Configurando la apariencia del control DataGrid

Luis Miguel Blanco Ancos

Mostrar datos en Windows

Cuando en una aplicación Windows se requiere visualizar información en formato tabular, en la mayor parte de las ocasiones recurriremos al control DataGrid para solventar la situación.

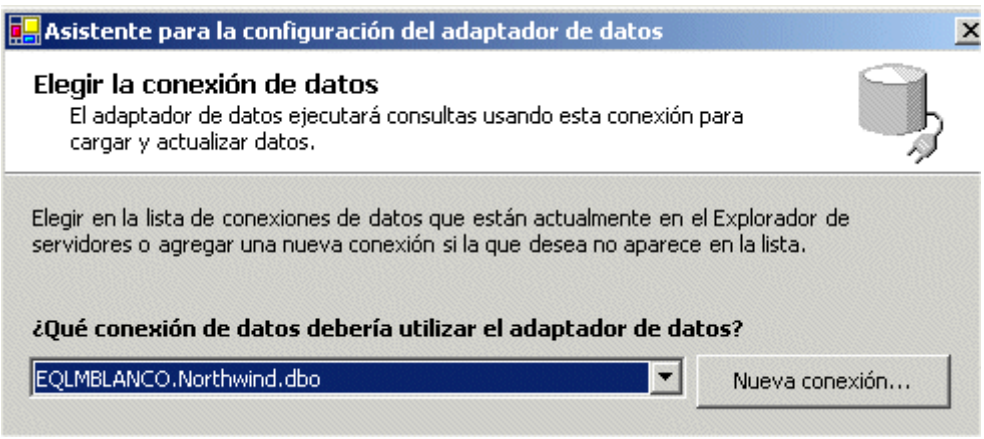
Se trata de un control que ha evolucionado notablemente desde los tiempos de Visual Basic 6, y aunque ya entonces ofrecía un importante abanico de características en cuanto a la presentación gráfica de los datos, con la llegada de la tecnología .NET, DataGrid, el sucesor del control DBGrid de Visual Basic 6, ha ampliado considerablemente la oferta de posibilidades en el apartado gráfico.

En el presente artículo vamos a explicar los pasos necesarios para establecer, en tiempo de diseño, la apariencia de los datos presentados por este control, dotándoles de una mayor vistosidad, y haciendo que al mismo tiempo, su consulta sea más fácil de llevar a cabo.

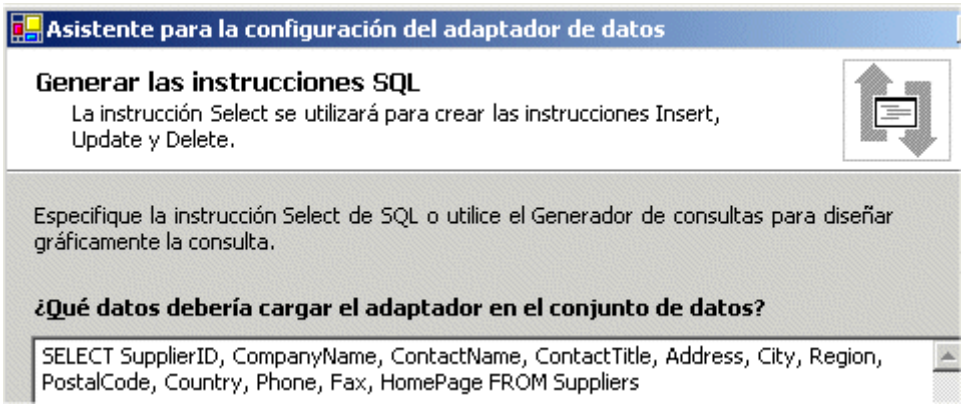
Creación del origen de datos para un DataGrid

Tras crear un nuevo proyecto Visual Basic .NET para Windows, al que daremos el nombre GridBasico, y antes de pasar a manipular un DataGrid, vamos a comenzar por hacer un repaso del modo de preparación del entorno de datos con los que alimentaremos a nuestro control. En los ejemplos de este artículo utilizaremos SQL Server como servidor de datos, y Northwind como base de datos.

