MultiBase Cosmos

Notas a la versión 5.0

BASE 100, S.A.
www.base100.com

Índice

[NTCO50v1]

1. Implementaciones

1.1 Runtime

En esta versión se han implementado métodos en la clase SimpleControl y eventos que permiten personalizar un control List Box de tipo *string*, árbol y sql, de forma que es posible crear estilos propios para celdas y columnas, personalizar los componentes del control mediante un fichero que indicará dónde están los elementos gráficos que se desean utilizar, así como añadir funcionalidades tales como la posibilidad de buscar cadenas de caracteres en todas las celdas de una lista o mostrar en una columna el resultado de una operación aritmética y lógica. Así mismo, también es posible añadir filtros en las cabeceras de las columnas, agrupar columnas bajo un mismo título y crear grupos y agregados.

Además, el usuario podrá seleccionar las columnas que formarán los grupos y las funciones de los agregados, seleccionar las columnas por las que desea ordenar y modificar el orden de éstas una vez elegidas. La apariencia de la lista después de crear grupos será la de una lista en árbol en la que se permitirá abrir y cerrar los nodos.

2. Mejoras

2.1 Runtime

- Se ha modificado el Runtime para que soporte la versión 3.4 de OpenOffice y permita la exportación a PDF, HTML y ODS.
- Posibilidad de mostrar/ocultar las líneas horizontales de separación de filas en controles lista en árbol.
- Posibilidad de asignar dinámicamente las propiedades Font, Foreground y Background a un control de las páginas de impresión. Para ello debe utilizarse el método SetProperty de la clase SimpleControl. Este valor se podrá consultar con el método GetProperty de la misma clase.
- Se ha ampliado el tamaño máximo del texto que retorna el método MsgText.
- Se han realizado las modificaciones necesarias para poder ejecutar aplicaciones desarrolladas en Cosmos sobre Windows 8.
- Se ha modificado el Runtime para que en la exportación a Excel desde el Preview de Cosmos se exporten las máscaras utilizadas en el diseño del listado.

3. Variables de entorno

COSMOSLISTSKIN

Esta variable de entorno indica al Runtime el path del fichero con los elementos gráficos que se emplearán para personalizar los controles tipo lista de la aplicación. Si se define esta variable todos los controles tipo lista se mostrarán con esa apariencia.

DRAWTREELISTNODELINES

Indica al Runtime si se dibujarán o no las líneas que unen los nodos de las listas en árbol.

Los valores posibles son:

TRUE o YES	Se dibujarán las líneas. Este es el valor por defecto.
FALSE o NO	No se dibujarán las líneas.

Si el tema seleccionado no es Windows XP o el valor de la variable de entorno COSMOSXPTHEMESTYLE es FALSE, no se dibujarán las líneas, independientemente del valor que se le haya asignado a variable.

Esta variable de entorno se define en la sección Environment del fichero de configuración del proyecto o en el fichero “cosmos.ini” de Cosmos. No se podrán modificar los valores de la variable en ejecución.

DRAWTREELISTLINES

Indica si se dibujarán o no las líneas que separan las filas en las listas en árbol.

Los valores posibles son:

TRUE o YES	Se dibujarán las líneas.
FALSE o NO	No se dibujarán las líneas. Este es el valor por defecto.

Esta variable de entorno se define en la sección Environment del fichero de configuración del proyecto o en el fichero “cosmos.ini” de Cosmos. No se podrán modificar los valores de la variable en ejecución.

ENABLEMENUOPTIONONENABLECOMMAND

Variable de entorno que indica si se habilitará o no la opción de menú asociada a un comando cuando éste se habilite. Se debe definir en el fichero de configuración del proyecto o en el fichero de configuración de Cosmos.

Los valores posibles son:

YES o TRUE	Cuando se habilita un comando, también se habilitará la opción de menú asociada al mismo.
NO o FALSE	Cuando se habilita un comando no habilitará la opción de menú asociada al mismo. Se tendrá que habilitar manualmente. Éste es su valor por defecto

ZOOMFORM

Indica el porcentaje de zoom que se desea aplicar en las pantallas, controles y fuentes de una aplicación.

SHOWCOSMOSTREEWALKDIALOG

Esta variable de entorno indica al Runtime de Cosmos que, al ejecutar el método TreeWalk, en lugar de mostrar el cuadro de diálogo de selección de fichero estándar de Windows para la versión del sistema operativo donde se está ejecutando la aplicación (e implementado por la API de dicho sistema), muestre un cuadro de diálogo con funcionalidad similar y aspecto idéntico en todas las versiones de Windows.

Los valores posibles son:

YES	Se llamará al nuevo cuadro de diálogo.
NO	El método funcionará como en las en versiones anteriores.

Esta variable de entorno se define en la sección Environment del fichero de configuración del proyecto o en el fichero “cosmos.ini” de Cosmos. No se podrán modificar los valores de la variable en ejecución.

SHOWCOSMOSTREEWALKNETWORKDRIVES

Indica si se muestran o no las carpetas compartidas en la red cuando en la llamada al método TreeWalk se utiliza el nuevo cuadro de diálogo en lugar de la función de la API de Windows.

Los valores posibles son: YES y NO, siendo este último su valor por defecto.

Esta variable debe definirse en la sección Environment del fichero Cosmos.ini o del fichero INI del proyecto.

SHOWCOSMOSTREEWALKLOCALDRIVES

Esta variable de entorno indica si se mostrarán o no las unidades locales cuando el método TreeWalk utilice el cuadro de diálogo implementado en la versión 5.0 de Cosmos en lugar de la API de Windows para la elección de un fichero.

Los valores posibles son: YES y NO. El valor por defecto es YES.

Esta variable debe definirse en la sección Environment del fichero Cosmos.ini o del fichero INI del proyecto.

4. Métodos

4.1 Métodos de la Clase Module

FtpPutFileEx

Permite enviar un fichero a un servidor FTP. A diferencia del FtpPutFile, este método no utiliza las funciones de la API de Windows.

Sintaxis:

```
FtpPutFileEx (remoteHost as Char ,remoteHostPort as Integer, remoteHostUser as
Char,remoteHostPassword as Char, remote-HostDirectory as Char, remoteHostFile
as Char,localFile as Char, showDialog as Boolean, allowCancelUpload as Boo-
lean) return Integer
```

Parámetros:

remoteHost	IP o nombre del Host.
remoteHostPort	Número del puerto del FTP. Si se indica un número negativo o el cero el número del puerto será el 21.
remoteHostUser	Usuario.
remoteHostPassword	Password.
remoteHostDirectory	Ruta completa del directorio destino.
RemoteHostFile	Nombre del fichero.
LocalFile	Ruta completa del fichero que se va a enviar.
ShowDialog	Indica si se mostrará la ventana de progreso o no.
AllowCancelUpload	Indica si se va a dar al usuario la posibilidad de cancelar la operación.

