

# ADO.NET

---

*Herramientas Avanzadas para el Desarrollo de  
Aplicaciones*

Escola Politècnica Superior  
Universitat d'Alacant

# Objectius

- ADO.net 2.0
- Creació d'una BD SQL des de Vstudio.net
- Accés connectat a BD
- Creació cadena connexió: Web.config
- Propietat DataDirectory

ADO.Net 2.0

1

---

## Objectes d'accés a dades (*ActiveX Data Objects*)

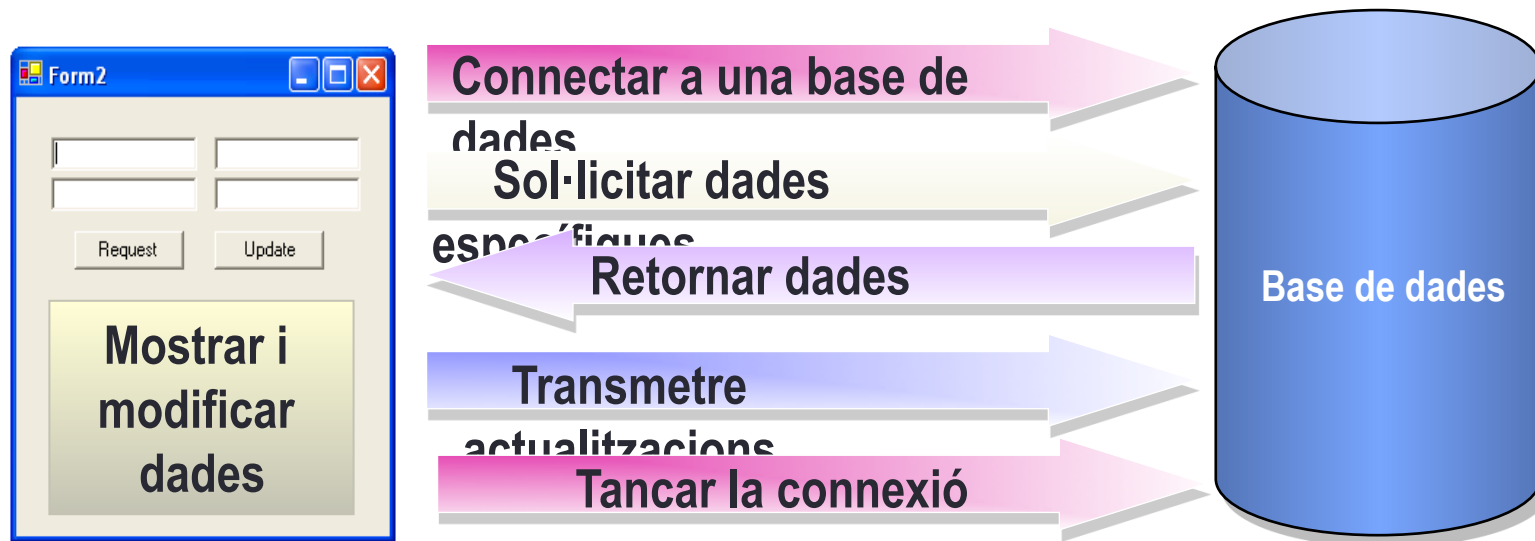
- ADO.NET és la tecnologia que les aplicacions asp.net utilitzen per comunicar-se amb la BD.
- Optimitzada per a aplicacions distribuïdes (com a aplicacions web).
- Basada en XML
- Model d'objectes completament nou.
- **Entorn connectat vs desconnectat.**

# Entorn connectat

- Un entorn connectat és aquell que els usuaris estan connectats contínuament a una font de dades
- Avantatges:
  - L'entorn és més fàcil de mantenir
  - La concurrència es controla més fàcilment
  - És més probable que les dades estiguin més actualitzats que en altres escenaris
- Inconvenients:
  - Ha d'existir una connexió de xarxa constant
  - Escalabilidad limitada

