

Entity Framework – Code First Approach – Part 1

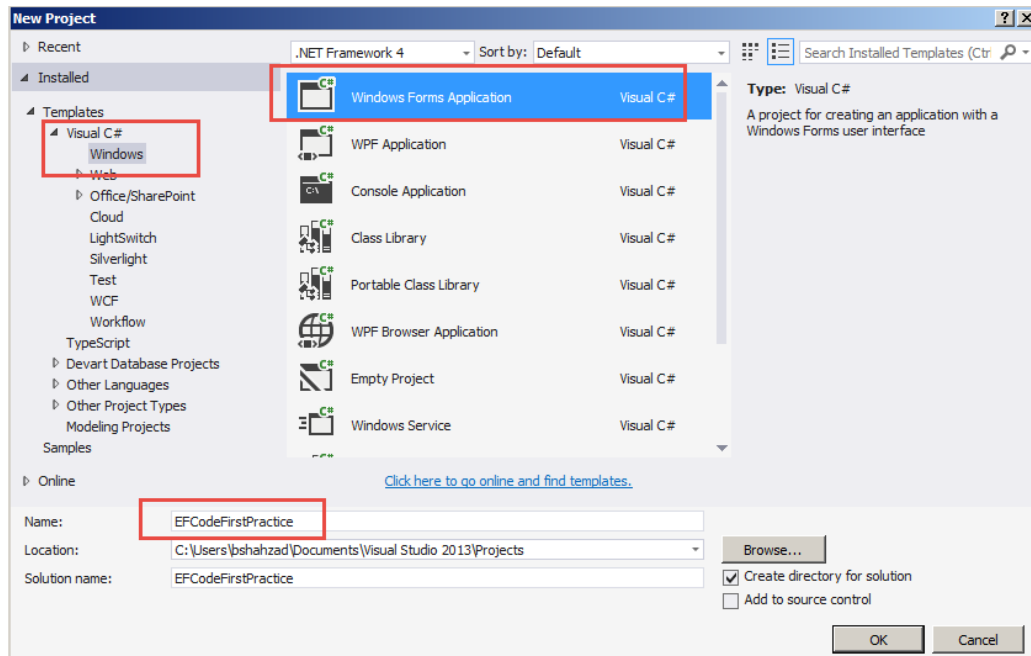
This guide will help you in creating a simple windows application which will interact with database (SQL Server) using Entity Framework.

This tutorial is prepared with Visual Studio 2013 + SQL Server.

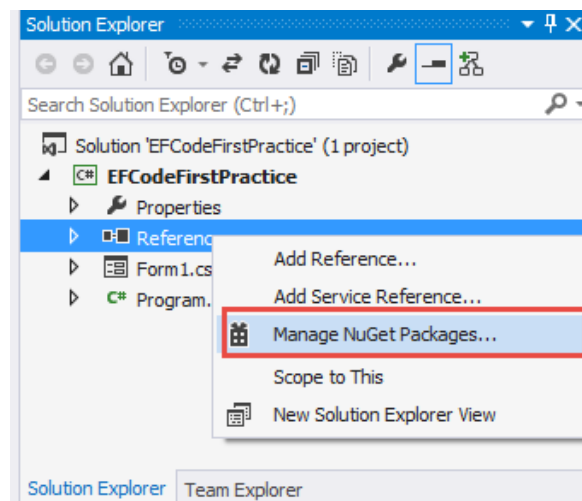
Version	Last updated	Comments	Modified By
V1.0	28-04-2016		Bilal Shahzad
V2.0	29-04-2016	Identity Column screen shot is added in Point 5.	Bilal Shahzad

Step by Step Walkthrough

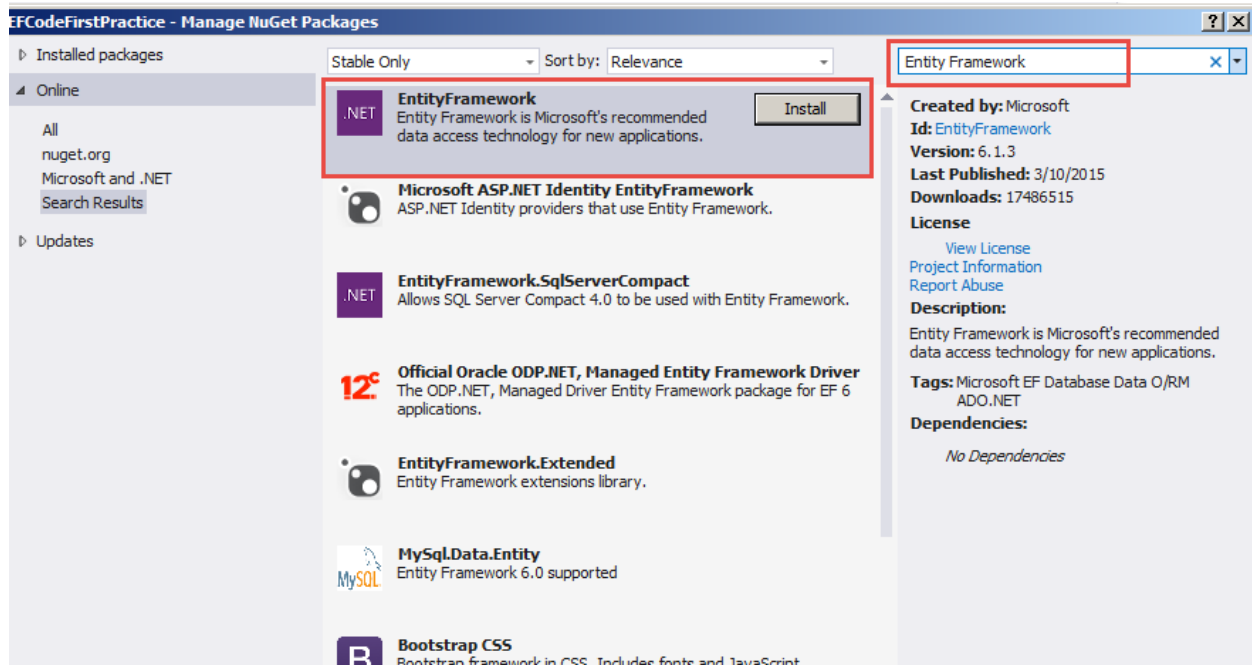
- 1- Create a new “Windows” Application.