Retorna los siguientes códigos:

0	Si el fichero se ha enviado correctamente.
-1	No existe fichero origen o no se puede acceder.
-2	No se puede conectar a la red.
-3	No se encuentra servidor o el número de puerto.
-4	No se puede acceder al directorio destino.
-5	Error en el envío del fichero.
-6	Abortado por el usuario.
-7	Timeout.

-8	Usuario/contraseña incorrectos.
-9	Parámetros host, remotehostfile y localfile no pueden ser nulos.

GetUrlFileEx

Permite descargar un fichero de la URL que se le indique. A diferencia del GetUrlFile, no utiliza las funciones de la API de Windows.

Sintaxis:

```
GetUrlFileEx(remoteFile as Char ,localFile as Char ,showDialog as Boolean
,allowCancelDownload as Boolean) return Boolean
```

Parámetros:

remoteFile	Indica el nombre del fichero.
LocalFile	Ruta completa, incluido el nombre del fichero que se descarga. El directorio donde se va a descargar el fichero debe haberse creado previamente.
ShowDialog	Indica si se mostrará la ventana de progreso o no.
AllowCancelDownload	Indica si se va a dar al usuario la posibilidad de cancelar la operación.

Retorna;

TRUE si ha conseguido descargar el fichero. FALSE en caso contrario.

ShowColorDialog

Muestra el cuadro de diálogo estándar de Windows de selección de color.

Sintaxis:

```
ShowColorDialog(parentWindow as Integer ,defaultColor as Integer ,
VAR retColor as Integer) return boolean
```

Parámetros:

parentWindows	Manejador de la ventana padre. Acepta el valor null.
defaultColor	Indica el color por defecto que esté seleccionado en el cuadro de diálogo.
retColor	Color seleccionado.

Retorna:

TRUE	Si se pulsa el botón aceptar en el cuadro de diálogo.
FALSE	Si se pulsa el botón cancelar.

ShowFontDialog

Muestra el cuadro de diálogo estándar de Windows que permite seleccionar una de las fuentes de Windows instaladas en el PC.

Sintaxis:

```
ShowFontDialog (parentWindow as Integer ,font as Char default "")
return char
```

Parámetros:

parentWindow	Manejador de la ventana padre. Acepta el valor null.
font	Fuente seleccionada por defecto.

Retorna:

Un string con la fuente seleccionada.

Un ejemplo del valor de retorno es:

Courier New;italic;Size=24

4.2 Métodos de la Clase SimpleControl

Dentro de este apartado, siempre que exista una referencia a un control lista o a una lista, se aplicará sólo a los controles List Box de tipo columns list (string, sql y/o árbol). Si algún método se puede aplicar a otro tipo de listas se indicará explícitamente.

AllowColumnHeaderFilter

Este método indica al Runtime de Cosmos que en la cabecera de la columna especificada se mostrará una flecha que, al ser pulsada, desplegará una lista para seleccionar el valor por el que se desea filtrar. Cuando se aplique este método, en la cabecera de la columna aparecerá una flecha, tal y como se muestra en la siguiente imagen:

Al pulsar sobre la fecha se habilitará un control tipo drop edit. En la lista desplegable se muestran los filtros que se enumeran a continuación:

- No vacías. Mostrar todas las filas con valores no nulos.
- Vacías. Mostrar todas las filas con valores nulos para las columnas.
- Todas. Mostrar todas filas.
- Lista con todos los valores únicos para esa columna.

En el campo de edición se podrán indicar los siguientes filtros:

- Para las columnas numéricas se pueden aplicar los siguiente operadores: “<”, “>”, “>=” y “!=”.
- Para las columnas alfanuméricas se podrá indicar el metacarácter “%” para filtrar con la cláusula like.

Sintaxis:

```
AllowColumnHeaderFilter(column as Smallint ,show as Boolean ,maxHeaderRows as
Integer default -1 caseSensitive as Boolean default FALSE).
```

Parámetros:

Column	Columna sobre la que se aplicará la acción.
Show	Indica si se permitirá o no aplicar un filtro sobre la columna que se pasa como parámetro.
maxHeaderRows	Indica el número máximo de valores únicos de filtrado que contendrá el control drop edit. Por defecto, mostrará todas las filas.
caseSensitive	Permite indicar si la búsqueda distinguirá o no entre mayúsculas y minúsculas.

> ***IMPORTANTE:*** *Si el valor que se indica en el parámetro casesensitive es FALSE y la lista es de tipo Sql, en instalaciones cliente-servidor la versión de motor deberá ser al menos la 3.4 0.2 en la versión para Windows. Para sistemas operativos UNIX/LINUX las releases de las versiones del motor tendrán que ser igual o superiores a la 3.2 0.2, 3.4 0.2 y 3.6 0.2.*

AllowExportListCellStyles

Este método permite indicar si la exportación de las listas Sql, String y Árbol con los métodos ExportToExcel, ExportToHTML, ExportToODS y ExportToPDF incluirá o no los estilos de las celdas y columnas creados con el método CreateListCellStyle. Por defecto, la exportación sí incluye los estilos creados con CreateListCellStyle.

Sintaxis:

```
AllowExportListCellStyles(allow as Boolean)
```

Parámetros:

allow	Indica si se exportan los estilos o no.

AlternateBackColor

Permite aplicar bandas de color a las filas de una lista alternando dos colores.

Sintaxis:

```
AlternateBackColor(BackColor1 as Integer ,BackColor2 as Integer ,nAlternate as
Integer)
```

Parámetros:

BackColor1	Valor RGB del color con el que se pintará la primera banda.
BackColor2	Valor RGB del color con el que se pintará la segunda banda.
nAlternate	Frecuencia con la que se va mostrar el segundo color en la lista.

BackupListRow

Este método hace una copia de los datos de la fila que se le indican como parámetro. Este valor se almacena en memoria y podrá ser consultado con el método GetBackupListRow y restaurado con el método RestoreBackupListRow. Este método es aplicable a List Box de tipo String y Árbol.

Sintaxis:

```
BackupListRow(row as Integer)
```

Parámetros:

row	Identificador de la fila.

ComputeListColumnTotals

Este método calculará los totales sobre las columnas que se han indicado con el método TotalizeColumn.

Sintaxis:

```
ComputeListColumnTotals()
```

CreateListCellStyle

Permite crear un estilo de visualización que posteriormente se aplicará a celdas individuales o a columnas de la lista.

