TAP

> Task-based Asynchronous Pattern
> 작업 기반 비동기 패턴

▼ Index

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 TcpClient connection;
        private IPEndPoint info;

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

        public Connection(TcpClient connection)
        {
            this.connection = connection;
            this.info = (IPEndPoint)connection.Client.RemoteEndPoint;
            Task.Factory.StartNew(Receive, this);

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

        private async void Receive(object obj)
        {
            while(connection.Connected)
            {
                int size = await connection.GetStream().ReadAsync(buffer, 0, buffer.Length).ConfigureAwait(false);
                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.
                        connection.Close();
                        return;
                    }

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

            // 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);
```

```
            connection.Client.Send(data, data.Length, SocketFlags.None);
        }
    }

    class Program : TcpListener
    {
        public Program()
            : base(IPAddress.Any, 1004)
        {
            base.Start(1);
            // 비동기 소켓으로 Accept 클래스로 대기한다.
            BeginAccept().Wait();
        }

        private async Task BeginAccept()
        {
            while (true)
            {
                // 비동기 Accept
                var client = new Connection(await AcceptTcpClientAsync().ConfigureAwait(false));
            }
        }

        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 Client : TcpClient
    {
        // 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();

        private TcpClient socket;
        private IPEndPoint remoteAddr;

        public Client(string ip, int port)
        {
            this.socket = new TcpClient(ip, port);
            remoteAddr = (IPEndPoint) socket.Client.RemoteEndPoint;

            // case 01. TAP
            // Task.Factory.StartNew(Receive, this);

            // case 02. APM 응용
            socket.Client.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);
        }

        private async void Receive(object obj)
        {
            while (socket.Connected)
            {
                int size = await socket.GetStream().ReadAsync(buffer, 0, buffer.Length).ConfigureAwait(false);
```

```
                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();
                }
            }

            Console.WriteLine($"Disconnected :  (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTime.N
        }


        //메시지가 오면 호출된다.
        private async void Receive(IAsyncResult result)
        {
            if (socket.Connected)
            {
                //EndReceive를 호출하여 데이터 사이즈를 받는다.
                // EndReceive는 대기를 끝내는 것이다.
                int size = socket.Client.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.
                socket.Client.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, Receive, this);
            }
            else
            {
                // 접속이 끊겼다..
                Console.WriteLine($"Disconnected :  (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTi
            }
        }

        public void Send(string msg)
        {
            byte[] data = Encoding.Unicode.GetBytes(msg);

            socket.Client.Send(data, data.Length, SocketFlags.None);
        }
    }

    class Program
    {
        public Program()
        {
            var client = new Client("127.0.0.1", 1004);
            while (true)
            {
                string k = Console.ReadLine();
                client.Send(k + "\r\n");
                if ("exit".Equals(k, StringComparison.OrdinalIgnoreCase))
                    break;
            }
        }

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