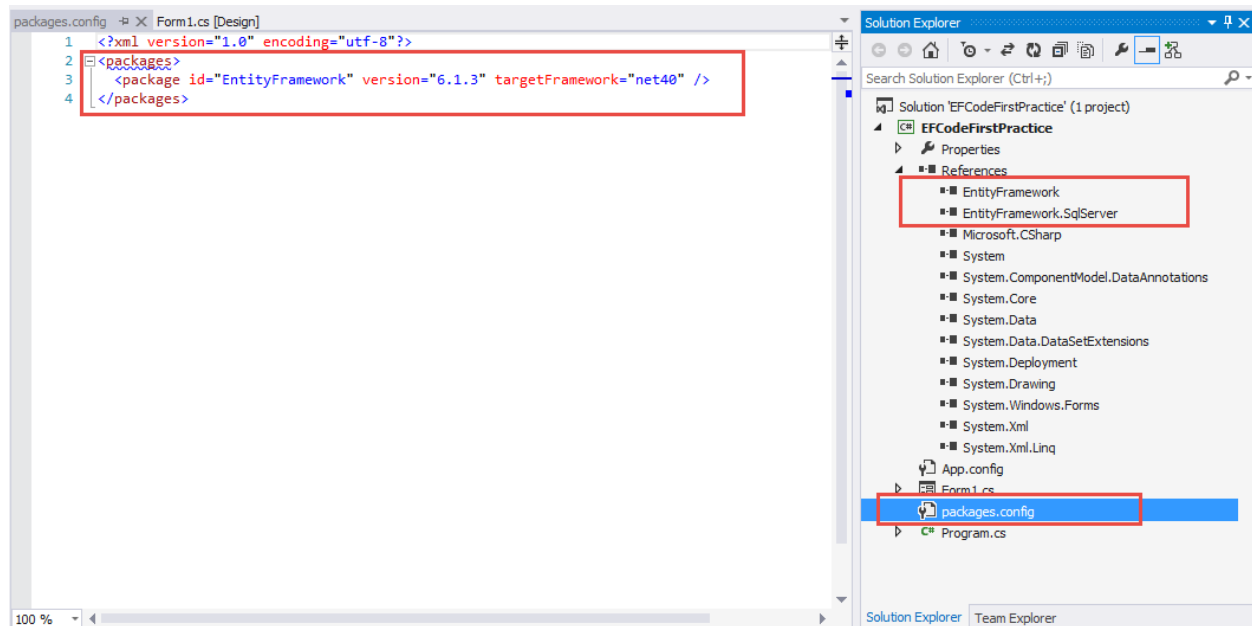
- 2- To add reference of “Entity Framework” package, right click on “References” and choose “Manage NuGet Packages”.



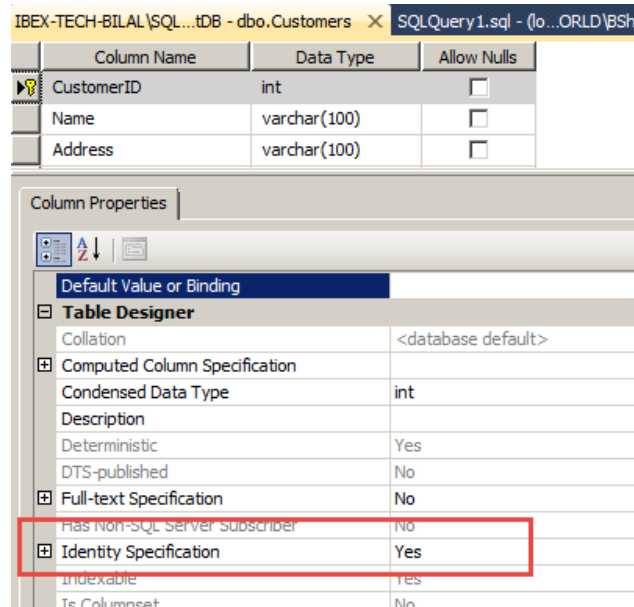
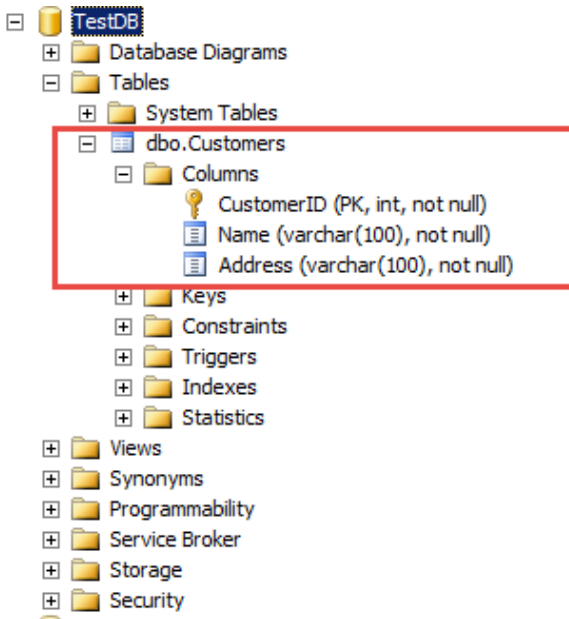
- 3- Search for “Entity Framework” and then install the highlighted package in following screenshot.



