Manual técnico ODBCConnect

Función

Esta es una librería DLL desarrollada en C# para poder hacer las conexiones a una base de datos a través de una ODBC ya configurada en la computadora. Nos ofrece métodos para poder hacer inserciones, modificaciones, eliminaciones de datos y lo más importante consultas a las diferentes tablas de nuestra base de datos. Esta librería trabaja a base de la DLL System.Data.Obc, ya que está es la que nos permite realizar las conexiones y las consultas a la propia base de datos. ODBCConnect nos sirve para poder hacer inserciones, actualizaciones, eliminaciones y consultas a la base de datos y nos brinda una forma fácil de adaptar cualquier DataGridView o cualquier ComboBox y poder desplegar los datos de alguna consulta.

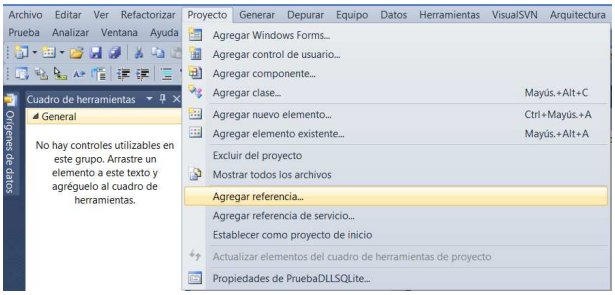
Contenido

Para poder utilizar está librería necesitamos tener en el mismo directorio el DLL Systme.Data.SQLite, por está razón para poder usarla en cualquier proyecto de Visual Studio solo necesitamos agregar la SQLiteConnect como referencia.

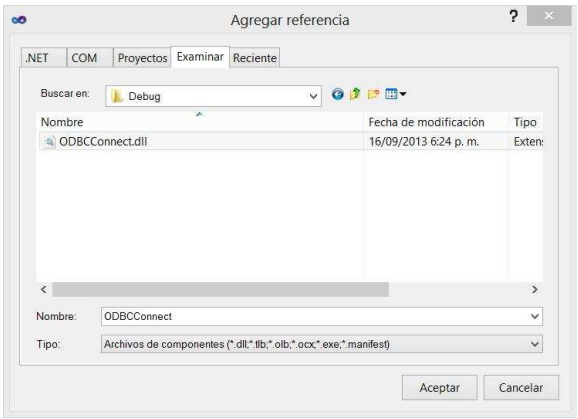
Instrucciones de uso

1. Agregar referencia

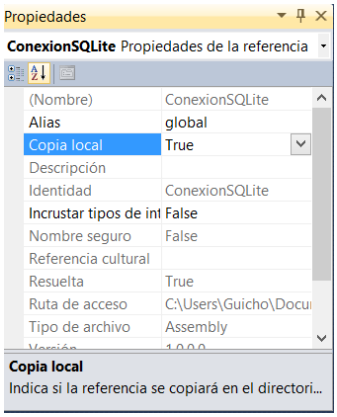
   Debemos ir a nuestro proyecto y en el menú de **Proyecto** darle click en **Agregar referencia.**



   En la pestaña **Examinar** buscamos la librería, donde la hayamos almacenado, la buscamos con el nombre de **ODBCConnect.dll**



Ya que la hemos agregado, en el **Explorador de soluciones** en el menú de **References**, seleccionar la librería y en las **Propiedades** verificamos que la opción de **Copia local** tenga **True**. De esta manera nuestra DLL se copiara en nuestro proyecto.



2. Agregar librería a nuestras clases

Ahora, en las clases o en los Forms donde utilizaremos la librería debemos llamarla en nuestra área de librerías. La llamamos con la instrucción **using ODBCConnect;**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ODBCConnect;
```

3. Instanciar la clase

ODBCConnect tiene una clase, para poder empezar a usar está clase donde se encuentran todas las funciones debemos de instanciarla y decirle el nombre del Odbc que estaremos utilizando.

```
DBConnect db = new DBConnect("Prueba");
```

De ahora en adelante podemos llamar a las funciones de la clase DBConnect, donde podremos consultar, insertar, actualizar o eliminar datos.

4. Inserciones (insertar(tabla, datos))

Para poder realizar inserciones debemos de apoyarnos en el método **insertar().** Este método requiere de dos parámetros, el primero es el nombre de la tabla a donde queremos insertar datos, esta variable es de tipo **string.** El segundo parámetro es un diccionario de datos, en él debemos de especificar el campo y el valor de este que queremos insertar en la base de datos, este en de tipo **Dictionary<string,string>.**

```
string tabla = "cliente";
Dictionary<string, string> dict = new Dictionary<string, string>();
dict.Add("id", textBox3.Text);
dict.Add("nombre", textBox1.Text);
dict.Add("direccion", textBox2.Text);
db.insertar(tabla, dict);
```

5. Actualizaciones (actualizar(tabla, datos, condición))

Este es un método parecido al de inserción, pero necesitamos de tres parámetros para poder llamarlo, el nombre de la tabla, un diccionario de datos donde especificamos que campos vamos a afectar y cuáles serán sus valores y la condición que se tiene que cumplir para que se realice el update.

