

Visual Foxpro y WinSock II

Estableciendo la conexión

Es ya sabido que en una comunicación entre computadoras es necesario contar con un servidor y un cliente; el servidor necesita estar escuchando una petición de conexión por el cliente para poder entablar una conexión y a partir de que la conexión se ha realizado poder comunicarse. Para poder comunicarse, ambos deben de hablar el mismo lenguaje (protocolo de comunicación); dicho de otra forma, no se van a conectar con un cliente para ftp a un servidor de correo, ¿o sí?.

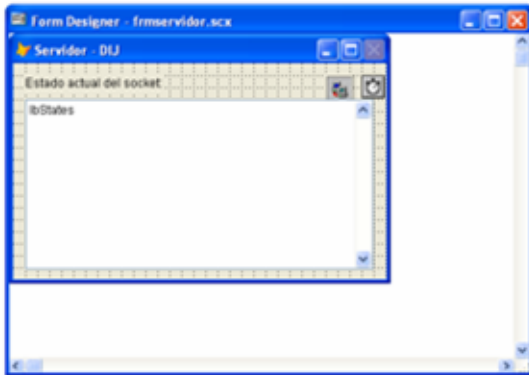
Existen varios protocolos ya establecidos (ver en www.rfc-es.org) y se pueden basar en alguno de ellos adaptándolos a sus necesidades.

Monitoreando los estados de la conexión...

Lo primero que tenemos que hacer es saber cómo se comporta (o que estados toma) el socket al momento de realizar una conexión por eso, con el primer ejemplo vamos a monitorear los estados que toma el control Winsock al entablar una conexión

El Servidor...

Creen un formulario y agréguele un control Winsock, y los objeto Timer, Label y ListBox, supondré que los nombres del Winsock, Timer, Label y ListBox son oWs, tmrState, lblState, lbStates respectivamente.



También hay que crear una propiedad en el formulario (digamos UltimoEstado) que nos indique el estado actual del socket, esto nos servirá para evitar ejecutar código si el estado del socket no ha cambiado.

El label indicará el estado actual del socket y el listbox los estados por donde a pasado el control.

Esto tampoco me lo dijeron: La única propiedad que se necesita definir en el servidor para que funcione correctamente es la de LocalPort; así que definan un número a utilizar (yo pondré el 1234).

En el evento ConnectionRequest del Winsock agreguen el código

```
IF This.State <> 0 THEN
    !!* Esto es necesario, como les comenté, por que el
    !!* socket no puede hacer dos cosas al mismo tiempo
    !!* o escucha o está conectado
    This.Object.Close()
ENDIF
This.Accept(RequestID)
```

En el evento Timer (del timer por supuesto) pongan el siguiente código

```
LOCAL lcEstado, strEstado
lcEstado = ThisForm.oWs.State

IF ThisForm.Ultimoestado = lcEstado THEN
    !!* No ha cambiado el estado, por tanto no tenemos nada que hacer aquí
    RETURN
ENDIF
```

```
This.Enabled = .F. &&<- No queremos que se ejecute nuevamente antes de terminar
```

```
#DEFINE sckClosed 0      && Socket is currently closed
#DEFINE sckOpen 1       && Socket is currently open
#DEFINE sckListening 2   && Socket is listening for requests
#DEFINE sckConnectionPending 3   && Socket has a pending request
#DEFINE sckResolvingHost 4   && Socket is resolving remote computer name
#DEFINE sckHostResolved 5   && Socket has resolved remote computer name
#DEFINE sckConnecting 6    && Socket is connecting to remote computer
#DEFINE sckConnected 7     && Socket has connected to remote computer
#DEFINE sckClosing 8      && Socket is closing connection to remote computer
#DEFINE sckError 9       && Socket has encountered an error
```

```
DO CASE
```

```
    CASE lcEstado = sckClosed
        strEstado = 'Socket cerrado'
        !!* Como es el servidor si no está conectado siempre deberá
        !!* estar escuchando
        Thisform.oWs.Listen()
    CASE lcEstado = sckOpen
        strEstado = 'Socket abierto'
    CASE lcEstado = sckListening
        strEstado = 'Esperando petición de conexión'
    CASE lcEstado = sckConnectionPending
        strEstado = 'Conexión pendiente '
    CASE lcEstado = sckResolvingHost
        strEstado = 'Resolviendo el host'
    CASE lcEstado = sckHostResolved
        strEstado = 'Host resuelto'
    CASE lcEstado = sckConnecting
        strEstado = 'Connectando'
    CASE lcEstado = sckConnected
        strEstado = 'Socket conectado'
    CASE lcEstado = sckClosing
        !!* Recuerden que el capítulo 1
        strEstado = 'Cerrando conexión '
        Thisform.oWs.Object.Close()
    CASE lcEstado = sckError
        strEstado = 'ERROR!!'
    OTHERWISE
        strEstado = 'Estado desconocido'
```

```
ENDCASE
```

```
ThisForm.Ultimoestado = lcEstado
ThisForm.lblState.Caption = 'Estado actual del socket: ' + strEstado
ThisForm.lbStates.AddItem(strEstado)
This.Enabled = .T.
```