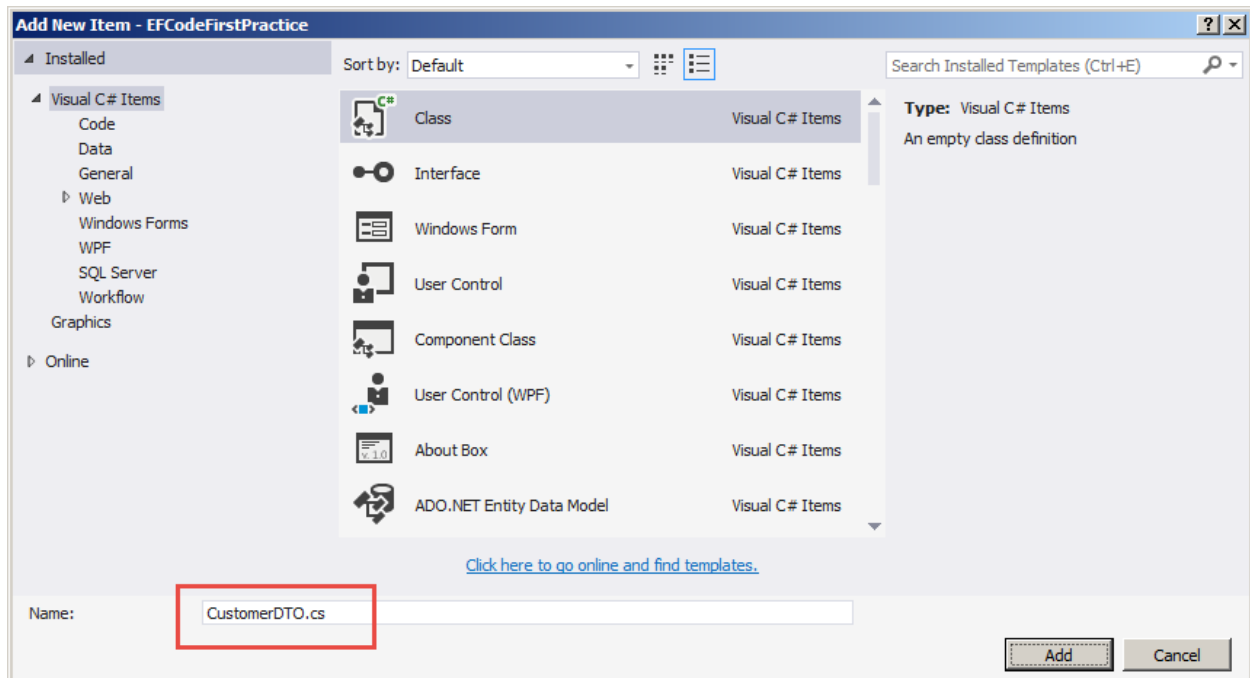
- 4- This will add highlighted DLLs, “packages.config” file and some configuration detail in “App.config”.



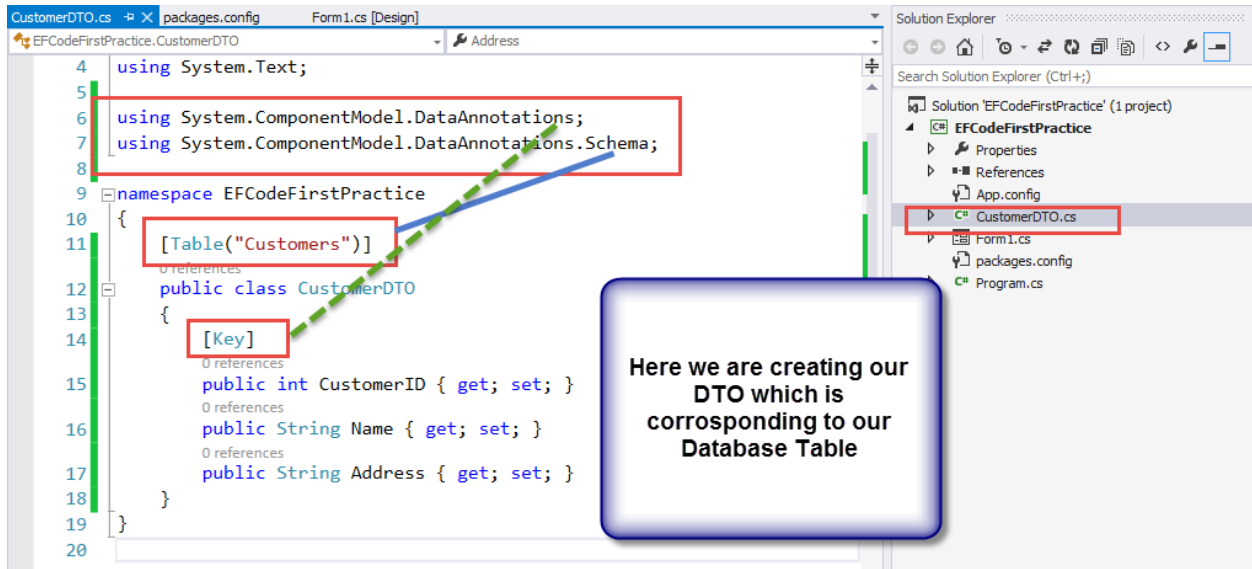
- 5- Now create a database (e.g. TestDB) in SQL Server and then create a table (e.g. dboCustomers) with schema highlighted in screenshot below. Please make sure that for “CustomerID” column “Identity” property is “Yes”.



