Curso de Programación Web .NET

Unidad 5

“Datos”

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Índice

Trabajando con Datos

Introducción a datos

Los datos son una parte esencial de las aplicaciones que desarrollamos y la manera en que definimos las estrategias de trabajo y gestión de la información va a determinar si nuestra aplicación sea un exitosa o no.

En un mundo más orientado a los datos y a la integración de los mismos con otros sistemas es fundamental hoy en día que cualquier desarrollador tenga conocimientos de Base de datos así también como los mecanismos de abstracción e Integración con otros sistemas.

Base de Datos

Introducción

Una base de datos es un sistema de colección de información que es organizada para poder ser accedida, administrada y actualizada de manera sencilla.

Los datos en estas bases están organizados en **Filas**, **Columnas** y **Tablas**, además están indexados para hacer que sea más rápido buscarlos.

Las bases de datos contienen, aunque no necesariamente, grandes volúmenes de datos de registros como ventas, catálogos , inventarios y perfiles de clientes. Aunque, Las bases de datos pueden contener cualquier tipo de dato que nos interese persistir para que nuestro negocio o aplicación pueda operar en el tiempo.

Un caso típico de uso de base de datos es una aplicación que ingresa datos de ventas en un sistema centralizado de base de datos y por otro lado tenemos otra aplicación de reportes que utiliza esa base de datos para generar reportes e informes automáticamente para enviar a los gerentes.

Tipos de Base de datos

Existen muchos tipos de base de datos las que vamos a estar cubriendo en el curso son las base de datos **relacionales** las cuales se centran en datos estructurados, es decir que conocemos su formato y podemos plasmarlo en una tabla. Por otro lado tenemos las base de datos **no estructuradas** o llamadas NoSQL que se especializan en casos donde necesitamos trabajar con datos de naturaleza no estructurada o semi-estructurada como Video o imágenes o casos de Big Data.

```
Select Nombre, Apellido From Usuarios Where Nombre = “Leonel”
```

El Lenguaje SQL

SQL es el lenguaje que se utiliza para operar sobre los datos de las base de datos relacionales. Es generalmente confundido con las base de datos ya que utilizan el nombre del lenguaje como nombre de producto: SQL Server, MySQL.

Para que no se confundan: el Lenguaje **SQL** me permite operar sobre los datos de las bases de datos **MySQL**, **SQL Server**, **Oracle** etc.

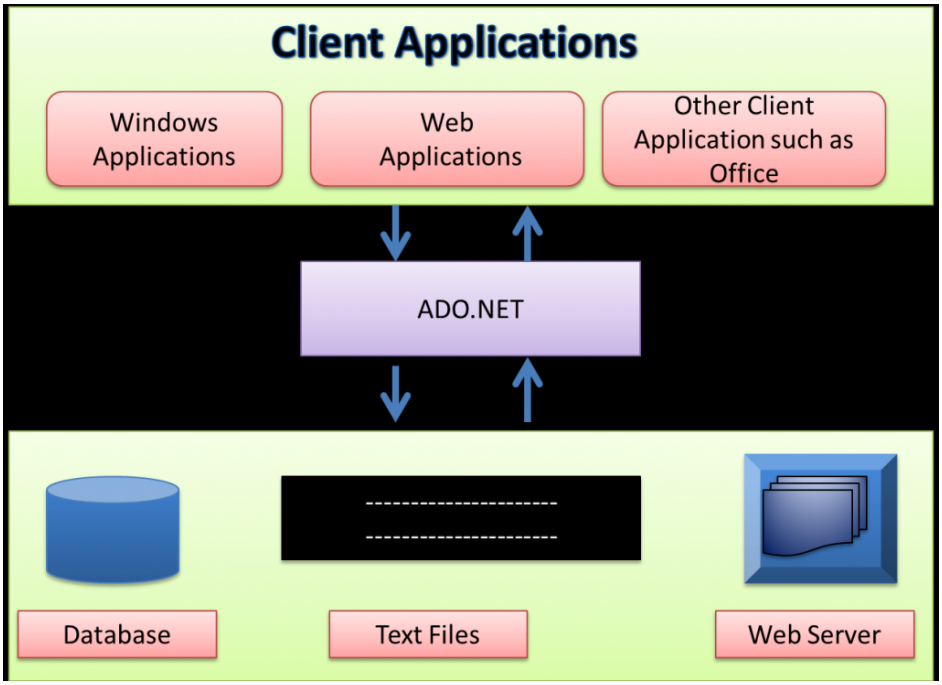
Nosotros vamos a ver lo básico de **SQL** para poder realizar consultas y actualizaciones en una base de datos Microsoft lo que nos va a permitir realizar y entender las aplicaciones con persistencia de datos. Expandir su conocimiento sobre base de datos y SQL siempre es una recomendación.

