Manipulación de imágenes con ADO.NET

Luis Miguel Blanco Ancos

Manipulando una imagen dentro de un mantenimiento de datos

Cuando nos enfrentamos al desarrollo de un mantenimiento de datos típico en VB.NET contra un proveedor de datos ADO.NET, pongamos como caso SQL Server, todos sabemos, al realizar por ejemplo una inserción en una tabla, el modo de pasar los datos tradicionales a las columnas de la nueva fila, entendiendo como datos tradicionales los correspondientes a cadenas, números y fechas.

Sin embargo la cuestión ya no resulta tan obvia cuando entran en juego tipos de datos como las imágenes, que salen del marco de trabajo de los tipos habituales, por lo que la pregunta a formular sería la siguiente: ¿cómo integrar una imagen en nuestra gestión de datos?

La solución más directa y sencilla pasaría por tener nuestras imágenes en archivos ubicados en una o varias rutas del servidor, y una tabla en la base de datos con un campo de tipo carácter, que contuviera la ruta del archivo.

Esta forma de resolver el problema es ampliamente utilizada y resulta una excelente solución, no obstante, en este artículo nos hemos propuesto complicarnos un poco la vida, y nuestra pretensión no es guardar una cadena con la ruta del archivo que contiene la imagen, sino el contenido de la propia imagen en la base de datos, lo que nos llevaría al siguiente interrogante: ¿cómo grabar una imagen en una base de datos?

El tipo Image de SQL Server

La pregunta que acabamos de plantear nos lleva a adentrarnos en la información que acerca de los tipos de datos existe en la documentación de SQL Server (sistema gestor de bases de datos que utilizaremos a lo largo del artículo), en donde encontramos un elemento específicamente diseñado para nuestros propósitos: el tipo Image, que consiste en un tipo de datos compuesto por información binaria de longitud variable, desde 0 hasta 2^{31}-1 (2.147.483.647) bytes.

Llegados al punto actual ya sabemos dónde hemos de depositar el contenido de una imagen en una base de datos SQL Server, pero ahora surge una nueva incógnita ¿cómo conseguirlo?

¿Qué es una imagen?

Ciñéndonos a la problemática planteada sobre la manipulación de imágenes dentro de la plataforma .NET Framework, podemos definir una imagen como una secuencia de datos binarios o un array de bytes. Detengámonos especialmente en la definición que

acabamos de dar: “secuencia de datos binarios”, y su correspondencia con los elementos que la plataforma .NET pone a nuestra disposición para su manejo. Dentro de la jerarquía de tipos de .NET, la clase que nos permite el trabajo con secuencias o flujos de datos es Stream, como clase abstracta, y todas las clases concretas que derivan de ella (FileStream, MemoryStream, etc.). Por lo tanto, si lo que necesitamos es extraer de un archivo gráfico la secuencia de bytes que contiene, abriremos el archivo con un objeto FileStream y volcaremos su contenido en un array de tipo Byte, este array será en última instancia lo que utilizaremos para grabar la imagen en la base de datos.

El modo de gestión de los datos

ADO.NET es una tecnología que permite el tratamiento de la información bajo una filosofía conectada o desconectada de la fuente de datos con la que trabajemos, en función de las clases que empleemos para dichas tareas de mantenimiento.

En este artículo abordaremos ambos modos de trabajo, de manera que el lector pueda tener una visión de cuál de los dos esquemas conviene mejor a sus necesidades, siendo incluso posible desarrollar un sistema que mezcle ambos.

Como ejemplos ilustrativos de todas las operaciones de mantenimiento de datos a realizar con imágenes y objetos de ADO.NET, hemos desarrollado un proyecto que contiene todos los casos que iremos exponiendo a lo largo de este artículo. Dicho proyecto puede encontrarlo el lector en los materiales de apoyo disponibles para este artículo en la dirección www.dotnetmania.com.