# Entorn connectat (II)

CONNEXIÓ OBERTA



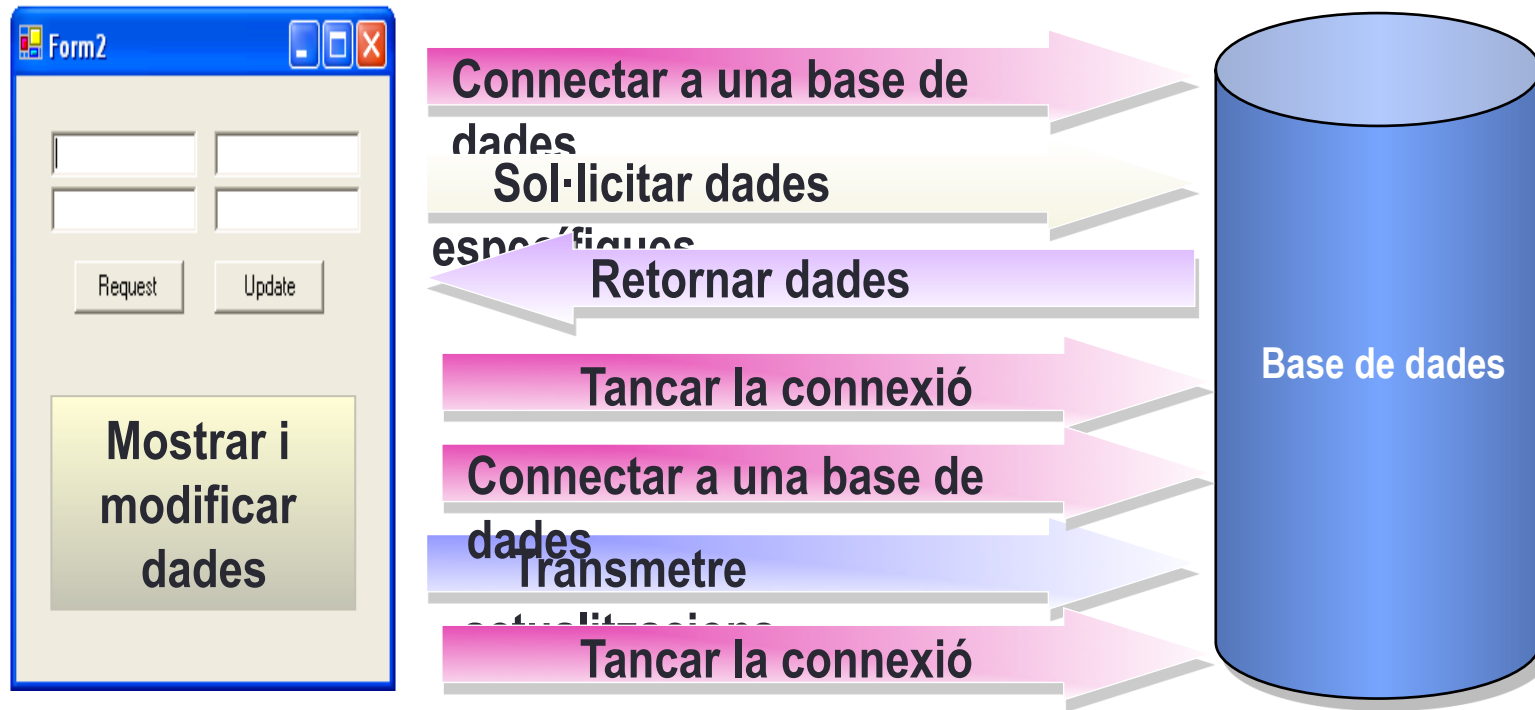
SENSE CONNEXIÓ

# Entorn desconnectat

- Un entorn desconnectat és aquell en el qual les dades poden modificar-se de forma independent i els canvis s'escriuen posteriorment en la base de dades.
- Avantatges:
  - Les connexions s'utilitzen durant el menor temps possible, permetent que menys connexions donin servei a més usuaris
  - Un entorn desconnectat millora la escalabilidad i el rendiment de les aplicacions
- Inconvenients:
  - Les dades no sempre estan actualitzats
  - Poden produir-se conflictes de canvis que han de solucionar-se

# Entorn desconnectat (II)

CONNEXIÓ OBERTA



SENSE CONNEXIÓ



# Exemples d'entorns

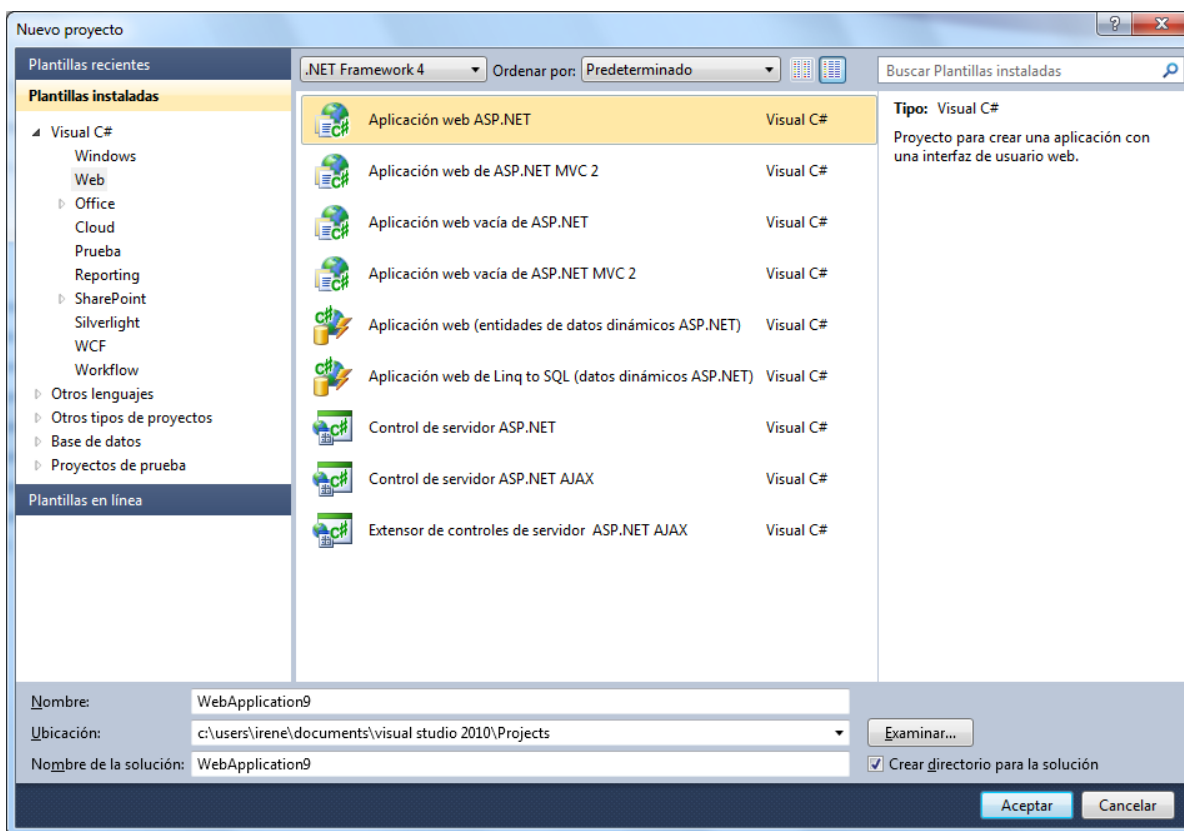
- Un supermercat, on en cada punt de venda (TPV) s'emmagatzemen les vendes realitzades. Cada cert temps les hi vendes s'han d'actualitzar en la BD central.
- Una fàbrica que requereix una connexió en temps real per controlar la sortida de producció i el magatzem.
- Una aplicació que manté dades de clients en un equip portàtil d'un representant.
- Una aplicació que fa un seguiment de pluges i precipitacions.
- Un agent de borsa que requereix una connexió constant als valors del mercat.
- Una aplicació que un granger utilitza per explicar el bestiar. L'aplicació està en el dispositiu basat en Microsoft Windows CE del granger que executa Microsoft SQL Server 2000 Windows CE Edition.

