Dominar el control Microsoft ListView

(5609 palabras totales en este texto)
(17785 lecturas) [1]

Dominar el control Microsoft ListView

Autor: Mike Lewis (www.ml-consult.co.uk [2])

Texto original:
-- Taming the Microsoft ListView control --
http://www.ml-consult.co.uk/foxst-28.htm [3]

Traducido por: Ana María Bisbé York (amby@telefonica.net [4])
Para: PortalFox (www.PortalFox.com [5])

Deje que el control SimpleList haga el duro trabajo de obtener listas de vistas. (List view)

La versión 1.1 de SimpleList apareció en Marzo 2003. Vea debajo un resumen de las nuevas posibilidades.

Si ha experimentado con los controles ActiveX que vienen con Visual FoxPro, probablemente haya pasado a través del control ListView. Este versátil control proporciona una forma atractiva para mostrar los datos en una lista, y ofrece amplias alternativas para los controles listbox (listas) y grids (cuadrículas), nativos de VFP.

Pero, mientras que ListView ofrece útiles ventajas, puede ser frustrantemente difícil de programar. A diferencia del control grid de VFP, no lo puede enlazar a una tabla simplemente estableciendo una propiedad, incluso una tarea tan mundana como es especificar un encabezado de columna requiere varias líneas de código.

Si le gusta la idea de tener un control ListView en su aplicación; pero verse libre de toda su complejidad, eche un vistazo a nuestro control SimpleList. Como sugiere su nombre, SimpleList toma el trabajo duro de utilizar y controlar el ListView. Es, en esencia una envoltura del control ListView, cuyo objetivo es proporcionar un método de llenar la lista basada en un simple cursor. De hecho, una vez que tenga su cursor preparado sólo necesitará tres líneas de comandos para ver aparecer su lista. (Observe que SimpleList requiere Visual FoxPro 7.0 o superior)

¿Qué es exactamente un ListView?

Para ver un ListView en acción, necesita ver solamente el panel derecho de la ventana del Explorador de Windows. Es el control ListView el que proporciona una lista de archivos y carpetas con sus ya familiares íconos grandes, íconos pequeños y ventana de detalles. El control ListView ActiveX puede proporcionar esta misma funcionalidad a sus aplicaciones.

Con una aplicación típica de bases de datos, probablemente esté interesado solamente en mostrar su lista en la ventana de detalles (también conocida como vista informes), ya que esta vista proporciona la vía más intuitiva de vista tabular de datos. Sin embargo, el ListView admite los otros tres tipos de vista de igual modo. La figura 1 muestra un ejemplo de vista de detalles, mientras la figura 2 muestra la vista con íconos grandes.

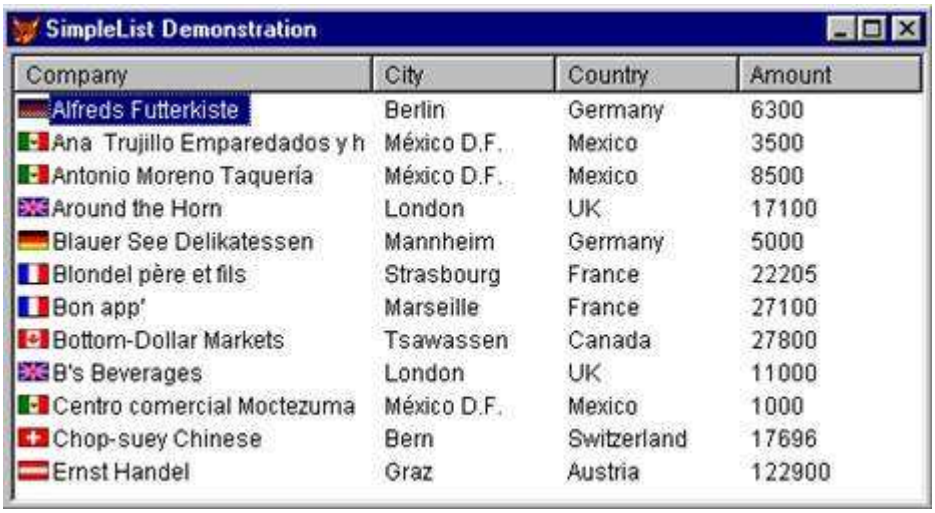

Figura 1: Un ListView con vista de detalle.

Figura 2: La misma lista en vista de íconos grandes.

