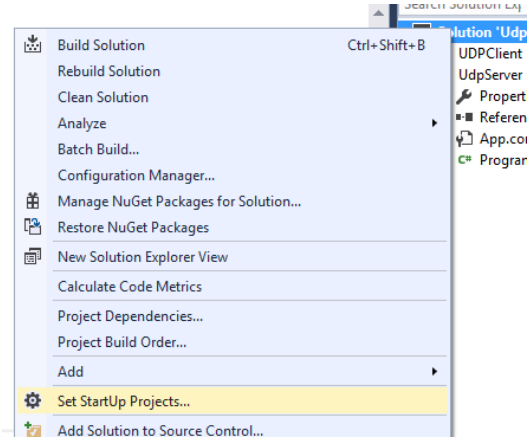
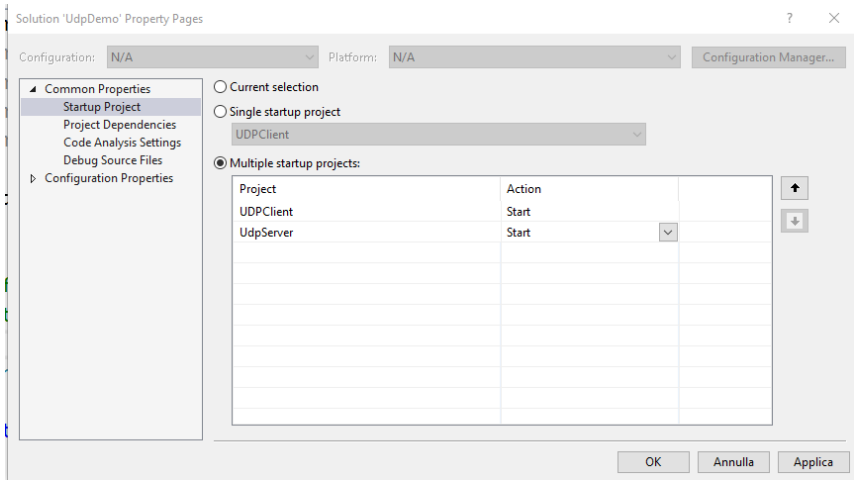
UDP in .NET

Per creare una Chat tramite UDP abbiamo bisogno di aprire due progetti: uno per il Client ed una per il server.

I due progetti devono essere avviati contemporaneamente. Per far ciò basta cliccare con il tasto desttro sulla solution e selezionare Set StartUp Projects.



Si aprirà una finestra che permetterà di stabilire un “Multiple StartUp Projects”.



A questo punto è possibile procedere con la scrittura del codice.

Pr prima cosa è necessario utilizzare i seguenti namespace:

```
using System.Net;
using System.Net.Sockets;
```

Affinchè due processi possano comunicare è necessario creare un EndPoint, anzi due, uno per il client

```
IPEndPoint epLocal = new IPEndPoint(IPAddress.Any, 4000);
```

ed uno per il server

```
IPEndPoint epLocal = new IPEndPoint(IPAddress.Any, 5000);.
```

Nel costruttore si specificano indirizzi ip e numeri di porta.

Un EndPoint è rappresentato da un indirizzo ip che identifica un host della rete e da una porta che identifica un processo di un'applicazione.

Il protocollo di trasporto UDP è implementato dalla classe UDP Client.

E' quindi necessario creare un oggetto UDPClient, specificando nel nel costruttore l'EndPoint

```
UdpClient cli = new UdpClient(epLocal);
```

Analogamente per il server

```
UdpClient serv = new UdpClient(epLocal);
```

Il client successivamente si connette all'indirizzo IP del server (nel nostro esempio client e server risideono sulla setta machina, pertanto l'indirizzo è localhost).

```
cli.Connect("127.0.0.1",5000);
```

Il messaggio viene trasformato un vettore di byte e codificato:

```
byte[] dati = Encoding.ASCII.GetBytes(messaggio);
```

Successivamente viene inviato insieme alla sua lunghezza.

```
cli.Send(dati, dati.Length);.
```

Per permettere più invii è sufficiente utilizzare un ciclo while. Ecco il codice completo.

```
        static void Main(string[] args)
        {
            Console.WriteLine("Client");
            //creare un oggetto UDPClient
            IPEndPoint epLocal = new IPEndPoint(IPAddress.Any, 4000);
            UdpClient cli = new UdpClient(epLocal);

            //connection-less: non è garantita la connessione
            // il client si vuole collegare al server

            cli.Connect("127.0.0.1",5000);

            //inserisco un ciclo while per inviare più messaggi

            while (true)
            {
                string messaggio = Console.ReadLine();
                //non posso inviare stringhe, devo codificarla
                //creo un vettore di byte
                byte[] dati = Encoding.ASCII.GetBytes(messaggio);

                //una volta codificati posso inviarli col metodo send
                //siccome invia un vettore,
                //devo passargli come parametro i dati e la langhezza del vettore

                cli.Send(dati, dati.Length);
                //se mando in esecuzione, il client non si accorge se è il server in ascolto

            }

            Console.ReadLine();
        } }
```

Passiamo al server che riceverà il vettore di byte e lo decodificherà.

```
byte[] dati = serv.Receive(ref ipe);

string s = Encoding.ASCII.GetString(dati);
```

Per ricevere è sufficiente inserire un ciclo while.

Ecco il codice completo:

```
    class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Server");
            IPEndPoint epLocal = new IPEndPoint(IPAddress.Any, 5000);
            UdpClient serv = new UdpClient(epLocal);
            //ipEndPoint ipe rappresenta il mittente
            IPEndPoint ipe= null;
            while(true)
            {
            byte[] dati = serv.Receive(ref ipe);
                //il Receive è un metodo bloccante
                //posso inviare solo dopo aver ricevuto
                //analogamente nel client ricevo dopo aver mandato
            string s = Encoding.ASCII.GetString(dati);
            Console.WriteLine(s);
            }

            Console.ReadLine();
        }
    }
```