ADO.NET

Explicación

ADO.NET es un conjunto de librerías orientadas a objetos que nos permiten interactuar con Data sources u orígenes de datos. La mayoría de las veces un Data Source es una base de datos aunque también lo puede ser un archivo de texto, un excel o un archivo xml.

La gran ventaja de ADO.NET es que nos permite abstraernos del origen de datos, por lo tanto podemos programar con SQL Server, luego cambiar a otra base de datos y en nuestro sistema solo deberíamos



modificar la capa de Datos, pero nuestra aplicación por ejemplo un WinForm quedaría intacta.

Dentro de lo que es Base de datos hay muchas “marcas”, Entre las más conocidas tenemos:

- Microsoft SQL Server
- MySQL
- ORACLE
- PostgreSQL

Cada una con sus ventajas particulares, en este curso estamos usando **Microsoft SQL Server** ya que obviamente se integra muy bien con .NET. También es posible conectarse con otras base de datos como MySQL, Oracle o Postgre instalando componente de terceros en Visual Studio.

Ejemplo de conexión a un base de datos

Para este ejemplo vamos a necesitar una base de datos con la tabla "**Usuarios**" en la base de datos "**master**" y la siguiente estructura de Columnas y Filas:

Id	Nombre	Apellido	Edad
1	Leonel	Clavero	29
2	Romina	Alvarez	35
3	Lautaro	Rivas	17

Para generar esta tabla en tu base debes correr este script en ella, el primer comando crea la estructura y el segundo llena los datos.

```
GO
CREATE TABLE [dbo].[Usuarios] (
    [Id]      INT       NOT NULL,
    [Nombre]   NCHAR (40) NULL,
    [Apellido] NCHAR (40) NULL,
    [Edad]    INT       NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);

GO
INSERT INTO Usuarios (Id, Nombre, Apellido,Edad)
VALUES
    (1,'Leonel', 'Clavero',29),
    (2,'Romina', 'Alvarez', 35),
    (3,'Lautaro', 'Rivas',17);
```

Código .NET:

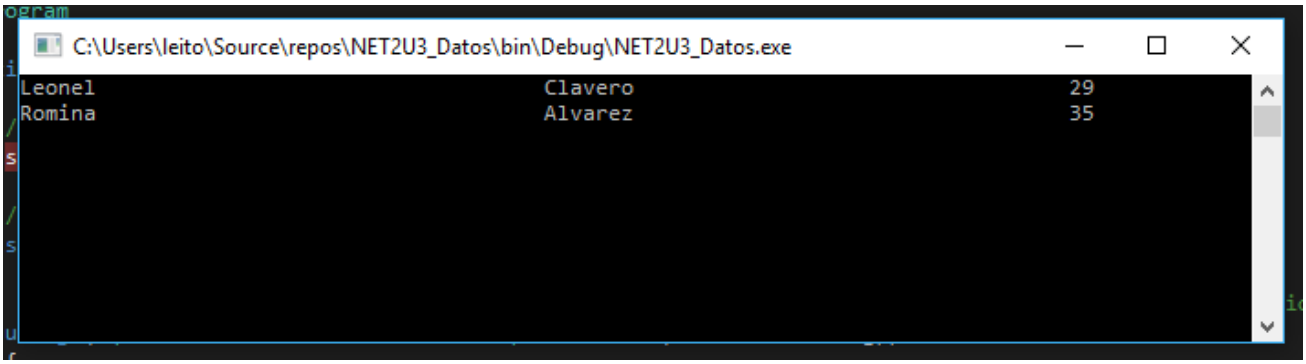
```
using System;
using System.Data;
using System.Data.SqlClient;

class Program
{
 static void Main()
  {
 // Armo el string de conexión para especificar que base conectarme
 string connectionString = "Data Source=(local);Initial Catalog=master;Integrated Security=true";

  // En este bloque “using” creo la conexión con la base de datos instanciando la clase SqlConnection
 using (SqlConnection connection = new SqlConnection(connectionString))
    {
 // Armo la query SQL
 string queryString ="SELECT Nombre, Apellido, Edad from dbo.Usuarios WHERE Edad > @ParametroEdad";
 // armo el comando con los parametros para traer solo a los mayores de 18
 SqlCommand command = new SqlCommand(queryString, connection);
 command.Parameters.AddWithValue("@ParametroEdad", 18);

  // Creo un DataReader para obtener los datos de la Base
  try
      {
          connection.Open();
          SqlDataReader reader = command.ExecuteReader();
    while (reader.Read())
          {
      Console.WriteLine("{0} {1} {2}", reader[0], reader[1], reader[2]);
          }
          reader.Close();
      }
   catch (Exception ex)
      {
          Console.WriteLine(ex.Message);
       }
       Console.ReadLine();
    }
 }
}
```

Resultado:



Vamos a ir por partes viendo los componentes claves.