Sintaxis:

```
CreateListCellStyle(font as Char ,foreColor as Integer ,backColor as Integer
,align as Smallint, icon as Smallint) return integer
```

Parámetros:

Font	Indica los atributos de formato para el texto. Los atributos que se pueden indicar para definir una fuente son: nombre, subrayado, rayado, negrita y tamaño. Los atributos deben ir separados por punto y coma.
ForeColor	Color del texto. Si el valor que se asigna a este parámetro es -1, al aplicar el estilo a la celda o a la columna no se modificará el color de texto de la misma.
BackColor	Color de fondo. Si el valor que se asigna a este parámetro es -1, al aplicar el estilo a la celda o a la columna no se modificará el color de fondo de la misma.
Align	Indica el tipo de alineación que se va a dar a los textos de la celda. Sus valores posibles son: 0. Texto alineado a la izquierda. 1. Texto centrado. 2. Texto alineado a la derecha.
Icon	Parámetro opcional que indica el icono que se dibujará en las celdas a las que se le asigne el estilo.

Retorna:

Un identificador del estilo creado para poder aplicarlo posteriormente a celdas individuales o a columnas de la lista.

FindColIntoString

Este método retorna una cadena de caracteres con los identificadores de las filas (separados por el carácter "|"), cuyo valor coincide con el literal indicado en el parámetro «text». La búsqueda comienza en el elemento de índice «indexStart». De esta manera será posible obtener todas las ocurrencias de un texto en una columna de la lista en una sola llamada a la función.

Sintaxis:

```
FindColIntoString(column as Smallint ,text as Char ,indexStart as Integer de-
fault 0 ,exact as Boolean default TRUE) return Char
```

Parámetros:

column	Identificador de la columna en la que se desea realizar la búsqueda.
text	Literal a buscar.
indexStart	índice del elemento de la lista a partir del cual comenzará la búsqueda. Su valor por defecto es cero, lo que indica que la búsqueda se realizará desde el primer elemento.
exact	Si es TRUE busca filas cuyo valor coincide exactamente con el conjunto de caracteres especificado. Si es FALSE busca filas cuyo valor empiece por la cadena de caracteres especificada.

Retorna:

Un string con los identificadores de todas las filas que cumplan la condición. Si no se ha encontrado ningún valor retorna null.

GetBakupListRow

Este método permite consultar los datos de la fila que se almacenaron en memoria con el método BackupListRow.

Sintaxis:

```
GetBakupListRow(row as integer) return Char
```

Parámetros:

row	Identificador de la fila

Retorna:

Un string con valor de la fila almacenado en memoria.

GetListCellSyles

Este método permite consultar el número de estilos, el ID de los estilos y las condiciones utilizadas para aplicar estilos a una celda.

Sintaxis:

```
GetListCellStyles(row as Integer ,col as Smallint ,VAR idxArray as
Array OF Numeric ,VAR conditionsArray as Array OF Char) return integer
```

Parámetros:

row	Identificador de fila.
col	Identificador de la columna.
idxArray	En este parámetro se retornarán los identificadores de los estilos asignados a las celdas.
ConditionArray	En este parámetro se retornarán las condiciones de los estilos asignados a la celda.

Retorna:

El número de estilos de la celda. Este valor indicará la dimensión con la que se han de definir los arrays que se pasan como tercer y cuarto parámetros al método.

GetListColumnStyles

Este método permite consultar el número de estilos, el ID de los estilos y las condiciones utilizadas para aplicar estilos a una columna.

Sintaxis:

```
GetListColumnStyles (col as Smallint ,VAR idxArray as Array OF Numeric ,VAR
conditionsArray as Array OF Char) return integer
```

Parámetros:

col	Identificador de la columna.
idxArray	En este parámetro se retornarán los identificadores de los estilos asignados a la columna.
ConditionArray	En este parámetro se retornarán las condiciones de los estilos asignados a la columna.

Retorna:

El número de estilos de la columna. Este valor indicará la dimensión con la que se han de definir los arrays que se pasan como tercer y cuarto parámetros al método.

GetStrListFilterBar

Este método permite consultar el texto asignado a los filtros creados con el método SetStrListFilterBar o bien el que el usuario haya introducido en la barra de filtrado mostrada en la ejecución del método ShowListFilterBar.

Sintaxis:

```
GetStrListFilterBar() return char
```

Retorna:

Un string con el texto.

GroupListColumns

Permite crear una nueva cabecera bajo la cual se agruparán varias columnas. La nueva cabecera tendrá un título que se pasará como parámetro al método junto con las columnas que formarán el grupo. Las columnas que formen parte de uno de estos grupos no podrán moverse fuera de él.

El máximo de columnas que se pueden agrupar es 10.

Sintaxis:

```
GroupListColumns(description as Char ,VAR argLst as ArgList OF Smallint) re-
turn integer.
```

Parámetros:

Description	Texto descriptivo del grupo de columnas.
Arglist	Indica la lista de columnas que se desean agrupar.

Retorna:

Retorna un integer cuyos posibles valores son:

-1	Alguna(s) columna(s) se ha indicado más de una vez. Código de error.
-2	Alguna(s) columna(s) pasada como parámetro pertenece(n) a otro grupo. Código de error.
-3	Indica que al menos una de las columnas que se ha pasado como parámetro no está en posición adyacente al resto de las columnas Código de error.
-4	Indica que se ha superado el máximo de columnas permitidas en un grupo. Código de error.
-5	Indica que el número de columnas pasado como parámetro es mayor al número de columnas que hay en la lista. Código de error.

Número entero positivo que identificará el grupo creado.

IsCheckedListCell

Este método permite consultar el valor de una celda de la lista en una columna tipo check. Es aplicable a controles List Box de tipo columns list (string y sql).

Sintaxis:

```
IsCheckedListCell(row as Integer ,col as Smallint ,VAR ischecked as Boolean)
```

Parámetros:

row	Identificador de la fila.

col	Identificador de la columna.
ischecked	Valor del control check. TRUE si su valor es 1. FALSE en caso contrario.

ListSetSkin

Este método indica al Runtime el fichero que contiene la descripción de los elementos gráficos que se utilizarán para cambiar la apariencia del control List Box de tipo columns list (string y sql).

Sintaxis:

```
ListSetSkin(skinFile as Char) return boolean
```

Parámetros:

SkinFile	Ruta del fichero.

Retorna:

TRUE si se ha cargado la lista de elementos y FALSE si no.

Los elementos gráficos que se pueden indicar en este fichero son:

SKIN_LIST_ITEM_HEADER_PART_BITMAP_FOCUSED
Indica el bitmap de la cabecera de la columna cuando el cursor del ratón está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_BITMAP_NORMAL
Indica el bitmap de la cabecera de la columna cuando el cursor del ratón no está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_BITMAP_DOWN
Indica el bitmap de la cabecera de la columna cuando se mantiene pulsado sobre ella el botón del ratón.

SKIN_LIST_ITEM_HEADER_PART_BORDER_FOCUSED
Indica el color del borde de la cabecera de la columna cuando el cursor del ratón está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_BORDER_NORMAL
Indica el color del borde de la cabecera de la columna cuando el cursor del ratón no está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_BORDER_DOWN
Indica el color del borde de la cabecera de la columna cuando se mantiene pulsado sobre ella el botón del ratón.

