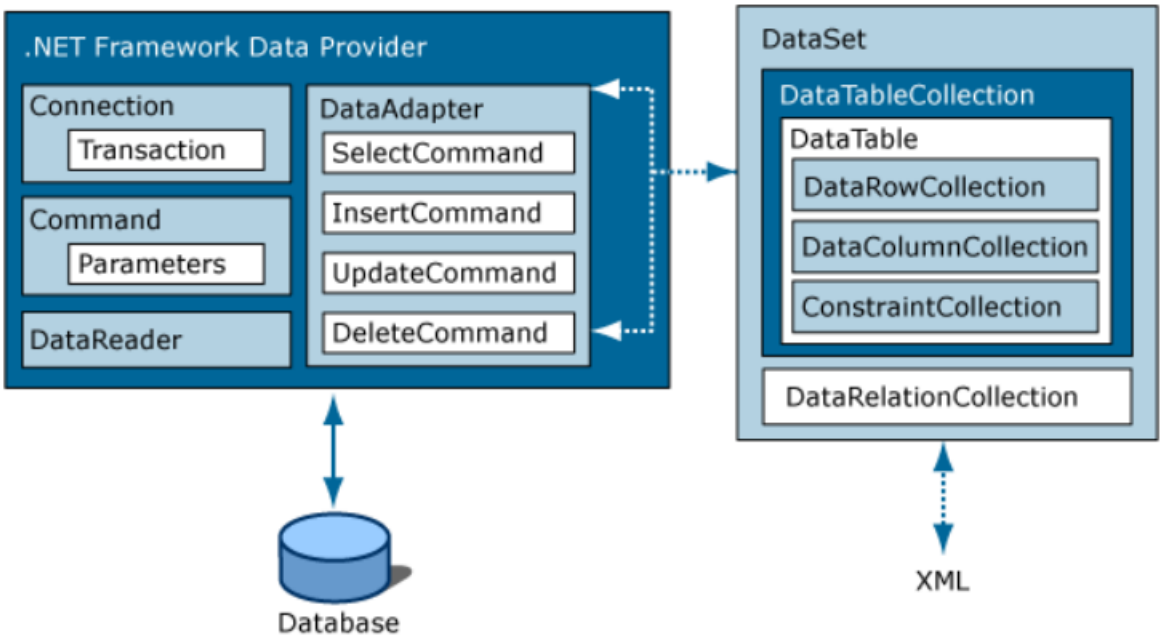
ARQUITECTURA DE CONEXIÓN A BASES DE DATOS



Introducción a la conexión a Bases de Datos en C#

Conectar una aplicación en C# a una base de datos en SQL Server es uno de los requerimientos más importantes de funcionalidad en un proyecto de tipo empresarial. En síntesis, consiste en: lo primero es definir una conexión al servidor de DB, después se abre o estable la conexión, luego se realizarla ejecución de comando SQL y por último se cierra la conexión.

1. **Crear una conexión con SqlConnection**

Consiste en realizar una conexión de red entre la aplicación y el servidor de bases de datos. Para ello se instancia la clase **SqlConnection** (p.e. con la línea: SqlConnection **objCon** = new SqlConnection();) y se debe usar (importar) el namespace **SqlClient**(System.Data.SqlClient;). Este espacio de nombres es el encargado de gestionar los datos en SQL Server hacia el Framework .NET y viceversa.

Para el objeto **objCon** es de vital importancia los datos que abren la conexión. La creación del este objeto (dos constructores) se puede realizar de dos formas: A) Sin parámetros: que se debe hacer uso del método llamado **ConnectionString**, para asígnales los valores para la conexión. Para establecerlo podemos usar el constructor o asignárselo luego de la inicialización, así:

```
String datosConexion = "Data Source=localhost;Initial Catalog=empresa;Integrated Security=true;";
SqlConnection objCon = new SqlConnection();
objCon.ConnectionString = datosConexion;
```

O B) con parámetros:

```
SqlConnection objCon = new SqlConnection("Data Source =l ocalhost;Initial Catalog = empresa ;Integrated Security=true”;);
```

La cadena de conexión indica las características necesarias para la conexión. Veamos la definición de algunos:

- Data Source: Se refiere al origen de datos, nombre del servidor o dirección donde se encuentra la base de datos.
- Initial Catalog: Es el nombre de la base de datos a la que deseamos acceder.
- Integrated Security: Si usas true el ingreso a la base de datos se autentica con los permisos del usuarios actual de Windows, si usas false, debes indicar en la cadena el nombre de usuario(UID) y la contraseña(PWD).

