XAMPP is a free and open source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

If you use xampp, you probably know how easy is to create and maintain databases with the integrated module of phpmyadmin. You may find easily to work with PHP, but, if you have .NET knowledge you can start working with it too.

MySQL offers a connector a easy connector that will allow you to execute queries to phpmyadmin from your winform.

In this article we'll learn how to access a database created in phpmyadmin using winforms in C# easily.

Requirements

- Visual Studio (any version).
- MySQL .NET Connector Extension.
- XAMPP Distribution (we'll assume that you know how to use mysql and xampp).

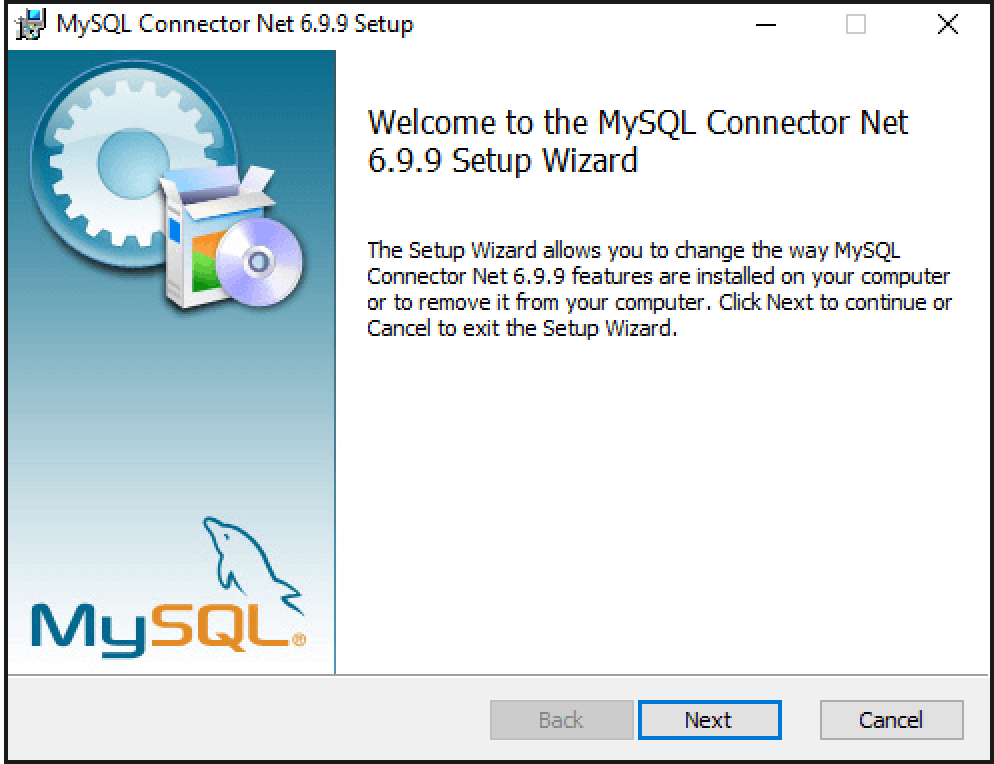
Implementation

This task is more easy than you think, to achieve a succesfully implementation follow these steps :

- Add the reference to the MySQL connector to your winform project.
- Create your database (ignore if you already have any) in MySQL with PHPMyAdmin.
- Start to execute queries.

