# MultiBase Cosmos

## Notas a la versión 8.6

BASE100

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

# Índice

# 1 Implementaciones

En la versión 8.6 de Cosmos se han realizado las siguientes implementaciones:

- Posibilidad de asignar una plantilla de colores a un control determinado y a sus hijos desde un fichero de configuración.
- Posibilidad de asignar atributos a un control determinado y a sus hijos desde un fichero de configuración.
- Posibilidad de asignar iconos personalizados a los controles tipo Check Box y Radio Button.
- Posibilidad de asignar iconos personalizados a los controles deshabilitados de tipo Push Button, Button Group y Tab Control.
- Posibilidad de asignar el color del texto de fila selecciona en los controles List Box (variable de entorno LISTCURRENTROWFORECOLOR).
- Método LoadFromCharEx y LoadFromFileEx de la clase Json.
- Posibilidad de ejecutar las pantallas de una aplicación Cosmos en formato HTML en el «Frame» de Cosmos.
- La versión del log4j que se distribuye con esta versión es la 2.17.1.
- Nuevos métodos de la clase FORM y JSON.

# 2 Mejoras

## 2.1 Runtime

### 2.1.1 Clase JSON

Optimización del proceso de recorrido de un array de un objeto JSON.

### 2.1.2 Reparador de índices

Se amplía el límite en la longitud al fichero de datos a 128 caracteres.

### 2.1.3 Apariencia en controles deshabilitados

Se ha modificado el comportamiento por defecto de Cosmos al mostrar las etiquetas de los controles Box, Radio Push Button, Check y Tab Control cuando estos están deshabilitados, mejorando la nitidez de las etiquetas.

Esta modificación tiene efecto cuando se ha definido un color para la etiqueta con el atributo LABELFORECOLOR y la variable COSMOSVISUALMODE está definida con valor igual a 7.

Si no se dan las circunstancias descritas en el párrafo anterior, el comportamiento será el habitual, es decir, las etiquetas deshabilitadas se mostrarán con efecto de sombreado.

### 2.1.4 Nuevas propiedades para la personalización de colores

Se han implementado nuevas propiedades que permiten personalizar el color de los controles. A continuación, se enumeran la lista de propiedades:

- Color_Text_Edit_Disabled. Permite definir el color de texto de los controles Edit Field cuando están deshabilitados.
- Color_Back_Edit_Disabled. Permite definir el color de fondo de los controles Edit Field cuando están deshabilitados.
- Color_Border_Edit_Disabled. Permite definir el color de los bordes e los controles Edit Field cuando están deshabilitados.
- Color_Text_DropEdit_Disabled. Permite definir el color del texto de los controles Drop Edit cuando están deshabilitados.
- Color_Back_DropEdit_Disabled. Permite definir el color de fondo de los controles Drop Edit cuando están deshabilitados.
- Color_Border_DropEdit_Disabled. Permite definir el color de los bordes de los controles Drop Edit cuando están deshabilitados.
- Color_Text_DropList_Disabled. Permite definir el color del texto de los controles Drop List cuando están deshabilitados.
- Color_Back_DropList_Disabled. Permite definir el color de fondo de los controles Drop List cuando están deshabilitados.
- Color_Border_DropList_Disabled. Permite definir el color de los bordes de los controles Drop List cuando están deshabilitados.
- Color_Back_Text. Permite definir el color del fondo de los controles tipo TEXT.
- Color_Text_Tab_Disabled. Permite definir el color del texto en pestañas deshabilitadas.
  Los textos de las etiquetas de los grupos de pestañas creados con el método SetTabControlHierarchy de la clase SimpleFormControl no modifican su color al no formar parte de las pestañas del control.

- Color_Back_Grid. Permite definir el color de fondo de un control Grid (Parent).
- Color_Back_Grid_Disabled. Permite definir el color de fondo de un control Grid (Parent) cuando está deshabilitado.
- Color_Back_List. Permite definir el color de fondo de un control List Box.
- Color_Text_List. Permite definir el color del texto de un control List Box.

