 Instituto Superior Santo Domingo	UNIDAD Nº :3 Recursos de ASP.Net.	TEMAS: Upload de archivos.	Clase 8
--	--	---	------------------------------

Objetivos:

- Subir archivos al servidor.
- Validar la existencia de archivos con el mismo nombre.
- Controlar propiedades del archivo.

Introducción

Una actividad muy común en un sitio web es el envío de archivos desde el cliente y su almacenamiento en el servidor.

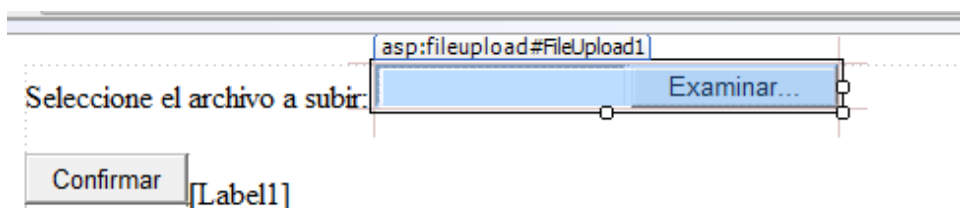
Upload

1 - Componente FileUpload

La componente FileUpload encapsula del envío y recepción de un archivo en el servidor web.

Confeccionaremos una serie de páginas web para aprender a utilizar los métodos y propiedades de la clase FileUpload.

Crear un webform e implementar la siguiente interface:



Disponemos en el webform un objeto de la clase FileUpload que se encuentra en la pestaña de componentes “Estándar”

Para el evento clic de botón confirmar implementamos el siguiente código:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        this.FileUpload1.SaveAs(Server.MapPath(".") + "/" +
this.FileUpload1.FileName);
        this.Label1.Text = "Archivo subido";
    }
}

```

El método SaveAs permite grabar el archivo que se subió al servidor. Debemos indicarle el camino del directorio donde se almacena y el nombre del archivo. Para obtener el path donde se almacena la página ASPX actual el objeto Server tiene el método MapPath y para obtener el nombre del archivo del objeto FileUpload1 tiene una propiedad llamada FileName.

Con esta única línea tenemos registrado en el servidor el archivo que se envió desde el navegador.

2 – Almacenar el archivo en un subdirectorio.

Es una buena costumbre evitar almacenar los archivos que suben los usuarios en la misma carpeta donde se encuentran las páginas dinámicas ASPX.

Para almacenar los archivos en otro directorio primero debemos crear el directorio (por ejemplo creamos una carpeta imágenes en el directorio donde se aloja el sitio)

Creamos un nuevo webform (Default2.aspx) y creamos el siguiente código para el evento clic del objeto Button:

```

protected void Button1_Click(object sender, EventArgs e)
{
    this.FileUpload1.SaveAs(Server.MapPath(".") + "/imagenes/" +
this.FileUpload1.FileName);
    this.Label1.Text = "Archivo subido";
}

```

La carpeta imágenes debemos crearla desde la ventana del "Explorador de soluciones" presionando el botón derecho del mouse y seleccionando la opción "Nueva carpeta". Luego cada archivo que se suba al servidor se almacenará en dicha carpeta.

3 – Mostrar propiedades del archivo subido.

Crearemos un tercer webform para ver como accedemos a distintas propiedades de la clase FileUpload.

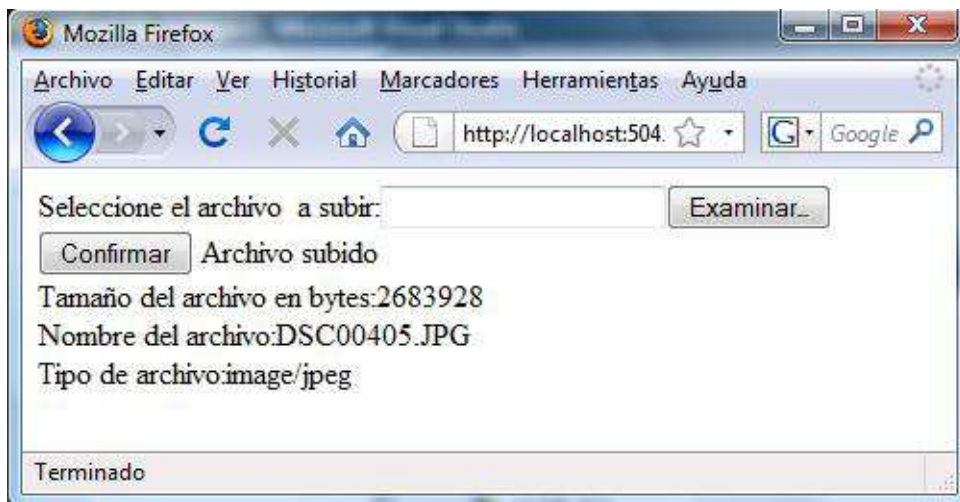
Disponemos sobre el webform un objeto de la clase FileUpload, un Button y cuatro Label.

Mostraremos el tamaño del archivo en bytes. El nombre del archivo y finalmente el tipo de archivo.

El código fuente para el evento clic es:

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.FileUpload1.SaveAs(Server.MapPath(".") + "/imagenes/" +
this.FileUpload1.FileName);
    this.Label1.Text = "Archivo subido";
    this.Label2.Text =
this.FileUpload1.PostedFile.ContentLength.ToString();
    this.Label3.Text = this.FileUpload1.FileName;
    this.Label4.Text = this.FileUpload1.PostedFile.ContentType;
}
```

La propiedad PostedFile del control FileUpload almacena en:
 ContentLength (el tamaño del archivo en bytes)
 FileName (El nombre del archivo dado en el cliente)
 ContentType (El tipo de archivo)



El tamaño del archivo nos puede ser útil si queremos limitar el peso del mismo.

4 – Validar la existencia de otro archivo con el mismo nombre.

Puede suceder que en la carpeta donde se almacena el archivo exista otro con el mismo nombre. Esto conlleva a que se pise el archivo antiguo por el nuevo.

Veamos otro webform donde validamos que no exista otro archivo con el mismo nombre, en caso afirmativo no permitimos su almacenamiento e informamos de tal situación:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
    }
}
```