Add the reference to the MySQL connector to the project

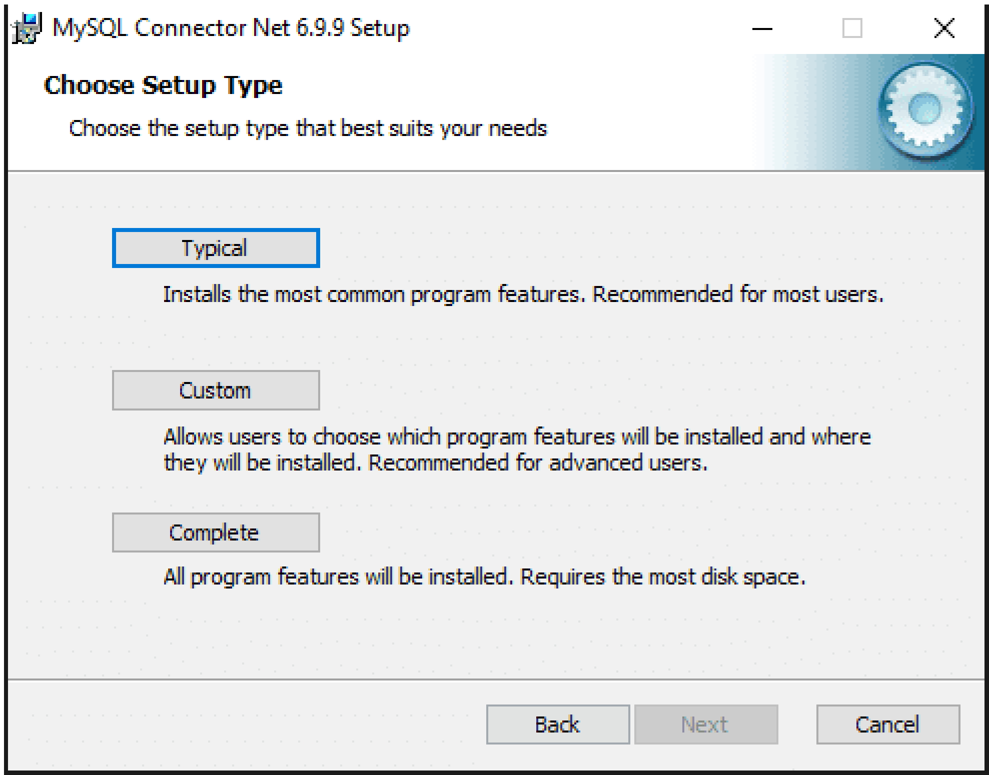
To get started, you need obligatory the .NET MySQL extension installed in your system as we need to add the reference in our project later. You can choose one of the latest version in the official website of MySQL.



Welcome to the MySQL Connector Net 6.9.9 Setup Wizard

The Setup Wizard allows you to change the way MySQL Connector Net 6.9.9 features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

You can choose wether a complete installation or typical.


MySQL Connector Net 6.9.9 Setup

Choose Setup Type

Choose the setup type that best suits your needs

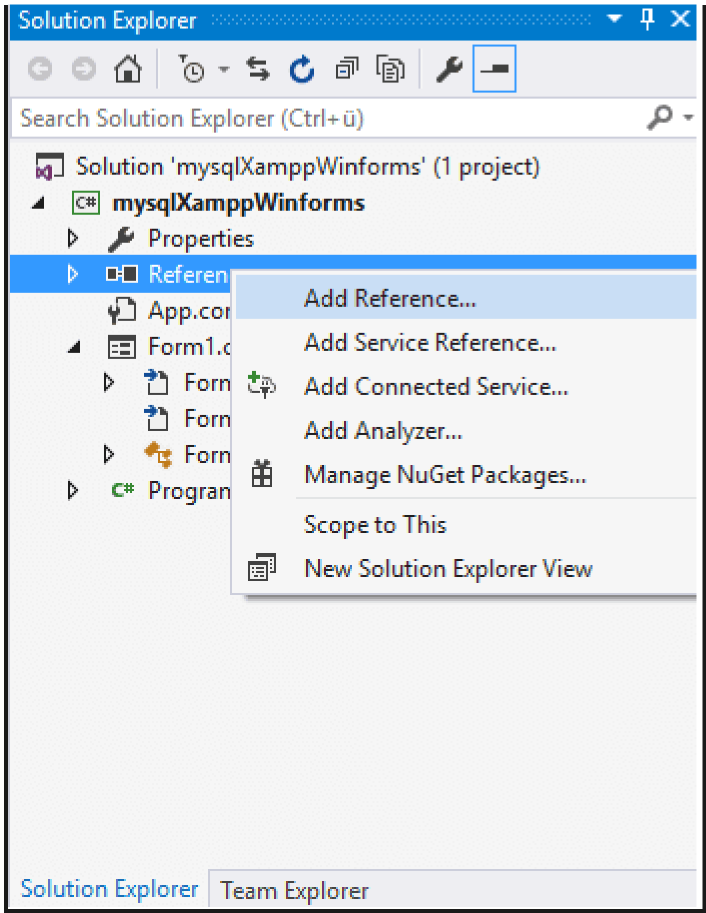
Installs the most common program features. Recommended for most users.