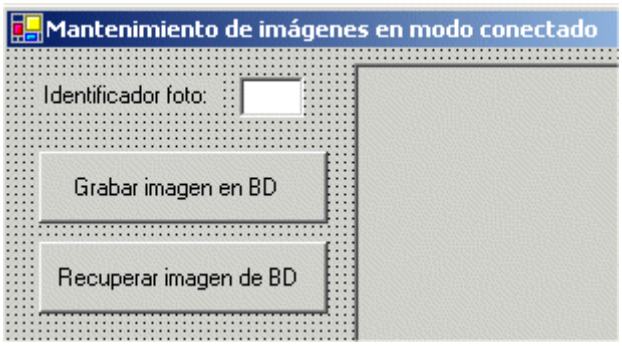
El mencionado proyecto consiste en una aplicación Windows con interfaz MDI, que contendrá dos formularios hijos, uno para los ejemplos con operaciones en modo conectado, y el otro para las operaciones desconectadas. En el código de ambos deberemos importar los siguientes espacios de nombre: System.Data.SqlClient y System.IO, que nos permitirán utilizar los objetos de ADO.NET para el proveedor de SQL Server y los diferentes tipos de stream respectivamente.

El objetivo de la aplicación consiste en almacenar y gestionar las fotos que tomamos con una cámara digital, y que tenemos repartidas en un numeroso conjunto de archivos. Para lo que crearemos una base de datos SQL Server que llamaremos InfoViajes, y que contendrá una tabla con el nombre Fotos y la estructura mostrada en la siguiente tabla.

Campo	**Tipo**
IDFoto	int
Nombre	varchar(50)
Foto	image

Una vez que hemos entrado en situación, comenzaremos con el formulario frmModoConectado, que usaremos para grabar un archivo gráfico en la base de datos,

y posteriormente recuperar una de esas imágenes almacenadas. La siguiente figura muestra una porción de este formulario.



Insertar imágenes en una base de datos utilizando un objeto Command

Tras introducir un número como identificador de foto en la caja de texto del formulario, pulsaremos su botón “Grabar imagen en BD”, que ejecutará el código fuente mostrado a continuación.

```
Private Sub btnGrabar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGrabar.Click
    Dim nIDFoto As Integer
    Dim oFileStream As FileStream
    Dim xDlgResult As DialogResult
    Dim aBytImagen() As Byte
    Dim cnConexion As SqlConnection
    Dim cmdComando As SqlCommand
    Dim parIDFoto As SqlParameter
    Dim parNombre As SqlParameter
    Dim parFoto As SqlParameter
    Dim sSQL As String
    Dim oFileInfo As FileInfo

    ' validaciones
    If Me.txtIDFoto.Text = String.Empty Then
        MessageBox.Show("Introducir identificador de foto")
        Return
    End If

    Try
        nIDFoto = Integer.Parse(Me.txtIDFoto.Text)
    Catch oException As Exception
        Me.txtIDFoto.Text = String.Empty
        MessageBox.Show("El identificador debe ser numérico")
        Return
    End Try
```

```
    ' cuadro de diálogo para seleccionar archivo
    Me.dlgAbrirArchivo.InitialDirectory = Application.StartupPath & "\ArchivosFotos"
    xDlgResult = Me.dlgAbrirArchivo.ShowDialog()

    If xDlgResult = DialogResult.OK Then
        ' abrir el archivo con un objeto stream
        oFileStream = New FileStream(Me.dlgAbrirArchivo.FileName, FileMode.Open)
        ' crear un array byte que tenga el tamaño del archivo
        aBytImagen = New Byte(oFileStream.Length - 1) {}
        ' leer con el stream el contenido del archivo
        ' y volcarlo al array
        oFileStream.Read(aBytImagen, 0, oFileStream.Length - 1)
        oFileStream.Close()

        oFileInfo = New FileInfo(Me.dlgAbrirArchivo.FileName)

        sSQL = "INSERT INTO Fotos VALUES (@IDFoto, @Nombre, @Foto)"
        cnConexion = New SqlConnection("data source=localhost;initial
catalog=InfoViajes;uid=sa;pwd='';")
        cmdComando = New SqlCommand
        cmdComando.Connection = cnConexion
        cmdComando.CommandType = CommandType.Text
        cmdComando.CommandText = sSQL

        parIDFoto = New SqlParameter("@IDFoto", SqlDbType.Int)
        parIDFoto.Value = Convert.ToInt32(Me.txtIDFoto.Text)
        cmdComando.Parameters.Add(parIDFoto)

        parNombre = New SqlParameter("@Nombre", SqlDbType.VarChar, 50)
        parNombre.Value = oFileInfo.Name
        cmdComando.Parameters.Add(parNombre)

        ' para pasar la imagen a la base de datos
        ' definimos un parámetro de tipo Image
        ' y le pasamos el array byte que contiene
        ' la información binaria de la imagen
        parFoto = New SqlParameter("@Foto", SqlDbType.Image)
        parFoto.Value = aBytImagen
        cmdComando.Parameters.Add(parFoto)

        cnConexion.Open()
        cmdComando.ExecuteNonQuery()
        cnConexion.Close()

        MessageBox.Show("Imagen grabada en la base de datos")
    End If
End Sub
```