SKIN_LIST_ITEM_HEADER_PART_TEXT_FOCUSED
Indica el color del texto de la cabecera de la columna cuando el cursor del ratón está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_TEXT_NORMAL
Indica el color del texto de la cabecera de la columna cuando el cursor del ratón no está sobre ella.

SKIN_LIST_ITEM_HEADER_PART_TEXT_DOWN
Indica el color del texto de la cabecera de la columna cuando se mantiene pulsado sobre ella el botón del ratón.

SKIN_LIST_ITEM_SELECTED_ROW_PART_BITMAP_FOCUSED
Indica el bitmap de la fila seleccionada en la lista.

SKIN_LIST_ITEM_SELECTED_ROW_PART_BORDER_FOCUSED
Indica el color del borde de la fila seleccionada en la lista.

SKIN_LIST_ITEM_SELECTED_ROW_PART_TEXT_FOCUSED
Indica el color del texto de la fila seleccionada en la lista.

SKIN_LIST_ITEM_SCROLLBAR_LEFT_ARROW_NORMAL
Indica el bitmap que se aplicará sobre la flecha desplazamiento a la izquierda en la barra de scroll horizontal de la lista.

SKIN_LIST_ITEM_SCROLLBAR_RIGHT_ARROW_NORMAL
Indica el bitmap que se aplicará sobre la flecha de desplazamiento a la derecha en la barra de scroll horizontal de la lista.

SKIN_LIST_ITEM_SCROLLBAR_UP_ARROW_NORMAL
Indica el bitmap que se aplicará sobre la flecha de desplazamiento hacia arriba en la barra de scroll vertical de la lista.

SKIN_LIST_ITEM_SCROLLBAR_DOWN_ARROW_NORMAL
Indica el bitmap que se aplicará sobre la flecha de desplazamiento hacia abajo en la barra de scroll vertical de la lista.

SKIN_LIST_ITEM_HORIZONTAL_SCROLLBAR_THUMB_NORMAL
Indica el bitmap que se aplicará sobre el botón de arrastre de la barra de scroll horizontal (thumb) de la lista.

SKIN_LIST_ITEM_VERTICAL_SCROLLBAR_THUMB_NORMAL
Indica el bitmap que se aplicará sobre el sobre el botón de arrastre en la barra de scroll vertical (thumb) de la lista.

SKIN_LIST_ITEM_HORIZONTAL_BITMAP
Indica el bitmap que se aplicará sobre la barra de scroll horizontal.

SKIN_LIST_ITEM_VERTICAL_BITMAP
Indica el bitmap que se aplicará sobre la barra de scroll vertical.

SKIN_LIST_ITEM_SORTBAR_BACKGROUND_PART_BITMAP
Indica el bitmap que se aplicará sobre la barra de ordenación de columnas.

SKIN_LIST_ITEM_HEADER_FILTER_DROPDOWN_ARROW_BITMAP
Indica el bitmap que se mostrará como flecha en los controles tipo lista en la cabecera de las columnas sobre las que se permite aplicar filtros.

Los bitmaps referidos en el fichero que se utiliza para la definición del skin deben estar en el mismo directorio.

RemoveColumnTotals

Este método permite eliminar los totales asignados a una columna con el método TotalizeColumn.

Sintaxis:

```
RemoveColumnTotals(col as Smallint).
```

Parámetros:

col	Identificador de la columna.

RemoveGroupListColumns

Permite eliminar la agrupación de columnas bajo una misma cabecera cuando la agrupación ha sido creada con el método GroupListColumns.

Sintaxis:

```
RemoveGroupListColumns(groupId as Integer).
```

Parámetros:

GroupId	Identificador del grupo retornado por el método GroupListColumns.

Retorna:

TRUE si se ha eliminado la columna y FALSE si se ha producido un error y no se ha podido eliminar la columna.

RemoveListGroupTreeView

Este método indica al control lista que muestre los registros de las listas en controles List Box de tipo columns list (string y sql) en su vista clásica y no en forma de árbol tras la llamada al método ShowMultiColumnGroupDlg.

Sintaxis:

```
RemoveListGroupTreeView()
```

ResetListCellStyles

Este método reinicia los estilos asignados a una celda de un control List Box de tipo Columns list (Strings, Árbol y Sql) con los métodos SetListCellStyle y SetListCellConditionalStyle. No reinicia los estilos asignados a las columnas.

Sintaxis:

```
ResetListCellStyles(row as integer, col as Smallint) return boolean
```

Parámetros:

row	Identificador de la fila.
col	Identicador de la columna.

Retorna:

TRUE si se ha podido realizar la acción y FALSE en caso contrario.

ResetListColumnStyles

Este método reinicia los estilos asignados a la columna de un control List Box de tipo Columns list (Strings, Árbol y Sql) con los métodos SetListColumnStyle y SetListColumnConditionalStyle.

Sintaxis:

```
ResetListColumnStyles(column as Smallint) return boolean
```

Parámetros:

column	Identicador de la columna.

Retorna:

TRUE si se ha podido realizar la acción y FALSE en caso contrario.

ResetListFilterBar

Este método reinicia el valor utilizado en el método SetStrLisFilterBar para filtrar los datos que se muestran en la lista.

Sintaxis:

```
ResetListFilterBar()
```

RestoreBackupListRow

Este método restaura el valor almacenado en memoria con el método BackupListRow en la fila indicada.

Sintaxis:

```
RestoreBackupListRow(row as integer)
```

Parámetros:

row	Identificador de la fila.

SetCheckListCell

Este método permite modificar el valor del control check en una celda de la lista de un List Box de tipo columns list (string y sql).

Sintaxis:

```
SetCheckListCell(row as Integer ,col as Smallint ,check as Boolean)
```

Parámetros:

row	Identificador de la fila.
col	Identificador de la columna.
check	Valor que se asignará al control. Los valores posibles son: TRUE y FALSE. Si se pasa TRUE, el valor de la celda será 1; si se pasa FALSE, el valor será 0.

SetComputedColumn

Este método indica que los valores de una columna numérica de un control List Box de tipo columns list (string y árbol) serán el resultado de una serie de operaciones lógicas/aritméticas que se realizarán cuando se añadan o modifiquen las filas de la misma. Los operandos podrán ser columnas de la misma fila, pero no se podrá utilizar el resultado de esta columna como operando de otra columna calculada.

Sintaxis:

```
SetComputedColumn(column as Smallint ,formula as Char) retun booelan
```

Parámetros:

Column	Identificador de la columna sobre la que asignará el resultado de la operación.
Fórmula	Operación que se va a realizar.

Retorna:

Boolean si se ha podido indicar la columna calculada.

NOTAS:

Si los operandos son columnas la notación que debe indicarse es:

```
$<n° de columna>$
```

La llamada a este método se debe realizar antes de cargar la lista.

