Cairo University
Faculty of Engineering
Computer Engineering Dept.

LAB #5
Accessing DB from C# Application - Part2
More GUI controls

Objectives

After this lab, the student should be able to:

- Create and fill database using SQL statements
- Connect C# application to a database
- Execute SQL queries on the database by calling direct SQL statements in C# application.

Database Creation using SQL statements

Create and Fill the Database

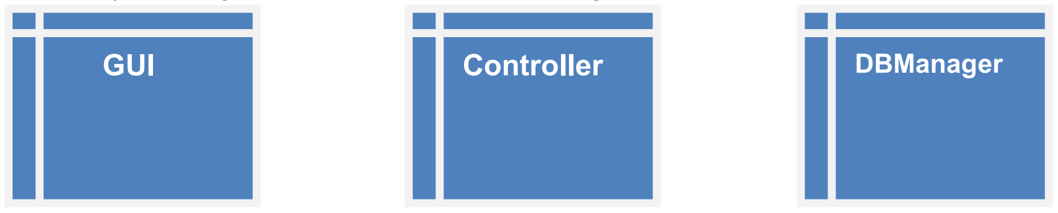
Script **CompanyDB.sql** does the following:

1- Creates a new DB called CompanyLab2 ***Create Database*** statement
2- Creates DB tables: Employee, Department, Dept_Locations, Project, and Works_On tables ***Create Table*** statement
3- Creates primary and foreign keys for each table ***Create/Alter Table*** statement
Why do we use Alter table to create Dno foreign key in table Employee?

4- Fills database tables with some sample data ***Insert Into*** statement
5- Completes tables filling ***Update*** statement
What are those nulls inserted in table Employee at first?

Windows Application Example

The application you are given consists of the following three modules:

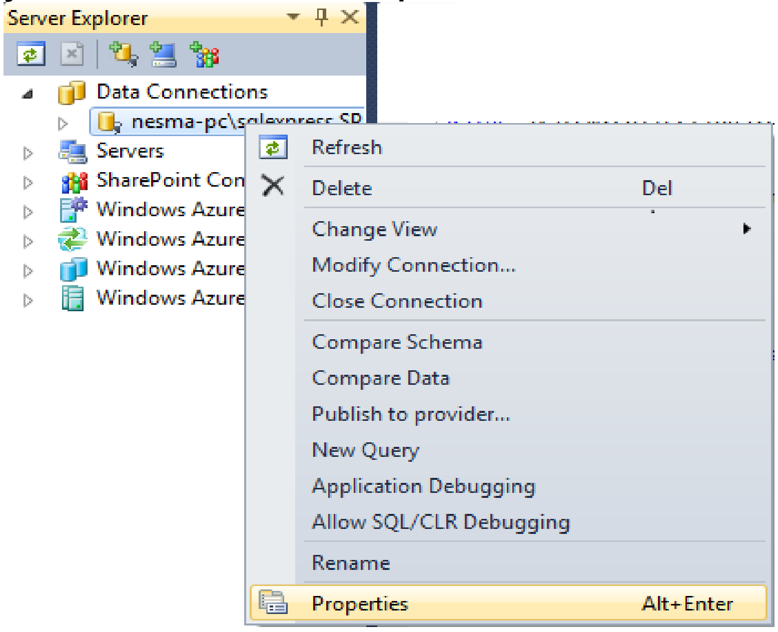


❑ User deals with the GUI to specify which functionality he wants to use.
❑ Controller class is responsible for controlling the flow of the program, therefore, the GUI deals with the controller class in order to specify what action to be done when the user for example clicks on a certain button.
❑ DBManager class is responsible for dealing with the database. Controller deals with this class when it finds any action involving DB.
❑ <u>**Note**</u>: Controller and DBManager and user-defined classes not library classes.

