

# informatizate

<http://www.informatizate.net>

## Almacenamiento Multimedia en Base de Datos Relacionales con ASP.NET - [Descargar demo](#)

Victor Hugo Lamadrid Mendoza (\*)

Analísta - Desarrollador Web - [OSIPTEL](#) [victor\\_lamadrid\(at\)informatizate\(dot\)net](mailto:victor_lamadrid(at)informatizate(dot)net)  
Ing. Informático  
Miembro Fundador de [informatizate](#)

[victor\\_lamadrid\(at\)informatizate\(dot\)net](mailto:victor_lamadrid(at)informatizate(dot)net)

Julio 5 del 2004.



### Introducción

Algunas veces se presenta la necesidad de almacenar cierto tipo de información como imágenes, documentos internos hechos en procesadores de texto como Microsoft Word, hojas de cálculo, diapositivas e incluso archivos de video de pequeño tamaño en un lugar que no sea el sistema de archivos del disco de un servidor de archivos, ya sea porque se carece de espacio en disco o porque es indispensable cumplir con un requerimiento de negocio. A menos que agrade se guste poner a prueba su servidor de base de datos para almacenar este tipo de información.

No son solo estas circunstancias las que nos hacen abrir los ojos para diferenciar que existe información de carácter estructurado e información no estructurada. Los términos "estructurada" o "no estructurada" son usados aquí y corresponden a información simple e información compleja. En términos de base de datos y almacenamiento podemos llamar información simple a un campo en una tabla que guarda el nombre de un empleado y para la cual existe un tipo de dato básico como una cadena de caracteres. Así en los términos anteriores el tipo de información compleja sería la correspondiente a la almacenada en un campo que guarda la fotografía del empleado y para la cual existen tipos de datos especiales implementados por las distintas base de datos del mercado ya sea de escritorio como las corporativas.

Las clases que constituyen la tecnología ASP.NET y ADO.NET, nos brindan la posibilidad de capturar este tipo de información no estructurada y poderla almacenar en una base de datos relacional ya sea de escritorio como Microsoft Access o en una corporativa como Microsoft SQL Server 2000 ó en su versión 7.0.

### Ventajas y desventajas

Seguro ha pasado por su mente la pregunta: ¿De que me serviría tener este tipo de información como archivos de MS Word, MS Excel, imágenes e incluso videos de pequeño tamaño almacenados en una tabla de una base de datos relacional?

Pues como toda solución para satisfacer los requerimientos de las personas a quienes nos debemos, es decir nuestros amigos usuarios, tiene sus beneficios. Uno de ellos es que tanto información "estructurada" como "no estructurada" que estén relacionadas lógicamente puedan estar almacenadas en un mismo repositorio y formando parte de un registro en una base de datos, así permanecerán siempre juntas aun si la base de datos cambiara de lugar o de servidor. Un ejemplo es la posibilidad de tener en una tabla llamada Empleado información como el nombre, apellidos, dirección, etc. y la fotografía del mismo en la misma tabla, incluso su hoja de vida originalmente en un archivo de MS Word almacenada junto con toda la información antes mencionada.

Otro beneficio es la posibilidad de almacenar documentos de carácter oficial como Normas, Resoluciones, etc. en su versión definitiva para asegurar la unicidad de los mismos y para que no estén expuestos a cambios o dañen su integridad, como ven, la seguridad de la información también entra a tallar en este aspecto. Para estos casos las bases de datos corporativas manejan mecanismos de protección que permiten autorizar o denegar el cambio de ciertos campos a determinados usuarios de base de datos, el tema de seguridad cubre varios niveles por lo que también se pueden implementar mecanismos de seguridad en el nivel de la aplicación web.

Habiendo enumerado algunos de los beneficios de esta solución, ahora toca ver el otro lado de la moneda y es que la principal desventaja de esta solución es el consumo de espacio en la base de datos por lo que es recomendable no abusar en almacenar tanta información "no estructurada" ya que generalmente esta se almacena como flujos de bytes en campos de tipo binarios para los cuales se mantiene metada en una proporción mayor a la usada para los tipos de datos simples.