2

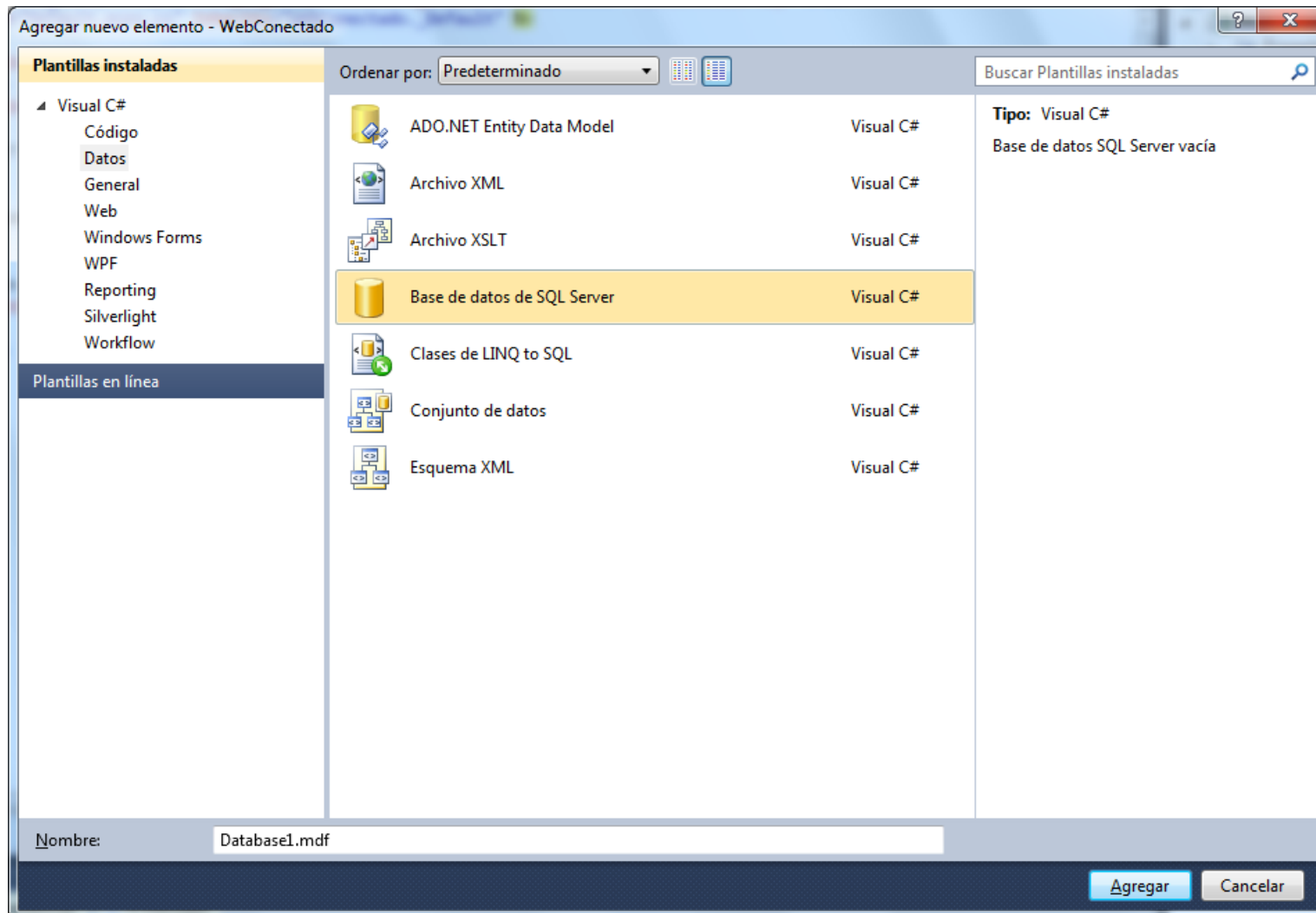
---

Creació BD des  
de Vstudio.net

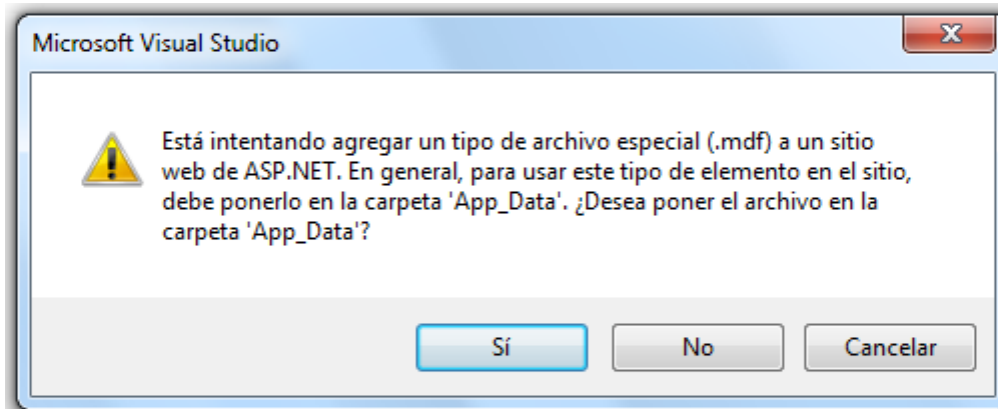