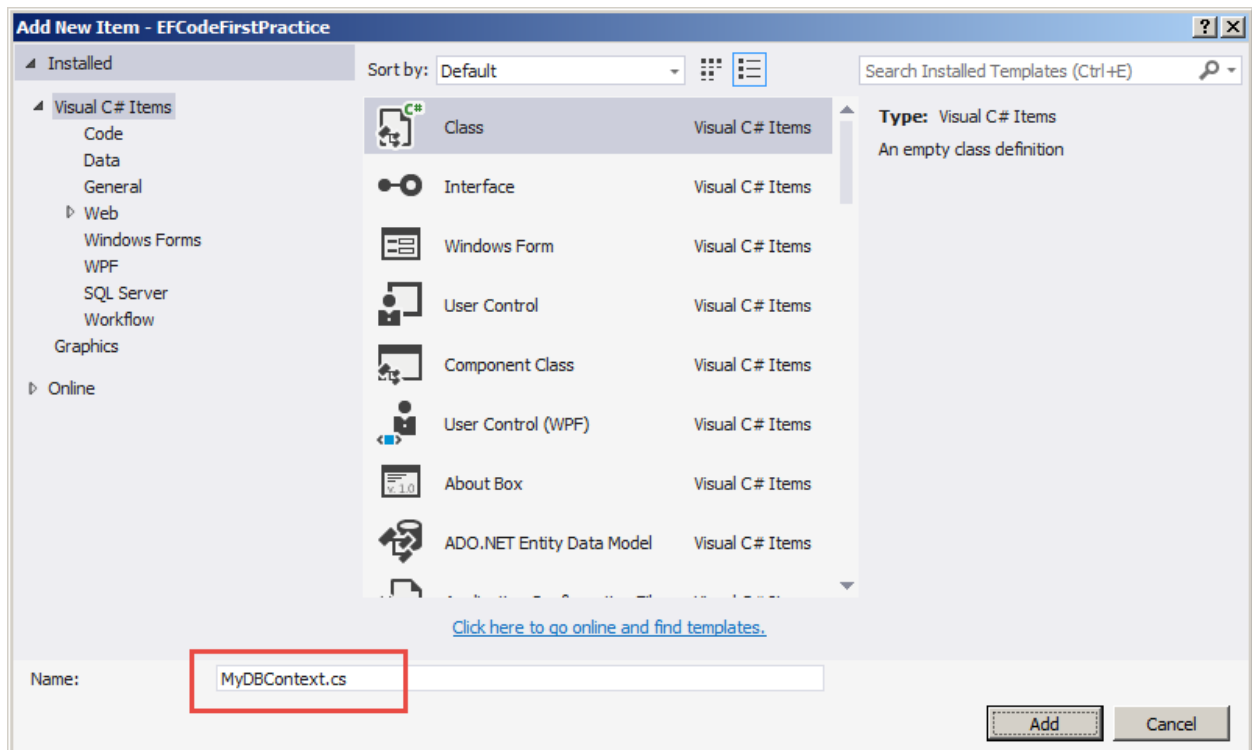
- 6- Now we need to add some classes to do interaction with database using Entity Framework. We'll add an entity (DTO) class to represent our table. We'll add a DbContext class to represent our database. And then we'll add connection string in our configuration file.
- 7- Add a new class file with name “CustomerDTO.cs”



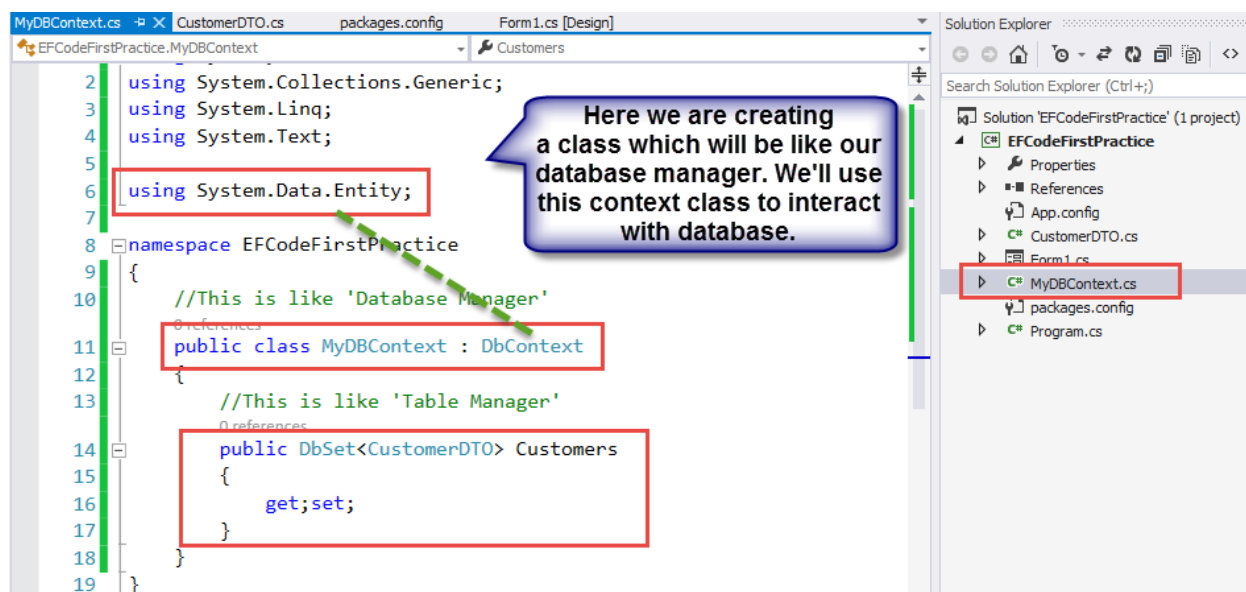
- 8- In newly created file, add references as highlighted below. Then add the code as shown in the screenshot. Here we are mapping our “CustomerDTO” class with table “Customers”. We are also annotating our property with “Key” attribute to declare that this is primary key column.