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

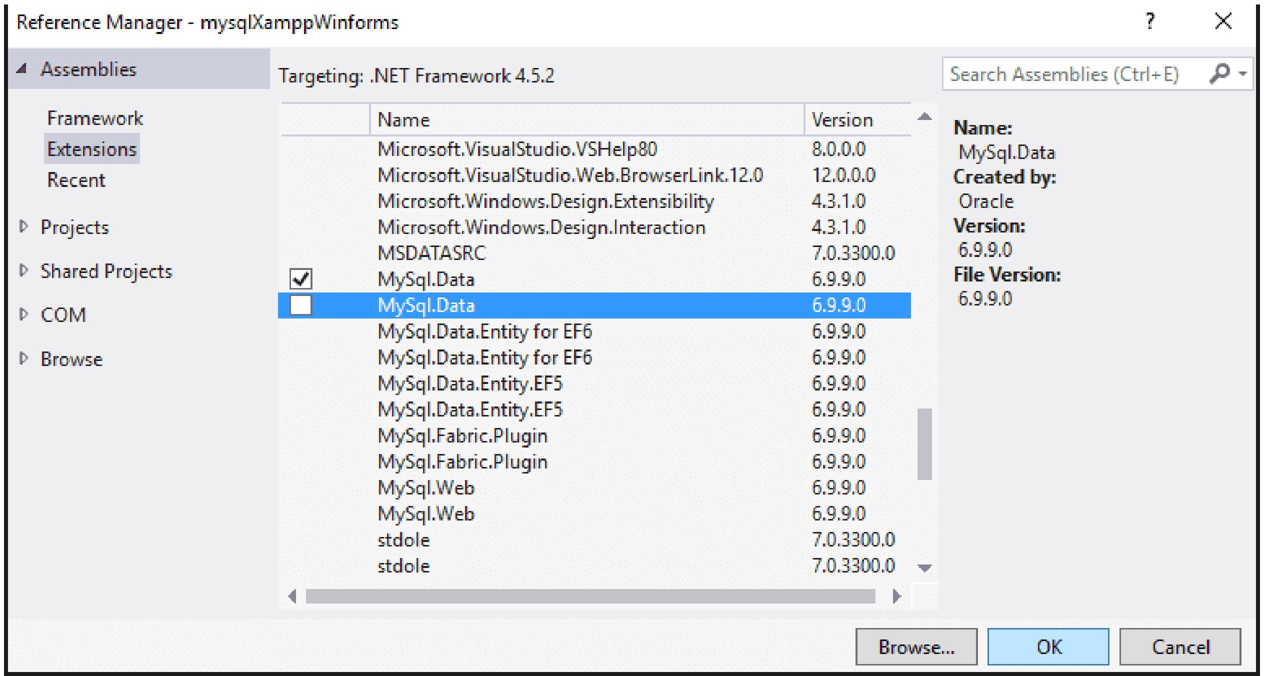
All program features will be installed. Requires the most disk space.

After the installation, we are going to proceed to create an empty Winforms project in Visual Studio as you usually do.

Now we need to add the reference to the mysql connector in our project. Locate the solution explorer in the right top corner of Visual Studio when your project is open, use right click on **References** and then select **Add Reference** from the context menu.