## 2.2 Entorno de desarrollo

Posibilidad de abrir el código fuente de un módulo de Cosmos dentro de la ventana MDI al abrir el entorno de desarrollo en el monitor secundario.

## 2.3 Cosmos WebServer

Se ha actualizado la versión de log4j que se incluye en Cosmos WebServer a la versión 2.17.1.

# 3 Personalización de controles desde fichero configuración

A partir de la versión 8.6 de Cosmos, se podrá personalizar las propiedades de un control y sus hijos a partir de ficheros de configuración sin tener que utilizar los métodos AddCustomControls/AddCustomColors y SetCustomControls/SetCustomColors como en versiones anteriores. Esto permitirá añadir controles con una personalización especifica sin necesidad de recompilar la aplicación.

**Qué necesitamos para poder cambiar la apariencia del control desde un fichero de configuración en Cosmos**

Debemos tener en cuenta que para definir la apariencia gráfica de un control en Cosmos se necesitan dos «secciones»: Una para definir los colores y la otra para definir borde, estilo del borde y estética de las etiquetas del control. Por ello, necesitaremos dos secciones nuevas. Estas dos secciones podrían definirse en un mismo fichero, pero para mayor claridad utilizaremos dos ficheros distintos.

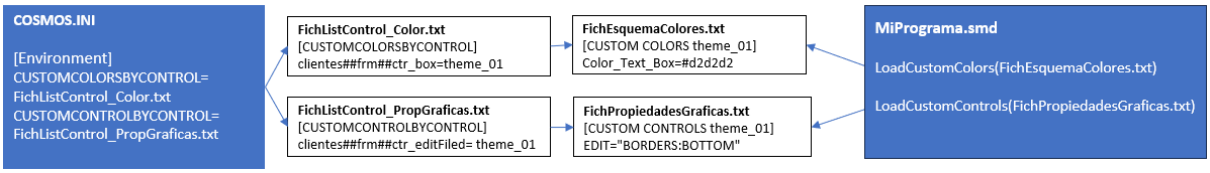
Cada uno de estos ficheros contendrán una lista de controles, identificados de manera inequívoca. Cada control tendrá asignado una etiqueta. El detalle de está etiqueta se definirá en un fichero distinto y dará nombre a nueva sección.

Si la etiqueta se utiliza para definir los colores, la nueva sección recibe el nombre de «Custom Colors etiqueta» y si se utiliza para definir las demás características, se llamará «Custom Controls etiqueta».

Los controles hijos heredan las propiedades definidas en la plantilla personalizada para el control contendor.

*¿Cómo busca Cosmos esta información?*

En el fichero de configuración del proyecto dentro de la sección «Environment» buscará dos variables de entorno CUSTOMCOLORSBYCONTROL y CUSTOMCONTROLSBYCONTROL. En estas variables se definirá la ruta al fichero que con las listas de controles y la etiqueta que define las propiedades de la apariencia gráfica de los controles. Estas etiquitas están definidas en otros ficheros de configuración. Estos ficheros son leídos por Cosmos cuando se ejecutan los métodos LoadCustomColors o LoadCustomControls.




## 3.1 Formato del fichero de configuración que describe la lista de controles.

Estos ficheros deberán incluir una nueva sección llamada «CustomColorsByControl» para la personalización de colores o una sección llamada «CustomControlsByControl» para la personalización de atributos de control. La ruta a este fichero se define con las variables de entorno CUSTOMCOLORSBYCONTROL y CUSTOMCONTROLSBYCONTROL respectivamente dentro de fichero de configuración de Cosmos o del proyecto (sección «Environment»).

La estructura del fichero de configuración es la siguiente:

```
[identificador de la sección]
Identificador_del_control= esquema_de_colores o atributos_de_control
```

| Personalización | Identificador de la sección | Identificador del control |
|---|---|---|
| Colores | CustomColorsByControl | módulo##clase##IdControl |
| Controles | CustomControlsByControl | módulo##clase##IdControl |

En necesario que a los controles se les haya asignado un nombre en diseño.