El proceso que llevamos a cabo al ejecutar este código es el siguiente: tras pasar las oportunas validaciones que comprueban si hemos tecleado un código para la imagen a grabar, abrimos un control OpenFileDialog que nos facilita la tarea de seleccionar el archivo gráfico. Una vez elegido este, lo abriremos haciendo uso de una secuencia representada mediante un objeto FileStream, y dimensionaremos un array de bytes con un tamaño igual a la longitud del stream. Seguidamente leeremos el contenido del stream y lo volcaremos en el array, con lo que ya tendremos la información binaria de la imagen en un formato que nos permita tratarlo para su inserción en la base de datos.

A continuación construiremos la conexión, el comando, y la cadena con la sentencia SQL a enviar a la base de datos. También definiremos tantos objetos Parameter como campos a grabar; en el caso concreto de la imagen, observemos que el parámetro lo creamos especificando como tipo de dato SqlDbType.Image, correspondiente a la enumeración que nos devuelve los tipos disponibles para SQL Server; el valor que pasamos al parámetro será el array de bytes que contiene la información binaria de la imagen.

Por último, abrimos la conexión, ejecutamos el comando y cerramos la conexión. Si todo ha funcionado correctamente, tendremos la imagen grabada en un registro de la tabla. Para comprobarlo podemos ejecutar una consulta hacia dicha tabla desde el Analizador de consultas de SQL Server, obteniendo un resultado como el que vemos en la siguiente figura.

Obviamente no veremos en el campo Foto la imagen tal y como estamos acostumbrados, sino como una sucesión de datos binarios, pero que son suficientemente indicativos de que el contenido del archivo ha sido grabado.

¿Y si utilizamos el proveedor de OLEDB?

Aunque este artículo está orientado hacia uso del proveedor de datos para SQL Server, si el lector se encuentra ante la necesidad de gestionar imágenes residentes en un origen de datos para el que deba de usar el proveedor de OLEDB, la mecánica a seguir sería la misma que acabamos de explicar, utilizando como es natural, los objetos de ADO.NET específicos para este proveedor de datos: OleDbConnection, OleDbCommand, OleDbParameter, etc. Veamos en el siguiente fuente una adaptación del ejemplo anterior pero usando el proveedor de OLEDB.

'....

```
' apertura archivo con FileStream y volcado a array byte
'....
sSQL = "INSERT INTO Fotos VALUES (@IDFoto, @Nombre, @Foto)"

cnConexion = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\Pruebas\InfoViajes.mdb;Password=;")

cmdComando = New OleDbCommand
cmdComando.Connection = cnConexion
cmdComando.CommandType = CommandType.Text
cmdComando.CommandText = sSQL

parIDFoto = New OleDbParameter("@IDFoto", OleDbType.Integer)
parIDFoto.Value = 5
cmdComando.Parameters.Add(parIDFoto)

parNombre = New OleDbParameter("@Nombre", OleDbType.VarChar, 50)
parNombre.Value = "karnak1.jpg"
cmdComando.Parameters.Add(parNombre)

parFoto = New OleDbParameter("@Foto", OleDbType.LongVarBinary)
parFoto.Value = aBytImagen
cmdComando.Parameters.Add(parFoto)

cnConexion.Open()
cmdComando.ExecuteNonQuery()
cnConexion.Close()
```

Recuperar imágenes desde una base de datos mediante los objetos Command y DataReader