In the shown menu, navigate to Extensions and select the checkbox from the list the MySql.Data (`MySql.Data.dll`) and then click on OK.


Reference Manager - mysqlXamppWinforms

Targeting: .NET Framework 4.5.2

	Name	Version
	Microsoft.VisualStudio.VSHelp80	8.0.0.0
	Microsoft.VisualStudio.Web.BrowserLink.12.0	12.0.0.0
	Microsoft.Windows.Design.Extensibility	4.3.1.0
	Microsoft.Windows.Design.Interaction	4.3.1.0
	MSDATASRC	7.0.3300.0
✓	MySql.Data	6.9.9.0
☐	MySql.Data	6.9.9.0
	MySql.Data.Entity for EF6	6.9.9.0
	MySql.Data.Entity for EF6	6.9.9.0
	MySql.Data.Entity.EF5	6.9.9.0
	MySql.Data.Entity.EF5	6.9.9.0
	MySql.Fabric.Plugin	6.9.9.0
	MySql.Fabric.Plugin	6.9.9.0
	MySql.Web	6.9.9.0
	MySql.Web	6.9.9.0
	stdole	7.0.3300.0
	stdole	7.0.3300.0

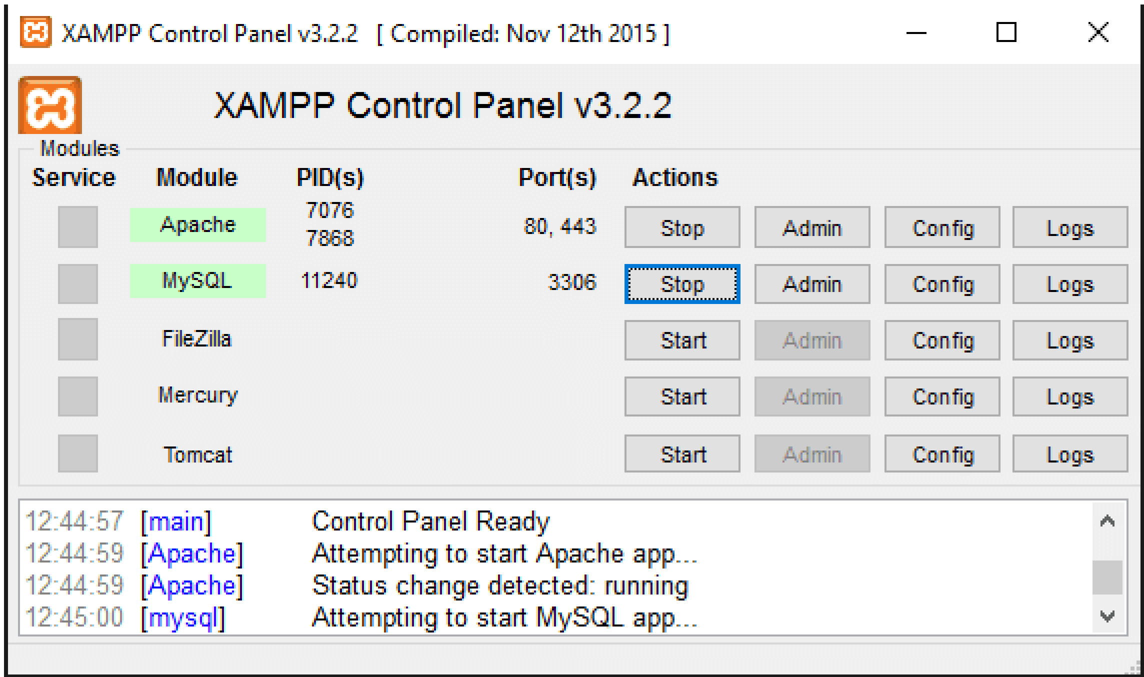
Name:
MySql.Data
Created by:
Oracle
Version:
6.9.9.0
File Version:
6.9.9.0

Now we'll be able to connect to execute queries to MySQL with C#.