# Nou projecte: Aplicació Web C#



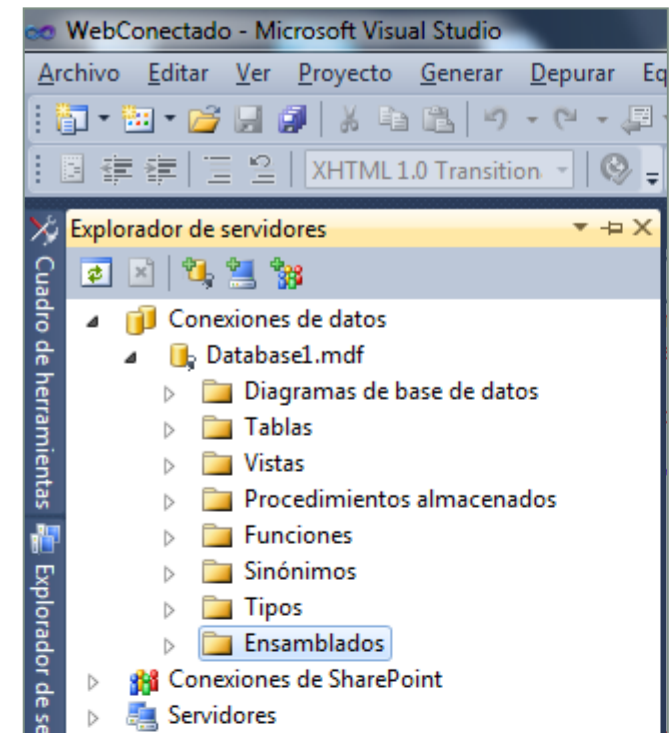
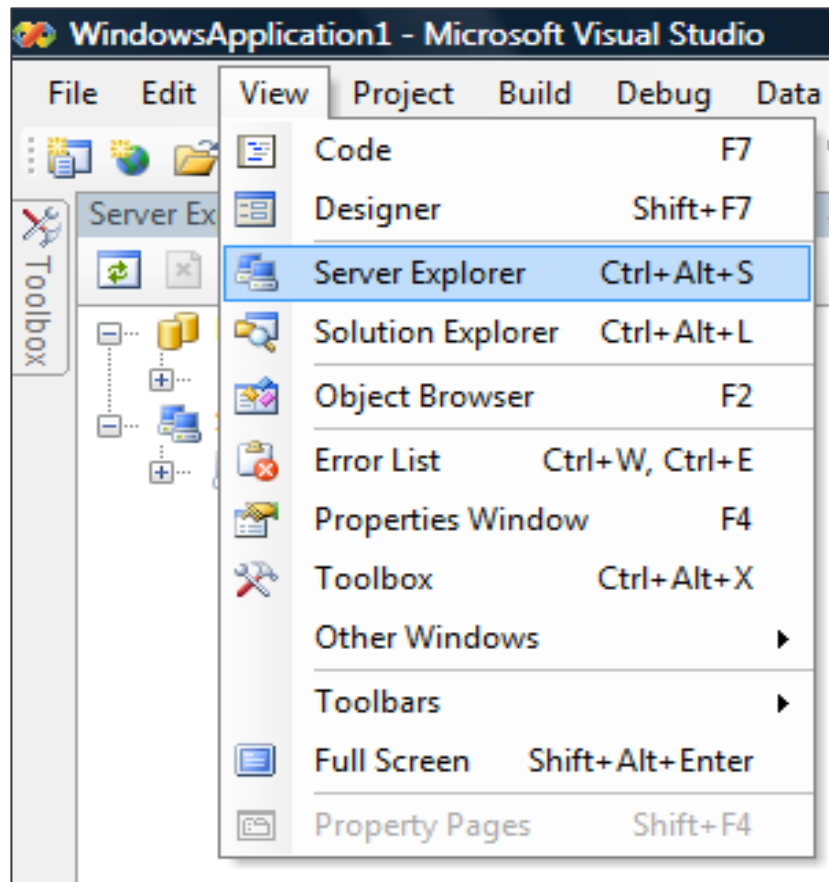
# Agregar nou element: BD SQL server



