EAP

> Event-based Asynchronous Pattern
> 이벤트 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;
using System.Threading.Tasks;

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

        // 메시지를 모으기 위한 버퍼
        private StringBuilder sb = new StringBuilder();
        private IPEndPoint remoteAddr;

        public Socket Socket => base.UserToken is Socket ? base.UserToken as Socket : null;

        public Connection(Socket socket)
        {
            // 이 비동기 소켓 작업과 연결된 사용자 또는 애플리케이션 개체를 가져오거나 설정합니다.
            base.UserToken = socket;

            // 메모리 버퍼를 초기화 한다. 크기는 1024이다
            base.SetBuffer(new byte[1024], 0, 1024);
            // * Message를 수신할 수 있도록 거는 것이 ReceiveAsync이고,
            //   수신한 메시지를 꺼내는 것이 Completed Event를 설정하는 것이다.
            // 메시지가 오면 이벤트를 발생시킨다. (IOCP로 꺼내는 것)
            base.Completed += Client_Completed;
            // 메시지가 오면 이벤트를 발생시킨다. (IOCP로 넣는 것)
            Socket.ReceiveAsync(this);

            // 접속 환영 메시지
            remoteAddr = (IPEndPoint) socket.RemoteEndPoint;
            Console.WriteLine($"Client : (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTime.Now})");
            this.Send("Welcome server!\r\n>");
        }

        // 메시지가 오면 발생하는 이벤트
        private void Client_Completed(object sender, SocketAsyncEventArgs e)
        {
            // 접속이 연결되어 있으면...
            if (Socket.Connected && base.BytesTransferred > 0)
            {
                // 수신 데이터는 e.Buffer에 있다.
                byte[] data = e.Buffer;

                // 메모리 버퍼를 초기화 한다. 크기는 1024이다
                // 새롭게 생성해서 대체하는 방식이며, Array.Clear(e.Buffer, 0x0, e.Buffer.Length); 해당 기능으로 초기화 가능
                base.SetBuffer(new byte[1024], 0, 1024);
                // 데이터의 공백은 없앤다.
                sb.Append(Encoding.Unicode.GetString(data).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: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time:
                        // 접속을 중단한다.
                        Socket.DisconnectAsync(this);
                        return;
                    }

                    sb.Clear();
                }

                // 메시지가 오면 이벤트를 발생시킨다. (IOCP로 넣는 것)
                Socket.ReceiveAsync(this);
            }
            else
            {
                // 접속이 끊겼다..
                Console.WriteLine($"Disconnected :  (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTi
            }
        }

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

    class Server : SocketAsyncEventArgs
    {
        public Socket Socket => base.UserToken is Socket ? base.UserToken as Socket : null;

        public Server(Socket socket)
        {
            base.UserToken = socket;
            // Client로부터 Accept이 되면 이벤트를 발생시킨다. (IOCP로 꺼내는 것)
            base.Completed += Server_Completed;
        }

        private void Server_Completed(object sender, SocketAsyncEventArgs e)
        {
            // 접속이 완료되면, Client Event를 생성하여 Receive이벤트를 생성한다.
            var client = new Connection(e.AcceptSocket);
            // 서버 Event에 cilent를 제거한다.
            e.AcceptSocket = null;
            // Client로부터 Accept이 되면 이벤트를 발생시킨다. (IOCP로 넣는 것)
            Socket.AcceptAsync(e);
        }
    }

    // 메인 Program은 Socket을 상속받고 서버 Socket으로 사용한다.
    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 Server 클래스를 선언한다. (IOCP로 집어넣는것)
            base.AcceptAsync(new Server(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 Client : SocketAsyncEventArgs
    {
        // 메시지는 개행으로 구분한다.
        private static char CR = (char)0x0D;
        private static char LF = (char)0x0A;

        // 메시지를 모으기 위한 버퍼
        private StringBuilder sb = new StringBuilder();

        public Socket Socket => base.UserToken is Socket ? base.UserToken as Socket : null;

        public Client(Socket socket, EndPoint pep)
        {
            base.UserToken = socket;
            RemoteEndPoint = pep;

            // 접속시 발생하는 이벤트를 등록한다.
            base.Completed += Connected_Completed;
        }

        private void Connected_Completed(object sender, SocketAsyncEventArgs e)
        {
            // 접속 이벤트는 해제한다.
            base.Completed -= Connected_Completed;

            // 버퍼 설정
            base.SetBuffer(new byte[1024], 0, 1024);
            // 수신 이벤트를 등록한다.
            base.Completed += Client_Completed;
            // 메시지가 오면 이벤트를 발생시킨다. (IOCP로 넣는 것)
            Socket.ReceiveAsync(this);
        }

        // 메시지가 오면 발생하는 이벤트
        private void Client_Completed(object sender, SocketAsyncEventArgs e)
        {
            // 접속이 연결되어 있으면...
            if (Socket.Connected && base.BytesTransferred > 0)
            {
                // 수신 데이터는 e.Buffer에 있다.
                byte[] data = e.Buffer;
                // 데이터를 string으로 변환한다.
                string msg = Encoding.Unicode.GetString(data);
                // 메모리 버퍼를 초기화 한다. 크기는 1024이다
                base.SetBuffer(new byte[1024], 0, 1024);
                sb.Append(msg.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] == '>')
                {
                    msg = sb.ToString();
                    // 콘솔에 출력한다.
                    Console.Write(msg);
                    // 버퍼 초기화
                    sb.Clear();
                }
                // 메시지가 오면 이벤트를 발생시킨다. (IOCP로 넣는 것)
                Socket.ReceiveAsync(this);
            }
            else
            {
                // 접속이 끊겼다..
                var remoteAddr = (IPEndPoint)Socket.RemoteEndPoint;
                Console.WriteLine($"Disconnected :  (From: {remoteAddr.Address.ToString()}:{remoteAddr.Port}, Connection time: {DateTi
            }
        }

        public void Send(String msg)
        {
            byte[] sendData = Encoding.Unicode.GetBytes(msg);
            Socket.Send(sendData, sendData.Length, SocketFlags.None);
        }
    }

    // 메인 Program은 Socket을 상속받고 클라이언트 Socket으로 사용한다.
    class Program : Socket
    {
        public Program()
            : base(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
        {
            var client = new Client(this, new IPEndPoint(IPAddress.Parse("127.0.0.1"), 1004));
            base.ConnectAsync(client);
            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();
        }
    }
}
```