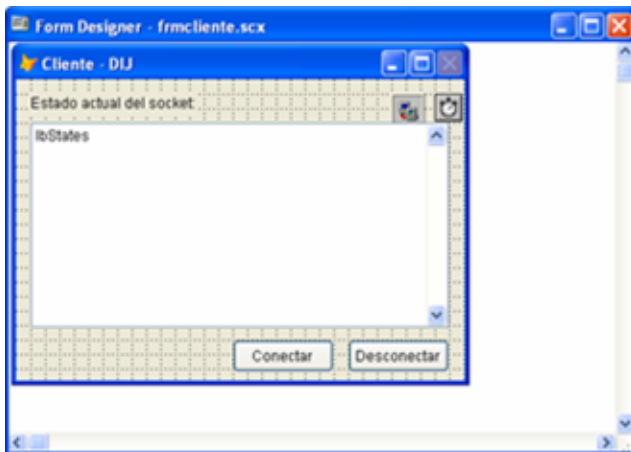
Y en el evento QueryUnload del formulario:

```
This.oWs.Object.Close()
```

Ahora vamos con

El Cliente...

Graben el formulario anterior con otro nombre (que indique que es el cliente) y agréguele dos botones (supondré los nombres btnConnect, btnDisconnect) como se muestra en la siguiente figura:



Del evento Timer, quiten la siguiente línea de código

```
Thisform.oWs.Listen()
```

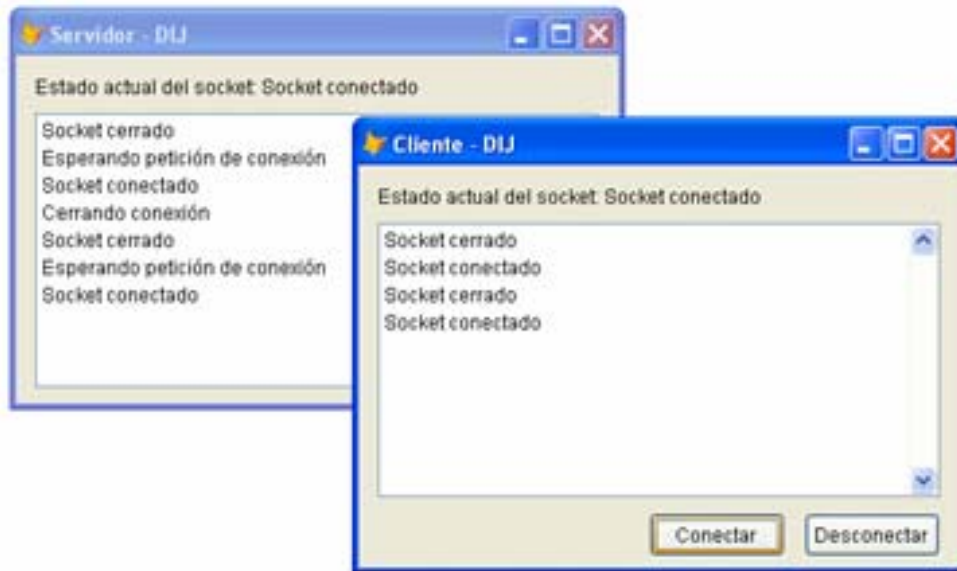
Por qué, porque éste es el cliente y se supone (en este caso) que no espera que alguien le solicite establecer una conexión.

Esto se lo voy a dejar a ustedes, pongan el código necesario en los botones para conectarse y desconectarse, no olviden el capítulo anterior de ésta serie, además de las propiedades necesarias en el control Winsock (son solamente 2).

Qué es lo que se supone debe hacer...

Al momento de ejecutar el servidor, éste debe entrar al estado de “escuchar”, en otras palabras está esperando que alguien le solicite una conexión.

Ejecuten el cliente, presionen el botón “Conectar”, al hacer esto el cliente intentará establecer la conexión con el servidor; en ambos casos en el listbox se irá indicando los estados de los sockets.



¿Errores?...