### 3.1.1 Formato del fichero de configuración del esquema de colores.

Fichero en el que se define el esquema de colores para uno o varios controles. Este fichero contendrá una sección llamada «Custom Colors <ID>», siendo <ID> el nombre de la sección de esquema de colores.
La información definida en este fichero se carga al llamar al método LoadCustomColors.

### 3.1.2 Formato del fichero de configuración de los atributos de control.

Fichero en el que definen los atributos para uno o varios controles. Este fichero contendrá una sección llamada «Custom Controls <ID>», siendo <ID> el nombre de la sección de atributos de control.
La información definida en este fichero se carga al llamar al método LoadCustomControls.

# 4 Métodos

## 4.1 Clase Form

### RunJavascriptFunction

Este método permite ejecutar una función JavaScript presente en el fichero HTML asociado a la clase Form.

El método RunJavascriptFunction de la clase Form permite ejecutar una función Javascript presente en el fichero HTML asociado al form.

Este método solo se utilizará cuando se ejecutan las ventanas de una clase Form en formato HTML dentro el Frame de Cosmos.

Sintaxis:

```
RunJavascriptFunction(function as Char, parameters as Char)
```

Parámetros:

| | |
|---|---|
| Function | Nombre de la función JavaScript. |
| Parameters | Parámetros que recibe la función.<br>Si la función JavaScript recibe más de un parámetro se recomienda que se pasen a través de un fichero JSON. En este caso, el valor del parámetro del método «RunJavascriptFunction» será la ruta a ese fichero. |

Cosmos solo permite la referencia a ficheros JS en directorio locales.

## 4.2 Clase JSON

### LoadFromCharEx

Carga un objeto de la clase JSON desde un objeto Char o una cadena de caracteres sin escapar caracteres.

Sintaxis:

```
LoadFromCharEx(string as Char) return boolean
```

Parámetros:

| | |
|---|---|
| string | Cadena de caracteres con el texto JSON. |

Retorna:

| | |
|---|---|
| TRUE | Si se ha cargado correctamente el objeto de la clase JSON desde el objeto Char. |
| FALSE | Si no ha podido cargar correctamente el objeto de la clase JSON desde el Char. Esto puede ocurrir si contiene algún error sintáctico. |

### LoadFromFileEx

Carga un objeto de la clase JSON desde un fichero sin escapar caracteres.

Sintaxis:

```
LoadFromFileEx(fileName as Char) return boolean
```

Parámetros:

| | |
|---|---|
| fileName | Ruta del fichero con formato JSON desde donde se desea cargar el objeto JSON. |

Retorna:

| | |
|---|---|
| TRUE | Si el objeto de la clase JSON se ha cargado correctamente desde el archivo. |
| FALSE | Si el objeto de la clase JSON no se ha podido cargar correctamente desde el archivo. |

Esto puede ocurrir si no existe el archivo, no tiene permisos de lectura o si contiene algún error sintáctico.

# 5 Variables de entorno

## CUSTOMCOLORSBYCONTROL

Esta variable de entorno permite personalizar la paleta colores de uno o varios controles de la aplicación.

El valor de esta variable de entorno será la ruta a un fichero. En este fichero se indicará el nombre de la sección de la paleta de colores personalizada y una ruta identificativa del control de la aplicación. Esta sección estará definida en el fichero que recibe como parámetro el método LoadCustomColors.

El método LoadCustomColors permite definir una plantilla de colores por cada tipo de control, es decir, los colores de texto, fondo, bordes, etc, de todos los controles de un mismo tipo. Todos los controles del mismo tipo compartirán esquema de colores. Al definir la variable CUSTOMCOLORSBYCONTROL podremos definir de una manera más detallada la plantilla de colores de cada control del cual necesitemos que sea distinta a la definida en la sección [Custom Colors] para los controles de su tipo.