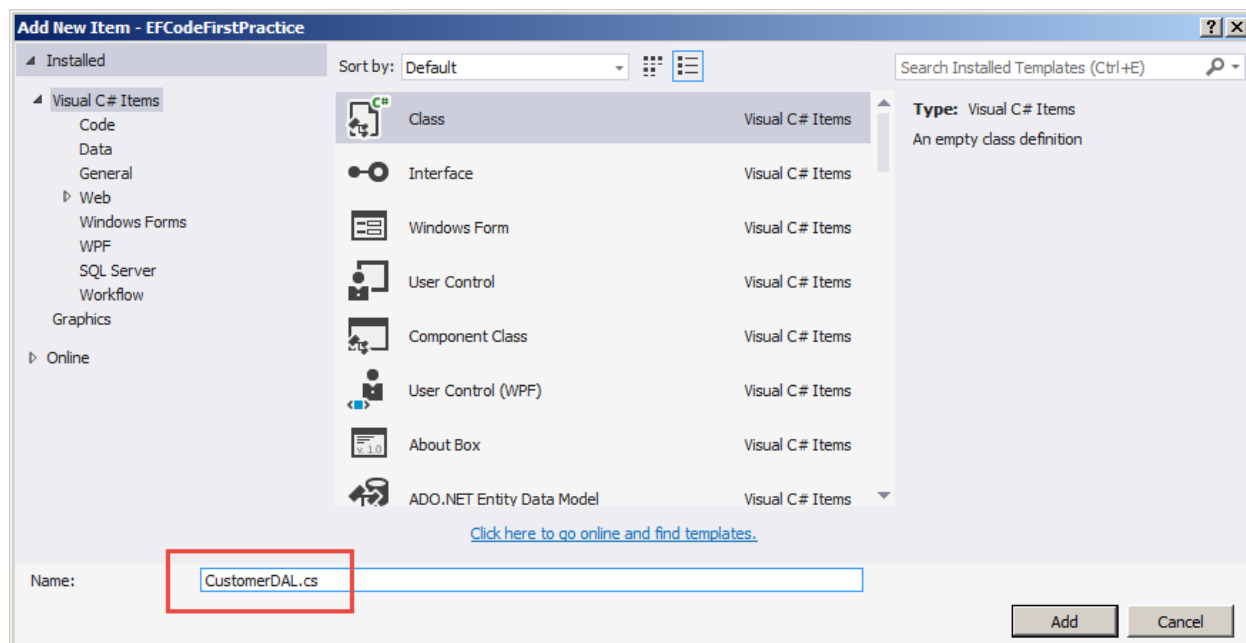
- 9- Now we need to create a class for our DataContext. Create a class file as shown below.



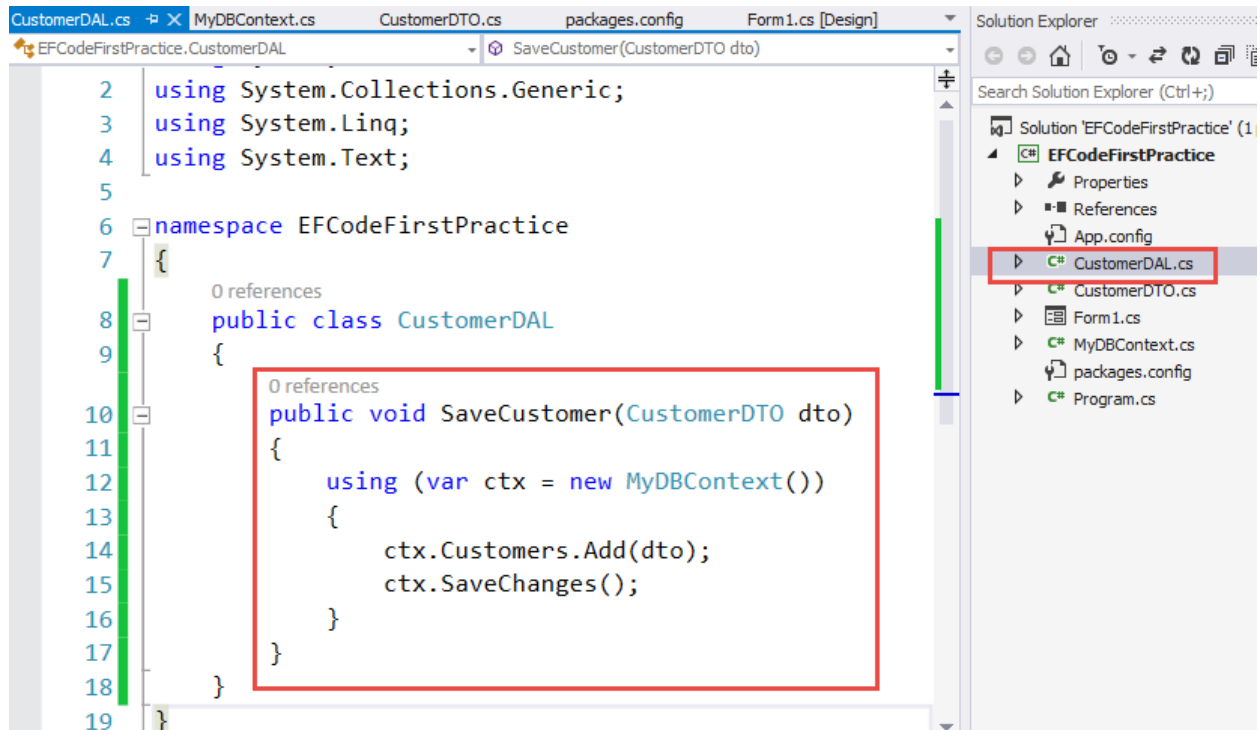
10- In newly created file, add code as shown below. Here we are creating a class of type (DbContext). This class will be responsible for managing all interactions with database. This class will contain “DbSet”. DbSet will correspond to our tables.