Si el cliente no establece la conexión puede ser por:

- No se ha especificado la propiedad RemoteHost del control Winsock, si la prueba la están haciendo en la misma computadora se soluciona asignando “localhost” a la propiedad.
- Los puertos LocalPort del servidor y RemotePort del cliente no son el mismo
- Si lo anterior está bien, quizá se nos haya pasado aceptar la conexión en el evento ConnectionRequest

El Servidor; pues éste se supone que no debe causar errores.

Un ejemplo más útil...

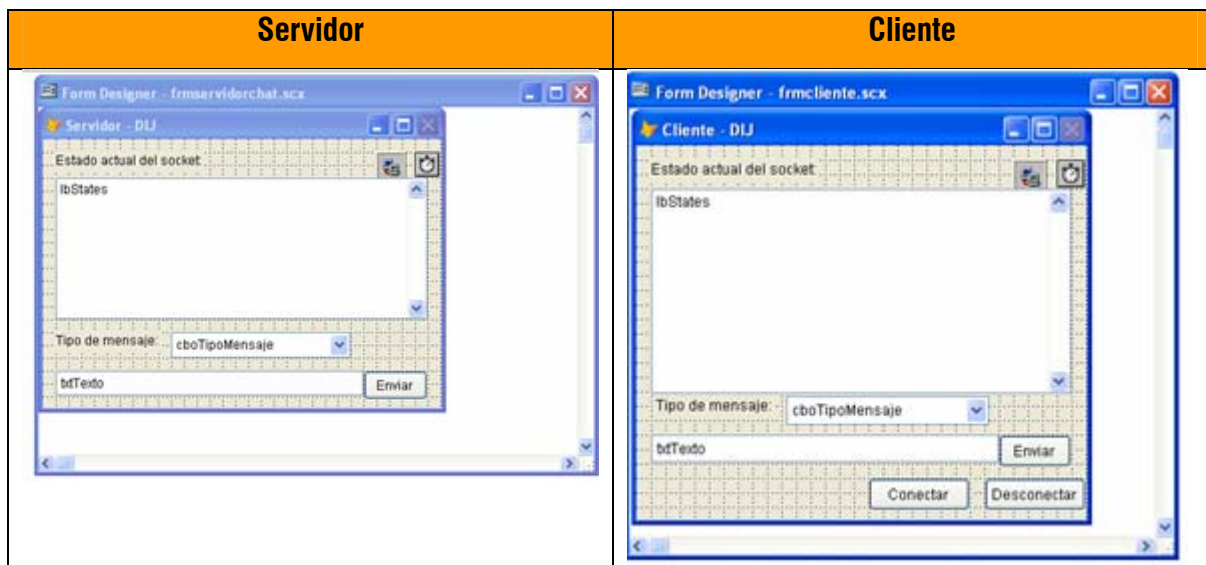
Primero que nada hay que definir que es lo que se va a hacer, vamos a empezar con el más típico, el del chat, pero vamos a hacer algo diferente, se podrán enviar dos tipos de mensajes, uno que aparecerá en el mismo formulario (en un control, claro) y otro que se muestre en un messagebox, es algo sencillo pero diferente.

En éste caso no es necesario un protocolo de comunicación así que, lo único que definiremos será los comandos que aceptarán el cliente y servidor.

| Comando | Acción |
|--------------|--|
| MSG<Texto> | Mostrará el texto en un control del formulario |
| MSI<Texto> | Presentará el mensaje con MessageBox y el icono de información |
| MSE<Texto> | Presentará el mensaje con MessageBox y el icono de error |
| SHL<Comando> | Comando válido para ejecutarse con ShellExecute |

Como el ejemplo es solo demostrativo con esos tres comandos es suficiente.

Tomando como base el servidor y cliente anteriores hagan las modificaciones necesarias para que el formulario quede como se muestran en las siguientes figuras:



El combobox tendrá los diferentes tipos de mensajes que se pueden enviar:

- Normal: que aparecerá en el control Listbox
- MessageBox – Información: que ejecutará el MessageBox con el icono de información
- MessageBox – Error: que ejecutará el MessageBox con el icono de error

El control text tendrá el texto que se va a enviar y el botón enviar tendrá el código necesario para enviar el texto.