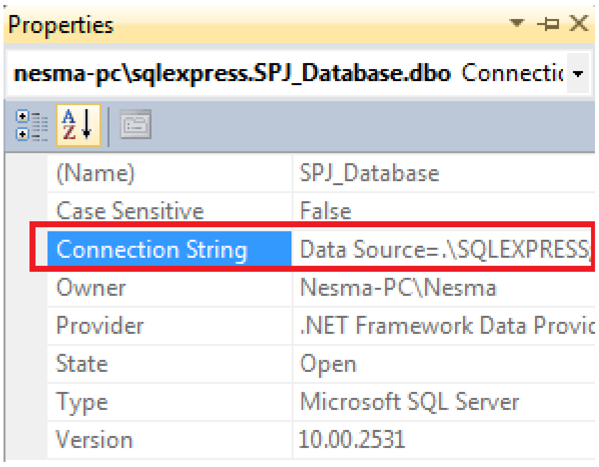
Connecting to the database (revision)

Steps

1. **In Server Explorer/Database Explorer, right click on the database connection you created and choose Properties**



2. **In the Properties list you will find a property called "Connection String", copy its value to use it in the c# code**


Properties

nesma-pc\sqlexpress.SPJ_Database.dbo Connectic

(Name)	SPJ_Database
Case Sensitive	False
Connection String	Data Source=.\SQLEXPRESS
Owner	Nesma-PC\Nesma
Provider	.NET Framework Data Provic
State	Open
Type	Microsoft SQL Server
Version	10.00.2531

3. **Use the following code to connect to the database**

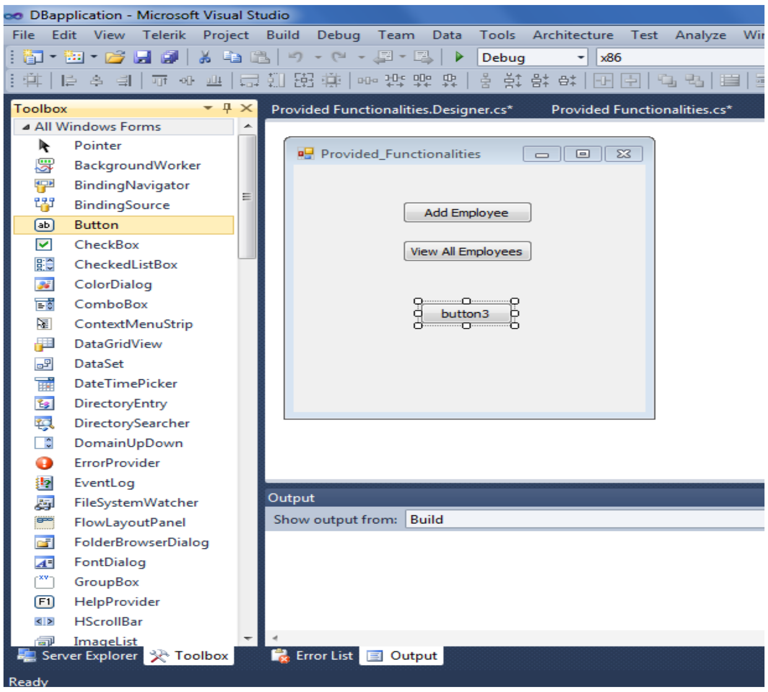
```
SqlConnection myConnection;
myConnection = new SqlConnection(DB_Connection_String);
try
{
     myConnection.Open();
     Console.WriteLine("The DB connection is opened successfully");
}
catch (Exception e)
{
     Console.WriteLine("The DB connection is failed");
}
```