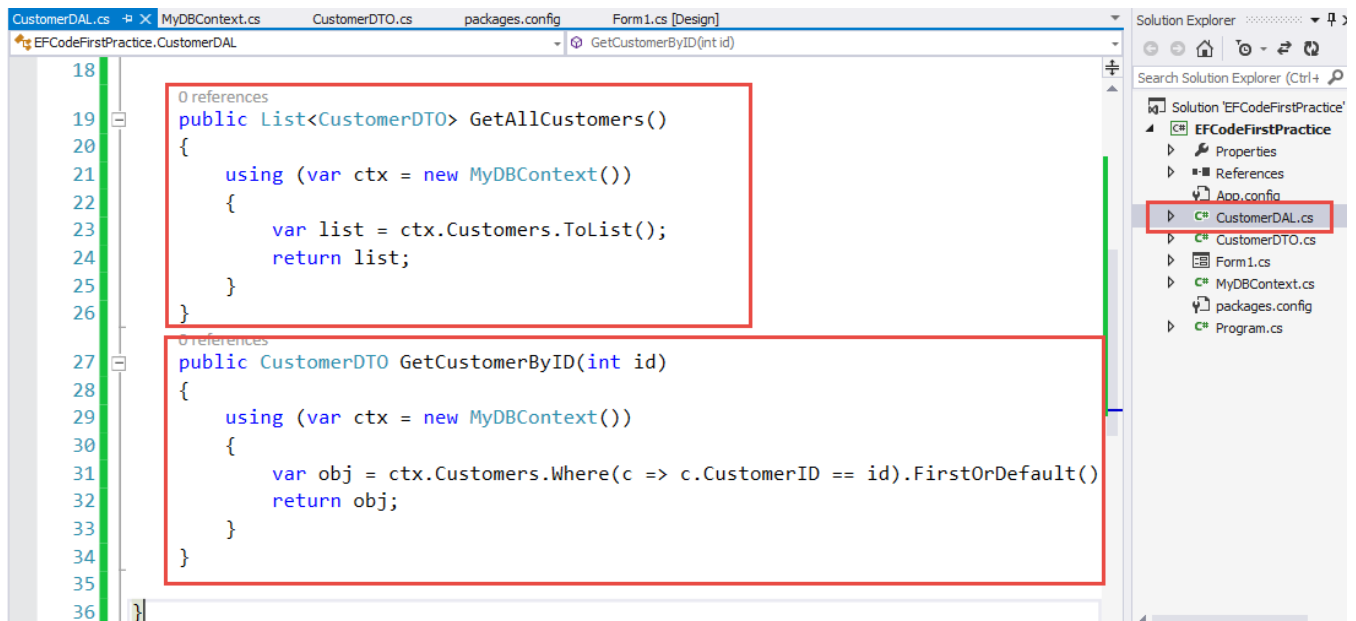
11- Now let's write a class to write our database access logic.



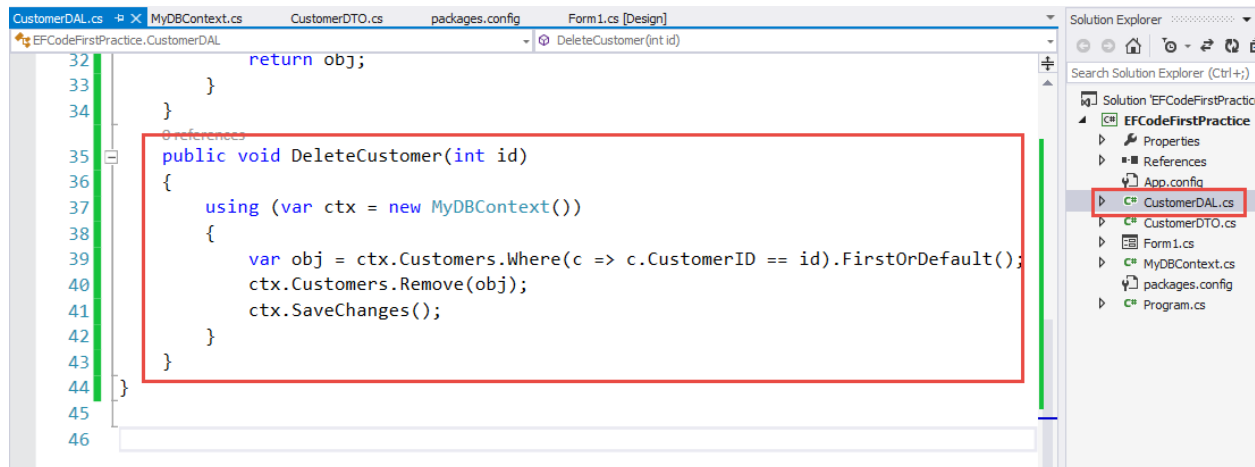
12- Here is the code to save an entity in database.



13- Here is the code to get all customers from database (as list of entities) and getting a single customer.



14- Here is code to delete a customer. To delete/update a record, we'll have to read it first as entity should be attached with context.