Además de proporcionar las cuatro vistas diferentes, el control ListView ofrece otras muchas ventajas:

- En la vista de detalle, el usuario, puede mover y redimensionar columnas.
- También en esta vista, el usuario puede ordenar los datos haciendo clic en un encabezado de columna. La primera vez que haga Clic se ordenará de manera ascendente, la segunda vez de manera descendente.
- Cada elemento de la lista puede tener asociado un ícono o gráfico. Esto es esencial en las vistas con íconos grandes y pequeños, pero quizás menos útil en las listas con vistas de detalles, (aunque los íconos son completamente opcionales en las cuatro vistas).
- Puede adicionar una casilla de verificación (checkbox) a cada elemento. Esto proporciona una vía sencilla de permitir al usuario hacer selección múltiple de la lista.
- En la vista de detalles, puede dar a cada elemento su propio tooltip. Esto es útil para mostrar información adicional que de otra forma sería muy ancha para mostrar en la lista.
- Puede permitir al usuario editar el texto de un elemento, simplemente haciendo clic sobre el, en la forma que pueda renombrar un archivo en el Explorador de Windows).
- Las listas pueden mejorar su apariencia. Por ejemplo, puede escoger mostrar o no las líneas de cuadrícula en la vista de detalle, puede optar por encender el rastreo caliente.
- Se admite la característica de arrastrar y soltar. Los usuarios pueden arrastrar un elemento de la lista y soltarlo en

otro control o aplicación.

Limitaciones

Aunque el SimpleList admite más de las características del ListView, tiene algunas limitaciones:

- Si selecciona utilizar íconos, debe utilizar el mismo ícono para un elemento determinado en todas las vistas.
- En la vista de detalles, el usuario puede redimensionar las columnas; pero no puede reconfigurarlas.
- Existen algunas limitaciones al atrapar eventos. SimpleList le permite atrapar Clicks, DobleClicks y cambios de posición del elemento resaltado; pero no puede distinguir entre el Clic derecho y el izquierdo.

Comenzar

Ahora ya conoce lo que puede y lo que no puede hacer el SimpleList, el siguiente paso es bajar una copia

- Vea más abajo **¿Cómo bajar SimpleList?**

SimpleList es completamente gratis, y le invitamos a entregar copias a sus amigos y colegas. Le pedimos solamente que no elimine nuestras notas de Copyright. Hemos probado el SimpleList en nuestras aplicaciones; pero no tenemos la garantía de que vaya a trabajar en las suyas, así que debe asegurarse bien de que le funcione antes de confiar en sus resultados.

En el resto de este artículo le damos un resumen de las características del SimpleList. Nuestro objetivo es darle sólo unas pinceladas de lo que este control puede hacer. Al bajarse su copia, encontrará una versión ampliada de este documento que le permitirá ver todos los detalles de estas características

Nota de la traductora: *A partir de aquí quedan integrados en un único resumen dos artículos que el autor - Mike Lewis - escribió por separado. Debido que gran parte de la información detallada sobre los métodos y propiedades se encuentra en un archivo que se anexa al control SimpleList, el que no se tradujo, porque su contenido se incorporó a este.*

Al menos necesita conocer:

Para poner a funcionar el SimpleList, siga estos pasos:

1. Crear un cursor que almacene los datos que desea mostrar en la lista (aunque nosotros vamos a utilizar el término cursor base en este artículo, puede ser también una tabla o una vista). Como mínimo el cursor debe contener un campo de caracteres, son los datos de este campo los que se emplearán para llenar la lista.
2. Arrastre el control a un formulario.
3. Para llenar la lista, escriba un código similar a este (como es típico, debe ir en el evento INIT del control)

```
WITH THIS
  .cAlias = "Customer"
  .cData = "Company"
  .PopulateList
ENDWITH
```

Es todo lo que necesita. Como un ejemplo, si el cursor contiene los datos que se muestran a continuación, su formulario tendrá una apariencia parecida a la figura 3.

```
Company
-----------------------------------
Alfreds Futterkiste
Ana Trujillo
Emparedados y h
Antonio Moreno Taquer¡a
Around the Horn
```

```
Blauer See Delikatessen
Blondel p□re et fils
Bon app'
Bottom-Dollar Markets
B's Beverages
Centro comercial Moctezuma
Chop-suey Chinese
Ernst Handel
```

Figura 3.- Una vista de lista de una sola columna

Vamos a cubrir un poco las propiedades y métodos. Por favor, no se agobie por lo grande del artículo ni la cantidad de propiedades y métodos. SimpleList es realmente simple, y mucha de la información que sigue a continuación es opcional. Puede llamar al SimpleList solo con unas pocas líneas

cAlias - El alias del cursor (o tabla o vista) que contenga los datos para la lista. Al llamar a PopulateList, el cursor debe estar en la sesión de datos actual.

cData - Es una lista de los campos del cursor, separados por coma que se utilizan para llenar la lista. Necesita especificar al menos un campo, y este debe ser un campo de caracteres. El contenido de este campo se mostrará en las cuatro vistas.

