C# MySQL

http://www.codeproject.com/Articles/43438/Connect-C-to-MySQL
MySQL
เลิกใช้ Appserv ได้แล้ว
HeidiSQL

```
54 SELECT * FROM `connectcsharptomysql`.`tableinfo` LIMIT 1000;
55 SHOW CREATE TABLE `connectcsharptomysql`.`tableinfo` ;
56 SELECT * FROM `connectcsharptomysql`.`tableinfo` LIMIT 1000;
57 SHOW CREATE TABLE `connectcsharptomysql`.`tableinfo` ;
```
Downloading Connector/Net

• First make sure you have downloaded and installed the MySQL Connector/NET from the MySQL


• ตอนนี้มีตัว update กว่านี้แล้ว

- HeidiSQL_8.3.0.4694_Setup.exe
- mysql-connector-net-6.8.3.msi
- xampp-win32-1.8.3-3-VC11-installer.exe
- wdexpress_full.exe
• ตัวอย่าง Browse ไปที่ C:\Program Files\MySQL\MySQL Connector Net 6.8.3\Assemblies\v4.5
Creating Database

- create database ConnectCsharpToMysql;
- use ConnectCsharpToMysql;
Creating Table

- Create table: tableinfo

create table tableInfo (  
    id INT NOT NULL AUTO_INCREMENT, 
    name VARCHAR(30), 
    age INT, PRIMARY KEY(id) );
In order to use the application on other computers that don't have the connector installed, we will have to create a DLL from the reference.
//Add MySql Library

using MySql.Data.MySqlClient;

Then declaring and initializing the variables that we will use:

- **connection**: will be used to open a connection to the database.
- **server**: indicates where our server is hosted, in our case, it's localhost.
- **database**: is the name of the database we will use, in our case it's the database we already created earlier which is connectcsharptomysql.
- **uid**: is our MySQL username.
- **password**: is our MySQL password.
- **connectionString**: contains the connection string to connect to the database, and will be assigned to the connection variable.
private MySqlConnection connection;

private string server;

private string database;

private string uid;

private string password;

public Form1()
{
  InitializeComponent();

  server = "localhost";

  database = "connectcsharptomysql";

  uid = "root";

  password = "root";

  string connectionString;

  connectionString = "SERVER=\" + server + \";\" + "DATABASE=\" + database + \";\" + "UID=\" + uid + \";\" + "PASSWORD=\" + password + \";\";

  connection = new MySqlConnection(connectionString);
}
private bool OpenConnection()
{
    try
    {
        connection.Open();
        return true;
    }
    catch (MySqlException ex)
    {
        // When handling errors, you can your application's response based on the error number.
        // The two most common error numbers when connecting are as follows:
        // 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;
    }
}
private bool CloseConnection()
{
    try
    {
        connection.Close();
        return true;
    }
    catch (MySqlException ex)
    {
        MessageBox.Show(ex.Message);
        return false;
    }
}
Working with Insert, Update, Select, Delete

Usually, Insert, update and delete are used to write or change data in the database, while Select is used to read data. For this reason, we have different types of methods to execute those queries.

The methods are the following:

- **ExecuteNonQuery**: Used to execute a command that will not return any data, for example Insert, update or delete.

- **ExecuteReader**: Used to execute a command that will return 0 or more records, for example Select.

- **ExecuteScalar**: Used to execute a command that will return only 1 value, for example Select Count(*).
Steps

Start with Insert, update and delete, which are the easiest. The process to successfully execute a command is as follows:

1. Open connection to the database.
2. Create a MySQL command.
3. Assign a connection and a query to the command. This can be done using the constructor, or using theConnection and the CommandText methods in the MySqlCommand class.
4. Execute the command.
5. Close the connection.
private void button1_Click(object sender, EventArgs e)
{
    string query = "INSERT INTO tableinfo (name, age) VALUES('John Smith', '33');"
    //open connection
    if (this.OpenConnection() == true)
    {
        //create command and assign the query and connection from the constructor
        MySqlCommand cmd = new MySqlCommand(query, connection);
        //Execute command
        cmd.ExecuteNonQuery();
        //close connection
        this.CloseConnection();
    }
}
private void button2_Click(object sender, EventArgs e)
{
    string query = "UPDATE tableinfo SET name='Joe', age='22' WHERE name='John Smith'";

    //Open connection
    if (this.OpenConnection() == true)
    {
        //create mysql command
        MySqlCommand cmd = new MySqlCommand();
        //Assign the query using CommandText
        cmd.CommandText = query;
        //Assign the connection using Connection
        cmd.Connection = connection;
        //Execute query
        cmd.ExecuteNonQuery();
        //close connection
        //close connection
        this.CloseConnection();
    }
}
private void button3_Click(object sender, EventArgs e)
{
    string query = "DELETE FROM tableinfo WHERE name='John Smith'";

    if (this.OpenConnection() == true)
    {
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.ExecuteNonQuery();
        this.CloseConnection();
    }
}
Steps for select

To execute a Select statement, we add a few more steps, and we use the ExecuteReader method that will return a dataReader object to read and store the data or records.

1. Open connection to the database.
2. Create a MySQL command.
3. Assign a connection and a query to the command. This can be done using the constructor, or using the Connection and the CommandText methods in the MySqlCommand class.
4. Create a MySqlDataReader object to read the selected records/data.
5. Execute the command.
6. Read the records and display them or store them in a list.
7. Close the data reader.
8. Close the connection.
private void button4_Click(object sender, EventArgs e)
{
    string query = "SELECT * FROM tableinfo";
    //Open connection
    if (this.OpenConnection() == true)
    {
        //Create Command
        MySqlCommand cmd = new MySqlCommand(query, connection);
        //Create a data reader and Execute the command
        MySqlDataReader dataReader = cmd.ExecuteReader();
        //Read the data and store them in the list
        while (dataReader.Read())
        {
            textbox1.text += dataReader["id"] +"" +dataReader["name"] +
""+dataReader["age"] + "\r\n";
        }
        //Close Data Reader
        dataReader.Close();
        //Close Connection
        this.CloseConnection();
    }
}
Sometimes, a command will always return only one value, like for example if we want to count the number of records, we have been using Select Count(*) from tableinfo;, in this case, we will have to use the method ExecuteScalar that will return one value.

- The process to successfully run and ExecuteScalar is as follows:
  1. Open connection to the database.
  2. Create a MySQL command.
  3. Assign a connection and a query to the command. This can be done using the constructor, or using the Connection and the CommandText methods in the MySqlCommand class.
  4. Execute the command.
  5. Parse the result if necessary.
  6. Close the connection.
private void button5_Click(object sender, EventArgs e)
{
    string query = "SELECT Count(*) FROM tableinfo";
    int Count = -1;

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

        //ExecuteScalar will return one value
        Count = int.Parse(cmd.ExecuteScalar() + "") ;

        //Close Connection
        this.CloseConnection();

        MessageBox.Show(Count.ToString()); // ต้องเพิ่ม using System.Windows;
    }
}
Workshop

- ให้จำลองร้านค้าเล็ก ๆ ที่มีของขาย อย่างน้อย 3 อย่างโดยที่ของแต่ละอย่างเก็บจำนวนและราคาต่อหน่วยที่สามารถเพิ่มลบแก้ไขได้