**FileUpload Control**

¿Cómo se guarda y muestra imágenes o archivos binarios que suben desde el cliente al servidor?

|  |  |
| --- | --- |
| **Ventajas y Desventajas del almacenamiento** | |
| **Almacenar en sistema de archivos (file system)** | **Almacenar dentro de la base de datos** |
|  |  |
|  |  |
|  |  |

**Usar OPENROWSET para insertar de forma masiva datos de archivo en una columna varbinary(max)**

En el ejemplo siguiente se crea una tabla pequeña como ejemplo y se insertan datos de archivo desde un archivo llamado Text1.txt ubicado en el directorio raíz C: en una columna varbinary(max).

SQLCopiar

CREATE TABLE myTable(FileName nvarchar(60),

FileType nvarchar(60), Document varbinary(max));

GO

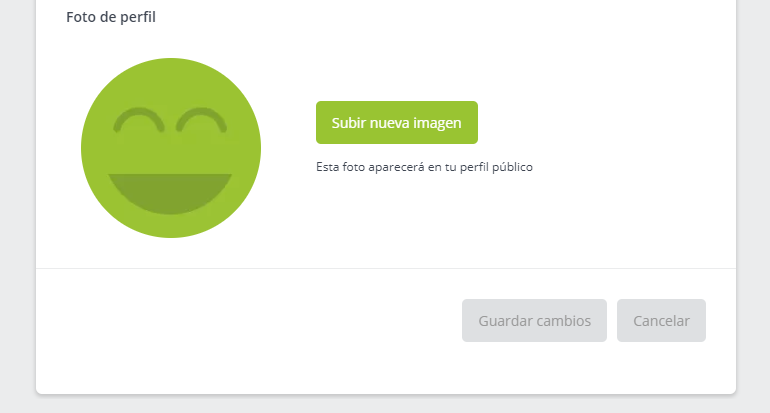
INSERT INTO myTable(FileName, FileType, Document)

SELECT 'Text1.txt' AS FileName, '.txt' AS FileType,

\* FROM OPENROWSET(BULK N'C:\Text1.txt', SINGLE\_BLOB) AS Document;

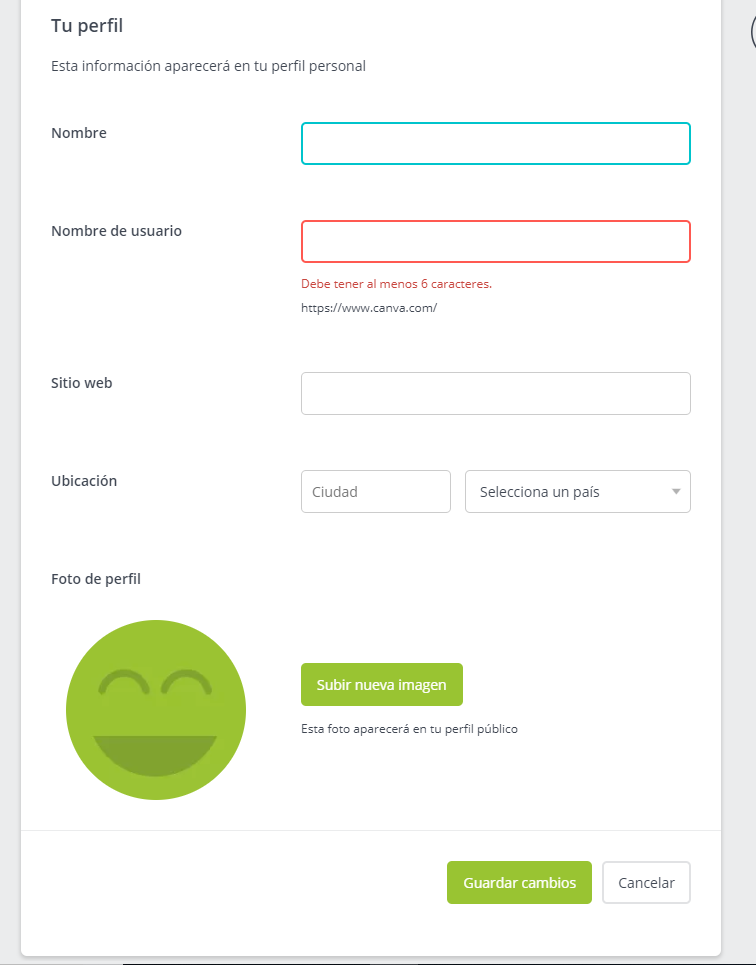
**Retos: Perfil de Usuario**

Actividad 1: Proporcionar la funcionalidad para subir un archivo y guardarlo en (1) un sistema de archivo y (2) una base de datos.



Actividad 2: Perfil más completo

Añadir más funcionalidad al perfil, incluyendo restricciones de (1) tipo de archivo, (2) tamaño máximo de archivo, (3) vinculando la tabla con una tabla de Usuarios….