SetListCellStyle

Este método permite aplicar un estilo a una celda. El estilo debe haber sido definido previamente con el método CreateListCellStyle. El estilo que se define para una celda prevalece sobre el que se define para una columna.

El estilo de la celda debe asignarse después de que la celda haya adquirido su valor. Es decir, no es posible asignar un estilo a una celda de una fila que no exista.

Sintaxis:

```
SetListCellStyle(row as Integer ,col as Smallint ,style as Integer) return
boolean.
```

Parámetros:

Row	Índice de la fila.
Col	Identificador de la columna.
Style	Identificador del formato que se va a aplicar a fila. Este identificador es el resultado de una llamada al método CreateListCellStyle.

Retorna:

Un booleano que indicará si ha habido error o no en la llamada al método.

SetListCellConditionalStyle

Este método permite aplicar un estilo a una celda si ésta cumple la condición que se indicará como parámetro. El estilo debe haber sido definido previamente con el método CreateListCellStyle. El estilo que se define para una celda prevalece sobre el que se define para una columna.

El estilo de la celda debe asignarse después de que la celda haya adquirido su valor. Es decir, no es posible asignar un estilo a una celda de una fila que no exista.

En el caso de controles List Box de tipo Sql, si se modifica el orden en que estaban los datos cuando se asignó el estilo, éste no se respetará, es decir, los estilos en estas listas se asignan a una fila/columna determinada, y si el dato de la celda cambia el estilo no varía.

Sintaxis:

```
SetListCellConditionalStyle(row as Integer ,col as Smallint ,style as Integer,
condition as Char).
```

Parámetros:

Row	Índice de la fila.
Col	Identificador de la columna.
Style	Identificador del formato que se va a aplicar a la fila. Este identificador es el resultado de una llamada al método CreateListCellStyle.
condition	Condición que deben cumplir las celdas para que se le asigne el estilo (ver *Anexo I*).

Retorna:

Un booleano que indicará si ha habido error o no en la llamada al método.

SetListColumnStyle

Este método permite asignar un estilo a todas las celdas de una columna de una lista.

Sintaxis:

```
SetListColumnStyle(col as Smallint ,style as Integer)
```

Parámetros:

Col	Identificador de la columna.
Style	Identificador del formato que se va a aplicar a la columna. Este identificador es el resultado de una llamada al método CreateListCellStyle.

SetListColumnConditionalStyle

Permite asignar un estilo visual a todas las celdas de una columnas de una lista que cumplan la condición que se indicará como parámetro.

Sintaxis:

```
SetListColumnConditionalStyle(col as Smallint ,style as Integer ,condition as Char)
```

Parámetros:

Col	Identificador de la columna.
Style	Identificador del formato que se va a aplicar a fila. Este identificador es el resultado de una llamada al método CreateListCellStyle.
condition	Condición que deben cumplir las celdas de la columna para que se le asigne el estilo (ver *Anexo I*).

SetListColumnEditType

Permite indicar el tipo de control con el que se visualizará y editará una celda. Por defecto este control será de tipo text y edit field.

Sintaxis:

```
SetListColumnEditType(col as Smallint ,Editstyle as Smallint ,Values as Char)
```

Parámetros:

Col	Identificador de la columna.
Editstyle	Identificador de estilo. Los valores posibles son: 1 Text. 2 Check. 3 Date. 4 Droplist. 5 Numeric.
Values	Si el control seleccionado es un drop list, en este parámetro se indicará las lista de valores que se van a mostrar en esta lista. Su valor es un string donde los elementos de la lista se encuentran separados por el carácter "|". Si el control seleccionado es un control check, en este parámetro se indicará en un string, separado por el carácter "|", cuál será el valor TRUE y cuál el valor FALSE. Si no se indica ninguno, se tomará 1 como TRUE y 0 como FALSE.

SetListTotalsCellStyle

Permite asignar un estilo a las celdas donde se mostrarán los totales creados con el método TotalizeColumn y calculados con el método ComputeListColumnsTotals.

Los estilos se crearán con el método CreateListCellStyle.

Sintaxis:

```
SetListTotalsCellStyle(style as Integer)
```

Parámetros:

style	identificador del estilo creado con el método CreateListCellStyle.

SetLockColumns

Permite bloquear el scroll horizontal de un número determinado de columnas de un control tipo lista. El orden en el que se aplicará el bloqueo de las columnas será de izquierda a derecha.

Sintaxis:

```
SetLockColumns(column as Smallint).
```

Parámetros:

Column	Indica el número de columnas a bloquear.

SetStrListFilterBar

Permite indicar un texto por el que realizar el filtro automático en la lista en curso. Como resultado del filtro se mostrarán todas las filas que contengan el texto indicado en cualquiera de las columnas que la componen. El resultado de esta operación es el mismo que la llamada al método ShowListFilterBar, con la diferencia de que no habrá interacción con el usuario.

Sintaxis:

```
SetStrListFilterBar(text as Char)
```

Parámetros:

Text	Literal por el que se va a filtrar.

ShowListFilterBar

Este método muestra un control de edición en el que se podrá indicar una cadena de caracteres para poder filtrar las filas de la lista.

Si se pulsa la tecla **[Esc]** se cancelará la acción y se lanzará el evento ListCancelEditFilterBar. Si se pulsa la tecla **[Intro]** se filtrará por los datos introducidos y lanzará el evento ListAcceptEditFilterBar.

La búsqueda se realiza de la siguiente manera: El literal indicado se busca en todas celdas de la lista. Si en el control de edición se indica más de un valor mostrará las filas que cumplan todas las condiciones.

Sintaxis:

```
ShowListFilterBar(show as Boolean, backColor integer, textColor
integer, ,casesensitive as Boolean default FALSE)
```

Parámetros:

Show	Indica si se va mostrar el control de edición. Los valores posibles son: TRUE: se pintará el control. FALSE: no se mostrará el control de edición, y si el control es visible cuando se llame a este método, éste se ocultará.

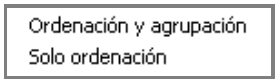
backColor	RGB que indica el color de fondo del control de edición.
textColor	RGB que indica el color del texto.
casesensitive	Permite indicar si la búsqueda distinguirá o no entre mayúsculas y minúsculas.

> ***IMPORTANTE:*** *Si el valor que se indica en el parámetro casesensitive es FALSE y la lista es de tipo Sql, en instalaciones cliente-servidor la versión de motor deberá ser al menos la 3.4 0.2 en la versión para Windows. Para sistemas operativos UNIX/LINUX las releases de las versiones del motor tendrán que ser igual o superiores a la 3.2 0.2, 3.4 0.2 y 3.6 0.2.*

ShowMultiColumnGroupDlg

Este método permite mostrar los registros de una lista de un control List Box de tipo Colimns List como una vista de registros agrupados, indicando las columnas por las que se agrupa y las columnas por las que se calculan valores agregados.

