Contenido

Acceso conectado a MySql para realizar select.

Como hemos mencionado anteriormente el acceso conectado no se puede hacer con asistentes por lo tanto hay que utilizar las clases por código. Los pasos que hay que dar son los siguientes:

1. Instanciar un objeto `MySqlConnection`.
2. Abrir la conexión
3. Instanciar un objeto `MySqlCommand`.
4. Instanciar un objeto `MySqlDataReader`.
5. Recorrer el `MySqlDataReader`.
6. Mostrar los datos al usuario.
7. Cerrar la conexión.

Para este ejemplo usaré la *tabla vendedor* de la base de datos *proveedores* creada con MySql. Crearemos un nuevo proyecto llamado ejemplo1. Añadiremos al proyecto un listbox y un botón.

```
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 MySql.Data.MySqlClient;


namespace ejemplo1
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {

            MySqlConnection conn = new MySqlConnection();               //  1.   Instanciar un
objeto MySqlConnection
            String connectionString;
            /*en cualquier conexión necesitamos una cadena de c
             * onexión el formato varía para cada base de datos*/
            connectionString ="server=localhost;UserId=root;password=root;database=proveedores";
            try
            {
                conn.ConnectionString = connectionString;
                conn.Open();                                            //2.     Abrir la
conexión
                label1.Text = "Proveedores...";
                String _sql = "Select nomvend, nombrecomer from vendedor";
                MySqlCommand sentencia = new MySqlCommand(_sql, conn);  //3.     Instanciar un
objeto MySqlCommand
                MySqlDataReader reader = sentencia.ExecuteReader();     //4.     Instanciar un
objeto MySqlDataReader.
                while (reader.Read()){                                  //5.     Recorrer el
MySqlDataReader.
                                                                        //6.     Mostrar los
datos al usuario
                    listBox1.Items.Add(reader.GetString("nomvend") + "" +
reader.GetString("nombrecomer"));
                }


                conn.Close();                                           //7.     Cerrar la
conexión
            }
            catch (MySqlException)
            {
                MessageBox.Show("Error en la conexión","ERROR");
            }
        }
```

*Otra forma de acceder a los datos

```
            conn.Open();                                        //2.         Abrir la conexión

            String _sql = "Select * from vendedor where numvend="+comboBox1.SelectedItem;
            MySqlCommand sentencia = new MySqlCommand(_sql, conn);  //3.          Instanciar un objeto MySqlCommand
            MySqlDataReader reader = sentencia.ExecuteReader();     //4.          Instanciar un objeto MySqlDataReader.
            while (reader.Read())
            {                                   //5.         Recorrer el MySqlDataReader.
                //6.        Mostrar los datos al usuario
                //listBox1.Items.Add(reader.GetString("nomvend") + "" + reader.GetString("nombrecomer"));
                textBox1.Text= reader[1].ToString();
                    textBox2.Text= reader[2].ToString();
                textBox3.Text=reader[3].ToString();
                textBox4.Text=reader[4].ToString();
                textBox5.Text=reader[5].ToString();
                textBox6.Text= reader[6].ToString();
            }
```

La cadena de conexión de Mysql es:

```
connectionString ="server=localhost;UserId=root;password=root;database=proveedores";
```

Para ver las distintas formas de formarlas podéis consultar, por curiosidad, en:

http://www.connectionstrings.com/mysql

Acceso conectado a Mysql para SQL distinto de select.

```
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 MySql.Data.MySqlClient;

namespace ejemplo1
{
    public partial class alta : Form
    {
        MySqlConnection conn = new MySqlConnection();               //  1.      Instanciar un
objeto MySqlConnection
        String connectionString;

        public alta()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            connectionString = "server=localhost;User
Id=root;password=root;database=proveedores";
            try
            {
                conn.ConnectionString = connectionString;
                conn.Open();
                String _sql = "insert into vendedor
values("+textBox7.Text+",'"+textBox1.Text+"','"+textBox3.Text+"','"+

textBox2.Text+"','"+textBox4.Text+"','"+textBox5.Text+"','"+
                                                                textBox6.Text+"')";
                MySqlCommand sentencia = new MySqlCommand(_sql, conn);  //3.     Instanciar un
objeto MySqlCommand
                sentencia.ExecuteNonQuery();

            }
            catch (Exception error)
            {
                MessageBox.Show("Error..."+error);
            }
            conn.Close();
        }
    }
}
```

En estos ejemplos de acceso conectado hemos utilizado tres clases: Connection, Command y DataReader. Veamos ahora las propiedades y métodos más importantes de estas clases, dará igual para que base de datos sean pues son los mismos para todas, por ejemplo los métodos y propiedades de MySqlConnection son los mismos que los de OleDbConnection.

Connection.

Es el objeto que está más cercano a la base de datos, establece un canal de comunicación entre nuestra aplicación y la base de datos. La propiedad más importante es **connectionString**, cada gestor de bases de datos necesita una cadena de conexión distinta pero la información suele ser el controlador a usar, nombre de la base de datos y

usuario. A esta propiedad se le da valor cuando se llama al constructor de la clase. Los métodos más destacables son: Open() para abrir la conexión y Close() para cerrarla.

Command.