**Recursos:**

<https://docs.microsoft.com/es-es/dotnet/api/system.web.ui.webcontrols.fileupload?view=netframework-4.7.2>

<https://docs.microsoft.com/es-es/dotnet/api/system.web.ui.webcontrols.fileupload.postedfile?view=netframework-4.7.2>

<https://www.aspsnippets.com/Articles/Save-Files-to-SQL-Server-Database-using-FileUpload-Control.aspx>

NOTAS:

PASOS:

1. Image v binary v varbinary(max) tipo de datos (viejo, 8000kb, 2 GB)
2. Insertar (subir con código transact-SQL)
3. Página web
   1. Guardar en sistema de archivos
   2. Guardar en base de datos
   3. Ver archivo
      1. Con código subyacente
      2. Con GridView y SQLConnect control
4. FILESTREAM atributo

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

using System.Data;

using System.Data.SqlClient;

public partial class AlmacenarImagenesBD : System.Web.UI.Page

{

protected void cargarImagen\_Click(object sender, EventArgs e)

{

try

{

if (fileUploader1.HasFile)

{

// Se verifica que la extensión sea de un formato válido

string ext = fileUploader1.PostedFile.FileName;

ext = ext.Substring(ext.LastIndexOf(".") + 1).ToLower();

string[] formatos = new string[] { "jpg", "jpeg", "bmp", "png", "gif" };

if (Array.IndexOf(formatos, ext) < 0)

MensajeError("Formato de imagen inválido.");

else if (disco.Checked)

GuardarArchivo(fileUploader1.PostedFile);

else

GuardarBD(fileUploader1.PostedFile);

}

else

MensajeError("Seleccione un archivo del disco duro.");

}

catch (Exception ex)

{

MensajeError(ex.Message);

}

}

private void GuardarArchivo(HttpPostedFile file)

{

// Se carga la ruta física de la carpeta temp del sitio

string ruta = Server.MapPath("~/temp");

// Si el directorio no existe, lo creamos

if (!Directory.Exists(ruta))

Directory.CreateDirectory(ruta);

string archivo = String.Format("{0}\\{1}", ruta, file.FileName);

// Verificamos que el archivo no exista

if (File.Exists(archivo))

MensajeError(String.Format(

"Ya existe una imagen con el nombre\"{0}\".", file.FileName));

else

{

file.SaveAs(archivo);

}

}

private static void GuardarBD(HttpPostedFile file)

{

// Nombre de la imagen

string nombre = file.FileName.Substring(0, file.FileName.LastIndexOf("."));

// Tipo de contenido

string contentType = file.ContentType;

// Imagen convertida a un array de bytes

byte[] imagen = new byte[file.InputStream.Length];

file.InputStream.Read(imagen, 0, imagen.Length);

// Se insertan los valores en la base de datos

String strConnString = System.Configuration.ConfigurationManager.ConnectionStrings["IgnacioConnectionString"].ConnectionString;

SqlConnection cnx = new SqlConnection(strConnString);

try {

cnx.Open();

SqlCommand cmd = cnx.CreateCommand();

cmd.CommandText =

"INSERT INTO fotos (nombre, foto, formato) " +

"VALUES (@nombre, @imagen, @contentType)";

cmd.Parameters.AddWithValue("@nombre", nombre);

cmd.Parameters.AddWithValue("@imagen", imagen);

cmd.Parameters.AddWithValue("@contentType", contentType);

cmd.ExecuteNonQuery();

}

catch (Exception ex) {

throw ex;

}

finally {

if (cnx != null) {

if (cnx.State == ConnectionState.Open)

cnx.Close();

cnx.Dispose();

}

}

}

void MensajeError(string mensaje)

{

lblmensaje.Text = mensaje;

}

}

NOTAS del profesor:

objetos binarios grandes (blobs)

BLOBS – superar 8kb (tipo de datos binario también para archivos pero se guarda poco).

BLOBs (binario), CLOBs (archivos con datos de caracters)

Image tipo de dato es viejo

Varbinary – max 2GB

Ventajas:  
1. Seguridad. Los archivos no están a simple vista, no se pueden ejecutar desde el sistema si se sube algún tipo de archivo malicioso.  
2. Único origen de datos. No tienes datos distribuidos en el sistema de archivos y la base de datos, todo está centralizado en un mismo punto y eso permite que sea más fácil importar y exportar tu información a diferente lado.  
3. Dependiendo del tipo de archivo puedes aprovechar las herramientas de búsqueda del motor de BD para buscar en el contenido del mismo.  
4. No dependes de una ruta para servir los archivos.  
Desventajas:  
1. Más información en la base de datos.  
2. La administración de archivos se puede volver engorrosa, sobre todo si quieres manejar paths virtuales.

**\* Tiempo de acceso a los archivos:** siempre es más rapido el acceso a filesystem que a una base de datos (la db requiere conexión, autenticación, autorización, etc).  
**\* Incremento en tamaño de la base de datos:** si debemos persistir una gran cantida de archivos y/o pocos pero de gran tamaño, esto hará que nuestra base de datos se incremente en tamaño, y en consecuencia su administración será más complicada (backups y restores mas lentos, tiempos de bloqueo de la base de dato mayores).  
**\* Acceso diferido a los archivos:** al persistir los archivos en una base de datos, para accederlos siempre se requerirá de un cliente de base de datos que lea el contenido y luego nos lo presente en forma de archivo. En el caso de filesystem podremos acceder directamente a los archivos físicos.  
**\* Mayor complejidad:** la persistencia en base de datos requiere un mayor conocimiento técnico y más código/procesos/validaciones que en el caso del filesystem.  
**\* Aumento de los requerimientos de infraestructura:** se requiere de mayor memoria RAM, procesador, espacio en disco, etc., requerimientos especificados por el DBMS a ser utilizado en la persistencia de archivos (en la práctica, por lo general ya estamos usando un DBMS para nuestro sistema, por lo que el incremento en infraestructura no suele ser considerable, distinto sería si tuvieramos que considerar implementar persistencia en base de datos en un sistema que no utiliza una base de datos).  
  
Ventajas de persistir archivos / imagenes en base de datos:  
**\* Concurencia:** acceso concurrente a los archivos, manejado por el DBMS (Database Management System).  
**\* Orden de los archivos:** los archivos quedan almacenados coherentemente en tablas de la base de datos, en el caso del filesystem se tiende al desorden en carpetas.  
**\* Mantenimiento de la aplicación:** al tener los datos separados del código de nuestra aplicación se facilita el mantenimiento / actualización de la misma.