```
string tabla = "cliente";
Dictionary<string, string> dict = new Dictionary<string, string>();
dict.Add("nombre", textBox1.Text);
string condicion = "id="+comboBox1.SelectedValue;
db.actualizar(tabla, dict, condicion);
```

6. Eliminación (eliminar(tabla,condición))

Este método solo necesita de dos parámetros, la tabla y la condición que se tiene que cumplir para poder realizar el Delete.

```
string tabla = "cliente";
string condicion = "id=" + comboBox1.SelectedValue ;
db.eliminar(tabla, condicion);
```

7. Consulta (consultar(query))

Este método nos devuelve una **ArrayList**, y dentro de este encontramos cada uno de los registros devueltos por la consulta dentro de un **Dictionary<string,string>**. Está consulta generalmente es usada para cuando utilizaremos los datos devueltos de la consulta fuera de un DataGridView. Los keys de los Dictionary estarán relacionados con el nombre de los campos que hayamos consultados, de esta manera será más fácil la forma de buscar algún valor específico dentro del diccionario.

```
string query = "select * from cliente where id="+comboBox1.SelectedValue;
System.Collections.ArrayList array = db.consultar(query);
foreach (Dictionary<string, string> dict in array)
{
    textBox1.Text = dict["nombre"];
    textBox2.Text = dict["id"];
}
```

8. Consulta solo un registro o fila(consultar_un_registro(query))

Esta función es una forma resumida de la instrucción anterior, la diferencia es que solo nos devolverá un registro o fila de alguna tabla que estemos consultando. Dicho registro será devuelto en un **Dictionary<string,string>** para poder referencia a los keys de cada columna. Este método debe ser utilizado solamente si queremos consultar una fila nada más, si es más de una fila es mejor utilizar el método del punto no. 7.

```
Dictionary<string, string> d = new DBConnect("factura").consultar_un_registro(query);
textBox3.Text = d["bodega"];
char a = d["serie"][0];
int i = (int)a;
textBox6.Text = i.ToString();
textBox7.Text = d["no"];
textBox4.Text = d["vendedor"];
textBox1.Text = d["nit"];
textBox2.Text = d["cliente"];
d["fecha"] = d["fecha"].Substring(0, 10);
DateTime dt = Convert.ToDateTime(d["fecha"]);
dateTimePicker1.Value = dt;
```

9. Consulta a DataGridView (consulta_DataGridView(query))

Este método nos es útil cuando queremos desplegar los datos de alguna consulta directamente a un **DataGridView**, ya que nos devuelve directamente un objeto tipo DataGridView. Solo debemos de mandar la consulta que deseamos y el devolverá los datos correspondientes. Para poder desplegar los datos solo debemos de igualar los DataSource de nuestro DataGridView y el objeto devuelto por este método que es un DataTable.

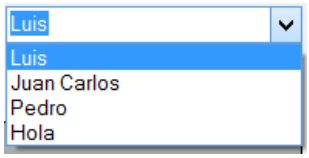
```
string query = "select id,nombre as 'Nombre de cliente' from cliente";
dataGridView1.DataSource = db.consulta_DataGridView(query);
```

	id	Nombre de cliente
▶	1	Luis
	2	Juan Carlos
	3	Pedro
		Hola

10. Consulta a ComboBox (consulta_ComboBox(query))

Este método nos devuelve una DataTable con el cual podemos llenar fácilmente el DataSource de cualquier ComboBox, con los datos de la consulta que hemos enviado. Solo debemos de especificar qué campo utilizaremos para el valor a mostrar y el valor que utilizaremos al hacer click en el ComboBox.

```
string sql = "select id,nombre from cliente";
comboBox1.DataSource = db.consulta_ComboBox(sql);
comboBox1.DisplayMember = "nombre";
comboBox1.ValueMember = "id";
```



11. Operaciones transaccionales

Estas operaciones están formadas por dos métodos diferentes que nos permiten hacer varias inserciones y/o actualizaciones al mismo tiempo manteniendo la integridad de los datos, en caso que exista algún problema se activará un Rollback que eliminará todos los datos que habían sido ingresados con anterioridad.

```
db.empezar_transaccion();
```

Con la instrucción anterior se inicializa la transacción.

```
a = db.insertar("factura", dict);
if (a == 0) error = true;
```

Estas instrucciones nos ayudarán a determinar si hubo un error en el momento de realizar una operación.

```
db.terminar_transaccion(error);
```

Por último indicamos la finalización de la transacción, en caso que exista algún error se activará la instrucción Rollaback, de lo contrario se hará el Commit correspondiente.