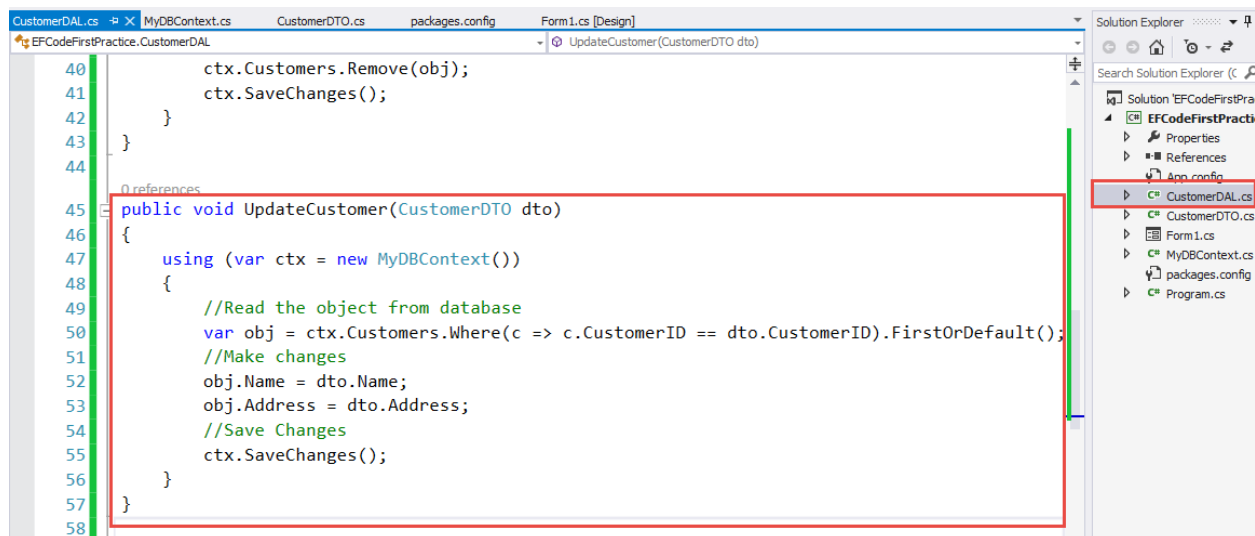


The screenshot shows the Visual Studio IDE with the `CustomerDAL.cs` file open. The `DeleteCustomer(int id)` method is highlighted with a red rectangle. The code inside the method is as follows:

```
32 return obj;
33 }
34 }
35 // references
36 public void DeleteCustomer(int id)
37 {
38     using (var ctx = new MyDBContext())
39     {
40         var obj = ctx.Customers.Where(c => c.CustomerID == id).FirstOrDefault();
41         ctx.Customers.Remove(obj);
42         ctx.SaveChanges();
43     }
44 }
45 }
46 }
```

The Solution Explorer on the right shows the project structure for `EFCodeFirstPractice`, with `CustomerDAL.cs` selected.

15- Here is code to update a customer. To delete/update a record, we'll have to read it first as entity should be attached with context.



The screenshot shows the Visual Studio IDE with the `CustomerDAL.cs` file open. The `UpdateCustomer(CustomerDTO dto)` method is highlighted with a red rectangle. The code inside the method is as follows:

```
40 ctx.Customers.Remove(obj);
41 ctx.SaveChanges();
42 }
43 }
44 // references
45 public void UpdateCustomer(CustomerDTO dto)
46 {
47     using (var ctx = new MyDBContext())
48     {
49         //Read the object from database
50         var obj = ctx.Customers.Where(c => c.CustomerID == dto.CustomerID).FirstOrDefault();
51         //Make changes
52         obj.Name = dto.Name;
53         obj.Address = dto.Address;
54         //Save Changes
55         ctx.SaveChanges();
56     }
57 }
58 }
```

The Solution Explorer on the right shows the project structure for `EFCodeFirstPractice`, with `CustomerDAL.cs` selected.

16- We need to define our connection string in our configuration file (app.config or web.config). With default settings, connection string name should match with our Data context class. So here we are providing following information (change this information according to your system)