Opcionalmente puede listar más campos en la propiedad cData. Si lo hace, su contenido se mostrará como columnas separadas en la vista de detalle; pero serán ignorados en las otras tres vistas. Estos campos adicionales pueden ser de cualquier tipo.

PopulateList (N.T: Populate List significa Llenar la lista) - Llame a este método para llenar la lista con los datos del cursor base. Cada vez que lo llame, la lista se configura con los valores predeterminados como: vista predeterminada, ancho de columnas, tipo de orden, etc. Si desea mantener esta configuración debe llamar en su lugar a:

RefreshList - Este método va a volver a llenar la lista; pero mantendrá la configuración actual del tipo de vista, ancho de columnas, tipo de orden, etc. Debe llamar a este método si los datos del cursor base cambian luego de haber llenado la lista.

Como otro ejemplo, considere este cursor:

```
Company                    City         Country  Amount
-------------------------- ------------ -------- -------
Alfreds Futterkiste        Berlin       Germany     6300
Ana Trujillo Emparedado..  México D.F.  Mexico      3500
Antonio Moreno Taquería    México D.F.  Mexico      8500
Around the Horn            London       UK         17100
```

```
Blauer See Delikatessen      Mannheim     Germany     5000
Blondel père et fils         Strasbourg   France     22205
Bon app'                     Marseille    France     27100
Bottom-Dollar Markets        Tsawassen    Canada     27800
B's Beverages                London       UK         11000
Centro comercial Moctez..    México D.F.  Mexico      1000
Chop-suey Chinese            Bern         Switzer    17696
Ernst Handel                 Graz         Austria   122900
```

El siguiente código creará la lista mostrada en la Figura 1. (pero sin los íconos):

```
WITH THIS
  .cAlias = "Customer"
  .cData = "Company,City,Country,Amount"
  .PopulateList
ENDWITH
```

Seleccionar la vista

De forma predeterminada, PopulateList muestra la lista en vista de detalles. Se pueden configurar las siguientes características:

nDefaultView - Utilice esta propiedad para especificar una vista inicial diferente (0 = íconos grandes, 1 = íconos pequeños, 2 = lista, 3 = detalle). Solo afectará la vista mostrada por PopulateView.

ChangeView - Llame a este método para cambiar a una vista diferente después de haber llenado la lista (pasando el correspondiente número de vista como parámetro).

nCurrentView Devuelve el número que identifica el tipo de vista. Es de sólo lectura.

Si desea que el usuario determine el tipo de vista que desea obtener, puede colocar el siguiente código en un botón del formulario. Cada vez que haga Clic, se mostrará la lista en el tipo de vista siguiente.

```
LOCAL lnView
WITH THISFORM.MySimpleList
  lnView = (.nCurrentView + 1) % 4
  .ChangeView(lnView)
ENDWITH
```

Columnas

De forma predeterminada los encabezados de columna van a mostrar los nombres de los campos del cursor. Las columnas serán alineadas a la izquierda y comenzarán todas con igual ancho. Puede variar este comportamiento como sigue:

cColumnHeaders Lista separada por comas de los encabezados que desea para cada columna. Los encabezados deben aparecer en el mismo orden en que aparecen los campos en cData. Si alguno de los encabezados de columnas está omitido, cColumnHeaders, el nombre del campo será utilizado en su lugar.

cColumnWidths - Lista separada por comas del ancho inicial, en pixels, de cada columna. Si se omite, las columnas saldrán todas con un tamaño equivalente. Puede establecer el ancho a cero, en tal caso la columna se ocultará (es útil para ordenar los datos no-visibles, vea más sobre esto en Ordenar, más adelante).

cColumnAlignments Lista separada por comas indicando la alineación de cada columna. Cada elemento de la lista será : L (Left - alineada a la izquierda), R (Right alineada a la derecha), C (Center alineada al centro) El valor predeterminado para cada columna es L. Sin embargo, la columna más a la izquierda está siempre alineada a la izquierda, sin importar cómo está fijada esta propiedad. Esta configuración afecta al encabezado de la columna y a los elementos de la misma.