Existen más características, pero estas son las básicas.

2. **Abrir la conexión**

Luego de establecer la cadena de conexión, se procede a abrirla con el método Open(): p.e. objCon.Open();

Al realizar todas las operaciones sobre la base de datos es importante cerrar la conexión con el método Close(): p.e. objCon.Close();

3. Ejecutar un comando SQL

Para ejecutar comandos T-SQL usaremos la clase **SqlCommand** (que hace parte de SqlClient). Solo se debe crear una instancia de esta clase y asociar una cadena que guarde el comando SQL (textoCmd) y el objeto que está gestionando la conexión, que para el ejemplo anterior es **objCon**:

SqlCommand cmd = new SqlCommand(textoCmd, objCon);

El primer parámetro es textoCmd (Instrucción SQL) y el segundo objCon (el objeto de la conexión asociada). Por ejemplo:

String textoCmd = "DELETE FROM CLIENTE WHERE IDCLIENTE = 1112;"
SqlCommand cmd = new SqlCommand (textoCmd, objCon);

Ejecutar modificaciones SQL

Para ejecutarlo usamos el método ExecuteNonQuery(), que ejecuta sentencias que no retornan filas o registro de la base de datos como: INSERT, UPDATE, DELETE ó SET. Lo que retorna es un número entero; que contiene la cantidad de filas o registros afectados o modificados de la tabla.

int n = cmd.ExecuteNonQuery();

Ejecutar consultas SQL

Para un SELECT se usa el método **ExecuteReader()** para leer cada fila del resultado de la consulta. Pero necesitamos una estructura de datos en donde guardar dicha información. Para ello usaremos la clase SqlDataReader (también contenida en SqlClient) que proporciona la forma de lectura ideal. Veamos como acondicionar el código para recibir el resultado de un SELECT:

textoCmd = "SELECT lista de campos FROM nombre tabla;";
SqlCommand cmd = new SqlCommand(textoCmd, objCon);
SqlDataReader reader = cmd.ExecuteReader();

Una vez referenciadas las filas de la consulta, procedemos a leer fila a fila mediante el método Read(). Este método cada vez que es invocado mueve la posición de lectura a la siguiente fila, por lo cual se usa un ciclo while para la lectura completa.

Ejemplos

1. Conexión sencilla en Consola

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace Base1
{
Class Program
    {
Static void Main(string[] args)
{ //esta es la cadena de conexion
SqlConnection conectar = new SqlConnection("Data Source=localhost;Initial
Catalog=empresa;Integrated Security=SSPI;");
try  {
       conectar.Open();
       Console.WriteLine("Se realizo la conexion...");
       SqlCommand comando = new SqlCommand("select * from Productos", conectar);
       SqlDataReader tabla = comando.ExecuteReader();
       Try {
              while (tabla.Read())  {
```

```
            Console.WriteLine(""+ tabla[0]+"\t"+ tabla[1]);
               }
            tabla.close();
           } catch (SqlException e)
           {
            Console.WriteLine("fallo la apertura. "+e.Message);
            }
       }
catch (Exception ex)  {
        Console.WriteLine("Mal conexión. " + ex.ToString());
        }
Console.ReadKey();
        }
    }
}
```

2. **Ejemplo Video conexión C schap con DataGridView**

https://www.youtube.com/watch?v=ZaDfBNSlMaI

3. **Conexión con modelo de datos**

```
Public partial class maestro : Form
    {
String nomBD = "ensayo";
Datos objcon = new Datos();
SqlConnection a;
SqlDataReader tabla;

public maestro ()         {
Initialize Component ();
            }

Private void maestro_Load(object sender, EventArgs e)
        {
Form1 objclave = new Form1();
        }

Private void button1_Click(object sender, EventArgs e)
{
       a= objcon.conectar(nomBD);
       string con =  "DELETE FROM Articulo WHERE Nombre ='arroz';";
       int n = objcon.operarar(con, a);
        }

Private void button2_Click(object sender, EventArgs e)
{
       a = objcon.conectar(nomBD);
       string con = "Insert into Articulo values (6, 'arroz',1000,1);";
       int n = objcon.operarar(con, a);
        }
private void button3_Click(object sender, EventArgs e)
{
       a = objcon.conectar(nomBD);
       string con = "update Articulo set Fabricantes = 5 where Nombre='arroz';";
       int n = objcon.operarar(con, a);
        }
```