Creating a test database in phpmyadmin (localhost)

As mentioned before, we assume that you already have Xampp installed on your system and you know how to use it.

First, do not forget to enable the apache and mysql services in the xampp panel (which is recommended in Administrator mode).

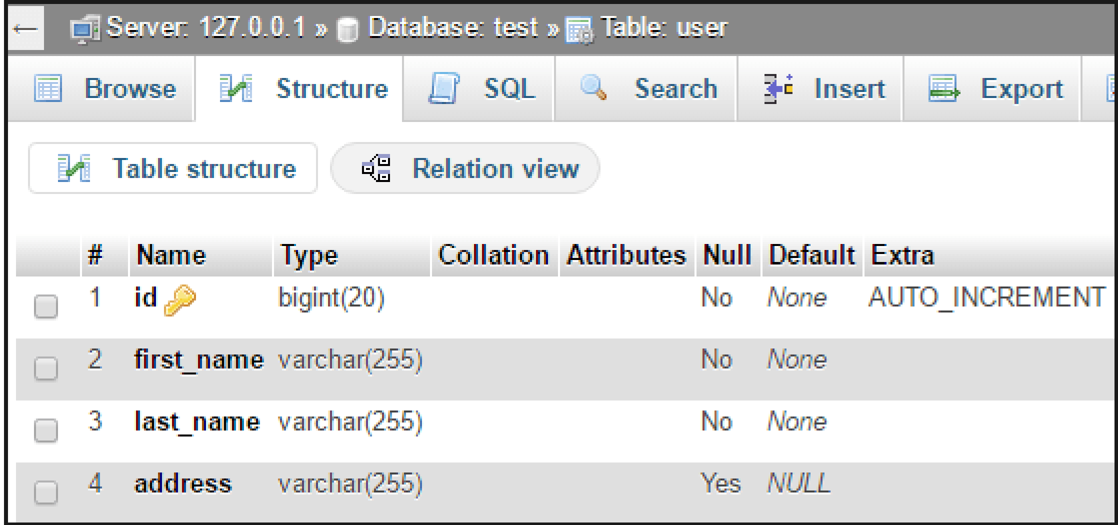

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules

Service	Module	PID(s)	Port(s)	Actions			
	Apache	7076 7868	80, 443	Stop	Admin	Config	Logs
	MySQL	11240	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
12:44:57  [main]     Control Panel Ready
12:44:59  [Apache]   Attempting to start Apache app...
12:44:59  [Apache]   Status change detected: running
12:45:00  [mysql]    Attempting to start MySQL app...
```

Now navigate in your browser to http://localhost/phpmyadmin and go to the databases area.

Create a database (in this example our database will be **test**) and create a table named `user`.



	#	Name	Type	Collation	Attributes	Null	Default	Extra
	1	**id**	bigint(20)			No	*None*	AUTO_INCREMENT
	2	**first_name**	varchar(255)			No	*None*	
	3	**last_name**	varchar(255)			No	*None*	
	4	**address**	varchar(255)			Yes	*NULL*	

Note

Remember to enable the autoincrementable option to the id field, otherwise you'll need to add an id everytime you insert a row.
Now that our database "test" contains at least one table "user", we can start executing queries.

Using C# to connect and execute queries

Now comes the fun part ! we'll write some code to interact with the MySQL database. Primary, don't forget to add the using statement of the reference in your class :

