Una semplice chat senza protocolli

http://www.vbsimple.net/cliserv/clser_01.htm

Difficoltà: 2 / 5

L'esempio più semplice in assoluto di programma Client/Server è la chat.
L'esempio che vedremo permetterà a due utenti di comunicare tra loro, mediante la connessione diretta tramite una specifica porta TCP.

La parte server di questa semplice chat aprirà la porta 2000 in modalità di ascolto ed attenderà la connessione da parte della parte client del progetto. L'esempio funzionerà senza alcun protocollo, ovvero non esisteranno regole che stabiliscono il formato dei messaggi o ne controllano la correttezza. Tutti i dati che un utente invierà all'altro saranno mostrati in una casella di testo.

Vediamo inizialmente la parte server: essa consiste soltanto in un Form sulla cui superficie inseriremo alcuni controlli.

Innanzitutto inseriremo un *CommandButton* di nome **Listen**. Inseriamo un controllo *Winsock* di nome **wskServer** che servirà da server di ascolto per comunicare con il client.

Inseriremo anche due caselle di testo: la prima sarà chiamata **txtReply** ed avrà la proprietà *Multiline* settata a ***True***; la seconda sarà chiamata **txtOut**. Il form si conclude con un altro pulsante di nome **Invia**.

Il funzionamento del form è molto semplice: l'utente del server prima di poter ricevere chiamate dal client deve porre il programma in attesa di connessione premendo il pulsante **Listen**. Una volta che l'altro utente si sarà collegato i messaggi provenienti dal client saranno mostrati nella casella di testo grande. L'altra casella più piccola con il pulsante **Invia** servirà per inviare messaggi al client.

Il codice è altrettanto semplice:

```
 1. Option Explicit
 2.
 3. Private Sub Listen_Click()
 4.     wskServer.Close
 5.     wskServer.LocalPort = 2000
 6.     wskServer.Listen
 7.     txtReply.Text = txtReply.Text & "Server in attesa..." & vbCrLf
 8.     txtReply.SelStart = Len(txtReply.Text)
 9. End Sub
10.
```


Come prima cosa porremo il server in attesa (in ascolto).
Alla riga 4, per evitare problemi in caso che l'utente clicchi più di una volta il pulsante, chiudiamo la connessione del server. Se essa era aperta sarà chuisa; se essa non era aperta, l'istruzione non ha alcun effetto, né genera errori.

Alla riga 5 stabiliamo la porta locale sulla quale il server ascolterà (riga 6). Non appena il server si sarà messo in attesa sarà aggiunto alla **txtReply** il testo ***"Server in attesa..."*** ed il cursore sarà spostato alla fine del testo (righe 7-8).

Da questo momento in poi il server non potrà fare nulla fino a quando il client non si connette ad esso.

```
11. Private Sub wskServer_ConnectionRequest(ByVal requestID As Long)
12.     If (wskServer.State <> sckClosed) Then wskServer.Close
13.     wskServer.Accept requestID
14.     txtReply.Text = txtReply.Text & "Connessione accettata..." & vbCrLf
15.     txtReply.SelStart = Len(txtReply.Text)
16. End Sub
17.
```

Quando un qualunque computer tenta di collegarsi al server, scatterà l'evento *ConnectionRequest* sul socket **wskServer**. Sarà fornito anche un numero identificativo della chiamata. Per confermare la chiamata è necessario che venga accettata la chiamata a tale numero identificativo.

Così, alla riga 11, viene innanzitutto controllato se il server era precedentemente collegato. Nel caso che lo fosse, sarà disconnesso prima di effettuare una nuova connessione.

Alla riga 12 viene accettata la connessione mediante il metodo **Accept** ed il parametro identificativo della connessione *requestID*. Fatto questo sarà mostrato il messaggio ***"Connessione accettata..."*** ed il cursore sarà spostato alla fine del testo, per mostrare gli ultimi messaggi (righe 14-15).

```
18. Private Sub wskServer_DataArrival(ByVal bytesTotal As Long)
19.     Dim DATI As String
20.     wskServer.GetData DATI
21.     txtReply.Text = txtReply.Text & DATI & vbCrLf
22.     txtReply.SelStart = Len(txtReply.Text)
23. End Sub
24.
```

Nel momento in cui il server riceve dati, tramite il client oppure tramite una qualunque altra connessione, verrà eseguito l'evento *DataArrival*.
Pertanto, ogni volta che scatta tale evento, il server si dovrà occupare di leggere i dati in entrata e mostrarli nella casella di testo dedicata.

Per leggere i dati sarà necessaria una variabile di tipo Stringa che chiameremo **DATI**.
Mediante essa leggeremo i dati utilizzando il metodo **GetData** del socket server.

Sarà allora possibile mostrarne il contenuto aggiungendolo alla casella di testo **txtReply** e spostando il cursore alla fine del testo (righe 21-22).

```
25. Private Sub Invia_Click()
26.     If wskServer.State <> sckConnected Then
27.         txtReply.Text = txtReply.Text & "Non connesso" & vbCrLf
```



```
28.         txtReply.SelStart = Len(txtReply.Text)
29.         Exit Sub
30.     End If
31.     wskServer.SendData txtOut.Text & vbCrLf
32. End Sub
```

Le funzioni del server si completano concedendogli la possibilità di comunicare con il client. Abbiamo un pulsante **Invia** che invia il testo presente nella casella di testo **txtReply**.