```

        if (File.Exists(this.Server.MapPath(".") + "/" +
this.FileUpload1.FileName))
        {
            this.Label1.Text = "Existe un archivo con dicho nombre en el
servidor";
        }
        else
        {
            this.FileUpload1.SaveAs(Server.MapPath(".") + "/" +
this.FileUpload1.FileName);
            this.Label1.Text = "Archivo subido";
        }
    }
}

```

El método estático Exists retorna true si ya existe un archivo con el nombre que le pasamos como parámetro, en caso negativo retorna false

En caso que no exista el archivo procedemos a efectuar la registración del mismo en la carpeta de nuestro proyecto.

Para utilizar la clase File debemos importar el espacio de nombres System.IO

Ejercicio Resuelto

1 – Se tienen las tablas:

autos (#patente char(6), propietario varchar(50), precio float, codigomarca int, foto varchar(100), modelo int)

marcas (#codigo int identidad, descripcion varchar(30))

a – Confeccionar el mantenimiento de la tabla marcas (altas, bajas y modificaciones – utilizar un GridView)

b – Confeccionar una página para efectuar el alta de autos.

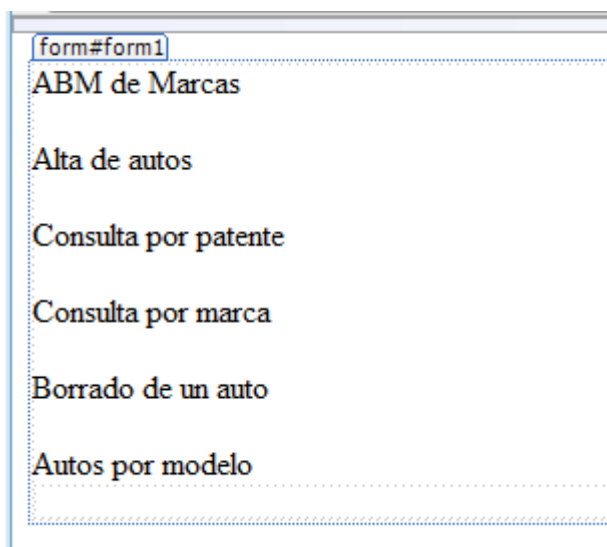
c – Consulta de un auto ingresando su patente (mostrar todos los datos, incluido la foto)

d – Seleccionar de un DropDownList una marca y luego mostrar todas las fotos de autos de dicha marca.

e – Implementar el borrado de un auto ingresando su patente.

f – Ingresar un rango de años y luego mostrar todas las fotos de autos en dicho rango.

La página Default.aspx será un menú de opciones:



The screenshot shows a web application interface. At the top, there is a title bar that says 'form#form1'. Below it, the main heading is 'ABM de Marcas'. Under this heading, there is a list of menu items: 'Alta de autos', 'Consulta por patente', 'Consulta por marca', 'Borrado de un auto', and 'Autos por modelo'. The menu is styled with a simple border and a light background.

a – Confeccionar el mantenimiento de la tabla marcas (altas, bajas y modificaciones – utilizar un

La página abmmarcas.aspx debe permitir implementar las altas, bajas y modificaciones de la tabla marcas.

Mediante un GridView implementamos las bajas y modificaciones. Codificamos por otro lado el alta.

Debemos disponer un SQLDataSource asociado a la tabla marcas y configurar las propiedades ConnectionString, InsertQuery y SelectQuery.

También disponemos un GridView e inicializamos la propiedad "Elegir origen de datos" con el SQLDataSource que hemos insertado.

asp:gridview#GridView1

codigo	descripcion
0	abc
1	abc
2	abc
3	abc
4	abc

Ingrese descripción de la marca:

Alta

SqlDataSource - DSMarcas

Para el evento click del botón "Alta" debemos:

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.DSMarcas.InsertParameters["descripcion"].DefaultValue =
this.TextBox1.Text;
    this.DSMarcas.Insert();
    this.TextBox1.Text = "";
}
```

Luego la interfaz en tiempo de ejecución debe quedar:

codigo	descripcion
1	Renault
2	Ford

Ingrese descripción de la marca:

Alta

b – Confeccionar una página para efectuar el alta de autos.

Debemos crear una interfaz visual similar a:

Configuramos el SqlDataSource de marcas : DSMarcas asociándolo con la tabla marcas, luego el DropDownList1 lo vinculamos a este DataSource para que muestre todas las marcas.

Por otro lado creamos el DSAutos y configuramos el InsertQuery.

Para el evento click del alta procedemos a inicializar los parámetros y proceder a efectuar el insert en la tabla autos.

Además hacemos el upload del archivo de la foto con el objetivo de que la foto quede almacenada en forma permanente en el servidor:

```
protected void Button1_Click(object sender, EventArgs e)
{
    FileUpload1.SaveAs(Server.MapPath(".") + "/" +
this.FileUpload1.FileName);
    this.DSAutos.InsertParameters["patente"].DefaultValue =
this.TextBox1.Text;
    this.DSAutos.InsertParameters["propietario"].DefaultValue =
this.TextBox2.Text;
    this.DSAutos.InsertParameters["precio"].DefaultValue =
this.TextBox3.Text;
    this.DSAutos.InsertParameters["modelo"].DefaultValue =
this.TextBox4.Text;
    this.DSAutos.InsertParameters["codigomarca"].DefaultValue =
this.DropDownList1.SelectedValue;
    this.DSAutos.InsertParameters["foto"].DefaultValue =
this.FileUpload1.FileName;
    this.DSAutos.Insert();
    this.Label1.Text = "Los datos fueron cargados";
    this.TextBox1.Text = "";
    this.TextBox2.Text = "";
}
```

```

        this.TextBox3.Text = "";
        this.TextBox4.Text = "";
    }