Una vez que hemos grabado varias imágenes en la base de datos, vamos a efectuar la operación inversa, por lo que tras introducir un número identificador de foto en la caja de texto del formulario, pulsaremos su botón “Recuperar imagen de BD”, que ejecutará el código fuente mostrado a continuación.

```
Private Sub btnRecuperar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRecuperar.Click
    Dim nIDFoto As Integer
    Dim xDlgResult As DialogResult
    Dim sSQL As String
    Dim cnConexion As SqlConnection
    Dim cmdComando As SqlCommand
    Dim drFotos As SqlDataReader
    Dim sNombreFoto As String
    Dim aBytImagen() As Byte
    Dim oFileStream As FileStream
```

```
    Dim oMemoryStream As MemoryStream
    Dim bmpImagen As Bitmap

    ' validaciones
    If Me.txtIDFoto.Text = String.Empty Then
        MessageBox.Show("Introducir identificador de foto")
        Return
    End If

    Try
        nIDFoto = Integer.Parse(Me.txtIDFoto.Text)
    Catch oException As Exception
        Me.txtIDFoto.Text = String.Empty
        MessageBox.Show("El identificador debe ser numérico")
        Return
    End Try

    ' crear sentencia, conexión y comando para obtener la imagen de la base de datos
    sSQL = "SELECT Nombre, Foto FROM Fotos WHERE IDFoto = " & Me.txtIDFoto.Text
    cnConexion = New SqlConnection("data source=localhost;initial
catalog=InfoViajes;uid=sa;pwd='';")
    cmdComando = New SqlCommand
    cmdComando.Connection = cnConexion
    cmdComando.CommandType = CommandType.Text
    cmdComando.CommandText = sSQL

    cnConexion.Open()
    drFotos = cmdComando.ExecuteReader(CommandBehavior.SingleRow)

    If drFotos.Read() Then
        sNombreFoto = drFotos("Nombre")
        ' recuperar datos binarios de la foto
        aBytImagen = drFotos("Foto")
    End If

    drFotos.Close()
    cnConexion.Close()

    If IsNothing(sNombreFoto) Then
        MessageBox.Show("No hay foto con ese identificador")
        Return
    End If

    ' abrir cuadro de diálogo para grabar la foto en un archivo
    Me.dlgGrabarArchivo.InitialDirectory = Application.StartupPath &
"\FotosRecuperadasBD\"
    Me.dlgGrabarArchivo.FileName = sNombreFoto
```

```
    xDlgResult = Me.dlgGrabarArchivo.ShowDialog()

    If xDlgResult = DialogResult.OK Then
        ' crear un objeto stream de tipo archivo y escribir en él
        ' el array byte que contiene los datos binarios de la imagen
        oFileStream = New FileStream(Me.dlgGrabarArchivo.FileName, _
            FileMode.CreateNew, FileAccess.Write)
        oFileStream.Write(aBytImagen, 0, aBytImagen.Length)
        oFileStream.Close()
    End If

    ' crear un objeto stream en memoria conteniendo los datos de la imagen,
    ' crear un bitmap con el stream y
    ' visualizar la imagen en un control PictureBox
    oMemoryStream = New MemoryStream(aBytImagen)
    bmpImagen = New Bitmap(oMemoryStream)
    Me.picFoto.Image = bmpImagen
End Sub
```

En esta ocasión, el proceso ejecutado consiste en crear una conexión y comando, cuya sentencia SQL contenga el identificador de fila a recuperar, que habremos introducido en el formulario. Observe el lector que al crear el comando, lo configuramos utilizando la enumeración CommandBehavior para optimizar su ejecución, de modo que devuelva un DataReader compuesto por una única fila.

Una vez obtenido el DataReader, su campo Foto contendrá una serie de bytes que representan los datos binarios de la imagen, y que volcaremos en un array, también de tipo byte.

A partir de aquí podemos tratar la imagen como necesitemos; en este ejemplo realizamos dos operaciones: en primer lugar pasamos el array a un objeto FileStream, creando de esta manera un archivo gráfico, ayudándonos de un cuadro de diálogo SaveFileDialog. En segundo lugar creamos un objeto MemoryStream con el array, y a partir del stream, creamos un objeto Bitmap, que asignamos al control PictureBox del formulario para visualizar la imagen; si no necesitamos transferir la imagen a un archivo, es mucho más efectivo tratarla en memoria con el stream disponible a tal efecto. La siguiente figura muestra este formulario en ejecución.

