

Visual Foxpro y WinSock I

Lo que nadie me dijo

Después de un tiempo sin escribir un gran amigo, Luis Rea, me invitó a colaborar con el sitio CodeFox (www.codefox.net) con artículos que sirvieran para la gran comunidad de Visual Foxpro y bueno aquí estamos iniciando con esta serie de artículos que tratarán de dar las bases a los foxeros para que exploten el potencial de los sockets.

Requisitos para entender el artículo.

Se supondrá que el lector tiene conocimientos en:

1. Programación utilizando Visual Foxpro
2. Uso de controles ActiveX
3. El control WinSock (propiedades, métodos y eventos)

Por dónde empezamos, esto es difícil de decidir, y sólo es por el título, que decirles que no les hayan contado: como establecer una conexión, ya, enviar texto a través de una conexión, ya, etc...ya.

Entonces voy a empezar por lo que a mí no me dijeron:

Los métodos no me hacen caso...

A más de alguno le ha de haber pasado lo que a mí cuando empecé a utilizar el control Winsock con VFP, bajé un ejemplo de algún lugar de internet y, según yo, lo seguí paso a paso, pero no me funcionó; después de algún tiempo investigando supe el por qué.

Ya han de haberse dado cuenta que el control Winsock tiene métodos y eventos con el mismo nombre, y la forma en que VFP maneja éste control en particular crea una serie de problemas de fácil solución pero problemas al fin.

Métodos	Eventos
Accept	
Bind	
Close	Close
Connect	Connect
ConnectionRequest	ConnectionRequest
DataArrival	DataArrival
Error	Error
GetData	
Listen	
PeekData	
SendComplete	SendComplete
SendData	
SendProgress	SendProgress

El problema surge al momento de llamar a los métodos, quién no intentado conectarse a otro Winsock sin ningún resultado (pues yo); esto es debido a que estamos llamando al evento en lugar del método, si no me creen hagan la prueba, pongan código en el evento Connect y hagan el intento de conectarse.... vieron los resultados, triste pero cierto pero no se preocupen, eso se arregla fácilmente.

Supongamos que oWs es el nombre de un objeto Winsock, para llamar a los métodos (no los eventos) se hace uso de la propiedad Object del control, por lo que la llamada a un método quedaría de la siguiente forma:

oWs.Object.NombreDelMétodo

Fácil verdad, éste pequeño detalle hizo que me tardara en poder hacer mi primera aplicación utilizando el Winsock; esto **solamente es necesario cuando** evento y método tiene el mismo

nombre, por ejemplo el método `SendData`, da el mismo resultado `oWs.SendData("")` que `oWs.Object.SendData("")`.

Qué pasa cuando el servidor desconecta a un cliente o viceversa

Con el que desconecta no pasa nada, pero el lado al que desconectaron queda en un estado de Cerrando (`sckClosing`) que no cambia a cerrado (al menos yo he esperado hasta 10 minutos y no lo ha hecho) por lo que el evento `OnClose` no se ejecuta; esto es de especial cuidado del lado del servidor ya que, si no hacen algo para que vuelva a “escuchar” las peticiones de los clientes no podrá establecer la conexión.

Como solucionarlo: una forma muy sencilla y práctica es colocar un objeto `Timer` que monitoree el estado del socket y al momento que entre al estado cerrando, cierre la conexión (`oWs.Object.Close()`).

La importancia de enviar una cadena de terminación

Esta, a mi ver, es una buena práctica porque facilita la vida cuando hablamos de transferencia de gran cantidad de información (como la transferencia de archivos), además, si el servidor es muy rápido en comparación con el cliente (muchas veces debido al tratamiento de la información recibida), el servidor puede enviar varias líneas de información y nosotros suponer que es solamente una ya que mientras no ejecutemos el `oWs.GetData(@DatosRecibidos)` el buffer se irá llenando, por ejemplo:

Servidor envía:	Buffer	Cliente	
		Acción	Información obtenida
Hello	Hello	GetData	Hello
Envío archivos	Envío archivos	Ninguna	
Archivo1.txt	Envío archivos Archivo1.txt	GetData	Envío archivos Archivo1.txt

Si en la segunda acción `GetData` del cliente se esperaba solamente “Envío archivos” entonces esto puede provocar un error.

Los sockets sólo pueden hacer una cosa a la vez y solamente con un cliente

Tan sencillo y la base para un sistema multiusuario, si escucho no me conecto, si estoy conectado no escucho peticiones de conexión, entonces que necesito para que un servidor acepte varias conexiones, esperen los próximos capítulos :o)

Creo que ésta información nos será suficiente para entender el por qué en los capítulos siguientes.

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.