Contenido

Programación por Capas

Una vez repasado el código podemos comprobar que siempre repetimos las mismas sentencias a la hora de hacer peticiones Sql, con muy buen criterio, siempre se repiten el mismo tipo de instrucciones, veamos los siguientes código que hacen referencia a la Select y al Insert…

```
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();
               }


               conn.Close();
```

```
            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();
```

Una buena práctica a la hora de programar es crearnos una **clase** donde controlemos todo el control sobre la base de datos, conexiones, abrir y cerrar así como las distintas operaciones que sobre la base de datos vayamos a realizar.

Modelo-Vista-Controlador

Esta manera de programar o mejor dicho esta arquitectura del software es conocida como **MVC** (Modelo-Vista-Controlador) que consiste en:

- Separar la lógica de negocio de la interfaz de usuario, así tendremos que:
    - Facilita la evolución por separado de ambos aspectos.

- Incrementa reutilización y flexibilidad.

En nuestro caso para gestionar los datos que debe utilizar la aplicación utilizamos un SGBD; dicha gestión corresponde al **modelo**. La unión entre *capa de presentación(lo que ve el usuario)* y *capa de negocio(el programa)* conocido en el paradigma de la *programación por capas* representaría la integración entre la **Vista** y su correspondiente **Controlador** de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa *visual gráfica* de su correspondiente *programación y acceso a datos*, algo que mejora el desarrollo y mantenimiento de la *Vista* y el *Controlador* en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

En resumidas cuentas. El Modelo Vista Controlador es un patrón para el desarrollo del software que se basa en separar los datos (por un lado), la interfaz del usuario (por otro) y la lógica interna (por un último lado). Es mayormente usado en aplicaciones web, dónde la vista es la página HTML, el modelo es el Sistema de Gestión de Base de Datos y la lógica interna, y el controlador es el responsable de recibir los eventos y darles solución.

Ejemplo

Dicho esto nuestra aplicación podría quedar así:

En la parte de la Vista

La interfaz de usuario

```
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 clase
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
      // List<string> vendedor = new List<string>();
      MySqlDataReader vendedor;

      DBConnect conn = new DBConnect();


      vendedor=conn.Select("Select * from vendedor");

      listView1.View = View.Details;
      listView1.GridLines = true;
      listView1.FullRowSelect = true;
      listView1.MultiSelect = true;
      listView1.Columns.Add("Numero");
      listView1.Columns.Add("Nombre");
      listView1.Columns.Add("Empresa");
      listView1.Columns.Add("Telefono");
      listView1.Columns.Add("Direccion");
      listView1.Columns.Add("Ciudad");
      listView1.Columns.Add("Provincia");
      ListViewItem elemento = new ListViewItem();
```

```
        //foreach(string s in vendedor)
         while (vendedor.Read())

        {
           elemento = listView1.Items.Add(vendedor[0].ToString());
           elemento.SubItems.Add(vendedor[1].ToString());
           elemento.SubItems.Add(vendedor[2].ToString());
           elemento.SubItems.Add(vendedor[3].ToString());
           elemento.SubItems.Add(vendedor[4].ToString());
           elemento.SubItems.Add(vendedor[5].ToString());
           elemento.SubItems.Add(vendedor[6].ToString());

        }
         vendedor.Close();
         conn.closeSelect();

    }

    private void button1_Click(object sender, EventArgs e)
    {
        DBConnect conn = new DBConnect();
        try{
        conn.Insert("insert into vendedor
values("+textBox7.Text+",'"+textBox1.Text+"','"+textBox3.Text+"','"+

textBox2.Text+"','"+textBox4.Text+"','"+textBox5.Text+"','"+
                                    textBox6.Text+"')");


        }catch(Exception y){
           MessageBox.Show("Error en el insert...");
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        DBConnect conn = new DBConnect();
        try
        {

           conn.Insert("update vendedor set nomvend='" + textBox1.Text +
                       "',nombrecomer='" + textBox2.Text +
                       "',telefono='" + textBox3.Text +
                       "',calle='" + textBox4.Text +
                       "',ciudad='" + textBox5.Text +
                       "',provincia='" + textBox6.Text +
                       "' where numvend=" + textBox7.Text);
        }
        catch (Exception y)
        {
           MessageBox.Show("Error en el update...");
        }
    }

    private void listView1_Click(object sender, EventArgs e)
    {
        foreach (ListViewItem item in listView1.SelectedItems)
        {

           MySqlDataReader reader;

           DBConnect conn = new DBConnect();



           reader = conn.Select("Select * from vendedor where numvend=" +
item.Text.ToString());
           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();
              textBox7.Text = item.Text.ToString();
           }
```