Las tres propiedades siguientes se aplican sólo cuando la lista se llena inicialmente, no cuando se refresca.

lHideColumnHeaders Si es .T. (Verdadero) no se mostrarán los encabezados. Esto significa que el usuario no podrá redimensionar las columnas u ordenar los datos haciendo clic en el encabezado. La configuración tiene efecto inmediato.

cAdjustedColumnWidths Contiene una lista separada por comas de los anchos de columnas actual. Difiere de cColumnWidths en que es de solo lectura, y contiene la configuración de ancho actual en cualquier momento, incluso si han sido ajustado los anchos por el usuario (cColumnWidths contiene siempre los anchos iniciales). Puede utilizar cAdjustedColumnWidths para guardar los anchos actuales de forma que puedan ser restaurados la siguiente vez que el SimpleList sea inicializado.

Iconos

Si tiene en plan mostrar su lista con íconos grandes o pequeños, probablemente desee asignar iconos a las listas de ítems que es desde luego una de las prestaciones más importantes de estas vistas. Si asigna los íconos, se mostrarán también en las vistas listas y detalles. Pero esta es su decisión. Si no está interesado en íconos, puede ignorar este punto.

SimpleList le permite especificar un ícono único para que aparezca junto a cada elemento de la lista. De forma alternativa, puede especificar un ícono diferente para cada item. Para hacer esto, debe almacenar el nombre de archivo tipo icono en un campo del cursor.

cIconField Este es el nombre del campo en su cursor que contiene el nombre del archivo de la imagen de icono. Si puede asignar un icono diferente a cada elemento de la lista. Si el campo está vacío en el registro actual del cursor mostrará una imagen predeterminada. (vea más adelante)

cDefaultIcon El nombre del archivo de la imagen a mostrar por defecto. Esta será utilizada si cIconField está vacía, o si el artículo actual del cursor tiene ese campo vacío. Si ambas propiedades cIconField y cDefaultIcon están vacías no se mostrará ninguna imagen.

El archivo de imagen puede ser un archivo BMP, GIF, JPG, o ICO. Es su responsabilidad asegurarse que el archivo puede ser encontrado en tiempo de ejecución (SimpleList no lo chequea) (N.T: Si no existe el archivo de imagen en la ruta predeterminada, genera un error El archivo no existe , luego devuelve un error del objeto OLE - Código de excepción OLE dispatch 0 de ListImages: Index out of bounds y finalmente no se genera la lista.)

A diferencia del control ListView, nuestro control SimpleList no le permite asignar diferentes tamaños de íconos para diferentes vistas. Cualquier imagen que especifique, puede ser establecida como 48 x 48 pixels (para un ícono grande) o 16 x 16 pixels (para las otras tres vistas). Para mejores resultados, utilice un archivo ICO que contiene dos imágenes, una para cada tamaño requerido.

Tooltips

Utilice las siguientes propiedades si desea que aparezca un tooltip para cada elemento de la lista. El texto puede ser diferente para cada elemento. Esto es útil si desea permitir al usuario ver información adicional que de otra forma no cabe en la lista. Estos tooltips sólo aparecerán en la vista de detalle, y sólo cuando el ratón esté sobre la columna más a la izquierda. Un tooltip está limitado a 75 80 caracteres aproximadamente. (N.T: El campo del cursor base puede tener mayor tamaño; pero el texto que se mostrará en el tooltip será truncado, sin que se genere ningún error.)

cTooltipField El nombre del campo del cursor base que contiene los tooltips para los elementos. Este debe ser normalmente de caracteres; pero no es imprescindible.

lShowTooltips Determina si se mostrarán o no los tooltips. (Predeterminado .F.)

Ordenar

Sólo en la vista detalle, puede permitir opcionalmente al usuario ordenar la lista haciendo clic en el encabezado de

columna. Puede especificar además un tipo de orden y dirección para la lista, donde este es diferente desde el orden del cursor.

lSorted Determina si el usuario puede ordenar o no las columnas. Se establece .T. (Verdadero) si decide permitirle al usuario ordenar las columnas. Predeterminado .F.

nInitialSortColumn El número de las columnas en la que la lista está inicialmente ordenada. El valor predeterminado es 1, la cual es la situada más a la izquierda. Esta propiedad tiene efecto si lSorted (vea arriba) es .T. (Verdadero). Si no es así, la lista se mostrará en la misma secuencia del cursor.

nInitialSortDirection La dirección en la cual la lista está inicialmente sorteada (0 = ascendente, 1 = descendente, el predeterminado es 0). La configuración es sólo aplicable si lSorted (vea arriba) es .T.

cSortColumns Utilice esta propiedad si desea que la lista sea ordenada por una columna diferente a la que el usuario haya hecho clic. Esto es útil si el contenido de la columna no refleje su secuencia natural. Por ejemplo, si la columna contiene datos en formato DD-MM-YY (italian), ordenarlos en columnas puede simplemente ordenar los datos en secuencia ASCII, no como secuencia de datos.