El comportamiento de esta variable de entorno equivale a ejecutar los métodos AddCustomColors y SetCustomColors. La diferencia radica en que de esta manera no será necesario modificar el código fuente de la aplicación cuando se añadan nuevos controles, o cuando deseemos cambiar el esquema de colores de un control determinado. Será suficiente con añadir o modificar la entrada a un control determinado en el fichero de configuración.

La estructura del fichero de configuración indicado en la variable CUSTOMCOLORSBYCONTROL será la siguiente:

- Sección [CustomColorsByControl]

  Dentro de esta sección se definirá, por cada control que deseemos tener una plantilla de colores personalizada, una línea con el siguiente formato:

  ```
  módulo##clase##control=esquema
  ```

  Cada control será identificado por el módulo donde se encuentra definido, la clase y el nombre del control, separados por doble almohadilla.

- El esquema será una sección llamada [Custom Colors ESQUEMA] que se deberá definir en el fichero que ha sido pasado como parámetro en el método LoadCustomColors.

**Ejemplo:**

1. En la sección [Environment] de Cosmos.ini o del INI del proyecto, se define la siguiente variable de entorno:

   ```
   CUSTOMCOLORSBYCONTROL=themes\colorsbycontrol.ini
   ```

2. El valor de variable de entorno es la ruta de un fichero de configuración con la siguiente estructura:

   ```
   [CustomColorsByControl]
   main##frm##button_01=theme_01
   main##frm##button_02=theme_02
   ```

   | | |
   |---|---|
   | main | Nombre del módulo donde está definido el control. |
   | frm | Nombre de la clase donde está definido el control. |

| | |
|---|---|
| Button_01 y button_02 | Nombre de los controles a los que se le indica la plantilla de colores personalizada. |
| theme_01 y theme_01 | Esquemas de colores que serán asignados a los controles anteriores. |

3. Dentro del fichero indicado como parámetro en LoadCustomColors, habrá en este caso dos secciones: una para “theme_01” y otra para “theme_02”:

```
[Custom Colors theme_01]
Color_Text_Button=Rgb(255,0,0)
[Custom Colors theme_02]
Color_Text_Button=Rgb(0,255,0)
```

El control “button_01” será dibujado con los atributos indicados en [Custom Colors theme_01], y el control “button_02” será dibujado con los atributos indicados en [Custom Colors theme_02]. El resto de controles del mismo tipo, en este caso, Push Button, serán dibujados con los atributos definidos en la sección [Custom Colors] del fichero indicado en el método LoadCustomColors.

Nota:

Se recomienda no utilizar simultáneamente esta variable de entorno con los métodos AddCustomColors y SetCustomColors, ya que son dos maneras distintas de realizar la misma operación y dependiendo del orden de ejecución de los métodos dentro de la aplicación, el comportamiento será distinto. Prevalecerán los esquemas de colores indicados en el último método ejecutado (AddCustomColors o LoadCustomColors).

La variable de entorno COSMOSVISUALMODE debe estar definida con valor igual a 7.

## CUSTOMCONTROLSBYCONTROL

Esta variable de entorno permite personalizar los atributos de uno o varios controles de la aplicación.

El valor de esta variable de entorno será la ruta a un fichero. En este fichero se indicará el nombre de la sección donde se definirán los atributos personalizados de uno o varios controles y una ruta identificativa del control de la aplicación. Esta sección estará definida en el fichero que recibe como parámetro el método LoadCustomControls.

El método LoadCustomControls carga un fichero con los atributos de cada tipo de control, es decir, la estética de las etiquetas, de los bordes del control, la forma de las esquinas (redondeadas o cuadradas) y su relieve (normal, bajo y sobre relieve), etc., de todos los controles de un mismo tipo. Todos los controles del mismo tipo compartirán estos atributos. Al definir la variable CUSTOMCONTROLSBYCONTROL podremos definir de una manera más detallada los atributos de cada control del cual necesitemos que sean a distintos a los definidos en la sección [Custom Controls] para los controles de su tipo.

El comportamiento de esta variable de entorno equivale a la ejecución de los métodos AddCustomControls y SetCustomControls. La diferencia radica en que de esta manera no será necesario modificar el código fuente de la aplicación cuando se añadan nuevos controles, o cuando deseemos cambiar los atributos de un control determinado. Será suficiente con añadir o modificar la entrada a un control determinado en el fichero de configuración.