Executing complex queries through the application

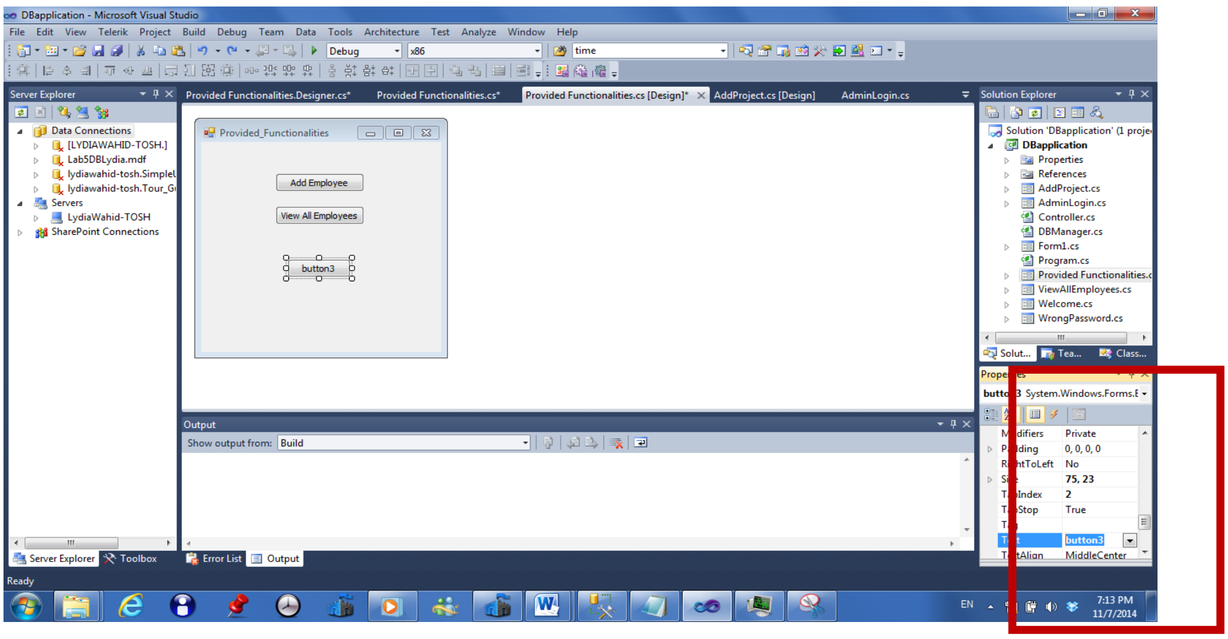
Example: If we want to add the following functionality to our application:
"Select project name and department name for all projects that are in departments located in a certain location" (location will be specified by the user):

Steps:

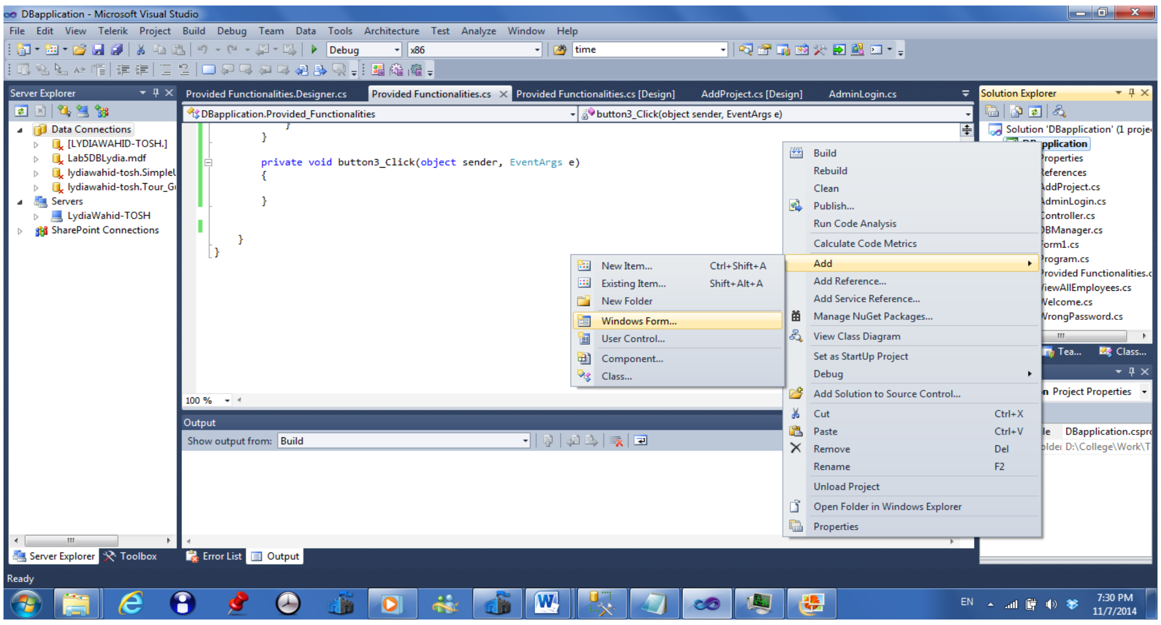
1. Add a button for this functionality in the "Provided_Functionalities" form