Pertanto l'esecuzione dell'evento *Click* del pulsante dovrà inviare i dati al socket collegato. Prima di effettuare quest'operazione, però, è necessario verificare se il server è connesso (riga 26). Se esso non è connesso, il tentativo di inviare dati genera un errore.
Quindi, prima di inviare i dati, controlliamo lo stato del server mediante la proprietà *State* del socket **wskServer**. Se esso non è connesso mostreremo il messaggio ***"Non connesso"*** ed usciremo dalla Sub (righe 27-29).

Il raggiungimento della riga 31 presuppone la connessione del server con il client. Potremo pertanto inviare i dati mediante il metodo **SendData** e specificando come testo il contenuto della casella di testo **txtOut**.

Il server termina qui. Possiamo provare il nostro server e collegarci ad esso mediante un qualunque programma di Telnet specificando come server l'indirizzo del computer in cui viene eseguito il programma server e come porta il numero 2000.

Alternativamente al programma di Telnet possiamo utilizzare un client molto semplice disegnato appositamente per questa applicazione. Il client si compone di un solo form.

Similmente al form server, questo contiene un pulsante di nome **Connetti**, una *Label* di nome **HostNameLabel**, una casella di testo di nome **txtHostName**. È necessario ovviamente anche un socket Winsock di nome **wskClient**.

Per comunicare abbiamo bisogno anche di una casella di testo di nome **txtReply** con la proprietà *Multiline* impostata a ***True***, una casella di testo di nome **txtOut** ed un pulsante di comando di nome **Invia**.

Il funzionamento è estremamente semplice: l'utente deve immettere l'indirizzo IP o il DNS del server in ascolto sulla porta 2000 e premere il pulsante **Connetti**. Stabilita la connessione sarà possibile scrivere messaggi nella casella piccola e inviarli con il tasto **Invia**. Il codice è persino più semplice di quello del server:

```
1. Option Explicit
2.
3. Public Sub Connetti_Click()
4.     txtReply.Text = txtReply.Text & "Connessione in corso..."
5.     txtReply.SelStart = Len(txtReply.Text)
6.     wskClient.Close
7.     wskClient.LocalPort = 0
```



```
 8.     wskClient.Connect txtHostName.Text, 2000
 9. End Sub
10.
```

Al click sul pulsante Connetti il client dovrà tentare la connessione al server, il cui indirizzo è specificato nella casella txtHostName.

Alle righe 4 e 5 viene mostrato un messaggio all'utente che avverte della richiesta connessione. Alla riga 6 viene chiuso l'eventuale socket aperto; se l'utente preme il pulsante **Connetti** mentre è connesso, sarà eseguita la disconnessione, l'impostazione della porta locale 0 (riguardo lo stato TIME_WAIT di NETSTAT) per far scegliere al sistema operativo una porta TCP libera (riga 7).

Possiamo adesso effettuare la connessione al server utilizzando il metodo **Connect** del socket **wskClient** richiedendo come indirizzo a cui collegarsi quello specificato nella casella di testo **txtHostName** e numero della porta 2000.

```
11. Private Sub wskClient_DataArrival(ByVal bytesTotal As Long)
12.     Dim DATI As String
13.     wskClient.GetData DATI
14.     txtReply.Text = txtReply.Text & DATI & vbCrLf
15.     txtReply.SelStart = Len(txtReply.Text)
16. End Sub
17.
```

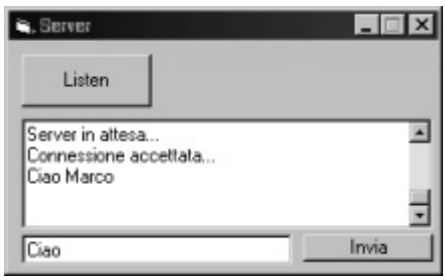
In maniera perfettamente identica al server, quando arrivano dati al socket **wskClient**, sarà necessario estrarre i dati, salvarli in una variabile di nome **DATI** e visualizzarli nella casella di testo **txtReply** (righe 12-15).

```
18. Private Sub Invia_Click()
19.     If wskClient.State <> sckConnected Then
20.         txtReply.Text = txtReply.Text & "Non connesso"
21.         txtReply.SelStart = Len(txtReply.Text)
22.         Exit Sub
23.     End If
24.     wskClient.SendData txtOut.Text & vbCrLf
25. End Sub
```

Anche qui il processo è del tutto analogo a quello del server.
Prima di inviare i dati mediante il pulsante **Invia**, viene verificata se la connessione è attiva. Se non lo è viene mostrato un messaggio all'interno della casella di testo **txtReply**, altrimenti mediante il metodo SendData viene inviato al socket server il contenuto della casella **txtOut** (righe 19-24).

Per provare i due programmi, se si dispone di una LAN, installare il server su un computer ed il client su un altro; se si dispone invece di un solo computer e non si ha la possibilità di far installare il programma ad un amico connesso su internet, è possibile eseguire entrambi i computer sulla stessa macchina.

Fatto questo sarà possibile eseguire il programma client, immettere l'indirizzo IP della macchina che funge da server (se entrambi vengono eseguiti sulla stessa macchina, utilizzare l'indirizzo di loopback 127.0.0.1 per chiamare il server della stessa macchina) e premere il pulsante connetti. Sarà facilmente possibile comunicare tra i due programmi.



Lanciare innanzitutto il programma server e premere il pulsante **Listen**.

Il progetto è molto semplice e contiene molti errori legati alla disconnessione di uno dei computer. Inoltre non esistendo protocollo non sarà possibile scambiare comandi tra i due programmi.

Questo esempio intende soltanto introdurre il funzionamento di base del controllo Winsock e mostrare quali sono le procedure di base da eseguire per controllare qualsiasi programma Client/Server.

Alberto Alagna
7 Marzo 2001

Torna all'indice Client / Server