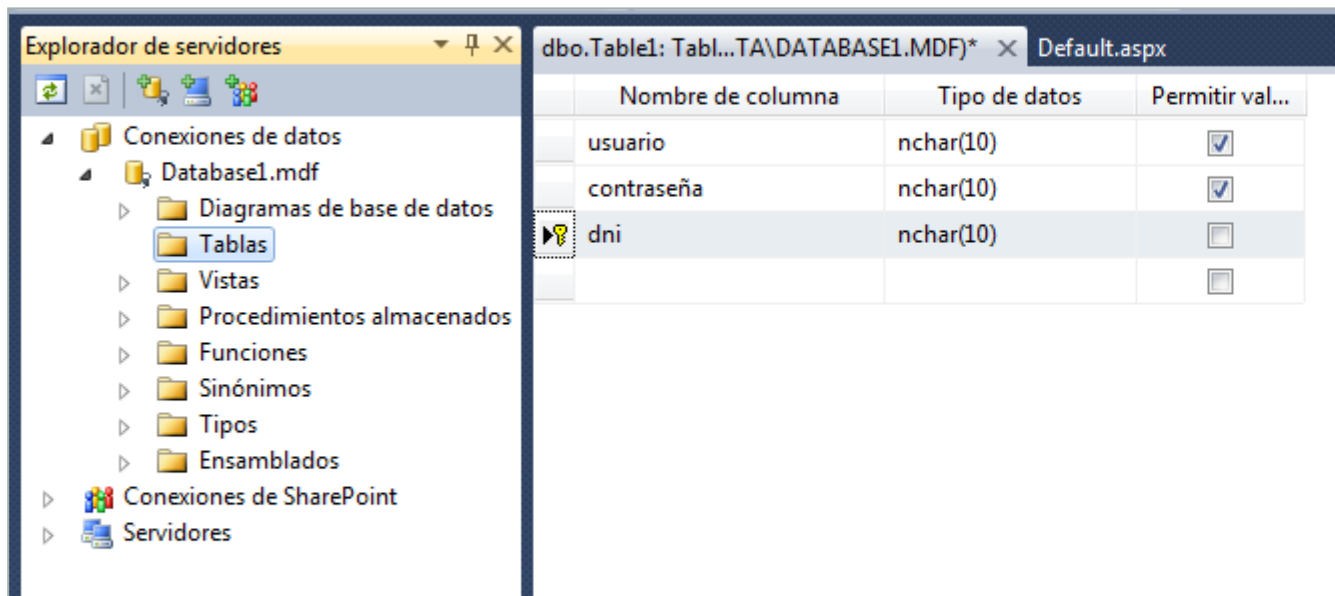
# Carpeta especial App\_Data



# Veure → explorador de servidores



# Afegim taules, claus... i dades!



# Possibles problemes de configuració

- Instal·lar SQLServer Management Studio Express
  - En l'opció connexions → permetre connexions remotes
- En SQL Configuration Manager
  - Protocols de SQLEXPRESS
    - Habilitar TCP/IP i canalitzacions amb nom
- Parar el servei SQLEXPRESS i reiniciar





3

---

Accès Connectat

# 1. Objectes Connection i Command

- **Connection:** s'utilitzen per establir les connexions al proveïdor de dades adequat (mètode Open).
- **Command:** serveixen per executar sentències SQL i procediments emmagatzemats.

# Objectes Connection i Command

- `System.Data.OleDb` i `System.Data.SqlClient`:  
classes responsables de l'accés a dades des de fonts SQL Server i OLE DB.
- Inclouen classes que en treballar amb SQL portaran el prefix `Sql` i en emprar Ole DB portaran `OleDb`:
  - `SqlConnection` i `OleDbConnection`
  - `SqlCommand` i `OleDbCommand`

## 2. Objecte DataReader

- Proporcionen un flux de dades ferma.
- Proporcionen un cursor de només lectura que avança pels registres només cap a davant.
  - Mantenen una **connexió viva** amb l'origen de dades, però no permeten realitzar cap canvi.

# Espais de noms de dades

- `System.Data`
- `System.Data.Common`
- `System.Data.OleDb` → Ms access, Oracle.. DB
- `System.Data.SqlClient` → Ms SQL Server 7.0 DB
- `System.Data.SqlTypes` → conté classes per treballar amb tipus de dades natives de SQL Server

# EXAMPLE: Connexió a una BD en Sql Server

## Importar namespaces

```
using System.Data;  
using System.Data.Common;  
using System.Data.SqlClient;  
using System.Data.SqlTypes;
```

## Cadena de Connexió (I)

Paràmetre	Descripció
Connection TimeOut	Defineix el temps d'espera màxim que ha d'esperar una connexió per intentar connectar amb èxit amb el servidor de base de dades. En cas de superar aquest temps es genera una excepció. El temps per defecte definit és de 15 segons.
Data Source	Rep el nom del servidor SQL Server utilitzat en la connexió, o en cas d'utilitzar bases de dades d'usuari Access el nom de l'arxiu utilitzat.
Initial Catalog / Database	Nom de la base de dades amb la qual anem a treballar.
Integrated Security	Configura nostra connexió d'una manera segura o no. Rep com a valors True, False i SSPI, sent True i SSPI la mateixa manera de seguretat.

## Cadena de Connexió (II)

Paràmetre	Descripció
AttachDBFilename	Si s'utilitza un nom d'arxiu per connectar amb la base de dades, se simplifica la implementació de la base de dades amb l'aplicació (especifiquem l'arxiu mdf)
Persist Security Info	Si rep el valor True, es retorna la contrasenya juntament amb la connexió si ha estat oberta o roman oberta, això suposa un risc de seguretat, per la qual cosa se sol deixar com està configurada per defecte, False.
Password	Contrasenya per a la identificació d'inici de sessió en SQL Server.
Provider	Utilitzada únicament per a connexions OleDbConnection, estableix o retorna el nom del proveïdor.
User ID	Nom d'usuari per a l'inici de sessió en SQL Server 2005, login.



# Provider

- SQLOLEDB: proveïdor OLEDB de SQL
- MSDAORA: proveïdor OLEDB per una bd Oracle
- Microsoft.Jet.OLEDB.4.0: proveïdor OLEDB de Access

## Crear la connexió

- `string s = "data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\Database1.mdf;User Instance=true";`

`SqlConnection c=new SqlConnection(s);`

## Obrir la connexió

`c.Open();`



# Propietat DataDirectory

# Propietat DataDirectory

- **DataDirectory** és una cadena de substitució que indica la ruta d'accés de la base de dades.
- **DataDirectory** facilita l'ús compartit d'un projecte i la impressió d'una aplicació en eliminar la necessitat de definir la ruta d'accés completa. Per exemple, en comptes de tenir la següent cadena de connexió:
  - "AttachDbFilename= c:\program files\MyApp\Mydb.sdf"
- En usar **|DataDirectory|**, pot tenir la següent cadena de connexió:
  - "AttachDbFilename= |DataDirectory|\Mydb.sdf"
- La propietat **DataDirectory** s'estableix en **AppDomain** cridant a **AppDomain.SetData**.
  - AppDomain.CurrentDomain.SetData("DataDirectory", newpath);

# Propietat DataDirectory

- Si no s'estableix la propietat **DataDirectory**, s'aplicaran les següents regles predeterminades per tenir accés a la ruta de la carpeta de la base de dades:
  - Per a les aplicacions que es col·loquin en un directori en l'equip client, la ruta d'accés de la base de dades serà la carpeta en la qual es col·loqui l'aplicació. Per exemple, si MyApp.exe es col·loca a la carpeta /MyDir, es tindrà accés a la carpeta /MyDir.
  - **Para aplicacions Web, es tindrà accés a la carpeta App\_Data.**