```
Private void button4_Click(object sender, EventArgs e)
{
       a = objcon.conectar(nomBD);
       stringconSQL = "select * from Articulo";
       tabla = objcon.consulta(conSQL, a);
       string mensaje="";
       try   {
       while (tabla.Read())
              {
              mensaje = mensaje+ (""+ tabla[0]+"\t"+ tabla[1]+"\n");
              }
       MessageBox.Show(mensaje);
       tabla.Close();
                     }
       catch (SqlExceptionve)
              {
              Console.WriteLine(ve.Message);
              }
       }
    }
}

class datos
    {

public SqlConnection conectar (string nomBD)
{
SqlConnection conectar = new SqlConnection("Data Source=localhost;Initial
Catalog="+nomBD+";Integrated Security=SSPI;");
try
            {
conectar.Open();
MessageBox.Show("Se realizo la conexion...");
return conectar;
            }
catch (Exception ex)
            {
MessageBox.Show("Fallo la conexión " + ex.ToString());
return null;
            }
        }

public SqlDataReader consulta(string conSQL, SqlConnection conector)
{
try
            {
SqlCommand comando = new SqlCommand(conSQL, conector);
SqlDataReader tabla = comando.ExecuteReader();
return tabla;
            }
catch (Exception ex)
            {
MessageBox.Show("Fallo la consulta " + ex.ToString());
return null;
            }
        }

public int operar(string conSQL, SqlConnection conector){
```

```
int num = 0;
try  {
SqlCommand comando = new SqlCommand( conSQL, conector);
num = comando.ExecuteNonQuery();
return num;
}
catch (SqlException e) {
MessageBox.Show("Fallo la consulta" + e.ToString());
return num;
       }
  }

public void cerrar(SqlConnection conector)
       {
try
           {
conector.Close();
           }
catch (SqlException eq) { }
       }
    }
}
```

4. Para realizar conexión a procedimiento almacenados de la base de datos se realiza las siguientes modificaciones:

```
string conSQL = "conNombre";
SqlCommand comando = new SqlCommand(conSQL, a);
comando.CommandType = CommandType.StoredProcedure;
comando.Parameters.Add("@Nom", SqlDbType.VarChar).SqlValue = txtNom.Text;
SqlDataReader tabla = comando.ExecuteReader();
```

5.

```
/* Formato general de un Script para crear bases de datos relacionadas*/

CREATE DATABASE nombreBD
GO
USE nombreBD
GO
CREATE TABLE NombreTabla1 (NombreClaveprimaria tipo PRIMARY KEY,
    ListaCampos Tipo)
GO
CREATE TABLE NombreTabla2 (NombreClaveprimaria tipo PRIMARY KEY,
    ListaCampos Tipo, CampoDeRelacion Tipo
    FOREIGN KEY(CampoDeRelacion) REFERENCES NombreTablaARelacionar(CampoArealcionar)
GO
INSERT INTO NombreTabla1 VALUES
(valor1Reg1, valor2Reg1),
(valor1Reg2, valor2Reg2),
(valor1Reg3, valor2Reg3),
(valor1RegN, valor2RegN) ;
GO
INSERT INTO NombreTabla2 VALUES
valor1Reg1, valor2Reg1, valor3Reg1),
(valor1Reg2, valor2Reg2, valor3Reg2),
(valor1Reg3, valor2Reg3, valor3Reg3),
(valor1RegM, valor2RegM, valor3RegM) ;
GO
```

```
/* Ejemplo de un Script para crear bases de datos relacionadas*/
USE master;
GO
create database Liga;
go

use Liga;
go

create table equipos (
      codigo int primary key not null,
      Nombre nvarchar(80) not null,
      puntos int not null default 0,
);
go

create table jugadores (
      id int primary key not null,
      nombre nvarchar(50) not null,
      valor money not null default 0,
      codigo_equipo int foreign key references equipos(codigo)
);
go

insert into equipos
values (1, 'medellin', 20),
         (2, 'Nacional', 20);
go

insert into jugadores
values (1, 'Cano', 85000, 1),
      (2, 'David', 50000, 1),
      (3, 'Ricaute', 120000, 1),
      (4, 'Nacnely', 90000, 2),
      (5, 'Alexis', 53000, 2),
      (6, 'Juan', 12000, 2);
go
```