2. Rename the button by selecting it and then changing the "Text" in the properties window that is found at the bottom right. Notice that this changes only the text that is displayed on the button, not its name.



3. Add new windows form and name it "RetrieveProjects".

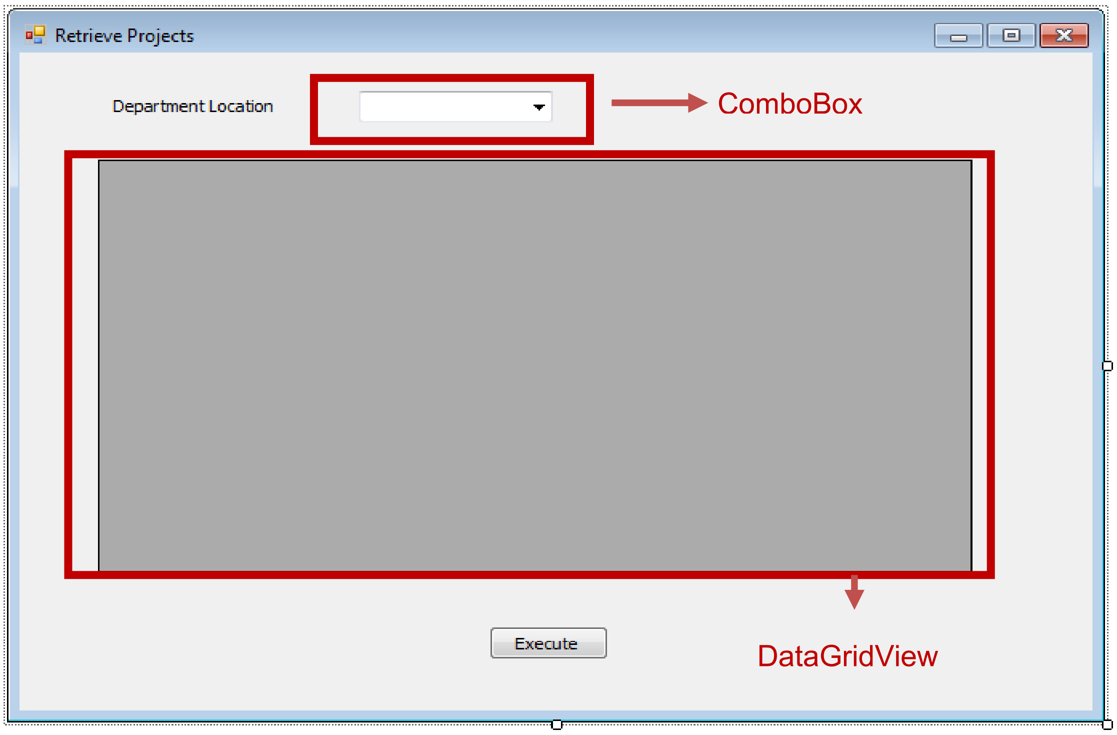


```
    }

private void button3_Click(object sender, EventArgs e)
{

}
}
}
```

4. Double click the button to write the code that will be executed when the button click event happens. Write the following code in order to show the new window added.

```
private void button3_Click(object sender, EventArgs e)
{
    RetrieveProjects RP = new RetrieveProjects();
    RP.Show();

}
```

5. In the “RetrieveProjects” windows form add a “combobox” and “dataGridView” from the toolbox. ComboBox will be used to select from it the department location that we want; it will get its data from the “Dlocation” colomn found in “Dept_locations” table. The “dataGridView” will be used to view the required information that will be retrieved from the database.