Conectando a la base de datos

SQL Connection

Comencemos con lo primero: conectarse a la base de datos. El componente SQLConnection es el encargado de generar la conexión con la base, para ello acepta un **connection string**

El connection string es una cadena de texto con un formato específico que especifica todos los datos necesarios para hacer la conexión entre los más importantes:

- **Server:** La dirección de nuestro Servidor de base de datos, si el mismo fue instalado en la misma PC se completa con **localhost**, si está en otra computadora se introduce el **nombre del servidor**
- **DataBase:** Un servidor de base de datos puede tener varias bases de datos, cada una de ellas tiene un nombre. Esta campo me ayuda a definir a cual de esas base de datos conectarnos.
- **TrustedConnection:** Si el servidor esta dentro de nuestra PC o nos conectamos en una organización por Active directory entonces utilizo este parámetro y no requiero de usuario y contraseña.
- **User Id:** Si en cambio estoy accediendo a un servidor remoto en internet o local pero que requiere usuario, lo voy a introducir en esta campo.
- **Password:** El password del usuario especificado arriba. Por motivos de seguridad nunca conviene dejar el password expresado en un conection string. Otras alternativas son pedir el password por teclado al momento de la conexión y guardarlo encriptado

Para buscar ejemplos de connection strings les recomiendo la siguiente web:
https://www.connectionstrings.com/

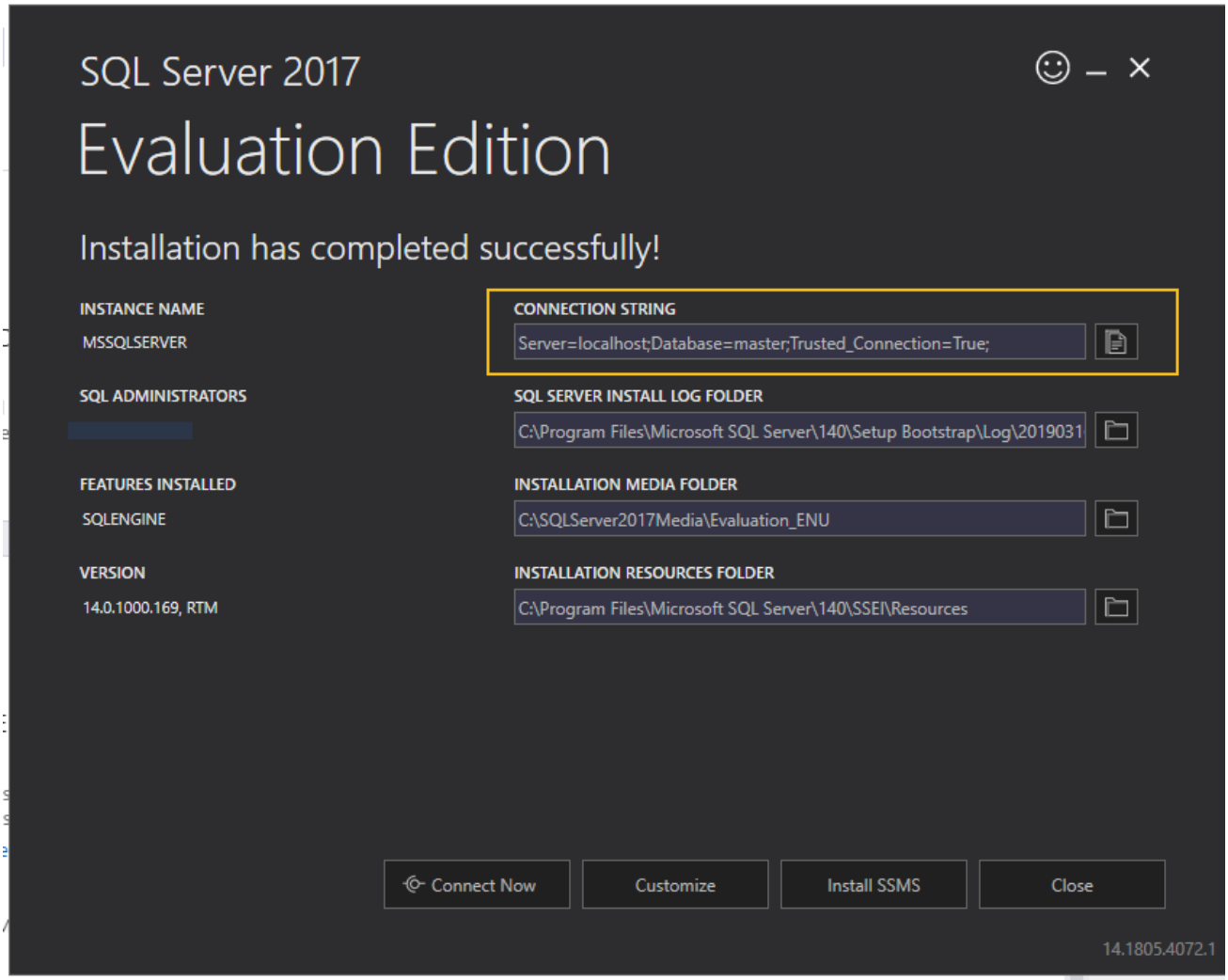
Ejemplo 1:

```
Server=myServerAddress;Database=myDataBase;Trusted_Connection=True;
```

