals(msg, StringComparison.OrdinalIgnoreCase))
                    {
                        Console.WriteLine($"Disconnected : (From: {info.Address.ToString()}:{info.Port}, Connection time: {DateTime.No
                        // 접속을 중단한다.
                        socket.Close();
                        return;
                    }

                    // 버퍼를 비운다.
                    sb.Clear();
                }

                // buffer로 메시지를 받고 Receive함수로 메시지가 올 때까지 대기한다.
                this.socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);
            }
            else
```

```
            {
                // 접속이 끊겼다..
                Console.WriteLine($"Disconnected :  (From: {info.Address.ToString()}:{info.Port}, Connection time: {DateTime.Now})");
            }
        }

        private void Send(string msg)
        {
            byte[] data = Encoding.Unicode.GetBytes(msg);
            socket.Send(data, data.Length, SocketFlags.None);
        }
    }

    class Program : Socket
    {
        public Program()
            : base(AddressFamily.InterNetwork, System.Net.Sockets.SocketType.Stream,
                System.Net.Sockets.ProtocolType.Tcp)
        {
            base.Bind(new IPEndPoint(IPAddress.Any, 1004));
            base.Listen(1);
            // 비동기 소켓으로 Accept 클래스로 대기한다.
            BeginAccept(Accept, this);
        }

        private void Accept(IAsyncResult result)
        {
            // EndAccept로 접속 Client Socket을 받는다. EndAccept는 대기를 끝나는 것이다.
            // Client 클래스를 생성한다.
            var client = new Connection(EndAccept(result));
            // 비동기 소켓으로 Accept 클래스로 대기한다.
            BeginAccept(Accept, this);
        }

        static void Main(string[] args)
        {
            new Program();

            Console.WriteLine($"Press the q key to exit.");
            while (true)
            {
                string k = Console.ReadLine();
                if ("q".Equals(k, StringComparison.OrdinalIgnoreCase))
                    break;
            }
        }
    }
}
```

2. Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace Client
{
    class Program : Socket
    {
        // 메시지는 개행으로 구분한다.
        private static char CR = (char)0x0D;
        private static char LF = (char)0x0A;

        private byte[] buffer = new byte[1024];
        private StringBuilder sb = new StringBuilder();

        public Program()
            : base(System.Net.Sockets.AddressFamily.InterNetwork, System.Net.Sockets.SocketType.Stream,
                System.Net.Sockets.ProtocolType.Tcp)
        {
            // 접속 요청
            base.BeginConnect(new IPEndPoint(IPAddress.Parse("127.0.0.1"), 1004), Connect, this);
            while (true)
            {
                string k = Console.ReadLine();
                Send(k + "\r\n");
                if ("exit".Equals(k, StringComparison.OrdinalIgnoreCase))
```

```
                    break;
            }
        }

        // 접속되면 호출된다.
        private void Connect(IAsyncResult result)
        {
            // 접속 대기를 끝낸다.
            base.EndConnect(result);
            // buffer로 메시지를 바독 Recive함수로 메시지가 올 때까지 대기한다.
            base.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);
        }

        //메시지가 오면 호출된다.
        private void Receive(IAsyncResult result)
        {
            if (Connected)
            {
                //EndReceive를 호출하여 데이터 사이즈를 받는다.
                // EndReceive는 대기를 끝내는 것이다.
                int size = this.EndReceive(result);

                sb.Append(Encoding.Unicode.GetString(buffer, 0, size).Trim('\0'));
                // 메시지의 끝이 이스케이프 \r\n와 >의 형태이면 클라이언트에 표시한다.
                if (sb.Length >= 3 && sb[sb.Length - 3] == CR && sb[sb.Length - 2] == LF && sb[sb.Length - 1] == '>')
                {
                    string msg = sb.ToString();
                    // 콘솔에 출력한다.
                    Console.Write(msg);
                    // 버퍼 초기화
                    sb.Clear();
                }

                // buffer로 메시지를 바독 Recive함수로 메시지가 올 때까지 대기한다.
                base.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);
            }
            else
            {
                // 접속이 끊겼다..
                var remoteAddr = (IPEndPoint)RemoteEndPoint;
                Console.WriteLine($"Disconnected :  (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTi
            }
        }

        private void Send(string msg)
        {
            byte[] data = Encoding.Unicode.GetBytes(msg);
            Send(data, data.Length, SocketFlags.None);
        }

        static void Main(string[] args)
        {
            new Program();
        }
    }
}
```