

Visual Foxpro y WinSock II- B

Enviar mensajes a través de la red y mostrándolos usando el control MDMSNPopup

Por qué “II – B”; porque es una “extensión” del capítulo II.

Varios compañeros me preguntaban si era posible enviar mensajes a través de la red utilizando el MDMSNPopup; mi respuesta era (y sigue siendo) no, el control solamente muestra un mensaje no lo envía.

Lo que se puede hacer es crear una aplicación que envíe, y reciba, los mensajes y mostrarlos con el control.



Voy a suponer que ya leyeron los artículos anteriores (y tienen los formularios de ejemplo) para no entretenernos más de la cuenta; si no tienen los capítulos anteriores los pueden encontrar en www.codefox.net en la sección descargas y el control **MDMSNPopup** en www.portalfox.com.

Basándonos en el ejemplo del “chat” visto en el capítulo II de la serie; se harán las siguientes modificaciones:

1. Agregar la propiedad oPopup a ambos formularios (el cliente y el servidor)
2. En el evento INIT, de ambos formularios, cambiar el código por:

```
WITH Thisform
    DIMENSION .Comandos(4) &&Línea modificada
    .Comandos(1) = 'MSG'
    .Comandos(2) = 'MSI'
    .Comandos(3) = 'MSE'
    .Comandos(4) = 'MDP' &&Línea agregada

    WITH .cboTipoMensaje
        .AddItem('NORMAL')
        .AddItem('MESSAGEBOX - Información')
        .AddItem('MESSAGEBOX - Error')
        .AddItem('MDMSNPopup Control') &&Línea agregada
        .ListIndex = 1
    ENDWITH
ENDWITH

*** Líneas agregadas
*** Creamos una referencia al control MDMSNPopup
This.oPopup = CREATEOBJECT("MDMSNPopup.MyDMSNPopup")
*** Fin de líneas agregadas ***
```

3. Y en el evento OnDataArrival de ambos formularios cambiar el código por:

```
*** ActiveX Control Event ***
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
    *** Líneas agragadas ***
    CASE lcCmd = 'MDP'
        With ThisForm
            .oPopup.Titulo = "Mensaje recibido"
            .oPopup.Texto = lcTexto
            .oPopup.TiempoEspera = 10
            .opopup.Muestra()
        EndWith
    *** Fin de líneas agragadas ***
```

```

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

```

Y esas son todas las modificaciones; si comparan los códigos se darán cuenta de que lo que hicimos fue

- Agregar una nuevo comando “MDP”
- Agregar una nueva propiedad “oPopup” a los formularios
- Crear un objeto MDMSNPopup en la propiedad “oPopup”
- Agregar el código para mostrar el mensaje en el control

Utilizar el control MDMSNPopup para recibir mensajes

1. Seleccionen el tipo de mensaje a mostrar

The image shows a sequence of steps in Visual Studio to configure a message display control. It includes screenshots of the 'Cliente - IU' and 'Servidor - IU' windows, a callout for selecting the 'MDMSNPopup Control' from the 'Tipo de mensaje' dropdown, and a final screenshot of the 'Properties - Control' window showing the 'Message received' event.

Selección del tipo de mensaje

El mensaje se muestra utilizando el control MDMSNPopup

Hasta aquí es cómo mostrar los mensajes recibidos con el control MDMSNPopup; pero queda algo pendiente; en la práctica, quién tendrá el servidor y quién el cliente.

Para que puedan conectarse con alguien ustedes necesitan el cliente y la otra persona tiene que estar esperando una conexión (servidor); y para que se puedan conectar con uds tienen que estar esperando una conexión; en otras palabras, y la respuesta al dilema, necesitan tener en una misma aplicación el cliente (para conectarse con alguien) y el servidor (para que puedan conectarse con uds).

Si leyeron el artículo anterior cuando vimos la parte de monitorear el estado del socket se habrán dado cuenta que la única diferencia, de código, entre el cliente y el servidor era que el servido tenía una línea más (`Thisform.oWs.Listen()`).

En otras palabras la única diferencia entre el cliente y el servidor es que el servidor está esperando una conexión; por lo que se me ocurre que:

Mientras no se vaya a enviar un mensaje el socket estará esperando una conexión (servidor) y cuando necesiten enviar un mensaje primero dejan de escuchar, se conectan y envían el mensaje.

Aunque no es difícil de hacer se los voy a dejar a uds.

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.