### Implementación

Pasemos ahora a desarrollar la solución exponiendo y explicando a grandes rasgos porciones de código de la demo, la cual está disponible para descargar aquí [Código Fuente](#) en toda su integridad, cabe resaltar que el lenguaje utilizado es Visual Basic.NET y la base de datos de ejemplo es de Microsoft Access XP (Access 2002) aunque pueden probar con Access 2000 manteniendo claro está, la estructura de la tabla de ejemplo. Los requisitos mínimos para su correcto funcionamiento son los mismos que para cualquier aplicación web hecha en .NET. Es decir, se necesita el Framework NET 1.0 sobre Windows 2000 Profesional o Server mínimo con SP2 y tener correctamente instalado como mínimo el IIS 5.0.

En la figura 1. se muestra el formulario usado para insertar el archivo que se desea almacenar, este puede ser: un documento de MS Word, una hoja de cálculo de MS Excel, una presentación de MS Power Point, un documento PDF, una página html, archivos de imágenes (GIF, JPG, BMP) o un video de pequeño tamaño para este caso ya sea en formato AVI o MPG.

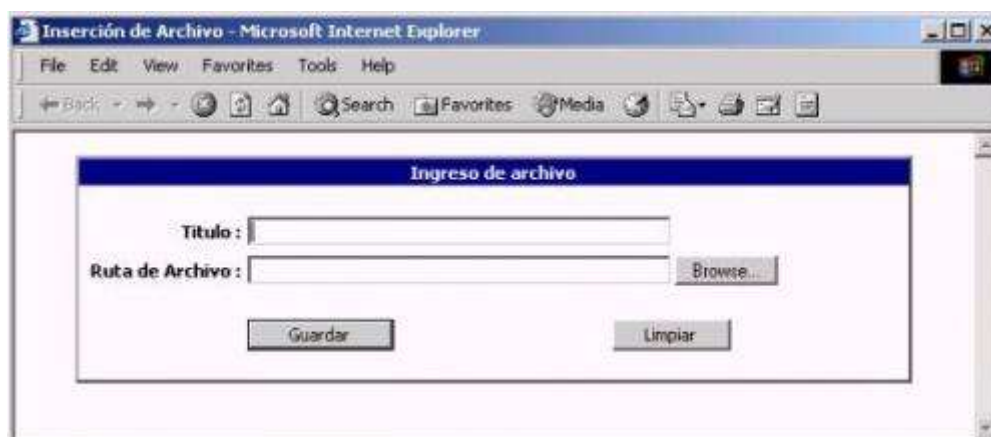


Figura 1. Formulario web usado para insertar contenido multimedia en la base de datos.

Es el turno de describir el código utilizado para almacenar el contenido del archivo especificado en el formulario de ingreso y el resto de datos ingresados. Este se encuentra en el archivo frmInsertar.aspx.vb.

```
Imports System.Data.OleDb
Imports System.IO
```

Las dos líneas anteriores nos permiten hacer uso de los namespaces OleDb para el acceso a datos y de IO para manipular el contenido del archivo a almacenar. El código que se presenta a continuación es el ejecutado cuando se lanza el evento click del botón Guardar

```
Private Sub cmdGuardar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles cmdGuardar.Click

Dim SQL As String
Dim Titulo As String
Dim Ruta As String
Dim Tipo As String
Dim FlujoBinario As System.IO.Stream

Titulo = txtTitulo.Value.ToString
Ruta = txtRuta.Value.ToString

Dim constr As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
Server.MapPath(".") & "\" & "BDTEST.mdb;"
Dim con As OleDbConnection = New OleDbConnection(constr)
Dim cmd As OleDbCommand = New OleDbCommand()
Dim prm As OleDbParameter = New OleDbParameter()

con.Open()

Try

Ruta = Request.Files(0).FileName
Ruta= Request.Files(0).FileName.Substring(Request.Files(0).FileName.LastIndexOf("\") + 1)
Tipo = Ruta.Substring(Ruta.LastIndexOf(".") + 1, Ruta.Length - (Ruta.LastIndexOf(".") + 1))

FlujoBinario = Request.Files(0).InputStream

Dim Contenido As Byte()
ReDim Contenido(FlujoBinario.Length)

FlujoBinario.Read(Contenido, 0, Convert.ToInt32(FlujoBinario.Length))

SQL = "INSERT INTO Documento (Titulo, Tipo ,Contenido) VALUES (' & _
Titulo & "', ' & Tipo & "', ContenidoDB )"

cmd.CommandType = CommandType.Text
cmd.CommandText = SQL
cmd.Connection = con
prm.ParameterName = "ContenidoDB"
prm.DbType = DbType.Binary
prm.Value = Contenido

cmd.Parameters.Add(prm)
cmd.ExecuteNonQuery()

Response.Redirect("frmListar.aspx")

Catch er As OleDbException

Response.Write("Descripción del error : " & er.Message)

Finally

con.Close()

End Try

End Sub
```