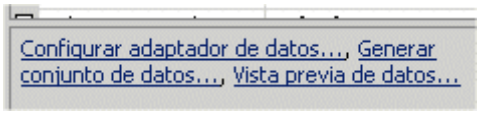
Desde la ficha Datos, del Cuadro de herramientas, añadiremos al formulario un componente SQLDataAdapter, en cuyo asistente de configuración crearemos una conexión contra la base de datos mencionada anteriormente.



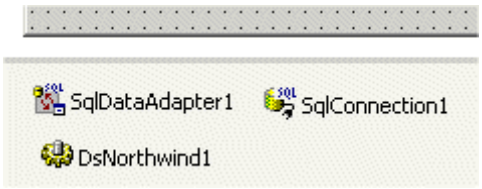
Seguidamente elegiremos el tipo de consulta basado en instrucciones SQL, y escribiremos una sentencia que obtenga las filas de la tabla Suppliers.



A continuación, en la parte inferior de la ventana de propiedades de este adaptador de datos, haremos clic en el enlace *Generar conjunto de datos*, y un nuevo asistente nos permitirá crear un DataSet al que llamaremos dsNorthwind, que contendrá la tabla Suppliers.



Como resultado, el componente SQLDataAdapter generará en el proyecto un esquema de DataSet, y a partir del mismo, se creará un DataSet con el nombre DsNorthwind1, que aparecerá en el panel de componentes del diseñador del formulario.



Configuración de un DataGrid desde el diseñador del formulario

Seguidamente arrastraremos sobre el formulario un control DataGrid, al que daremos el nombre grdDatos. Para indicar a este control de dónde debe obtener los datos, usaremos las siguientes propiedades.

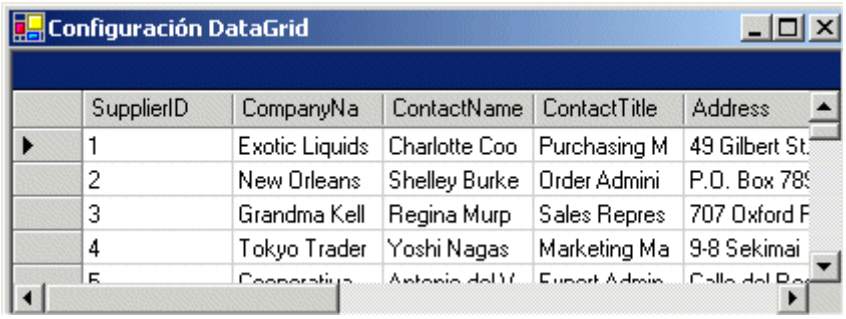
- DataSource. En esta propiedad estableceremos el DataSet que contiene toda nuestra estructura de datos: DsNorthwind1.
- DataMember. Aquí asignaremos la tabla del DataSet establecida en DataSource que queremos visualizar en el DataGrid: Suppliers.

Llegados a este punto, debemos tener en cuenta un aspecto importante que muchas veces pasa desapercibido. Si ejecutamos en este momento el proyecto, el control DataGrid estará vacío. Esto es debido a que el DataSet con el que está enlazado se