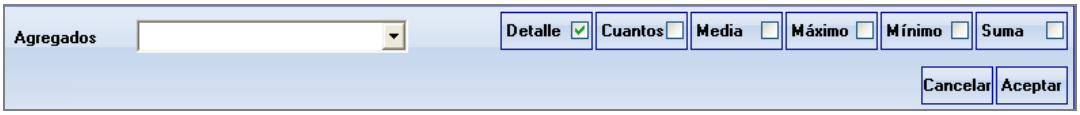
Para ello mostrará una barra en la parte inferior de la lista sobre la que se podrán arrastrar columnas. Sobre las columnas añadidas se podrán realizar dos acciones: agrupar y ordenar. Si sobre algunas de ellas sólo se quiere ordenar los registros habrá que desplegar el menú secundario que se muestra a continuación y elegir la opción "Solo ordenación".



Además de las dos acciones mencionadas se podrán aplicar las siguientes funciones sobre las columnas agregadas:

Detalle	Muestra los registros. Si se desmarca la opción sólo muestra los grupos y los valores agregados.
Cuántos	Esta función muestra el número de registros agrupados bajo un mismo grupo.
Media	Obtiene la media aritmética de la columna en los registros del grupo.
Máximo	Obtiene el valor máximo de la columna en los registros del grupo.
Mínimo	Obtiene el valor mínimo de la columna en los registros del grupo.
Suma	Obtiene la suma del conjunto de valores de la columna en los registros del grupo.

Para indicar la columna agregada ésta debe seleccionarse en el control drop list de la barra que se muestra la siguiente figura.



El aspecto de la lista después de aplicar este método será similar al de una lista en árbol desplegada. Cada nodo indica un punto de ruptura de un grupo. Los registros aparecerán ordenados según el orden en que se han añadido las columnas en la barra, y éstas se moverán del lugar que ocupaban originalmente a las primeras posiciones.

Sintaxis:

```
ShowMultiColumnGroupDlg(show as Boolean ,iconOpen as Smallint default 0
,iconClose as Smallint default 0 ,lockGroupColumns as Boolean
default TRUE ,countTitleTemplate as Char default "Total:" ,maxTitleTemplate as
Char default "Máximo:" ,minTitleTemplate as Char default "Minimo:"
,sumTitleTemplate as Char default "Suma:" ,avgTitleTemplate as Char default
"Media:")
```

Parámetros:

Show	Indica si se va a mostrar o no la barra. Los valores posibles son: TRUE y FALSE.
iconOpen	Identificador del icono que se mostrará en la vista de la lista para indicar que el grupo está desplegado.
iconClose	Identificador del icono que se mostrará en la vista de la lista para indicar que el grupo no está desplegado.
lockGroupColumns	Permite fijar las columnas que forman el grupo después de haberse movido para que éstas no se desplacen al hacer scroll horizontal. Los valores posible son: TRUE y FALSE.
countTitleTemplate	Literal que aparecerá como etiqueta de la función count().
maxTitleTemplate	Literal que aparecerá como etiqueta de la función max().
minTitleTemplate	Literal que aparecerá como etiqueta de la función min().
sumTitleTemplate	Literal que se mostrará como etiqueta de la función sum().
avgTitleTemplate	Literal que se mostrará como etiqueta de la función avg().

NOTAS:

El máximo de columnas a agrupar es 8.

ShowMultiColumnSortDlg

Este método muestra una barra de ordenación en la parte inferior de la lista. A ésta se arrastrarán las columnas por las que se desea ordenar.

Una vez añadidas la columnas a la barra se podrá cambiar el orden, así como la ordenación ascendente/descendente. Para aplicar la ordenación habrá que pulsar el botón **[Aceptar]** que se muestra a la derecha de la barra.

Las columnas seleccionadas se moverán del lugar que ocupaban originalmente a las primeras posiciones.

La columnas seleccionadas con este método podrán ser consultas con el método GetOrderBy.

Sintaxis:

```
ShowMultiColumnSortDlg(show as Boolean, changeView as Boolean).
```

Parámetros:

Show	Indica si se va a mostrar la barra o no. Los valores posibles son: TRUE y FALSE.
changeView	Hace que las columnas indicadas en la ordenación se muestren las primeras de la lista.

TotalizeColumns

Este método permite añadir totales en las columnas de un control List Box de tipo String o de tipo Sql.

Sintaxis:

```
TotalizeColumn(column as Smallint ,aggType as Smallint ,condition as Char de-
fault NULL ,title as Char default NULL)
```

Parámetros:

column	Índice de la columna.
aggType	Tipo de totales. Los posibles valores son: 1 (suma), 2(mínimo), 3 (máximo), 4 (media), 5 (total registros).
condition	Condición por la que se desea restringir la función totalizadora (ver *Anexo I*).
title	Etiqueta que se indicará en el agregado.

NOTA:

Los totales no se calculan automáticamente. Para que se realice el cálculo habrá que llamar al método ComputeListColumnTotals.

UpdateBackupListRow

Este método permite modificar el valor de la fila almacenado en memoria con el método BackupListRow.

Sintaxis:

```
UpdateBackupListRow(row as Integer, Text as Char)
```

Parámetros:

row	Indentificador de la fila.
text	Texto que se la asignará a la fila.

5. Eventos

ListAcceptEditColumnFilter

Este evento se lanzará cuando se pulse la tecla **[Intro]** después de ejecutar el método AllowColumnHeaderFilter.

ListAcceptEditFilterBar

Este evento se lanzará cuando se pulse la tecla **[Intro]** después de mostrar el campo de edición para filtrar los datos introducidos.

ListCancelEditColumnFilter

Este evento se lanzará cuando se pulse la tecla **[Cancel]** después de ejecutar el método AllowColumnHeaderFilter y el foco no esté en la cabecera.

ListCancelEditFilterBar

Este evento se lanzará cuando se pulse la tecla **[Cancel]** después de mostrar el campo de edición para filtrar los datos introducidos.