Ahora se pasa a explicar algunas porciones de código del bloque anterior

```
FlujoBinario = Request.Files(0).InputStream

Dim Contenido As Byte()
ReDim Contenido(FlujoBinario.Length)

FlujoBinario.Read(Contenido, 0, Convert.ToInt32(FlujoBinario.Length))
```

En la variable de objeto FlujoBinario almacenamos el contenido del archivo transferido, este contenido será depositado en un arreglo cuyos elementos son octetos de bits llamado Contenido, para ello se utiliza la sentencia ReDim para ajustar el tamaño del arreglo y poder almacenar todo el contenido, el traspaso del contenido se realiza en la última línea con el uso del método Read.

```
SQL = "INSERT INTO Documento (Titulo, Tipo ,Contenido) _
VALUES ('" & Titulo & "', '" & Tipo & "', ContenidoDB )"

cmd.CommandType = CommandType.Text
cmd.CommandText = SQL
cmd.Connection = con

prm.ParameterName = "ContenidoDB"
prm.DbType = DbType.Binary
prm.Value = Contenido

cmd.Parameters.Add(prm)
cmd.ExecuteNonQuery()
```

Una vez llena la variable Contenido se pasa a elaborar la consulta de inserción, usando para ello una consulta parametrizada, para ello se crean objetos Command y Parameter. Con el último objeto se puede especificar el nombre del parámetro, el tipo de dato de este y su valor. Como verán el valor ha sido establecido con el valor de la variable Contenido en donde se depositó el contenido binario.

Realizado lo anterior, ya podemos adjuntar el parámetro al comando y ejecutar la sentencia de inserción. Para tal efecto el campo en la base de datos Access que almacena el contenido ha sido fijado con el tipo de dato Objeto OLE. Junto con el contenido del archivo almacenamos el título del archivo ingresado y el tipo de archivo, este último dato nos será de utilidad para la tarea de recuperación.

Si se tratase de una base de datos SQL Server, la convención para la denominación de un parámetro impone el uso del carácter @ como prefijo para cualquier parámetro, así en ese caso el parámetro sería @ContenidoDB y el tipo de dato recomendado a usar en la base de datos para almacenar la información sería Image. A parte se supone que el namespace utilizado para acceder a datos ya no sería OleDb sino SqlClient.

Oracle ofrece un tipo de dato denominado BLOB para almacenar este tipo de información. Dejamos como tarea reescribir el código si se deseara utilizar como base de datos una de Oracle.

Una vez almacenado el contenido del archivo transferido a la base de datos, podemos recuperarlo y mostrarlo en una página web. El código que realiza la tarea de recuperación se encuentra en el archivo frmRecuperar.aspx.vb y es el que se muestra a continuación:

```

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

Dim ID As String
Dim SQL As String

Dim constr As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
Server.MapPath(".") & "\" & "BDTEST.mdb;"

Dim con As OleDbConnection = New OleDbConnection(constr)

con.Open()

Try
    ID = Request.QueryString("ID")

    SQL = "SELECT Contenido FROM Documento WHERE IDDocumento=" & ID

    Dim cmd1 As OleDbCommand = New OleDbCommand()

    cmd1.CommandText = SQL
    cmd1.Connection = con
    cmd1.CommandType = CommandType.Text

    SQL = "SELECT Tipo FROM Documento WHERE IDDocumento=" & ID

    Dim cmd2 As OleDbCommand = New OleDbCommand()
    Dim extension As String
    Dim tipo As String

    cmd2.CommandText = SQL
    cmd2.Connection = con
    cmd2.CommandType = CommandType.Text
    cmd2.ExecuteNonQuery()

    Dim da As OleDbDataAdapter = New OleDbDataAdapter(cmd2)
    Dim ds As DataSet = New DataSet()

    da.Fill(ds)
    extension = ds.Tables(0).Rows(0)("Tipo")

    Select Case extension
    Case "txt"
        tipo = "text/html"
    Case "doc"
        tipo = "application/msword"
    Case "xls"
        tipo = "application/msexcel"
    Case "pdf"
        tipo = "application/pdf"
    Case "avi"
        tipo = "video/avi"
    End Select

    Dim dr As OleDbDataReader = cmd1.ExecuteReader

    If (dr.Read) Then
        Response.Clear()
        Response.Buffer = True
        Response.ContentType = tipo
        Response.AddHeader("Content-Type", tipo)
        Response.AddHeader("Content-Disposition", "filename=" & Now().ToString & "." &
extension)
        Response.BinaryWrite(CType(dr("Contenido"), Byte()))
    End If

Catch er As OleDbException

    Response.Write("Mensaje de error : " & er.Message)

Finally
    con.Close()
End Try

End Sub