```

c – Consulta de un auto ingresando su patente (mostrar todos los datos, incluido la foto)

Implementamos una interfaz que nos permita ingresar la patente de un auto y nos muestre mediante una Label y un objeto de la clase Image los datos de dicho vehículo:

Configuramos la propiedad SelectQuery del DSAutos con el siguiente comando SQL para recuperar los datos:

Nombre	Valor
patente	

Recordar que cada vez que creamos un parámetro en el comando SQL debemos presionar el botón "Actualizar parámetros".

El código del evento click de "Consultar" queda definido por:

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.DSAutos.SelectParameters["patente"].DefaultValue =
this.TextBox1.Text;
    this.DSAutos.DataSourceMode = SqlDataSourceMode.DataReader;
    SqlDataReader registro =
(SqlDataReader)this.DSAutos.Select(DataSourceSelectArguments.Empty);
    if (registro.Read())
    {
        this.Imagel.ImageUrl = registro["foto"].ToString();
        this.Label1.Text = "Propietario" + registro["propietario"] +
"<br>" +
                                "Precio:" + registro["precio"] + "<br>" +
                                "Modelo:" + registro["modelo"] + "<br>" +
                                "Marca:" + registro["descripcion"];
    }
    else
    {
        this.Label1.Text = "No existe un auto con dicha patente";
    }
}
```

Y tenemos como resultado una página en tiempo de ejecución similar a:

Ingrese patente:

PropietarioLopez Pablo

Precio:210000

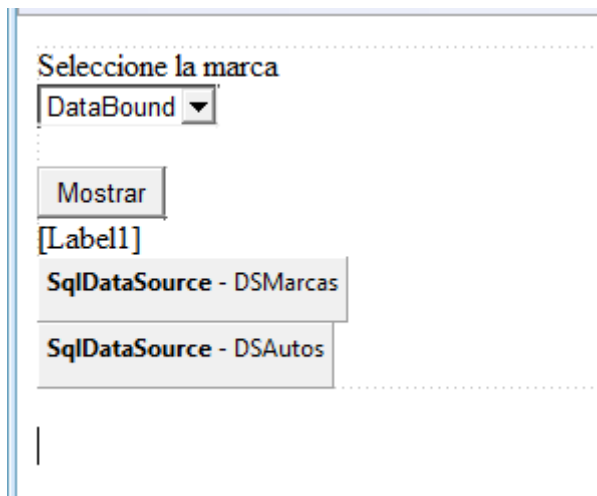
Modelo:2014

MarcaRenault



d – Seleccionar de un DropDownList una marca y luego mostrar todas las fotos de autos de dicha marca.

Para implementar este punto debemos crear una interfaz similar a la siguiente:



Configuramos el SelectQuery del DSMarcas con la consulta de la tabla marcas:
select codigo, descripcion from marcas

Por otro lado configuramos el origen de datos del DropDownList1 para que se muestren todas las marcas y podamos posteriormente recuperar el código de la marca seleccionada.

Por otro lado configuramos la propiedad SelectQuery del objeto DSAutos:
select foto from autos where codigomarca=@codigomarca

Luego cuando se presiona el botón "Mostrar" debemos recuperar todas las fotos de autos que pertenecen a la marca seleccionada:

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.DSAutos.SelectParameters["codigomarca"].DefaultValue =
this.DropDownList1.SelectedValue;
    this.DSAutos.DataSourceMode = SqlDataSourceMode.DataReader;
    SqlDataReader registros =
(SqlDataReader) this.DSAutos.Select(DataSourceSelectArguments.Empty);
    this.Label1.Text = "";
    while (registros.Read() == true)
    {
        this.Label1.Text=this.Label1.Text+"<img src=\"\" +
            registros["foto"] + "\"><br>";
    }
}
```