Ejemplo 2:

```
Server=myServerAddress;Database=myDataBase;User Id=myUsername;
Password=myPassword;
```

Al instalar SQL Server en nuestra PC al finalizar la instalación nos va a aparecer el Connection String



El objeto que devuelve **SQLConnection** si la conexión al a base es exitosa es el que voy a utilizar para realizar las subsiguientes consultas a la base de datos:

```
SqlConnection connection =  new SqlConnection(connectionString)
connection.Open();
```

Leyendo datos de una base de datos

SQL Command y SQL Parameter

Para leer datos de una base de datos necesitamos un objecto llamado **SQLDataReader** pero antes de utilizarlo tenemos que preparar la consulta con sus parámetros usando un **SqlCommand** así:

```
// Armo la query SQL
   string queryString ="SELECT Nombre, Apellido, Edad from dbo.Usuarios WHERE Edad > @ParametroEdad";
   // armo el comando con los parametros para traer solo a los mayores de 18
   SqlCommand command = new SqlCommand(queryString, connection);
   command.Parameters.AddWithValue("@ParametroEdad", 18);
```

SQL Data Reader

A esta altura ya tenemos un comando preparado para ejecutar contra la base de datos con una conexión abierta a la base de datos entonces ejecutemos este Select para que nos devuelva nuestro objeto **SQLDataReader**.

```
   SqlDataReader reader = command.ExecuteReader();
   while (reader.Read())
   {
      Console.WriteLine("{0} {1} {2}", reader[0], reader[1], reader[2]);
   }
```

Una vez que tenemos el SQLDataReader puedo iterar sobre el llamando a Read() y recorriendo cada una de las filas de mi base de datos. El SQLDataReader devuelve un array para acceder a cada columna basta con poner el numero de columna entre los corchetes

Insertando, Borrando y Actualizando datos

Para realizar estas acciones se sigue el mismo principio anterior lo que cambia es la consulta SQL que ejecuto y el comando de ejecución de las mismas

Insertando

```
string InsertString = "INSERT INTO Usuarios (Id, Nombre, Apellido,Edad) VALUES (4, 'Alan', 'Roman', 11);";
SqlCommand cmdinsert = new SqlCommand(InsertString, conexion);
cmdinsert.CommandType = CommandType.Text;
cmdinsert.ExecuteNonQuery();
```

Modificando

```
string updateString = "UPDATE Usuarios SET Nombre = 'Alancito' WHERE Nombre = 'Alan';";
SqlCommand cmdupdate = new SqlCommand(updateString, conexion);
cmdupdate.CommandType = CommandType.Text;
cmdupdate.ExecuteNonQuery();
```

Borrando

```
string DeleteString = "DELETE FROM Usuarios WHERE Nombre = 'Alancito';";
SqlCommand cmddelete = new SqlCommand(DeleteString, conexion);
cmddelete.CommandType = CommandType.Text;
cmddelete.ExecuteNonQuery();
```

Conclusión

Esta unidad es el punto de partida para bases datos y el desarrollo de aplicaciones integradas. Es recomendable antes de seguir expandiéndose en este campo comprender y dominar los temas aquí expuestos ya que Datos es una carrera completa y muy enriquecedora que conlleva años de estudio.