Nota de la traductora: *Por ejemplo, si tiene las fechas: 25-01-01 / 02-03-01 / 30-10-02 / 02-01-03 / 20-03-03. El orden en que le aparecerán al hacer clic en el encabezado de columna será 02-01-03 / 02-03-01 / 20-03-03 / 25-01-01 / 30-10-02 Esto deja ver un orden no deseado en los elementos de la lista.*

Al utilizar cSortColumns, puede especificar una columna alternativa para ordenar. Esta puede contener los mismos datos en formato YYYYMMDD. Al establecer el ancho para esta columna alternativa en cero, el usuario no notará de su presencia.

cSortColumns debe contener una lista delimitada por comas de números enteros, uno por columna, en la secuencia que las columnas son listadas en cData. Si un dato dado es cero, entonces haciendo Clic sobre su encabezado de columna va a ordenar la lista en esa columna. Si esta contiene un valor positivo, entonces el entero representa el número de la columna a ser ordenada.

Considere este ejemplo:

```
SELECT OrdenID, OrdenFecha, OrdenCantidad ;
DTOS(OrdenFecha) AS OrdenYMD ;
FROM Ordenes INTO CURSOR ListaOrdenes
WITH THISFORM.MySimpleList
  .cAlias = "ListaOrdenes"
  .cData = "OrdenID,OrdenFecha,OrdenCantidad,OrdenYMD"
  .cColumnHeaders = "No. Orden,Fecha,Cantidad"
  .cColumnWidths = "200,80,100,0"
  .lSorted = .T.
  .cSortColumns = "0,4,0"
  .PopulateList
ENDWITH
```

En este caso va a mostrar una lista de 4 columnas, que contienen respectivamente el número de orden (OrdenID), la fecha de la orden (OrdenFecha - en el formato SET DATE actual), Cantidad (OrdenCantidad), y Fecha de la orden nuevamente (OrdenYMD - en formato YYYYMMDD).

Nota de la traductora: *Al Hacer Clic en el encabezado de la columna Fecha SimpleList advierte que es la columna 2 (FechaOrden) y que tiene un número 4 asociado, por tanto, busca la columna 4 (Orden YMD) y establece el orden por los valores de esta columna 4, esté la misma visible o no.*

Casillas de verificación (CheckBox)

Si ha deseado alguna vez permitir a los usuarios una lista (scrolling list) de casillas de verificación, puede encontrar esta

característica indispensable. Esencialmente, le permite mostrar una casilla de verificación para cada elemento de la lista. Esto proporciona una vía intuitiva de hacer selecciones múltiples, por ejemplo, para permitir al usuario seleccionar los clientes que deben aparecer en un informe, o los informes que desea enviar a un proceso de impresión por lotes. (Como ejemplo, vea la figura 4)

Para ir más lejos, puede marcar las casillas de verificación a campos lógicos en el cursor. El campo será actualizado para ver el estado actual de la casilla de verificación.

Las siguientes propiedades son relevantes si desea trabajar con casillas de verificación:

lCheckBoxes Es la propiedad que define si existirán casillas de verificación. Establezca a .T. (verdadero) si desea que se muestren las casillas para cada elemento de las columnas, en las cuatro vistas. El valor predeterminado es .F. (falso)

cCheckField Nombre del campo lógico del cursor base que se enlaza con cada elemento de la casilla. Si, para algún registro, el campo contiene .T., la correspondiente casilla aparecerá seleccionada. Si la propiedad está vacía, todas las casillas de verificación se mostrarán desmarcadas. (N.T: El nombre de este campo no debe aparecer en la lista de campos indicado en la propiedad cData, ya que en tal caso aparece además de la casilla de verificación marcada o no, el valor del contenido del campo, es decir .T. ó .F. lo cual no es deseado.)

lUpdateCheckbox De forma predeterminada, SimpleList no actualiza el cursor al ser seleccionadas las casillas. Establezca esta propiedad en .T. (Verdadero) si desea que el campo lógico del cursor (aquel que ha sido especificado en cCheckField) será actualizado con el estado de las casillas. Es su responsabilidad garantizar que el cursor sea Lectura Escritura.

Nota de la traductora: *Si el cursor base está establecido como de sólo lectura, al intentar modificar una casilla de verificación SimpleList devuelve un mensaje de error No se puede actualizar el cursor nombre del cursor base ya que es de sólo lectura.*

Para determinar si está marcada la casilla para un elemento determinado llame al método GetChecked (vea Recuperar información, más adelante en este documento).