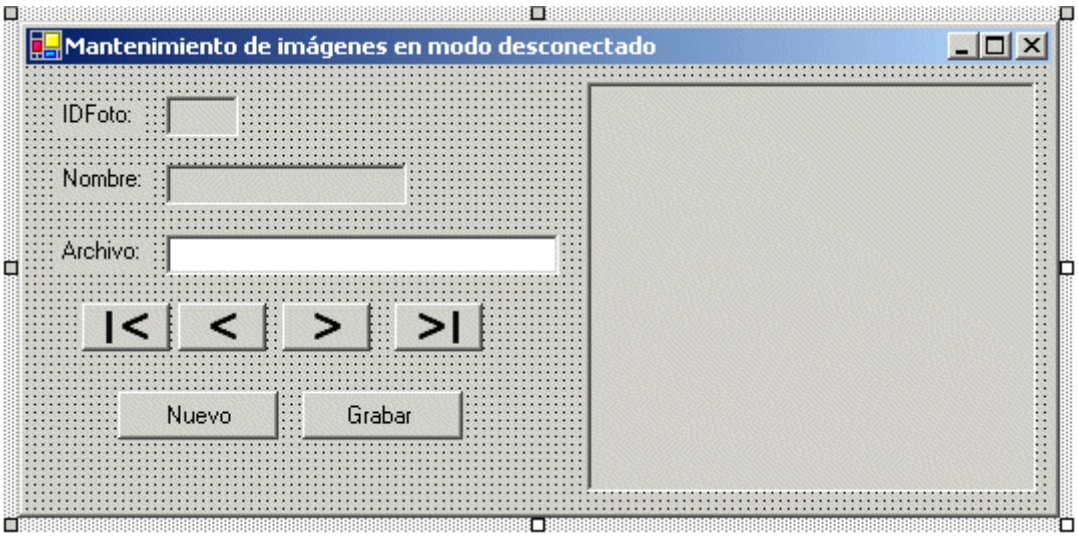
Llegados a este punto, podemos dar por concluida la fase de gestión de imágenes en ADO.NET desde un enfoque conectado a la fuente de datos, es hora pues de abordar una técnica diferente: el modo desconectado.

Manipulando imágenes mediante un DataSet

DataSet representa el objeto central sobre el cual se sustenta la arquitectura de gestión de datos desconectados en ADO.NET. La forma de tratar las imágenes es igual que con los objetos Command y DataReader, salvando claro está, las diferencias entre estos tipos de objeto en función de su natural operativa (conectada o desconectada).

El ejemplo con DataSet que acompañamos a continuación, además de las operaciones de edición de imágenes, contiene la capacidad de navegación por las filas de la tabla, constituyendo un factor adicional que le diferencia del anterior ejemplo basado en comandos; el resultado conseguido consiste en un visualizador de imágenes, que aunque sencillo, cumple correctamente su misión.

El formulario de ejemplo encargado de tratar con el DataSet es frmDesconectado, que podemos ver en la siguiente figura.



En este artículo asumimos que el lector conoce la mecánica básica para crear y llenar un DataSet con datos, por lo que obviaremos aquellas líneas de código pertenecientes a este formulario encargadas de esta tarea y de la navegación de registros, centrándonos exclusivamente en los procesos que competen al tratamiento de imágenes con este objeto. Debemos aclarar también, que para facilitar la programación de las operaciones en el formulario, algunos elementos tales como el DataSet, las variables de control de la posición de fila, filas totales, etc., han sido declarados con ámbito a nivel de la clase.