encuentra también vacío, puesto que el llenado del mismo no se produce automáticamente, siendo el programador quien debe realizarlo, ejecutando el método Fill del objeto DataAdapter. Podemos hacer esta operación en múltiples lugares, por ejemplo, en el evento Load del formulario, como vemos en el siguiente fuente.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
   Me.SqlDataAdapter1.Fill(Me.DsNorthwind1, "Suppliers")
End Sub
```

Ahora sí que podremos visualizar los datos en el control tal y como esperábamos.



Como truco para conseguir una mejor visualización del grid en el formulario, podemos asignar a la propiedad Dock de este control el valor Fill, de modo que cada vez que se redimensione el formulario, el control ajustará automáticamente su tamaño, ocupando toda la superficie de la ventana, así visualizaremos una mayor cantidad de filas y columnas.

Comenzando a decorar los datos

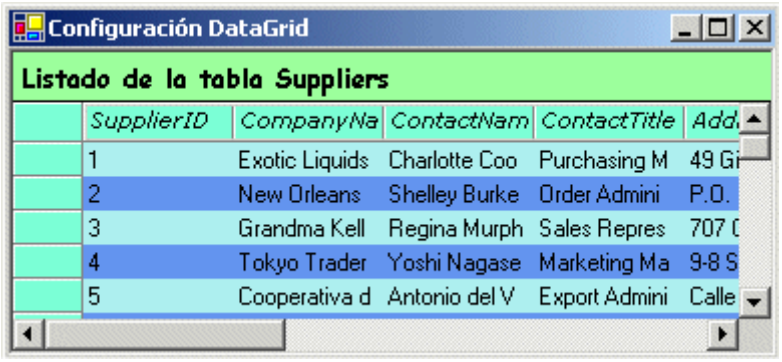
Aunque el resultado de la presentación de información realizada por el DataGrid es correcto, no deja de ser un tanto parco en el apartado gráfico. Nuestra misión a partir de ahora es solucionar dicho punto débil, así que pongámonos manos a la obra para aportar un poco de atractivo a nuestros datos.

Primeramente vamos a cambiar la combinación de colores y tipos de letra predeterminada, asignando valores a las propiedades relacionadas con este aspecto del control. La siguiente tabla describe algunas de estas propiedades.

Propiedad	**Descripción**
CaptionText	Título del control mostrado en su parte superior
CaptionBackColor, CaptionForeColor	Color de fondo y frente, respectivamente, del título del control
HeaderFont	Tipo de letra para los títulos de las columnas
GridLineStyle	Muestra u oculta las líneas de cuadrícula
AlternatingBackColor	Color para las filas alternas

HeaderBackColor	Color de fondo de los títulos de las columnas, y los indicadores de fila

La siguiente figura muestra el control resultante tras modificar los valores por defecto de algunas de estas propiedades.



De esta forma podemos personalizar notablemente la apariencia de los datos mostrados por el DataGrid, aunque, no obstante, en muchas ocasiones esto no será suficiente. Supongamos por un momento que tenemos en nuestro formulario un DataSet que contiene dos tablas, y queremos mostrar cada una de ellas en el DataGrid con una configuración de colores y fuente distinta, también es posible que sólo queramos mostrar determinados campos de una tabla, con configuraciones de formato independiente para cada uno. A continuación veremos como abordar esta problemática.

Estilos de tabla y columna. Datos y presentación por separado

La maquinaria subyacente del control DataGrid, a través de las clases DataGridTableStyle y DataGridTextBoxColumn, permite implementar en este control una efectiva separación entre los datos y su presentación, aspecto este cada vez más presente en todos los ámbitos del desarrollo de aplicaciones.

DataGridTableStyle es un tipo de la plataforma .NET Framework, cuya misión consiste en almacenar la configuración de estilo para tabla de datos mostrada dentro de un control DataGrid.