Modificar el texto de un elemento

De forma predeterminada, su lista será de sólo lectura. Aunque, puede opcionalmente dar al usuario la habilidad para modificar el texto de un elemento. (en la vista de detalle, solo la columna más izquierda puede ser editada). Puede escoger además si escribir su propio código para procesar la modificación, hacer que el SimpleList actualice automáticamente el campo correspondiente al cursor.

Para permitir modificar, utilice estas propiedades:

lReadOnly – Establezca esta propiedad en .F. para permitir modificar (el predeterminado es .T.)

lUpdate – Establezca esta propiedad a .T. (Verdadero) para permitir modificar el elemento actual del cursor (el predeterminado es .F.) Es su responsabilidad garantizar que el cursor sea de Lectura – Escritura.

Nota de la traductora: *Si el cursor base está establecido como de sólo lectura y lUppdate = .T., al intentar guardar la modificación realizada en el texto de la primera columna a la izquierda, SimpleList devuelve un mensaje de error – No se puede actualizar el cursor – nombre del cursor base– ya que es de sólo lectura.–*

Si lReadOnly y lUpdate son ambas .F., el usuario será capaz de modificar el elemento; pero la modificación no será procesada en ningún momento, (N.T: Tampoco se emitirá mensaje de error alguno, en caso que el cursor base sea de sólo lectura).

En ese caso, puede escribir su propio código para procesar las modificaciones en el evento nativo AfterLabelEdit del control ListView. (N.T. Para ver su comportamiento, puede probar colocando este código en el método AfterLabelEdit del control oleList contenido en el SimpleList.)

```
=MESSAGEBOX('Ha cambiado el contenido del texto; pero no se desea actualizar el cursor')
```

Arrastrar y soltar

SimpleList admite arrastrar y soltar. Sin embargo, no está activa de forma predeterminada. Cuando está habilitado, permite al usuario arrastrar un elemento de la lista y soltarlo en algún control o aplicación que conozca como recibirlo. Por ejemplo, puede arrastrar el texto desde un cuadro de texto nativo de Visual FoxPro a un documento de Word. En cada caso el control o aplicación destino va a recibir una cadena de texto.

SimpleList no puede actuar por sí mismo como un destino para soltar. En otras palabras, puede arrastrar y soltar elementos desde la lista; pero no puede soltar nada dentro de el.

Para utilizar arrastrar y soltar, establezca las siguientes propiedades:

lDragEnabled – Establezca esta propiedad a .T. para permitir arrastrar y soltar.

nDragText – Define el texto que recibirá la aplicación o control destino, como sigue:

- 0 (predeterminado): El texto asociado con el elemento actual (en vista de detalle, es el texto de la columna más a la izquierda)
  entero positivo – El texto de la columna correspondiente para el elemento seleccionado (sin embargo las columnas sólo mostrarán en la vista de detalle, esto establece que funcione de la misma forma en las cuatro vistas).
- 1 (con signo negativo): El texto del tooltip del elemento seleccionado (si hay alguno)
- 2 (con signo negativo): El número de registro del elemento actualmente seleccionado en el cursor base.

Si desea que el control nativo de Visual FoxPro reciba el texto arrastrado, debe establecer la propiedad oleDropMode en 1.

Configurar embellecimiento

Puede modificar la apariencia y comportamiento de una lista de varias maneras. Por ejemplo, puede decidir si mostrará las líneas de la cuadrícula en la vista detalles y si permitirá o no el seguimiento. Puede utilizar estas características para modificar la apariencia de su lista:

lFullRowSelect - Si es .T. la fila entera será resaltada cuando el usuario haga clic sobre un item en la vista de detalle. Si es .F. (que es la predeterminada), sólo la columna más a la izquierda será iluminada.

lGridLines Establezca como .T. si desea líneas de cuadrícula aparezcan en la vista de detalle. El valor predeterminado es .F.

lHotTracking Si es .T. el puntero del ratón se mostrará con un dedo índice, y el texto del elemento será de sólo lectura (independientemente del valor de lReadOnly). Esto se aplica a las cuatro vistas. El valor predeterminado es .F.