Empezemos, **los cambios son en ambos formularios, cliente y servidor**

1. Agreguen la propiedad Comandos a los formulario (ésta propiedad debe ser un array)
2. En los evento INIT pongan el código:

```
WITH ThisForm
    DIMENSION .Comandos(3)
    .Comandos(1) = 'MSG'
    .Comandos(2) = 'MSI'
    .Comandos(3) = 'MSE'

    WITH .cboTipoMensaje
        .AddItem('NORMAL')
        .AddItem('MESSAGEBOX - Información')
        .AddItem('MESSAGEBOX - Error')
        .ListIndex = 1
    ENDWITH
ENDWITH
```

3. En los botones enviar:

```
LOCAL lcMsg
WITH ThisForm
    lcMsg = .Comandos(.cboTipoMensaje.ListIndex) + ALLTRIM(.txtTexto.Value)
    .oWs.SendData(lcMsg)
ENDWITH
```

4. Borren del evento Timer la línea:

```
*ThisForm.lbStates.AddItem(strEstado)
```

5. y en el evento DataArrival

```
LPARAMETERS bytestotal
LOCAL lcDatosRecibidos, lcCmd, lcTexto

lcDatosRecibidos = REPLICATE(CHR(0), BytesTotal)
This.GetData(@lcDatosRecibidos)
lcCmd = LEFT(lcDatosRecibidos, 3)
lcTexto = RIGHT(lcDatosRecibidos, BytesTotal - 3)
```

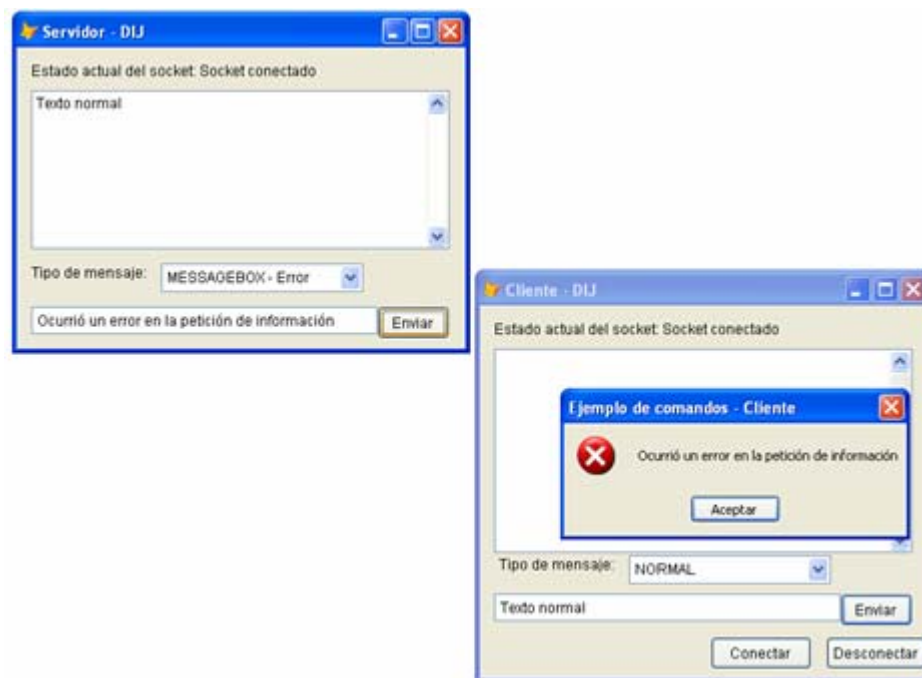
```

DO CASE
    CASE lcCmd == 'MSG'
        Thisform.lbStates.AddItem(lcTexto)
    OTHERWISE
        MESSAGEBOX(lcTexto, IIF(lcCmd = 'MSI', 64, 16), "Ejemplo de comandos
- Servidor")
ENDCASE

```

Solo un detalle en éste último punto, en el comando MessageBox en el evento DataArrival del cliente, cambien el título de la ventana (texto en rojo y negritas del código) por el del cliente para que distingan quien está recibiendo el mensaje.

Deberá verse algo como esto:



Observen los mensajes, el cliente envió un mensaje de tipo “Normal” que aparece en el listbox del servidor y el servidor envió un mensaje de tipo “MessageBox – Error” que se muestra en la ventana que está encima de la del cliente.

Recuerden que se deben conectar primero, lo digo por experiencia.

A ustedes les dejo que habiliten y/o deshabiliten los controles en función del estado de los sockets, en otras palabras si está desconectado el botón enviar (en ambos formulario) y el de

desconectar (en el caso del cliente) deberían estar deshabilitados y habilitarse al momento de establecerse la conexión.

Hasta aquí este capítulo...

Saludos y hasta el próximo capítulo

Denny Infante

denny_infante@hotmail.com

ADVERTENCIA: El código es proporcionado “como esta”, sin garantía de ningún tipo, implícita o explícita, ni me hago responsable por su mal uso y/o daños que se pudieran atribuir al uso del mismo.