# |DataDirectory|

```
<connectionStrings>
```

```
<add name="miconexion"
```

```
connectionString="Data Source=localhost\SQLEXPRESS;
```

```
AttachDbFilename=|DataDirectory|\bd.mdf;Integrated
```

```
Security=True;Connect Timeout=30;User Instance=True"
```

```
providerName="System.Data.SqlClient" />
```

```
</connectionStrings>
```

## Definició d'un comando Select

- Per recuperar les dades es necessita:
  - Una sentència SQL que seleccioni la informació desitjada
  - Un objecte Command que executi la sentència SQL
  - Un objecte DataReader que capturi els registres recuperats

# Objecte Command

- Els objectes Command representen sentències SQL. Per utilitzar un objecte Command es defineix la sentència SQL a utilitzar i la connexió disponible i s'executa el comando:

```
SqlCommand com= new SqlCommand("Select *  
from clientes",c);
```



## Objeto DataReader (I)

- S'utilitza el mètode ExecuteReader de l'objecte Command:
  - `SqlDataReader dr= com.ExecuteReader();`
- Es recupera una fila amb el mètode Read
  - `dr.Read()`

## Objecte DataReader (II)

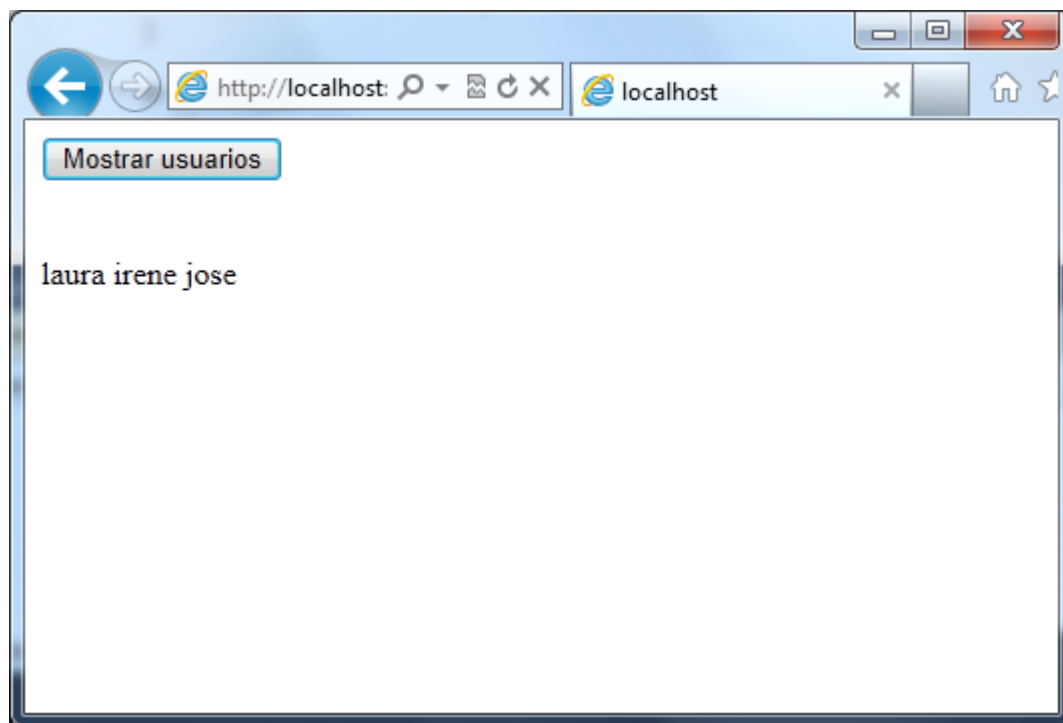
- Podem accedir als valors d'aquesta fila amb el nom del camp corresponent:
  - `MyDataReader["nombrecampo"];`
- Per llegir la següent fila, tornem a usar el mètode `Read`
  - retorna `true` si s'ha recuperat una fila d'informació correctament
  - si retorna `false` és perquè hem intentat llegir després del final del conjunt de resultats

## Com es faria en l'exemple que estem veient??

- Escriure el codi necessari per mostrar en una etiqueta el nom dels clients (taula client, columna usuari)

cliente: Query(ir...qlxpress.prueba1)			
	usuario	contraseña	dni
▶	laura	123	45698
	irene	123	45896
	jose	258	85964
*	NULL	NULL	NULL

# Execució



## En l'exemple...

```
while (dr.Read())  
{  
    this.label1.Text+=dr["usuario"].ToString();  
    label1.Text += " ";  
}
```

## Tancar els objectes DataReader i Connection

- `dr.Close();`
- `c.Close();`

# Important

- Utilitzar maneig d'excepcions per a connexió a BD:
  - Try/catch

# On afeim try/catch?

```
string s = "data source=.\SQLEXPRESS;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\  
Database1.mdf;User Instance=true";
```

```
SqlConnection c=new SqlConnection(s);  
c.Open();  
SqlCommand com= new SqlCommand("Select * from cliente",c);  
SqlDataReader dr= com.ExecuteReader();
```

```
while (dr.Read())  
{    this.label1.Text+=dr["usuario"].ToString();  
    label1.Text += " ";  
}
```

```
dr.Close();  
c.Close();
```



```
string s = "data source=.\SQLEXPRESS;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\  
Database1.mdf;User Instance=true";
```

```
SqlConnection c=new SqlConnection(s);
```

```
try
```

```
{
```

```
c.Open();
```

```
SqlCommand com= new SqlCommand("Select * from cliente",c);
```

```
SqlDataReader dr= com.ExecuteReader();
```

```
while (dr.Read())
```

```
{    this.label1.Text+=dr["usuario"].ToString();
```

```
    label1.Text += " ";
```

```
}
```

```
dr.Close();
```

```
}
```

```
catch (Exception ex) { label2.Text = ex.Message; }
```

```
finally
```

```
{
```

```
    c.Close();
```

```
}
```



I com seria el codi tenint en compte les 3  
capes?