La configuración anterior toma efecto inmediatamente (no necesita llamar a los métodos PopulateList, RefreshList o RedrawList)

Recuperar información

SimpleList proporciona una variedad de vías para obtener información sobre el contenido de la lista y qué elementos ha seleccionado el usuario. Puede recuperar el texto, número de registro, status de la casilla de verificación o cualquier elemento, y puede seleccionarlos programáticamente. Puede optar además por mantener el cursor en sincronía con la lista (de esta forma, cuando el usuario mueva el registro resaltado, el puntero de registro en el cursor se moverá automáticamente al registro correspondiente). Veamos cómo toma la información de la lista:

cSelectedText Esta propiedad contiene el texto del elemento actualmente seleccionado (o está en blanco si la lista está vacía).

nSelectedIndex Contiene el número de índice del elemento seleccionado actualmente (o cero si la lista está vacía). El índice es el número secuencial (empezando por 1) del elemento, en el tipo de orden actual (es decir, el número de índice va a variar cada vez que el usuario haga clic en el encabezado de la columna para ordenar la lista)

nSelectedRecno Esta propiedad contiene el número de registro en el cursor base correspondiente al elemento actualmente seleccionado (o cero si la lista está vacía). A diferencia de nSelectedIndex, no va a cambiar si la lista está ordenada.

nCount Contiene la cantidad de elementos de la lista.

Observe que estas cuatro propiedades son de sólo lectura.

SelectItem Llame a este método para seleccionar programáticamente un elemento. Pase como parámetro el número de índice del elemento dado. Para asegurarse de que el elemento será mostrado inmediatamente en la lista sin scrolling, pase .T. (verdadero) como segundo parámetro (este es el comportamiento predeterminado).

GetText Este método devuelve el texto del elemento del que se ha pasado el número de índice como parámetro.

GetRecno - Este método devuelve el número de registro asociado del elemento del que se ha pasado el número de índice como parámetro.

GetChecked Y este método devuelve el estado de la casilla de verificación (.T. o .F.) del elemento del que se ha pasado el número de índice como parámetro.

Estos tres últimos métodos le permiten mirar los elementos de la lista, definir acciones para cada uno de ellos. Por ejemplo, teniendo una lista como la mostrada en la figura 4, el código siguiente imprimirá los informes seleccionados:

```
LOCAL lnI, lcReport
WITH THISFORM.MySimpleList
  FOR lnI = 1 to .nCount
    IF .GetChecked(lnI)
      * Item is checked
      lcReport = .GetText(lnI)
      REPORT FORM "&lcReport" TO PRINTER NOCONSOLE
    ENDIF
  ENDFOR
ENDWITH
```

Sincronizar el cursor

De forma predeterminada, no existe conexión entre el elemento resaltado en la lista y el puntero de registro del cursor base. Sin embargo, puede cambiar este comportamiento configurando las siguientes propiedades:

lLink Establezca a .T. (Verdadero) para sincronizar el cursor y la lista. Cuando el usuario cambia el elemento resaltado en la lista SimpleList mueve el puntero al correspondiente registro del cursor base. Sin embargo lo contrario no es válido. Si mueve el puntero de registro en el cursor no se reflejará el cambio en el elemento resaltado en la lista.

Atrapar eventos

SimpleList le permite atrapar eventos Click y Doble Clic y cambiar de posición el elemento resaltado. Sin embargo no puede distinguir entre el Clic derecho e izquierdo o entre los Doble Clic derecho e izquierdo.

Click El evento Click del SimpleList se dispara cuando el usuario hace clic derecho o izquierdo en cualquier lugar del control (no necesariamente en un elemento). Para determinar que elemento, si alguno fue seleccionado en el momento de hacer Clic, utilice cSelectedText, nSelectedIndex o nSelectedRecno (descritos anteriormente)

DblClick De forma similar, este evento se dispara cuando el usuario hace doble clic sobre el control.

InteractiveChange Este evento se dispara cuando el usuario selecciona un elemento. Más específicamente, es disparado, cuando el usuario hace clic con el ratón o presiona alguna de las teclas de movimiento. Estas teclas son: arriba, Abajo, Izquierda, Derecha, Inicio, Fin, Av Pag, Re Pag (en cada caso, el evento disparado en dependencia del estado de las teclas Shift, Alt y Ctrl). No es estrictamente un evento InteractiveChange, ya que se dispara si el usuario mueve el cursor clave sin cambiar nada (por ejemplo si Arriba es presionado cuando el registro seleccionado está ya en el primer elemento).