```
using MySql.Data.MySqlClient;
```

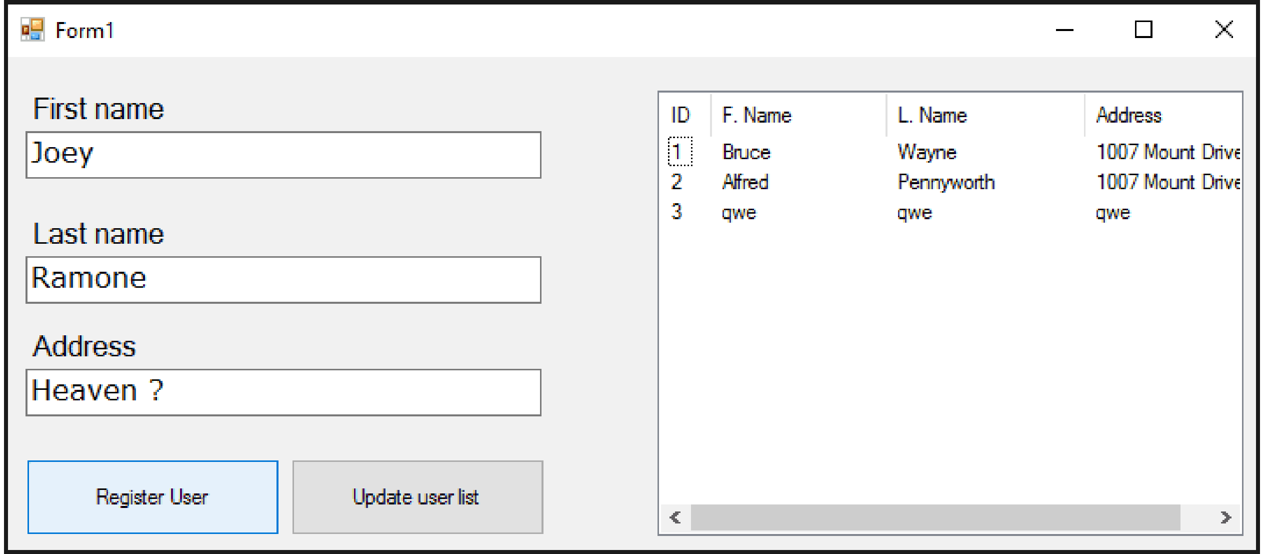
Copy snippet
You can understand how works the connection and how to execute a query with the following snippet :

```
// Change the username, password and database according to your needs
// You can ignore the database option if you want to access all of them.
// 127.0.0.1 stands for localhost and the default port to connect.
string connectionString =
"datasource=127.0.0.1;port=3306;username=root;password=;database=test;";
// Your query,
string query = "SELECT * FROM user";

// Prepare the connection
MySqlConnection databaseConnection = new MySqlConnection(connectionString);
MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
commandDatabase.CommandTimeout = 60;
MySqlDataReader reader;

// Let's do it !
try
{
    // Open the database
    databaseConnection.Open();
    // Execute the query
    reader = commandDatabase.ExecuteReader();

    // All succesfully executed, now do something

    // IMPORTANT :
    // If your query returns result, use the following processor :

    if (reader.HasRows)
    {
        while (reader.Read())
        {
            // As our database, the array will contain : ID 0, FIRST_NAME
1,LAST_NAME 2, ADDRESS 3
            // Do something with every received database ROW
            string[] row = { reader.GetString(0), reader.GetString(1),
reader.GetString(2), reader.GetString(3) };
        }
    }
    else
    {
        Console.WriteLine("No rows found.");
    }

    // Finally close the connection
    databaseConnection.Close();
}
catch (Exception ex)
{
    // Show any error message.
    MessageBox.Show(ex.Message);
}
```
Copy snippet

And that's it ! Basically, you just need to change the query and start testing. You can read more about the connection string and all the available properties here.

Basic examples of queries

In these examples we are going to execute the most basic tasks to execute (CRUD):



First name: Joey

Last name: Ramone

Address: Heaven ?

ID	F. Name	L. Name	Address
1	Bruce	Wayne	1007 Mount Drive
2	Alfred	Pennyworth	1007 Mount Drive
3	qwe	qwe	qwe

Note that we'll use a simple listView component (with 4 columns : id,first name, last name and address), 3 textBox and 2 Buttons.

Create

In the following snippet, we'll create a register in the test database :

```
private void SaveUser()
{
    string connectionString =
"datasource=127.0.0.1;port=3306;username=root;password=;database=test;";
```