Por otra parte, el objetivo de la clase DataGridColumnStyle reside en contener el estilo de visualización de una columna perteneciente a un estilo de tabla del DataGrid

Gracias a este eficaz diseño, un único control DataGrid puede disponer de varios estilos de visualización diferentes para las distintas tablas que muestra, y cada uno de estos estilos puede mostrar las columnas de su tabla correspondiente con un alto grado de personalización.

El acceso a los estilos de tabla se realiza a través de la propiedad TableStyles, que es una colección de objetos DataGridTableStyle; y dentro de cada elemento de esta colección, el acceso a los estilos de columna se efectúa mediante la propiedad

GridColumnStyles, que igualmente es una colección de objetos DataGridColumnStyle, pertenecientes a la tabla.

DataGridColumnStyle es una clase abstracta de la que hereda DataGridTextBoxColumn, que será la que usemos para la creación de los estilos para las columnas.

Creación de un estilo de tabla

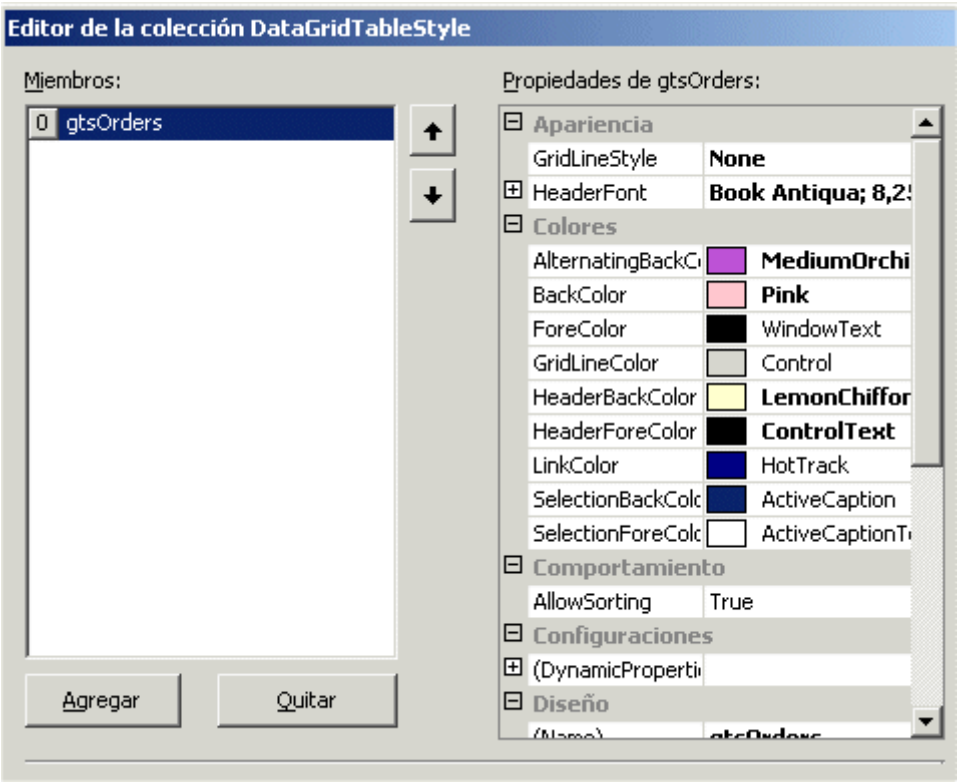
Pero dejémonos de tanta verborrea y pasemos a la acción. Vamos a crear un nuevo proyecto con el nombre GridEstilos, y en el formulario agregaremos un DataGrid que llenaremos con las filas de la tabla Orders, en la forma descrita al principio del artículo. No es necesario establecer una configuración de colores inicial para el grid, aunque sí sería recomendable para comprobar mejor el efecto de cambio al aplicar los estilos de tabla y columna que posteriormente realizaremos.

Como siguiente paso, haremos clic en el botón de puntos suspensivos de la propiedad TableStyles del DataGrid, que abrirá el cuadro de diálogo *Editor de la colección DataGridTableStyle*, en el que crearemos un estilo de formato para la tabla Orders.

Pulsando el botón Agregar, se añadirá un nuevo estilo de tabla vacío en el panel izquierdo, al que daremos el nombre gtsOrders, y cuyas propiedades estableceremos en el panel derecho. La siguiente tabla describe algunas de las propiedades de este tipo de objeto.

Propiedad	**Descripción**
MappingName	Nombre de la tabla del DataSet enlazado al DataGrid que se muestra cuando el estilo de tabla es seleccionado
GridColumnStyles	Colección de objetos DataGridColumnStyle, que contiene los estilos de formato para cada una de las columnas mostradas por el estilo de la tabla en el DataGrid
HeaderFont	Tipo de letra para los títulos de las columnas
AlternatingBackColor	Color para las filas alternas
HeaderBackColor	Color de fondo de los títulos de las columnas, y los indicadores de fila

La siguiente figura muestra el cuadro de diálogo de creación de estilos para tablas.



Como habrá observado el lector, muchas de las propiedades de un objeto DataGridTableStyle son comunes a las del DataGrid, ya que cuando el estilo de la tabla es aplicado sobre el control, los valores de estilo sustituyen a sus homólogos del grid.

Creación de estilos para las columnas

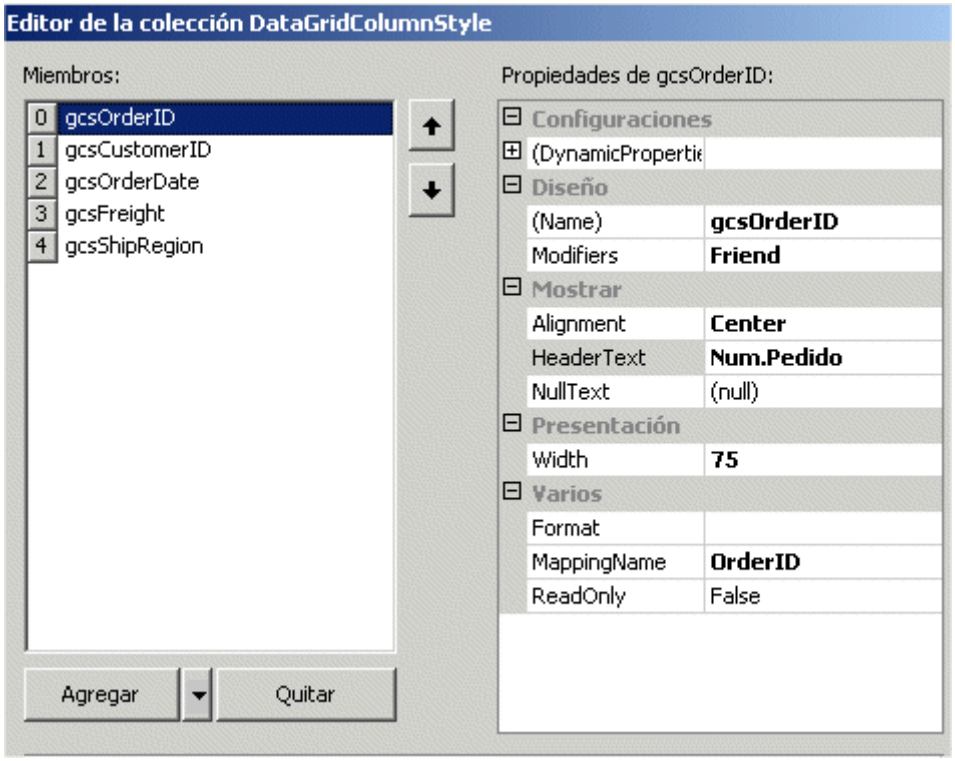