Cosas que pueden ir mal

SimpleList realiza un control básico de errores al chequear los valores que coloca en sus propiedades. Si todo está bien, PopulateList y RefreshList devolverán cero. Si alguno de estos métodos detecta un error, su comportamiento dependerá de cómo haya establecido la siguiente propiedad:

lError - Si es .F (Falso - predeterminado), PopulateList y RefreshList retornará un número diferente de cero para indicar un error (vea debajo). Si es .T. (Verdadero), SimpleList pasa el error a su manipulador de errores o método Error, o al controlador de errores interno de VFP. El mensaje de error variará en dependencia de l naturaleza del error, pero simpre va a comenzar con SimpleList Error:

Los códigos de error retornados por PopulateList y RefreshList son como siguen:

1. cAlias contiene un nombre no válido de cursor, o no se puedo acceder al control.
2. cData contiene nombres de campos no válidos.
3. cData contiene nombres de campos que no existen en el cursor.
4. cIconField contiene nombre no válido de campo.
5. cTooltipField contiene nombre no válido de campo.
6. cCheckFile nombre no válido de campo.

Elevar el dominio

Si está interesado en aprender sobre el trabajo dentro del SimpleList, puede abrirlo desde el Diseñador de clases y estudiar su código.

Notará que el control SimpleList es en realidad un contenedor de VFP, que contiene un control ActiveX ListView. Todas las propiedades y métodos que describimos en este artículo pertenecen al contenedor y no al ListView. (N.T: Recuerde que

únicamente se hizo mención del método AfterLabelEdit nativo del ListView)

La razón para este proceder es sencilla. Si en algún momento decide utilizar un tipo diferente de control para mostrar sus listas, puede eliminar el ListView desde el contenedor y poner un nuevo control en su lugar. Es verdad que tendría que re-escribir la mayoría o todo el código del control SimpleList; pero la cuestión es no modificar las características externas del control su interfaz pública. Este es un buen ejemplo de separación de la interfaz de un componente desde su implementación siempre un objetivo deseado. Esto significa que por mucho que cambiemos SimpleList internamente, no afectará ninguno de los programas que lo utiliza.

Distribuir su aplicación

Para instalar SimpleList en sus aplicaciones, asegúrese de distribuir el control ActiveX Microsft ListView, que es el MSCOMCTL.OCX. Debe ser instalado en la carpera System o Sistemas del usuario y registrada como un control ActiveX. Si no está seguro de cómo hacerlo, revise la ayuda del InstallShield Express.

Nuevas características de la versión 1.1

Aquí tiene un resumen de las nuevas características del SimpleList versión 1.1. Todas estás descritas en la documentación del SimpleList y en este documento.

- Posibilidad de sincronizar el cursor con la lista (Propiedad Link)
- Posibilidad de asegurar que un elemento está visible sin desplazar la lista cuando se selecciona por programación. (Método SelectItem)
- Opción de controlar un error de VFP cuando el SimpleList detecte que una propiedad no válida ha sido establecida (Propiedad lError)
- Marcar las casillas de verificación con campos lógicos del cursor (estaba documentado previamente; pero no implementado).
- Más ayuda para modificar el texto de un elemento. (Propiedad lUpdate)
- Actualizar los campo lógicos marcados con las casillas de verificación.(Propiedad lUpdateCheckbox)
- Posibilidades de arrastrar y soltar.(Propiedades lDragEnabled y nDragText)
- El método Redraw no será requerido cuando ciertas propiedades de embellecimiento son cambiadas.
- Se han reparado problemas menores y errores en la documentación.

¿Cómo descargar el control SimpleList?

Haga Clic en el enlace que se encuentra debajo para descargar SIMPLELIST.ZIP. El archivo .zip contiene una biblioteca de clases, la que contiene a su vez, el control SimpleList. Contiene además la versión original de este artículo (en formato HTLM y texto); esto constituye la documentación entera para el control. El tamaño total de l descarga es 66 KB.

Descargar ahora [6]

Esperamos que el SimpleList sea de utilidad en su aplicación tanto como lo es para nosotros. Como siempre, estaremos agradecidos de recibir sus sugerencias (vea nuestra **Página de contactos** [7])

Mike Lewis Consultants Ltd. Septiembre 2002. Revisado Marzo 2003.

[**Volver a Mike Lewis** [8] | **Índice de secciones** [9]]

Enlaces

[1] http://www.portalfox.com/index.php?name=Sections&req=viewarticle&artid=39&allpages=1&theme=Printer
[2] http://www.ml-consult.co.uk
[3] http://www.ml-consult.co.uk/foxst-28.htm
[4] http://www.portalfox.com/mailto:(amby@telefonica.net
[5] http://www.portalfox.com
[6] http://www.portalfox.com//secciones/ml/simplelist/simplelist.zip

[7] http://www.ml-consult.demon.co.uk/MLCCntct.htm
[8] http://www.portalfox.com/index.php?name=Sections&req=listarticles&secid=11
[9] http://www.portalfox.com/index.php?name=Sections