```
    string query = "INSERT INTO user(`id`, `first_name`, `last_name`,
`address`) VALUES (NULL, '"+textBox1.Text+ "', '" + textBox2.Text + "', '" +
textBox3.Text + "')";

    // Which could be translated manually to :

    // INSERT INTO user(`id`, `first_name`, `last_name`, `address`) VALUES
(NULL, 'Bruce', 'Wayne', 'Wayne Manor')


    MySqlConnection databaseConnection = new
MySqlConnection(connectionString);

    MySqlCommand commandDatabase = new MySqlCommand(query,
databaseConnection);

    commandDatabase.CommandTimeout = 60;


    try

    {

        databaseConnection.Open();

        MySqlDataReader myReader = commandDatabase.ExecuteReader();


        MessageBox.Show("User succesfully registered");


        databaseConnection.Close();

    }

    catch (Exception ex)

    {

        // Show any error message.

        MessageBox.Show(ex.Message);

    }

}
```

Copy snippet
Show

In the following snippet we'll list all the users in the test database in a listview (if available or show in the console) :

```
private void listUsers()

{
    string connectionString =
"datasource=127.0.0.1;port=3306;username=root;password=;database=test;";
    // Select all
    string query = "SELECT * FROM user";

    MySqlConnection databaseConnection = new
MySqlConnection(connectionString);
    MySqlCommand commandDatabase = new MySqlCommand(query,
databaseConnection);
    commandDatabase.CommandTimeout = 60;
    MySqlDataReader reader;

    try
    {
        databaseConnection.Open();
        reader = commandDatabase.ExecuteReader();
        // Success, now list

        // If there are available rows
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                                     ID                                First
name                  Last Name                  Address
                Console.WriteLine(reader.GetString(0) + " - " +
reader.GetString(1) + " - " + reader.GetString(2) + " - " +
reader.GetString(3));
                // Example to save in the listView1 :
                //string[] row = { reader.GetString(0), reader.GetString(1),
reader.GetString(2), reader.GetString(3) };
                //var listViewItem = new ListViewItem(row);
                //listView1.Items.Add(listViewItem);
            }
        }
        else
        {
            Console.WriteLine("No rows found.");
        }

        databaseConnection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Copy snippet

Update

Update the fields of a row with id :

```
private void updateUser()
{
    string connectionString =
"datasource=127.0.0.1;port=3306;username=root;password=;database=test;";
    // Update the properties of the row with ID 1
    string query = "UPDATE `user` SET
`first_name`='Willy',`last_name`='Wonka',`address`='Chocolate factory' WHERE
id = 1";

    MySqlConnection databaseConnection = new
MySqlConnection(connectionString);
    MySqlCommand commandDatabase = new MySqlCommand(query,
databaseConnection);
    commandDatabase.CommandTimeout = 60;
    MySqlDataReader reader;

    try
    {
        databaseConnection.Open();
        reader = commandDatabase.ExecuteReader();
        // Succesfully updated

        databaseConnection.Close();
    }
    catch (Exception ex)
    {
        // Ops, maybe the id doesn't exists ?
        MessageBox.Show(ex.Message);
    }
}
```

Copy snippet

Delete

Delete a row with ID (x) :

```
private void deleteUser()
{
    string connectionString =
"datasource=127.0.0.1;port=3306;username=root;password=;database=test;";
    // Delete the item with ID 1
    string query = "DELETE FROM `user` WHERE id = 1";

    MySqlConnection databaseConnection = new
MySqlConnection(connectionString);
    MySqlCommand commandDatabase = new MySqlCommand(query,
databaseConnection);
    commandDatabase.CommandTimeout = 60;
    MySqlDataReader reader;

    try
    {
        databaseConnection.Open();
        reader = commandDatabase.ExecuteReader();
        // Succesfully deleted

        databaseConnection.Close();
    }
    catch (Exception ex)
    {
        // Ops, maybe the id doesn't exists ?
        MessageBox.Show(ex.Message);
    }
}
```