Como podemos ver debemos generar las marcas HTML img con cada una de las fotos (no podemos utilizar un objeto Image ya que pueden haber varios autos que pertenecen a la misma marca)

En pantalla en tiempo de ejecución se debe mostrar algo similar a:

Seleccione la marca

Renault ▼

Mostrar



e – Implementar el borrado de un auto ingresando su patente.
Construimos una interfaz similar a esta:

Ingresa la patente del auto a borrar:

SqlDataSource - DSAutos

Borrar [Label1]

Debemos consultar la tabla autos para recuperar el nombre del archivo almacenado del auto, para ello inicializamos la propiedad SelectQuery con la siguiente consulta:

```
select foto from autos where patente=@patente
```

Luego también debemos implementar la propiedad DeleteQuery:

delete from autos where patente=@patente

Cuando se presiona el botón borrar procedemos a ejecutar los dos comandos SQL:

```
this.DSAutos.DeleteParameters["patente"].DefaultValue =
this.TextBox1.Text;
int cant;
cant = this.DSAutos.Delete();
if (cant == 0)
    this.Label1.Text = "No existe un auto con el valor de patente
ingresado";
else
{
    this.Label1.Text = "Se borro el auto con dicha patente";
    this.DSAutos.SelectParameters["patente"].DefaultValue =
this.TextBox1.Text;
    this.DSAutos.DataSourceMode = SqlDataSourceMode.DataReader;
    SqlDataReader registro;
    registro =
(SqlDataReader)this.DSAutos.Select(DataSourceSelectArguments.Empty);
    if (registro.Read())
        File.Delete(Server.MapPath(".") + "/" + registro["foto"]);
}
}
```

f – Ingresar un rango de años y luego mostrar todas las fotos de autos en dicho rango.

Creamos una interfaz similar a:

The screenshot shows a web form with two text input fields. The first field is labeled 'Ingrese año inicial:' and the second is labeled 'Ingrese año final:'. Below these fields is a button labeled 'Mostrar vehículos'. At the bottom, there is a label '[Label1]' and a text box containing the text 'SqlDataSource - DSAutos'.

Configuramos la propiedad SelectQuery del SqlDataSource con el comando SQL:

select foto from autos where modelo>=@modelo1 and modelo<=@modelo2

Para el evento click del botón procedemos a mostrar todas las fotos de autos cuyos modelos se encuentran comprendidos entre los dos valores ingresados:

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.DSAutos.SelectParameters["modelo1"].DefaultValue =
this.TextBox1.Text;
    this.DSAutos.SelectParameters["modelo2"].DefaultValue =
this.TextBox2.Text;
    this.DSAutos.DataSourceMode = SqlDataSourceMode.DataReader;
```

```
        SqlDataReader registros =  
(SqlDataReader)this.DSAutos.Select(DataSourceSelectArguments.Empty);  
        this.Label1.Text = "";  
        while (registros.Read() == true)  
        {  
            this.Label1.Text = this.Label1.Text + "<img src=\"\" +  
                registros[\"foto\"] + \"\"><br><br>";  
        }  
    }
```

Ejercicio Propuesto

1 – Se tienen las siguientes tablas:

libros (codigo, titulo, descripcion, fototapa, codigotema)

temas (codigo, descripcion)

a – Confeccionar el alta de las tablas libros y temas (permitir seleccionar el tema mediante una lista) Hacer el Upload de la imagen al servidor. Validar que se ingresen obligatoriamente el titulo, descripción y selección de un archivo.

b – Ingresar por teclado el código de un libro y luego mostrar la foto de la tapa.