Una vez que hemos finalizado la creación del formato para la tabla, y permaneciendo en el cuadro de diálogo del estilo de tabla, haremos clic en el botón de puntos suspensivos de la propiedad GridColumnStyles; esto nos llevará a un nuevo diálogo, que con el nombre *Editor de la colección DataGridColumnStyle*, será el que usaremos para crear cada uno de los estilos de columna de la tabla.

Pulsando el botón Agregar, se añadirá un nuevo estilo de columna vacío, al que daremos formato asignando valores en sus propiedades. En la siguiente tabla destacamos algunas de estas propiedades.

Propiedad	**Descripción**
MappingName	Nombre de la columna de la tabla del DataSet, seleccionada al crear el estilo de la tabla.
Format	Cadena con la expresión de formato para aplicar a los datos de la columna
Alignment	Permite alinear el contenido de la columna a la izquierda, centro o derecha
HeaderText	Cadena con el título a mostrar en la columna

NullText	Cadena con el valor a mostrar cuando el campo de la tabla tenga valor nulo

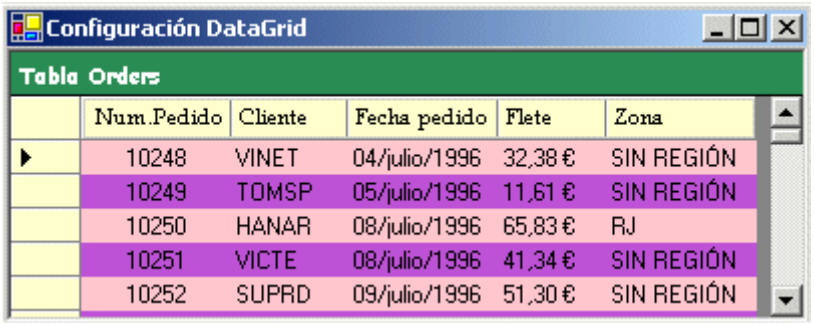
La siguiente figura muestra el cuadro de diálogo de creación de estilos para columnas.



Para este ejemplo hemos creado estilos de columna sólo para algunos campos de la tabla Orders, por lo que sólo estos serán mostrados en el grid. En general, a todas hemos asignado un nuevo título de cabecera en la propiedad HeaderText, y en las mencionadas a continuación, hemos establecido valores particulares:

- OrderDate. Al ser una fecha, en la propiedad Format hemos asignado la cadena de formato dd/MMMM/yyyy.
- Freight. Al ser un valor numérico monetario, en la propiedad Format hemos asignado el valor C2, para que muestre el campo con el símbolo de moneda y dos decimales.
- ShipRegion. Como se trata de un campo que tiene valores nulos en algunas filas de la tabla, hemos asignado a la propiedad NullText la cadena *SIN REGIÓN*, que será mostrada en lugar del literal *null* que utiliza el DataGrid por defecto.

Una vez que hemos terminado de crear los estilos de columna, aceptaremos todos los cuadros de diálogo, y ejecutaremos el proyecto. La siguiente figura muestra el resultado.



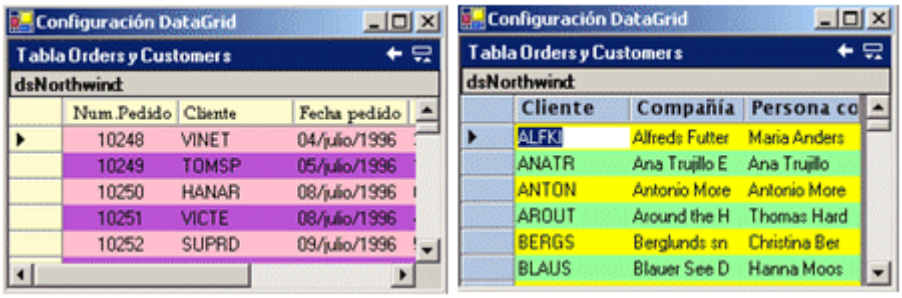
Utilizando un estilo diferente para cada tabla

Como habrá deducido el lector tras la demostración anterior, si el DataSet enlazado al DataGrid contiene más de una tabla, podemos crear un estilo distinto para cada una, con su combinación particular de colores y formato.

Para probar este comportamiento, eliminaremos los valores de las propiedades DataSource y DataMember del grid del formulario; después agregaremos un nuevo DataAdapter con una consulta contra la tabla Customers; seguidamente borraremos del panel de componentes del formulario el DataSet DsNorthwind1, y volveremos a crearlo, pero esta vez incluyendo las dos tablas correspondientes a los DataAdapter que tenemos en el formulario. No olvidemos, por último, añadir al DataSet desde el código la nueva tabla con el método Fill del último DataAdapter creado, tal y como muestra el siguiente código fuente.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
   Me.SqlDataAdapter1.Fill(Me.DsNorthwind1, "Orders")
   Me.SqlDataAdapter2.Fill(Me.DsNorthwind1, "Customers")
End Sub
```