```

El código del bloque anterior se ejecuta al ser invocada la página frmRecuperar.aspx por lo que está incluido en el evento Page\_Load, a esta página se le pasa un parámetro que viene a ser el código del documento que queremos consultar para ver su contenido.

Explicaremos a continuación algunas porciones de código del bloque anterior.

```
ID = Request.QueryString("ID")
SQL = "SELECT Contenido FROM Documento WHERE IDDocumento=" & ID

Dim cmd1 As OleDbCommand = New OleDbCommand()
cmd1.CommandText = SQL
cmd1.Connection = con
cmd1.CommandType = CommandType.Text
```

Recuperamos el código pasado del documento que queremos visualizar y formamos nuestro comando para recuperar el contenido del archivo almacenado, pero aún no lo ejecutamos.

```
SQL = "SELECT Tipo FROM Documento WHERE IDDocumento=" & ID

Dim cmd2 As OleDbCommand = New OleDbCommand()
Dim extension As String
Dim tipo As String

cmd2.CommandText = SQL
cmd2.Connection = con
cmd2.CommandType = CommandType.Text
cmd2.ExecuteNonQuery()

Dim da As OleDbDataAdapter = New OleDbDataAdapter(cmd2)
Dim ds As DataSet = New DataSet()
da.Fill(ds)
extension = ds.Tables(0).Rows(0)("Tipo")
```

Luego se pasa a recuperar el tipo de archivo que se ha almacenado, para ello usamos otro objeto comando, ejecutamos la sentencia, obtenemos el dato y lo almacenamos en la variable extension. Según el tipo de extensión se deduce el tipo de archivo, así podemos especificar el tipo de contenido que devolverá el servidor web al browser del cliente.

```
Select Case extension
Case "txt"
    tipo = "text/html"
Case "doc"
    tipo = "application/msword"
Case "xls"
    tipo = "application/msexcel"
Case "pdf"
    tipo = "application/pdf"
Case "avi"
    tipo = "video/avi"
End Select
```

El tipo de contenido a devolver queda especificado en la variable tipo, esta variable será usada para especificar en la cabecera de la respuesta el tipo de contenido que el servidor web devolverá. Para mayor información sobre información de cabecera en donde especificar el tipo de contenido a devolver, visitar: <http://reliableanswers.com/contenttype/CType.asp>. Allí se puede encontrar una surtida lista de los tipos de contenido que se pueden especificar.

```
Dim dr As OleDbDataReader = cmd1.ExecuteReader

If (dr.Read) Then

    Response.Clear()
    Response.Buffer = True
    Response.ContentType = tipo
    Response.AddHeader("Content-Type", tipo)
    Response.AddHeader("Content-Disposition", "filename=" & Now().ToString &
"." & extension)

    Response.BinaryWrite(CType(dr("Contenido"), Byte()))

End If
```

Es el momento de ejecutar el primer comando que definimos y obtener de este un objeto DataReader, comprobamos si tiene información, armamos la información de cabecera de la respuesta y finalmente devolvemos el contenido que tenemos en el campo "Contenido" en forma de un arreglo de octetos al usar Byte() y con el uso del método BinaryWrite.

De esta forma hemos cubierto tanto la forma como almacenar y recuperar este tipo de información, así como las ventajas y desventajas de esta solución.

¡ Feliz Programación!

## Referencias

File Uploading Using ASP.NET  
(<http://www.developer.com/net/vb/article.php/3097661>)

Content-Type Listings (<http://reliableanswers.com/contenttype/CType.asp>)