Establecida una conexión con una base de datos, la siguiente operación lógica consiste en enviarle sentencias SQL. A través de un objeto Command podremos obtener un conjunto de resultados del almacén de datos. En este caso, los resultados se pasarán a otros objetos de ADO .NET, como DataReader o DataAdapter.

Un objeto Command lo vamos a crear a partir de una conexión ya existente, y va a contener una sentencia SQL para ejecutar sobre la conexión establecida con el origen de datos.

Entre las ***propiedades*** cabe destacar las siguientes.

- **CommandText**. Contiene una cadena de texto que va a indicar la sentencia SQL o procedimiento almacenado que se va a ejecutar sobre el origen de los datos.
- **CommandTimeout**. Tiempo de espera en segundos que se va a aplicar a la ejecución de un objeto Command. Su valor por defecto es de 30 segundos.
- **CommandType**. Indica el tipo de comando que se va a ejecutar contra el almacén de datos, es decir, indica como se debe interpretar el valor de la propiedad CommadText. Puede tener los siguientes valores:
    - StoredProcedure, para indicar que se trata de un procedimiento almacenado.
    - TableDirect se trata de obtener una tabla por su nombre (únicamente aplicable al objeto OleDbCommand); y …
    - Text que indica que es una sentencia SQL. EL valor por defecto es Text.
- **Connection**. Devuelve el objeto Connection utilizado para ejecutar el objeto Command correspondiente.
- **Parameters**. Colección de parámetros que se pueden utilizar para ejecutar el objeto Command, esta colección se utiliza cuando deseamos ejecutar sentencias SQL que hacen uso de parámetros, esta propiedad devuelve un objeto de la clase ParameterCollection. Estas colecciones contendrán objetos de la clase Paramter para representar a cada uno de los parámetros utilizados.

Estos parámetros también son utilizados para ejecutar procedimientos almacenados.

Los principales ***métodos*** de estas clases.

- CreateParameter. Crea un parámetro para el que después podremos definir una serie de características específicas como pueden ser el tipo de dato, su valor, tamaño, etc. Devolverá un objeto de la clase Parameter.
- ExecuteNonQuery. Ejecuta la sentencia SQL definida en la propiedad ComandText contra la conexión definida en la propiedad Connection. La sentencia a ejecutar debe ser de un tipo que no devuelva un conjunto de registros, por ejemplo ***Update, Delete o Insert***. Este método *devuelve un entero* indicando el número de filas que se han visto afectadas por la ejecución del objeto Command.
- ExecuteReader. Ejecuta la sentencia SQL definida en la propiedad ComandText contra la conexión definida en la propiedad Connection. En este caso, la sentencia sí devolverá un conjunto de registros. El resultado de la ejecución de este será un objeto de la clase DataReader, que nos va a permitir leer y recorrer los resultados devueltos por la ejecución del objeto Command correspondiente.
- ExecuteScalar. Este método se utiliza cuando deseamos obtener la primera columna de la primera fila del conjunto de registros, el resto de datos no se tendrán en cuenta. La utilización de este método tiene sentido cuando estamos ejecutando una sentencia SQL del tipo Select Count(*). Este método devuelve un objeto de la clase genérica Object.
- Prepare. Este método construye una versión compilada del objeto Command dentro del almacén de datos.

DataReader.

Un objeto DataReader permite la navegación hacia delante y de sólo lectura, de los registros devueltos por una consulta. Los DataReader permanecen conectados durante todo el tiempo que realizan el recorrido por los registros que contienen.
Para obtener un DataReader, ejecutaremos el método ExecuteReader( ) de un objeto Command basado en una consulta SQL o procedimiento almacenado.
Las principales ***propiedades***:

- FieldCount. Devuelve el número de columnas (campos) presentes en el fila (registro) actual.
- IsClosed. Devolverá los valores True o False, para indicar si el objeto DataReader está cerrado o no.

- Item. Devuelve en formato nativo, el valor de la columna cuyo nombre le indicamos como índice en forma de cadena de texto.

Una vez vistas las propiedades, vamos a comentar los ***métodos*** más destacables.

- **Close( )**. Cierra el objeto DataReader liberando los recursos correspondientes.
- **GetXXX( ).** El objeto DataReader presenta un conjunto de métodos que nos van a permitir obtener los valores de las columnas contenidas en el mismo forma de un tipo de datos determinado, según el método GetXXX empleado. Existen diversos métodos GetXXX atendiendo al tipo de datos de la columna, algunos ejemplos son GetBoolean(), GetInt32(), GetString(), GetChar(), etc. Como parámetro a este método le debemos indicar el número de orden de la columna que deseamos recuperar.
- **NextResult( ).** Desplaza el puntero actual al siguiente conjunto de registros, cuando la sentencia es un procedimiento almacenado de SQL o una sentencia SQL que devuelve más de un conjunto de registros.
- **Read( ).** Desplaza el cursor actual al siguiente registro permitiendo obtener los valores del mismo a través del objeto DataReader. Este método devolverá True si existen más registros dentro del objeto DataReader, False si hemos llegado al final del conjunto de registros. La posición por defecto del objeto DataReader en el momento inicial es antes del primer registro, por lo tanto para recorrer un objeto DataReader debemos comenzar con una llamada al método Read(), y así situarnos en el primer registro.