Volviendo al DataGrid, asignaremos otra vez el DataSet a su propiedad DataSource, pero no estableceremos valor en DataMember, así, al ejecutar se mostrará un icono de selección de tabla. Añadiremos un estilo para la nueva tabla del DataSet, que configuraremos en la forma mostrada anteriormente, con una combinación de colores y letra distinta, y al ejecutar la aplicación, cada vez que seleccionemos en el grid una de las tablas, esta será mostrada con su estilo correspondiente como vemos en la siguiente figura.



Una tabla, dos estilos (o más)

La arquitectura con la que está diseñado el control DataGrid es tan flexible, que partiendo de un DataSet con una única tabla para mostrar en el grid, podemos crear tantas configuraciones de estilo visuales como sea necesario. Otra de sus ventajas reside en que cada estilo creado puede visualizar un grupo distinto de campos de la misma tabla, escenario de trabajo este con el que nos podemos encontrar en alguna ocasión. Los pasos para conseguir un DataGrid con estas características se describen seguidamente.

Supongamos que tenemos un proyecto con el nombre GridTablaDosEstilos, y un formulario con un DataSet que contiene la tabla Suppliers. Partimos del hecho de que en este ejemplo ya hemos creado un primer estilo de tabla en un DataGrid , y vamos a crear uno adicional, usando ambos la misma tabla del DataSet como origen de datos.