Tras poblar el DataSet con el contenido de la tabla Fotos en el evento Load del formulario, llamaremos al método CargarDatos, que toma la fila en la que el DataSet se encuentra posicionado, y la muestra en los controles del modo que vemos en el siguiente código fuente.

```
Private Sub CargarDatos()
   Dim drFila As DataRow
   Dim aBytImagen() As Byte
   Dim bmpImagen As Bitmap
   Dim oMemoryStream As MemoryStream

   ' obtener el objeto DataRow de la fila actual,
   ' y recuperar los valores de sus campos
   drFila = Me.dsInfoViajes.Tables("Fotos").Rows(nFilaActual)
   Me.txtIDFoto.Text = CType(drFila("IDFoto"), String)
   Me.txtNombre.Text = CType(drFila("Nombre"), String)
   ' el campo de imagen consiste en información binaria (bytes),
   ' volcarlo a un array de tipo byte
   aBytImagen = CType(drFila("Foto"), Byte())

   ' crear un objeto stream en memoria a partir del array byte
   oMemoryStream = New MemoryStream(aBytImagen)
   ' crear un objeto bitmap a partir del stream
   bmpImagen = New Bitmap(oMemoryStream)
   ' pasar el objeto bitmap al control PictureBox del formulario
   Me.picFoto.Image = bmpImagen
   oMemoryStream.Close()
End Sub
```

Como acabamos de comprobar, la operación realizada para obtener la imagen es la misma que para un DataReader: pasamos el contenido del campo de la tabla a un array de tipo byte, y utilizamos este para crear un stream en memoria; finalmente, creamos un bitmap con el stream, y asignamos la imagen a un control PictureBox. A partir de aquí, al pulsar los botones de navegación, se actualizará el número de fila de la tabla del DataSet a mostrar, volviendo a llamar a este mismo método para visualizar la imagen, como vemos en la siguiente figura.

En lo que respecta a la grabación de un archivo gráfico empleando este DataSet, pulsaremos el botón Nuevo para habilitar los controles correspondientes del formulario. Después de teclear el número identificador y la ruta del archivo, pulsaremos el botón Grabar, que ejecutará el siguiente código fuente.

```
Private Sub btnGrabar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGrabar.Click
    Dim oFileStream As FileStream
    Dim aBytImagen() As Byte
    Dim drFila As DataRow

    ' abrir el archivo con un stream y volcar en un array
    oFileStream = New FileStream(Me.txtArchivo.Text, FileMode.Open)
    aBytImagen = New Byte(oFileStream.Length - 1) {}
    oFileStream.Read(aBytImagen, 0, oFileStream.Length - 1)
    oFileStream.Close()

    ' crear un objeto DataRow, asignar valores a sus campos
    ' y añadirlo al dataset
    drFila = dsInfoViajes.Tables("Fotos").NewRow()
    drFila("IDFoto") = Integer.Parse(Me.txtIDFoto.Text)
    drFila("Nombre") = Path.GetFileName(Me.txtArchivo.Text)
    drFila("Foto") = aBytImagen
    dsInfoViajes.Tables("Fotos").Rows.Add(drFila)

    ' actualizar el dataset en la base de datos
    ' y volverlo a llenar para que el orden de las
    ' filas sea el adecuado
    cnConexion.Open()
    daAdaptador.Update(dsInfoViajes, "Fotos")
    dsInfoViajes.Clear()
    daAdaptador.Fill(dsInfoViajes, "Fotos")
    cnConexion.Close()
```

```
    ' actualizar las variables de control del número de filas
    nFilasTotales = Me.dsInfoViajes.Tables("Fotos").Rows.Count - 1
    nFilaActual = nFilasTotales

    Me.ControlesNavegar()
    Me.CargarDatos()
End Sub
```

La manera de grabar el archivo gráfico sobre la base de datos es, en esencia, la misma que en el ejemplo con objetos conectados, ya que abrimos el archivo con un FileStream, y volcamos este a un array de bytes; a continuación creamos un nuevo objeto DataRow y asignamos los valores para sus campos, añadiendo este objeto a la colección de filas de la tabla del DataSet. Finalmente actualizaremos la base de datos física con el DataSet y rellenaremos este último de nuevo, para que el orden de filas sea el correcto durante la navegación de registros por el formulario.

Y llegamos al final

En efecto estimado lector, por mucho que nos pese toda narración tiene su fin, y nuestro artículo no iba a ser menos, esperamos que el tema tratado en esta ocasión le sea de utilidad si se encuentra ante la tesitura de crear un mantenimiento de datos en el que intervengan imágenes.