Como podéis observar se ha simplificado el código, invocamos a la clase DBConnect y ahí tendremos toda la lógica para la conexión con la base de datos.

```
            reader.Close();
            conn.closeSelect();
        }
    }


    private void button3_Click(object sender, EventArgs e)
    {
        DBConnect conn = new DBConnect();
        try
        {
            conn.Delete("delete from vendedor where numvend=" + textBox7.Text);
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
        }
        catch (Exception y)
        {
            MessageBox.Show("Error en el borrado");
        }
    }
  }
}
```

Este método lo he tenido que implementar a parte. Si cierro el datareader, en la clase, no podría acceder a el.

En la parte del Controlador

tendremos la clase con los accesos a la BBDD

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace clase
{
  class DBConnect
  {
    private MySqlConnection connection;
    private string server;
    private string database;
    private string uid;
    private string password;

   /// <summary>
    /// Constructor variables abrimos la conexion con la
    /// base de datos
   /// </summary>

    public DBConnect()
    {

      server = "localhost";
      database = "proveedores";
      uid = "root";
      password = "root";
      string connectionString;
      connectionString = "SERVER=" + server + ";" + "DATABASE=" +
      database + ";" + "UID=" + uid + ";" + "PASSWORD=" + password + ";";

      connection = new MySqlConnection(connectionString);
    }

    public void closeSelect()
    {
      CloseConnection();
    }
    /// <summary>
    /// Abrir conexion
    /// </summary>
    /// <returns>true o false</returns>

    private bool OpenConnection()
```

Constructor para crearnos la conexión con la base de datos. Esto me permitirá poder cambiar de base de datos o usuarios.

Método privado para abrir la conexión

```
{
    try
    {
        connection.Open();
        return true;
    }
    catch (MySqlException ex)
    {

        //0: Cannot connect to server.
        //1045: Invalid user name and/or password.
        switch (ex.Number)
        {
            case 0:
                MessageBox.Show("Cannot connect to server.  Contact administrator");
                break;

            case 1045:
                MessageBox.Show("Invalid username/password, please try again");
                break;
        }
        return false;
    }

}

/// <summary>
/// Close conexion
/// </summary>
/// <returns>true o false</returns>
private bool CloseConnection()
{
    try
    {
        connection.Close();
        return true;
    }
    catch (MySqlException ex)
    {
        MessageBox.Show(ex.Message);
        return false;
    }

}


/// <summary>
/// Insert
/// </summary>
/// <param name="query"> La cadena para hacer el insert </param>

public void Insert(String query)
{
    //open
    if (this.OpenConnection() == true)
    {

        MySqlCommand cmd = new MySqlCommand(query, connection);


        cmd.ExecuteNonQuery();

        //close
        this.CloseConnection();
    }

}

/// <summary>
/// Update
/// </summary>
/// <param name="query">La cadena para Actualizar</param>

public void Update(String query)
{
    //Open
    if (this.OpenConnection() == true)
    {

        MySqlCommand cmd = new MySqlCommand(query,connection);
```

Método privado para cerrar la conexión

En los métodos públicos Insert, update y delete si os fijais todos tienen el mismo conjunto de instrucciones, se hubiese podido crear uno solo y le pasamos la operación, pero prefiero hacerlos independientes por claridad.

Exceptuando el Select que nos tiene que devolver un conjunto de datos, este es más configurable dependiendo de las necesidades del usuario.

```
            //cmd.CommandText = query;

            //cmd.Connection = connection;


            cmd.ExecuteNonQuery();

            //close
            this.CloseConnection();
        }

    }

    /// <summary>
    /// Delete
    /// </summary>
    /// <param name="query">La cadena para borrar un registro</param>

    public void Delete(String query)
    {
        if (this.OpenConnection() == true)
        {
            MySqlCommand cmd = new MySqlCommand(query, connection);
            cmd.ExecuteNonQuery();
            this.CloseConnection();
        }

    }

    /// <summary>
    ///
    /// </summary>
    /// <param name="query">La consulta a la tabla</param>
    /// <returns> el resultado de la select en forma de datareader</returns>
    public MySqlDataReader Select(String query)
    {
        List<string> list = new List<string>();

        //Open
        if (this.OpenConnection() == true)
        {
            //Create
            MySqlCommand cmd = new MySqlCommand(query, connection);

            MySqlDataReader dataReader = cmd.ExecuteReader();

            return dataReader;
        }
        else
        {
            return null;
        }

    }

  }
}
```

La parte Modelo

tendríamos el SGBD.