# CAPA INTERFÍCIE (I)

The screenshot displays the Visual Studio IDE with the 'WebForm1.aspx' file open. The top toolbar shows the 'Diseño' (Design) view is active. The 'Eventos y objetos de cliente' (Client events and objects) pane is visible, showing '(No hay eventos)' (No events).

The code editor shows the following HTML markup:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebConectado.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>

</div>
<asp:Button ID="Button1" runat="server" Text="Mostrar usuarios"
onclick="Button1_Click" />
<br />
<br />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
</form>
</body>
</html>
```

The design view at the bottom shows a visual representation of the code. It features a button labeled 'Mostrar usuarios' and a label placeholder '[Label1]'. The design view is currently showing the 'body' container.

The bottom status bar indicates the current selection is on the 'Código' (Code) tab, and the breadcrumb path is: <html> <body> <form#form1> <asp:Label#Label1>.

## CAPA INTERFÍCIE (II)

```
using System;
```

```
...
```

```
using System.Collections;
```

```
namespace WebConectado
```

```
{
```

```
    public partial class WebForm1 : System.Web.UI.Page
```

```
    {
```

```
        ArrayList a = new ArrayList();
```

```
        protected void Page_Load(object sender, EventArgs e)
```

```
        {
```

```
            protected void Button1_Click(object sender, EventArgs e)
```

```
            {
```

```
                ENCliente en = new ENCliente();
```

```
                a=en.listarClientes();
```

```
                foreach (string s in a)
```

```
                    Label1.Text += s + " ";
```

```
            }}}
```

# CAPA EN

```
namespace WebConectado
{
    public class ENCliente
    {
        private string usuario;
        public string Usuario
        { get { return usuario; }
          set { usuario = value; }
        }
        private string dni;
        public string Dni
        { get { return dni; }
          set { dni = value; }
        }
        private string contrasenya;
        public string Contrasenya
        { get { return contrasenya; }
          set { contrasenya = value; }
        }
    }
}
```

```
public ArrayList listarClientes()
{
    ArrayList a = new ArrayList();
    CADcliente c = new CADcliente();
    a=c.ListarClientes();

    return a;
}

}
```

# CAPA CAD

```
public class CADcliente
{
    ArrayList lista = new ArrayList();
    string s = "data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|
DataDirectory|\\Database1.mdf;User Instance=true";

    public ArrayList ListarClientes()
    {
        SqlConnection c = new SqlConnection(s);
        c.Open();
        SqlCommand com = new SqlCommand("Select * from cliente", c);
        SqlDataReader dr = com.ExecuteReader();

        while (dr.Read())
        {
            lista.Add(dr["usuario"].ToString());
        }
        dr.Close();
        c.Close();
    }
    return lista;
}
```



On crear la cadena de connexió???

# L'arxiu web.config

- Arxiu de configuració de l'aplicació ASP.NET basat en XML
- Inclou les opcions de seguretat personalitzada, administració d'estat, administració de memòria..etc

```
<?xml version="1.0" encoding="utd-8" ?>  
  <configuration>  
    <system.web>  
      <!-- Sección de configuración -- >  
    </system.web>  
  </configuration>
```



## Important, on guardar la cadena de connexió?

- Per evitar emmagatzemar cadenes en el codi, es pot emmagatzemar en l'arxiu **web.config** en una aplicació ASP.NET.
- La cadena de connexió es pot emmagatzemar en l'arxiu de configuració en l'element **<connectionStrings>**. Les cadenes de connexió s'emmagatzemen com a parells de clau i valor, on el nom es pot utilitzar per buscar el valor emmagatzemat en l'atribut **connectionString** en temps d'execució.

# Web.config Arxivo configuració : basat en XML

```
<connectionStrings>  
  <add name="DatabaseConnection"  
    connectionString="Persist Security Info=False;Integrated  
Security=SSPI;database=Northwind;server=(local);"  
    providerName="System.Data.SqlClient" />  
</connectionStrings>
```

cadena de connexió anomenada DatabaseConnection que fa referència a una cadena de connexió que es connecta a una instància local de SQL Server.

# Recuperar cadenes de connexió d'arxius de configuració

- L'espai de noms **System.Configuration** proporciona classes per treballar amb informació de configuració emmagatzemada en arxius de configuració.

# C#