6. Correcciones

6.1 Runtime

- En un control lista tipo string, si la columna era numérica y alguno de los valores era un número negativo, al pulsar en la cabecera de la misma para ordenar la lista no se tenía en cuenta el signo.
  Corregido.
- Si un fuente de Cosmos tenía más de 32.768 líneas, y desde CEDIT.EXE se pulsaba la combinación de teclas **[Ctrl-Fin]**, el cursor no se posicionaba en la última línea del archivo.
  Corregido.
- Si las pestañas de un control de tipo tabControl tenían asociado un icono y se mostraban en pantalla con el método SetTabVerticalPages, en lugar de pintarse apiladas en vertical se mostraban con la vista clásica.
  Corregido.
- No se repintaba correctamente la celda de una lista que se había editado después de modificar su valor.
  Corregido.
- El método UnloadTo no tenía en cuenta los caracteres de escape.
  Corregido.
- Al realizar una búsqueda utilizando el método Query de la clase Formtable con condición, si el dato por el que se buscaba contenía la palabra order, y además existía una ordenación por esa columna, Cosmos no construía bien la sentencia.
  Corregido.
- Método AddColumnFilter. No funcionaba correctamente cuando el valor por el que se filtraba en la condición era "is not null".
  Corregido.
- Si se desplegaba un drop list en un PC con dos monitores conectados, si el drop list estaba en el monitor de la derecha, su ventana se mostraba en el monitor de la izquierda.
  Corregido.
- Lista SQL con filtros. Si se hacía un loadSelect de nuevo, no aplicaba los filtros existentes previamente hasta que se pulsaba el botón de ordenación de columna.
  Corregido.
- Error de protección general cuando el tamaño de retorno en parámetro headerSend del método CallWebService sobrepasaba los 1.000 caracteres.
  Corregido.
- Método FindCol. Si el texto con el que se realizaba la búsqueda tenía más de una ocurrencia en la lista, cuando se llegaba al final de la lista este método no retornaba 0, sino el primer elemento de la lista que cumpliera las condiciones de la búsqueda.
  Corregido.

- La exportación a Excel/CSV de variables de tipo smallint o integer con máscara #,### o #,##0 desde el Preview de Cosmos no era correcta.
  Corregido

- Al habilitar un comando, no se habilitaba la opción de menú asociada al mismo (ver variable de entorno ENABLEMENUOPTIONONENABLECOMMAND).
  Corregido

- Errores aleatorios de protección general al ejecutar el método SetErrorStr.
  Corregido

- Exportación a Excel, PDF, HTML y ODT de listas. Se ha modificado la exportación de las listas en árbol para que exporte cada nodo sólo si el padre no está contraído. Hasta ahora exportaba todas las filas, estuviera contraído o no el nodo padre.
  Corregido

- La propiedad Selected retornaba un smallint cuando debería retornar un integer.
  Corregido

  IMPORTANTE: *Este error persistirá si se ejecutan programas compilados con versiones anteriores. Es decir, cuando el elemento de la lista sea superior a 32.767, el dato retornado no será correcto.*

  Para que la corrección se aplique satisfactoriamente es necesario que cualquier objeto que se utilice para asignar o recoger el valor de la propiedad debe ser de tipo integer. Así mismo, será necesario recompilar la aplicación con esta versión.

  Si los objetos definidos como variables locales o globales, o como parámetros de una función definidos para asignar o recoger el valor de la propiedad selected ya son objetos de tipo integer sólo será necesario recompilar la aplicación.

6.2 CTSQL

- No ordenaba correctamente los datos de los campos decimales cuando la precisión era de 3 decimales o superior.
  Corregido

Anexo I

Operaciones aritméticas simples

Suma
Resta
Multiplicación
División

Operaciones aritméticas complejas

Potencias de e (exp)
Potenciación (^)
Logaritmo en base 2 (log2)
Logaritmo en base 10 (log)
Logaritmo neperiano (ln)
Raíz cuadrada (sqrt)
Signo (sign). Si < 0 retorna -1, si mayor 0 retorna 1
Redondeo al entero más próximo (rint)
Valor absoluto (abs)
Valor mínimo de una lista de valores (min)
Valor máximo de una lista de valores (max)
Suma de una serie de valores (sum)
Media aritmética (avg)

Operaciones trigonométricas

Seno (sin)
Coseno (cos)
Tangente (tan)
Arcoseno (asin)
Arcocoseno (acos)
Arcotangente (atan)
Seno hiperbólico (sinh)
Coseno hiperbólico (cosh)
Tangente hiperbólica (tanh)
Arcoseno hiperbólico (asinh)
Arcocoseno hiperbólico (acosh)
Arcotangente hiperbólica (atanh)

Operaciones lógicas

And (&&)
Or (||)
Menor o igual que (<=)
Mayor o igual que (>=)
Menor que (<)
Mayor que (>)
Igual que (==)
Distinto que (!=)
Operador ternario -if then else- (?:)

Comparación de cadenas

equalstring
equalstringnocase

Consultar valor nulo

isnull

Tabla de prioridades

Operador	Significado	Prioridad
=	Asignación	-1
&&	AND lógico	1
||	OR lógico	2
<=	Menor o igual que	4
>=	Mayor o igual que	4
!=	Distinto	4
==	Igual	4
>	Mayor que	4
<	Menor que	4
+	Suma	5
-	Resta	5
*	Multiplicación	6
/	División	6
^	Potenciación	7

Anexo II. Notas sobre estilos a nivel de celda y de columna en listas Cosmos

En Cosmos 5.0 se ha añadido la posibilidad de asignar estilos visuales a las listas a nivel de celda y a nivel de columna.

Creación y definición de estilos

Un estilo permite definir una serie de atributos visuales bajo un mismo identificador que posteriormente se podrá asignar a celdas y columnas de una lista.

Estos atributos son:

- Tipografía (nombre de fuente, tamaño, negrita, cursiva, subrayado, tachado, charset).
- Color del texto.
- Color del fondo.
- Alineación (izquierda, centro, derecha).
- Icono.

A la hora de crear un estilo no es obligatorio detallar todos sus atributos. Por ejemplo, en el campo que define la tipografía se puede indicar un valor nulo. En este caso no se cambiará la fuente del elemento al que se asigne el estilo, sino que se utilizará la fuente asignada a la lista o la fuente por defecto de la misma.

Si no se desea cambiar el color del texto o el del fondo se indicará el valor -1 (menos uno) en la definición del estilo en los parámetros foreground y/o background. En este caso, el elemento de la lista se dibujará con el color del fondo o del texto asignado con los métodos SetRowBackground, SetColumnBackground, SetRowForeground, SetColumnForeground y, si no se ha asignado color con ninguno de estos métodos, se usará el color asignado a la lista.

Si no se desea definir un icono en el estilo, se indicará el valor -1 (menos uno) en el parámetro icono.

Un estilo se puede asignar a más de una celda y a más de una columna de la misma lista, pero un estilo definido para un control lista no se puede utilizar en otro control lista. Es decir, los estilos son reutilizables para los elementos del control lista para el que se han creado, pero si se desean asignar esos estilos a elementos de otro control lista se deben redefinir para el nuevo control.

Estilos a nivel de celda y a nivel de columna

La diferencia entre estilos a nivel de celda y de columna radica en que un estilo a nivel de celda se asigna a una determinada celda de la lista, identificada por índice de fila y número de columna, y un estilo a nivel de columna es asignado a las celdas de una determinada columna para todas las filas de la lista.

Prioridad en estilos

Los estilos a nivel de celda son prioritarios sobre los estilos a nivel de columna.