6. In order to let the comboBox get its data from the the “Dlocation” column found in “Dept_locations” table, we will perform the following steps:

❑ Define the following member function in the controller class

```
public DataTable SelectDepLoc()
{
    string query = "SELECT DISTINCT Dlocation FROM Dept_Locations;";
    return dbMan.ExecuteReader(query);
}
```

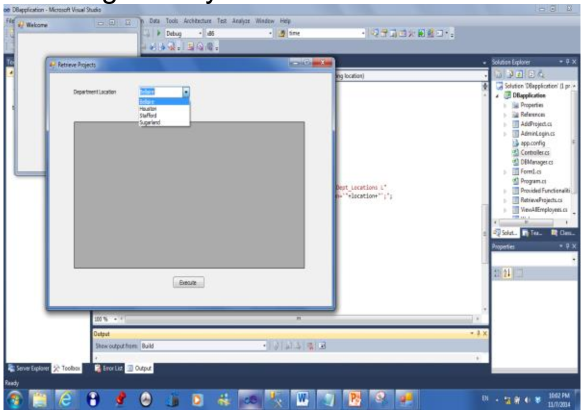
Notice the use of the word “DISTINCT” in the query to remove duplicate tuples.

❑ Then in the “RetrieveProjects” form code, add a “Controller” object ad specify the datasource of the comboBox as shown below in the form constrcutor:

```
Controller controllerObj;
public RetrieveProjects()
{
    InitializeComponent();
    controllerObj = new Controller();
    DataTable dt = controllerObj.SelectDepLoc();
    comboBox1.DataSource = dt;
    comboBox1.DisplayMember = "Dlocation";

}
```

Where the “comboBox1.DisplayMember” is the column in the database that the comboBox will get its data from.

❑ When you run the project and go to the “RetrieveProjects” form, you will see the following when you click the arrow of the combo box.



7. In order to execute the query and see the results, do the following:

❑ Add a dataGridView to display the resulting table

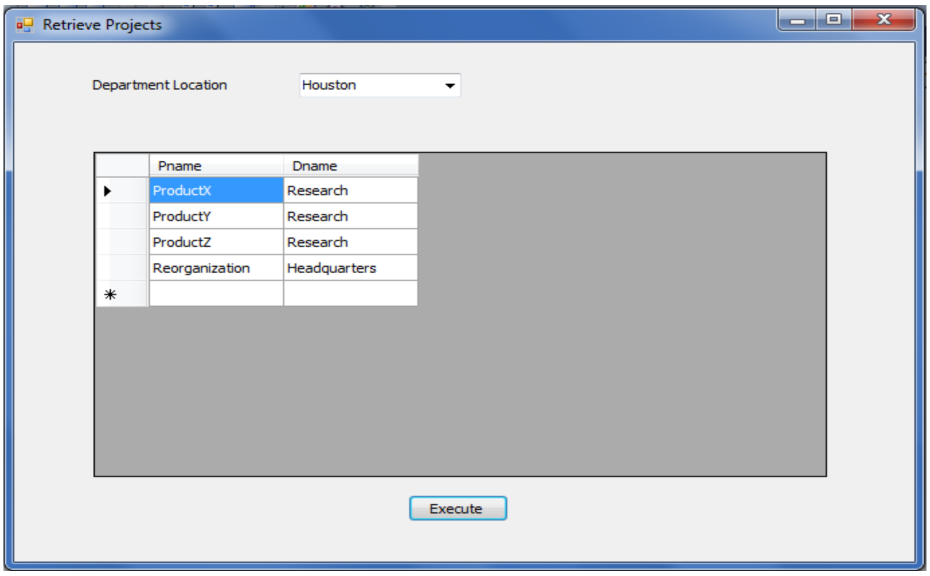
❑ Add the following function in the “Controller” class in order to form the query that we want to execute:

```
public DataTable SelectProject(string location)
{
    string query = "SELECT Pname,Dname FROM Department D, Project P, Dept_Locations L"
     +" where P.Dnum=D.Dnumber and L.Dnumber=D.Dnumber and L.Dlocation='"+location+"';";

    return dbMan.ExecuteReader(query);
}
```