La estructura del fichero de configuración indicado en la variable CUSTOMCONTROLSBYCONTROL será la siguiente:

- Sección [CustomControlsByControl]
  Dentro de esta sección incluirá una entrada por cada control del cual se desee que sus atributos sean distintos a los de mismo tipo (atributos definidos en [Custom Controls]). La sintaxis es la siguiente:

  ```
  módulo##clase##control=esquema
  ```

  Cada control será identificado por el módulo donde se encuentra definido, la clase y el nombre del control, separados por doble almohadilla.
- El esquema será una sección llamada [Custom Controls ESQUEMA] que se deberá definir en el fichero que ha sido pasado como parámetro en el método LoadCustomControls.

Ejemplo:

1. En la sección [Environment] de Cosmos.ini o del INI del proyecto, se define la siguiente variable de entorno:

```
CUSTOMCONTROLSBYCONTROL=themes\controlsbycontrol.ini
```

2. El valor de variable de entorno es la ruta de un fichero de configuración con la siguiente estructura:

```
[CustomColorsByControl]
main##frm##edit_01=theme_01
main##frm##edit_02=theme_02
```

| | |
|---|---|
| main | Nombre del módulo donde está definido el control. |
| Frm | Nombre de la clase donde está definido el control. |
| Edit_01 y edit_02 | Nombre de los controles a los que se le indica la plantilla de atributos personalizada. |
| theme_01 y theme_01 | Esquema de atributos que serán asignados a esos controles. |

3. Dentro del fichero indicado como parámetro en LoadCustomControls, habrá en este caso dos secciones: una para “theme_01” y otra para “theme_02”:

```
[Custom Controls theme_01]
EDIT="BORDERS:BOTTOM,TOP"
[Custom Controls theme_02]
EDIT="BORDERS:LEFT,RIGHT"
```

El control “edit_01” será dibujado con los atributos indicados en [Custom Controls theme_01], y el control “edit_02” será dibujado con los atributos indicados en [Custom Controls theme_02]. El resto de controles del mismo tipo, en este caso, Edit Field, serán dibujados con los atributos definidos en la sección [Custom Controls] del fichero indicado en el método LoadCustomControls.

Nota:

Se recomienda no utilizar simultáneamente esta variable de entorno con los métodos AddCustomControls y SetCustomControls, ya que son dos maneras distintas de realizar la misma operación y dependiendo del orden de ejecución de los métodos dentro de la aplicación, el comportamiento será distinto. Prevalecerán los esquemas de colores indicados en el último método ejecutado (AddCustomControls o LoadCustomControls).

La variable de entorno COSMOSVISUALMODE debe estar definida con valor igual a 7.

## CUSTOMCHECKRADIOICONS

Permite definir un fichero de iconos que sustituirá en tiempo de ejecución a los iconos por defecto en los controles Check Box y Radio Button de Cosmos.

El fichero de iconos estará definido en la sección ICONS de COSMOS.INI o del fichero INI del proyecto.

El tamaño de cada icono deberá ser de 13x13 píxeles.

El fichero de iconos deberá constar de 8 iconos, uno por cada posible estado, y deberán estar posicionados de la siguiente manera:

| Posición | Control y estado |
|---|---|
| **1** | Check desmarcado habilitado |
| **2** | Check desmarcado deshabilitado |
| **3** | Check marcado habilitado |
| **4** | Check marcado deshabilitado |
| **5** | Radio desmarcado habilitado |
| **6** | Radio desmarcado deshabilitado |
| **7** | Radio marcado habilitado |
| **8** | Radio marcado deshabilitado |

Ejemplo:

En fichero de configuración COSMOS.INI o INI de proyecto:

```
[Environment]
CUSTOMCHECKRADIOICONS=checkyradio_bmp
[Icons]
checkyradio_bmp=.\checkyradio_bmp.lst,13 ,13, RGB(255,0,255)
```