```
using System.Configuration;
```

```
string cadena;
```

```
cadena =
```

```
ConfigurationManager.ConnectionStrings["DatabaseConnection"].ToString  
();
```

Es recupera la cadena de connexió de l'arxiu de configuració passant el nom de l'aquesta cadena al  
ConfigurationManager

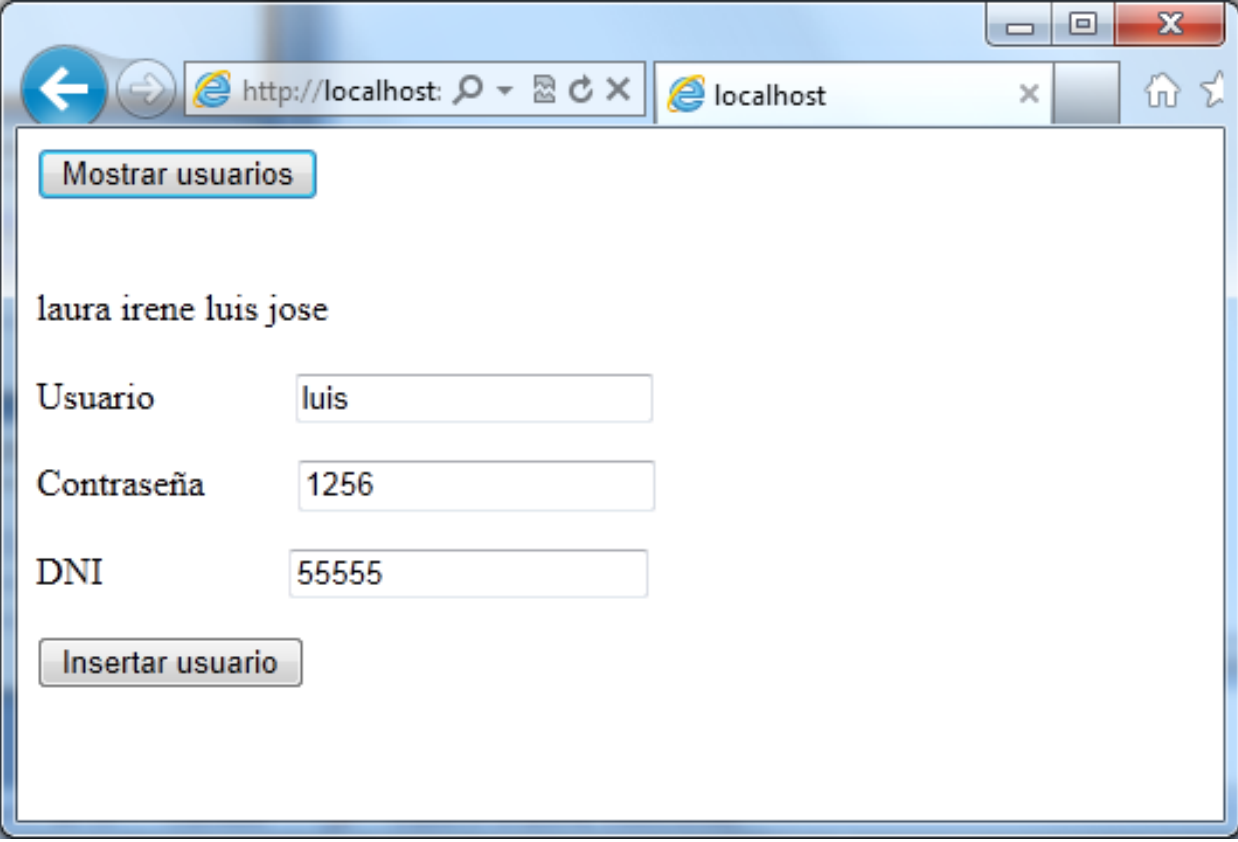
## Modificació de dades: Insert, Update, Delete

- En aquest cas no és necessari un objecte DataReader ja que no es recupera cap resultat.
- Tampoc un comando Select de SQL, sinó:
  - Update
  - Insert
  - Delete
- Necessitem crear un objecte Command per executar la sentència SQL apropiada.
- Mètode ExecuteNonQuery: obté el nombre de registres afectats.

# Execució de comandos

- ExecuteNonQuery
  - Executa un comando i no retorna cap resultat (obté el nombre de registres afectats.)
- ExecuteReader
  - Executa un comando i retorna un comando que implementa DataReader (Permet iterar a partir dels registres rebuts)

# Exemple, Inserir usuariis



A screenshot of a web browser window displaying a user registration form. The browser's address bar shows 'http://localhost:'. The page has a light blue header with a 'Mostrar usuarios' button. Below the header, the text 'laura irene luis jose' is displayed. The form contains three input fields: 'Usuario' with the value 'luis', 'Contraseña' with the value '1256', and 'DNI' with the value '55555'. At the bottom of the form is an 'Insertar usuario' button.

Mostrar usuarios

laura irene luis jose

Usuario

Contraseña

DNI

Insertar usuario

```
string s = "data source=.\SQLEXPRESS;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\  
Database1.mdf;User Instance=true";
```

```
SqlConnection c=new SqlConnection(s);
```

```
try
```

```
{
```

```
c.Open();
```

```
SqlCommand com= new SqlCommand("Select * from cliente",c);
```

```
SqlDataReader dr= com.ExecuteReader();
```

```
while (dr.Read())
```

```
{    this.label1.Text+=dr["usuario"].ToString();
```

```
    label1.Text += " ";
```

```
}
```

```
dr.Close();
```

```
}
```

```
catch (Exception ex) { label2.Text = ex.Message; }
```

```
finally
```

```
{
```

```
    c.Close();
```

```
}
```



```
string s = "data source=.\SQLEXPRESS;Integrated  
Security=SSPI;AttachDBFilename=|DataDirectory|\  
Database1.mdf;User Instance=true";
```

```
SqlConnection c=new SqlConnection(s);
```

```
try
```

```
{
```

```
c.Open();
```

```
SqlCommand com = new SqlCommand("Insert Into Cliente  
(usuario,contraseña,dni) VALUES ('" + textBox1.Text + "','" +  
textBox3.Text + "','" + textBox2.Text +'"', c);
```

```
com.ExecuteNonQuery();
```

```
}
```

```
catch (Exception ex) { label2.Text = ex.Message; }
```

```
finally
```

```
{
```

```
c.Close();
```

```
}
```

# Interfaz

```
protected void Button2_Click(object sender, EventArgs e)
{
    ENCliente en = new ENCliente();
    en.Usuario = TextBox1.Text;
    en.Contraseña = TextBox2.Text;
    en.Dni = TextBox3.Text;

    en.InsertarCliente();
}
```

# ENCliente

```
public void InsertarCliente()  
{  
    CADcliente c = new CADcliente();  
    c.InsertarCliente(this);  
}
```

# CADCliente

```
public void InsertarCliente(ENCliente cli)
{
    ENCliente cl = cli;
    SqlConnection c = new SqlConnection(s);
    c.Open();

    SqlCommand com = new SqlCommand("Insert Into Cliente
(usuario,contraseña,dni) VALUES ('" + cl.Usuario + "','" + cl.Contraseña + "','" +
cl.Dni + "')", c);

    com.ExecuteNonQuery();
    c.Close();
}
```