❑ Then in the “RetrieveProjects” form, in the event handler of “Execute” button click, will write the following code:

```
private void Execute_Click(object sender, EventArgs e)
{
    DataTable dt = controllerObj.SelectProject(comboBox1.Text);
    dataGridView1.DataSource = dt;
    dataGridView1.Refresh();
}
```

❑ When you run the project and choose “Houston” from the comboBox and click “Execute” button, the dataGridView will contain the result of the query as shown:



	Pname	Dname
▶	ProductX	Research
	ProductY	Research
	ProductZ	Research
	Reorganization	Headquarters
*		

Add Project Form

Form Constructor:

```
public AddProject()
{
    InitializeComponent();
    controllerObj = new Controller();

    // WRONG: DataTable dt = controllerObj.SelectDepNum(); XXXXXXXXXXX

    DataTable dt = controllerObj.SelectDepInfo();   // Here is the correct way
    // Note: make sure of the query to select at least the (Dname and Dnumber)
    // not only the Dname, to be able to use them in (.DisplayMember) and (.ValueMember) of combobox

    comboBox1.DataSource = dt;
    comboBox1.DisplayMember = "Dname";  // Here
    comboBox1.ValueMember = "Dnumber";  // Here
}
```

Event Handler:

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text=="" || textBox2.Text=="" || textBox3.Text=="")  // Validation part
    {
        MessageBox.Show("Please, insert all values");
    }
    else
    {
        // WRONG:
        // int r = controllerObj.InsertProject(textBox1.Text.ToString(), Convert.ToInt32(textBox2.Text),
        //                textBox3.Text.ToString(), Convert.ToInt32(comboBox1.Text)); XXXXXXXXXXXXX

        // Here: should use int.TryParse in a condition
        int r = controllerObj.InsertProject(textBox1.Text.ToString(), Convert.ToInt32(textBox2.Text),
                       textBox3.Text.ToString(), Convert.ToInt32(comboBox1.SelectedValue));
        // NOTE: we used comboBox1.SelectedValue to access the dnumber (not the displayed dname)
        // and we needed to convert it to int because SelectedValue returns type "object"

        if (r != 0)  // r contains the number of affected rows (r should be = 1 if inserted 1 row)
            MessageBox.Show("Project is inserted successfully");
        else
            MessageBox.Show("Project is NOT inserted. Failure.");
    }
}
```

Needed Controller Queries:

Note: For the combobox, you cannot only select department name or department number, you must at least select both to later set the name as the DisplayMember and the number as the ValueMember.

```
public DataTable SelectDepInfo()
{
    string query = "SELECT * FROM Department;";
    return dbMan.ExecuteReader(query);
}

public int InsertProject(string Pname, int pnumber, string Plocation, int Dnum)
{
    string query = "INSERT INTO Project (Pname, Pnumber, Plocation, Dnum)" +
                   "Values ('" + Pname + "'," + pnumber + ",'" + Plocation + "'," + Dnum + ");";
    return dbMan.ExecuteNonQuery(query);
}
```

Notes:

```
// TODO: it is good practice, to rename the controls with meaningful names before accessing them
// TODO: Don't forget to add any needed validations (the empty fields, int.TryParse, etc.)
// We omitted the TryParse here, but you MUST use it in any assessment
```