Fichero checkyradio_bmp.lst:

```
iconos\checkyradio_bmp\icon_0.bmp
iconos\checkyradio_bmp\icon_1.bmp
iconos\checkyradio_bmp\icon_2.bmp
iconos\checkyradio_bmp\icon_3.bmp
iconos\checkyradio_bmp\icon_4.bmp
iconos\checkyradio_bmp\icon_5.bmp
iconos\checkyradio_bmp\icon_6.bmp
iconos\checkyradio_bmp\icon_7.bmp
```

## DRAWCUSTOMDISABLEDICON

Al definir esta variable, el runtime permitirá el uso de ficheros de iconos alternativos para mostrar los iconos de los controles deshabilitados.

El runtime buscará, en el fichero «ini» de configuración de Cosmos o del proyecto, un identificador del fichero de iconos con igual nombre al utilizado para los controles habilitados y con prefijo «_disabled». El orden los iconos en ambos ficheros deber ser idéntico.

Ejemplo:

```
comida=.\icoComida.lst, 25,25, RGB192,192,192)

comida_disabled=.\icoComidaDeshabilitado.lst, 25,25, RGB(192,192,192)
```

## FREECOMINFOONEXIT

Esta variable de entorno permite no liberar los objetos COM (ActiveX) al salir de la aplicación.

Valores posibles:

| | |
|---|---|
| NO/FALSE | No se liberan los objetos COM al salir de la aplicación. |
| YES/TRUE | Se liberan los objetos COM al salir de la aplicación. |

## HTMLFORMSACTIVE.

Esta variable de entorno permite mostrar las pantallas de un Form en formato HTML si está definida con valor TRUE. Si no se define o se define con valor distinto a TRUE, todos los formularios Cosmos se mostrarán en modo tradicional.

## HTMLFORMSDISABLECONTEXTMENU.

Al visualizar una ventana de Cosmos en formato HTML y pulsar el botón secundario de ratón, se muestra un menú propio de Chromium (ver sección Motor de renderizado). Este menú es útil para depurar el aspecto visual de las aplicaciones.

Esta variable de entorno permite que no se muestre el menú si la variable de entorno se define con valor TRUE.

Se recomienda que en el entorno de producción, se defina la variable HTMLFORMSDISABLECONTEXTMENU con valor TRUE.

## LISTCURRENTROWFORECOLOR

Permite definir el color del texto de la fila seleccionada en un control List Box.

# 6 Corrección de errores

## 6.1 Runtime

- El color de fondo del botón auxiliar de un control Edit Field datetime Picker definido en el atributo «Color_Back_AuxButton», en la sección «Custom Colors» del fichero, no se mostraba cuando a este botón se le había asignado un icono con el método «SetGlobalEditDateTimePickerIcon» o en diseño.
- El control Rich text no soportaba el cambio de color con la propiedad Background.
- La imagen del código QR generada desde la API COSQRDLL en formato PNG al imprimirla a PDF desde la librería de Cosmos mostraba un código ilegible. **Se ha modificado la generación de la imagen en formato PNG desde la API COSQRDLL para que sea compatible con la librería utilizada por Cosmos para la impresión a PDF**.
- El método UnloadToDomDocument los campos de tipo char los truncaba a 10 caracteres en arquitectura x64. En arquitectura x86 la ejecución se abortaba cuando se llama al método.

## 6.2 Entorno de desarrollo

Se producía un error en ejecución al realizar una búsqueda en todo el proyecto desde la opción «Find String in Files» si la línea donde encuentra el texto tiene más de 200 caracteres de longitud.

La lista que se mostraba en el drw del control Edit Field para tipo DateTime no era correcta. No se correspondía la máscara de la lista con la mostrada en ejecución.

## 6.3 TTTOLS

- Si la longitud de la ruta de la base de datos de la tabla que se iba a reparar era superior a 64 caracteres mostraba el siguiente mensaje: «debug error». RunTime Check Failure #2 – Stack around the variable «name» was corrupted».