Comenzaremos eliminando del estilo de tabla existente en el grid, el valor de la propiedad MappingName; no se preocupe el lector, ya que esta acción no afectará a los estilos de columna definidos. El motivo de esta acción radica en que no podemos tener simultáneamente en un DataGrid dos estilos que apunten a la misma tabla del DataSet.

Continuaremos creando un nuevo estilo en el grid para la misma tabla Suppliers, pero con una configuración que sea distinta del primero; como propuesta, cada uno de los estilos puede estar basado en un diseño de colores diferente: tonos azules y verdes.

Una vez que terminemos el diseño de los estilos, añadiremos dos botones en el formulario para cambiar de uno a otro estilo, y un tercer botón para que el grid muestre los datos sin estilos, con la configuración por defecto del control. La clave de todo, como puede ver el lector en el siguiente código fuente, se encuentra en manipular la propiedad MappingName de la colección de estilos definidos en el DataGrid.

```
Private Sub btnVerdes_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnVerdes.Click
   ' quitar uno de los estilos y asignar el otro
   Me.grdDatos.TableStyles(1).MappingName = ""
   Me.grdDatos.TableStyles(0).MappingName = "Suppliers"
End Sub

Private Sub btnAzules_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAzules.Click
   ' quitar uno de los estilos y asignar el otro
   Me.grdDatos.TableStyles(0).MappingName = ""
   Me.grdDatos.TableStyles(1).MappingName = "Suppliers"
End Sub
```

```
Private Sub btnPredeterminados_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPredeterminados.Click
    ' no utilizar estilos en el grid
    Me.grdDatos.TableStyles(0).MappingName = ""
    Me.grdDatos.TableStyles(1).MappingName = ""
End Sub
```

La siguiente figura muestra el mismo DataGrid visualizando los datos con uno y otro estilo.

La clase DataGridBoolColumn

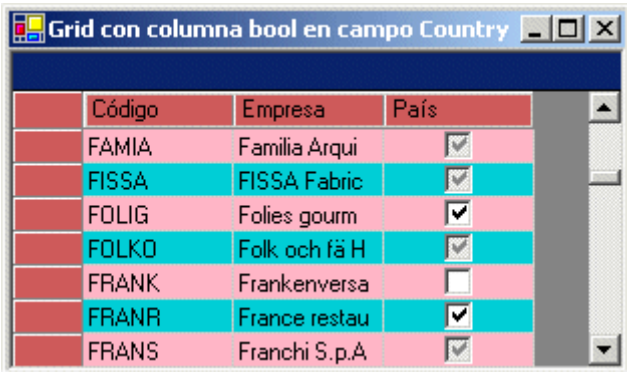
Cuando estamos añadiendo columnas a un estilo de tabla, habremos notado la presencia, junto al botón Agregar, de un botón con una flecha hacia abajo, que al ser pulsado nos ofrece elegir entre crear una columna del ya conocido DataGridTextBoxColumn o DataGridBoolColumn. Esta última clase, que al igual que la primera hereda de DataGridColumnStyle, nos permite mostrar una columna consistente en una casilla de verificación para indicar tres estados: seleccionado, no seleccionado e indeterminado.

Los campos de tipo bit de una base de datos, son un buen candidato para aplicar este estilo de columna, de modo que si mostramos un DataGrid de la tabla Products, el campo Discontinued se mostraría como vemos en la siguiente figura.

Pero no sólo los campos de tipo bit se pueden beneficiar del uso de este estilo de columna, ya que perfectamente podemos aplicarlo a campos de otros tipos, como cadena de caracteres.

Si por ejemplo llenamos un DataGrid con la tabla Customers, y creamos una columna de este estilo para el campo Country, asignando a la propiedad TrueValue el valor

France, y a FalseValue el valor Germany, los registros de la tabla coincidentes con la primera propiedad mostrarán la casilla marcada, mientras que para los coincidentes con la segunda propiedad, la casilla estará vacía; el resto de registros mostrarán la casilla en el estado indeterminado.



Unos estilos con mucha "clase"

Y llegamos al final de nuestro camino confiando en que el lector haya encontrado interesante todo lo que aquí acabamos de mostrar, así pues, esperamos que a partir de ahora, "estilicen" más y mejor sus datos dentro del control DataGrid.