Es decir, si se asigna un estilo A a una celda de la lista situada en la fila 10, columna 4 (estilo a nivel de celda), y se asigna otro estilo B definido en la lista a la columna 4 (estilo a nivel de columna), el estilo con el que se dibujará la celda 10-4 será el estilo asignado a la celda, el estilo A, y no con el estilo asignado a la columna 4, el estilo B.

Las celdas situadas en la columna 4 del resto de las filas se dibujarán con el estilo B.

Listas de estilos

Cuando se asigna un estilo a una celda o a una columna a la que previamente ya se le había asignado otro estilo, el nuevo estilo no sustituye al anterior, sino que se añade a la lista de estilos disponibles para la celda/columna.

Esto permite que una columna se dibuje con un estilo u otro dependiendo del valor de la misma.

Se podrá indicar, por ejemplo, que el texto de las celdas de una columna numérica se dibuje en color rojo cuando sean menores que 0, que se dibujen en color negro cuando sean igual a 0 y que se dibujen en color azul cuando sean mayores que 0.

Por ejemplo:

Se crean para la lista LST_1 los estilos ESTILO_1, ESTILO_2 y ESTILO_3.

Estos estilos tendrán una fuente, un color de texto, un color de fondo, una alineación y un icono determinados.

Se asigna a la columna 1 el estilo ESTILO_1. Éste es un estilo de asignación condicional que se aplicará a la columna cuando el valor de la celda sea menor que 0.

Se asigna a la misma columna el estilo ESTILO_2. Éste es un estilo de asignación condicional que se aplicará a la columna cuando el valor de la celda sea igual a 0.

Se asigna a la misma columna 1 el estilo ESTILO_3. Éste es un estilo de asignación condicional que se aplicará a la columna cuando el valor de la celda sea mayor que 0.

Una celda con valor 0 se dibujará con los atributos del ESTILO_2, ya que cumple la condición de valor igual a 0.

Una celda con valor -1 se dibujará con los atributos del ESTILO_1, ya que cumple la condición de valor menor que 0.

Una celda con valor 123 se dibujará con los atributos del ESTILO_3, ya que cumple la condición de valor mayor que 0.

El orden en que Cosmos comprobará el estilo que tiene que asignar a una celda/columna es el mismo en el que se asignaron los estilos a la celda/columna.

Este punto es importante, principalmente la razón siguiente: Se puede dar la circunstancia de que se haya asignado más de un estilo de asignación condicional a nivel de celda/columna en los que una celda cumpla la condición de dos estilos. Entonces el estilo que se le asignará será el primero que se haya asignado de los que cumpla la condición.

Por ejemplo:

Se crean los estilos ESTILO_1, ESTILO_2 y ESTILO_3 para la lista LST_1.

Primero se asigna el estilo ESTILO_1 a la columna 1. La condición de aplicación de este estilo será que su valor sea menor que 0.

Después se asigna el estilo ESTILO_2 a la columna 1. La condición de aplicación de este estilo será que su valor sea mayor o igual a 0 y menor que 100.

En tercer lugar, se asigna el estilo ESTILO_3 a la columna 1. La condición de aplicación de este estilo será que su valor sea mayor que 80.

En una celda de la columna 1 el valor es -3. Se dibujará con el estilo ESTILO_1 porque su valor es menor que 0 (ESTILO_1), no es mayor o igual que 0 y menor que 100 (ESTILO_2) y no es mayor que 80 (ESTILO_3).

En otra celda de la columna 1 el valor es 40. Se dibujará con el estilo ESTILO_2 porque su valor está entre 0 y 100, su valor no es menor que 0 ni mayor que 80.

En otra celda de la columna 1 el valor es 120. Se dibujará con el estilo ESTILO_3 porque su valor no es menor que 0 (ESTILO_1), ni mayor o igual que 0 y menor que 100 (ESTILO_2) y es mayor que 80 (ESTILO_3).

En otra celda de la columna 1 su valor es 90. Aquí nos encontramos con que no cumple la condición de asignación de ESTILO_1 (menor que 0), pero sí la de asignación de los estilos ESTILO_2 (entre 0 y 100) y ESTILO_3 (mayor que 80).

En este caso, el estilo con el que se dibujará será con el estilo ESTILO_2, ya que se ha asignado a la columna 1 antes que el ESTILO_3, que también cumple la condición.

En este caso, y para evitar problemas, lo estrictamente correcto hubiera sido que la condición de asignación del ESTILO_3 fuera que su valor fuese mayor o igual que 100.

Un estilo aplicado sin condición o aplicado con un valor nulo en el campo condición no significa que será el estilo que se aplique siempre a las celdas/columnas a las que se ha asignado el estilo, sino que su condición siempre será TRUE. El orden en que se comprobará el estilo con el que se dibujará una celda/columna siempre será el orden de asignación de estilos.

Si se desea cambiar el orden de comprobación de los estilos a nivel de celda/columna será necesario que éstos se reasignen en el orden de comprobación deseado tras haber reseteado los estilos a nivel de celda/columna con los métodos ResetListCellStyles o ResetListColumnStyles.

Prioridad en el color del texto y del fondo

Se puede dar el caso de que se haya asignado un color de texto o de fondo a nivel de fila y/o a nivel de columna a los elementos de una lista con los métodos SetRowBackground, SetColumnBackground, SetRowForeground o SetColumnForeground.

En ese caso, el orden de comprobación de color de texto/fondo con el que se dibujará la celda será del primero al último:

1. Color de texto/fondo definido en el estilo a nivel de celda (SetListCellConditionalStyle y SetListCellStyle).
2. Color de texto/fondo definido en estilo a nivel de columna (SetListColumnConditionalStyle y SetListColumnStyle).
3. Color de texto/fondo a nivel de fila (SetRowBackground y SetRowForeground).
4. Color de texto/fondo a nivel de columna (SetColumnBackground y SetColumnForeground).
5. Color de texto/fondo a nivel de control.

Funciones de creación, asignación, eliminación de asignación y consulta de estilos

- Creación de estilos
  CreateListCellStyle

- Asignación de estilos a nivel de celda
  SetListCellStyle — Sin condición.
  SetListCellConditionalStyle — Con condición.

- Asignación de estilos a nivel de columna
  SetListColumnStyle — Sin condición.
  SetListColumnConditionalStyle — Con condición.

- Eliminación de asignación de estilos a nivel de celda
  ResetListCellStyles — Para estilos asignados a nivel de celda con y sin condición.

- Eliminación de estilos a nivel de columna
  ResetListColumnStyles — Para estilos asignados a nivel de columna con y sin condición.

- Consulta de estilos a nivel de celda
  GetListCellStyles — Para estilos asignados a nivel de celda con y sin condición.

- Consulta de estilos a nivel de columna
  GetListColumnStyles — Para estilos asignados a nivel de columna con y sin condición.