Connection String Name: MyDBContext

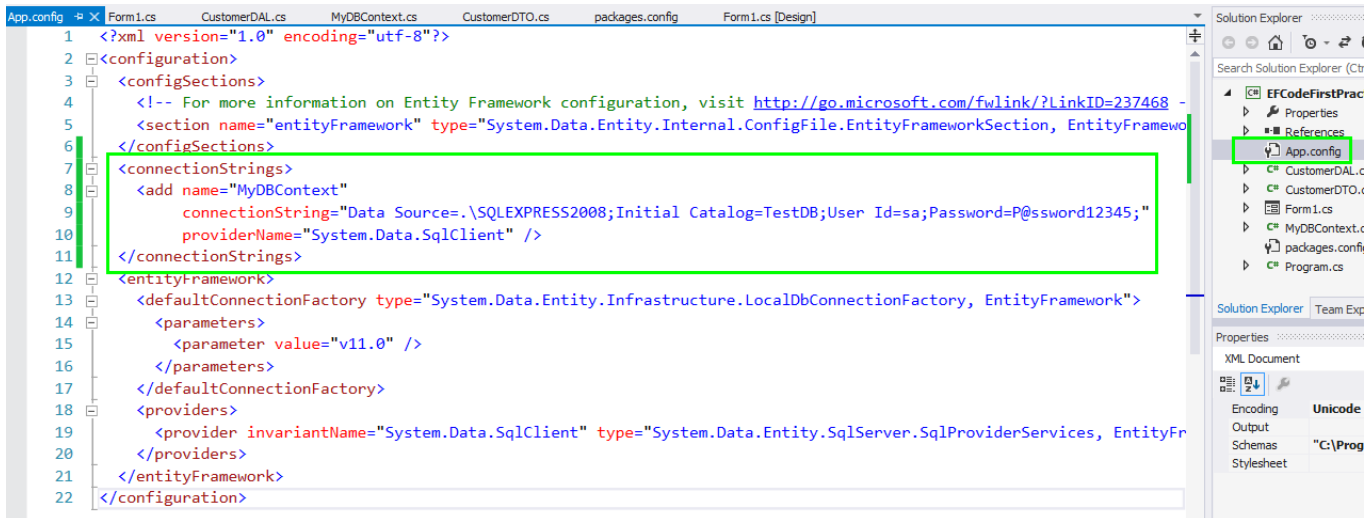
Server: =.\SQLEXPRESS2008

Database: TestDB

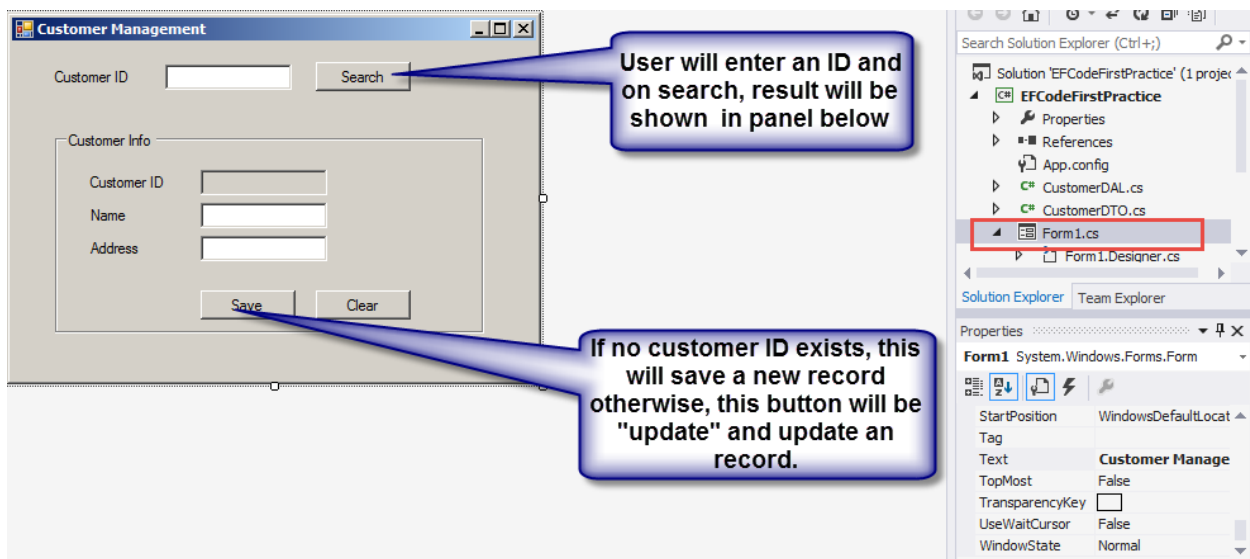
User Id: Sa

Password: P@ssword12345

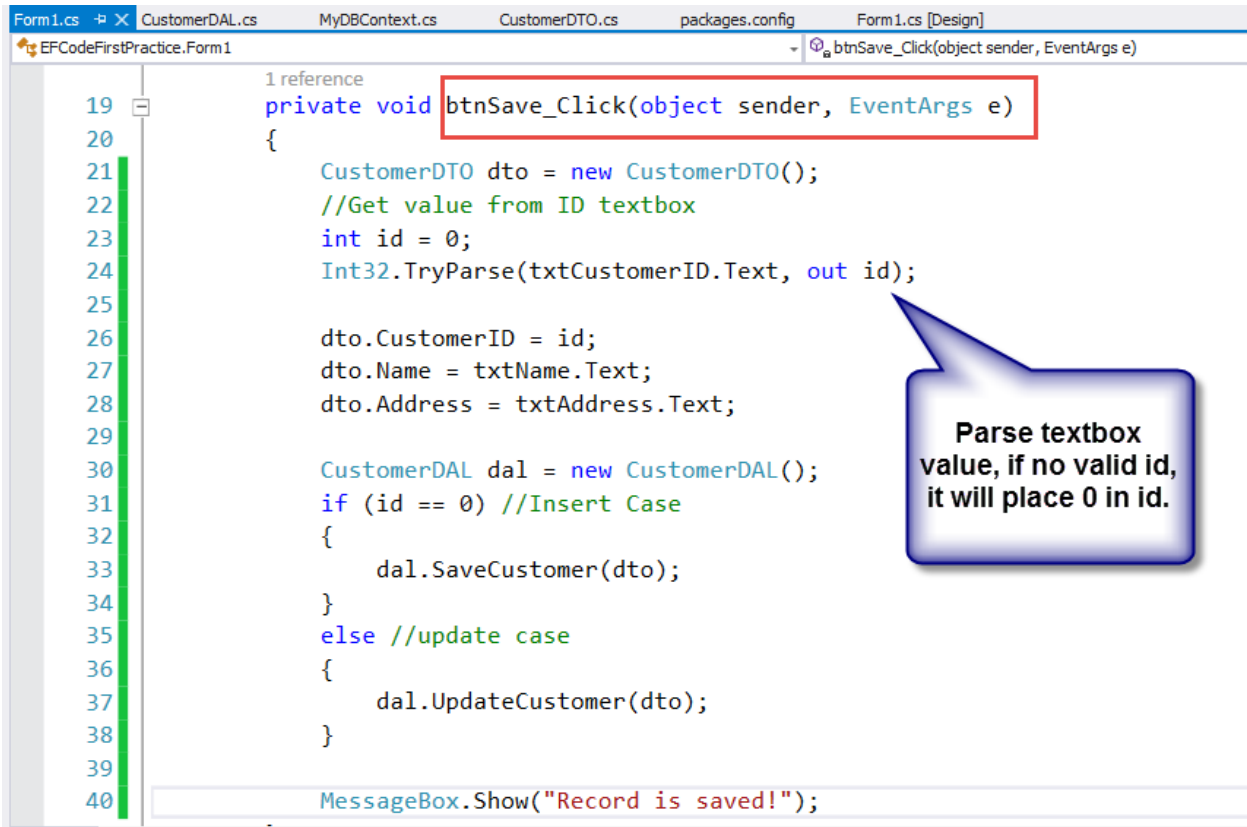
```
<add name="MyDBContext"
connectionString="Data Source=.\SQLEXPRESS2008;Initial Catalog=TestDB;User
Id=sa;Password=P@ssword12345;"
providerName="System.Data.SqlClient" />
```



17. Now are done with all logic to interact with database. Let's design your windows form as shown in the screenshot below. Name your controls properly. Here we are allowing the user to save a new customer. If user searches a customer by providing the ID, record will be loaded in the panel and record will be updated if user saves it.



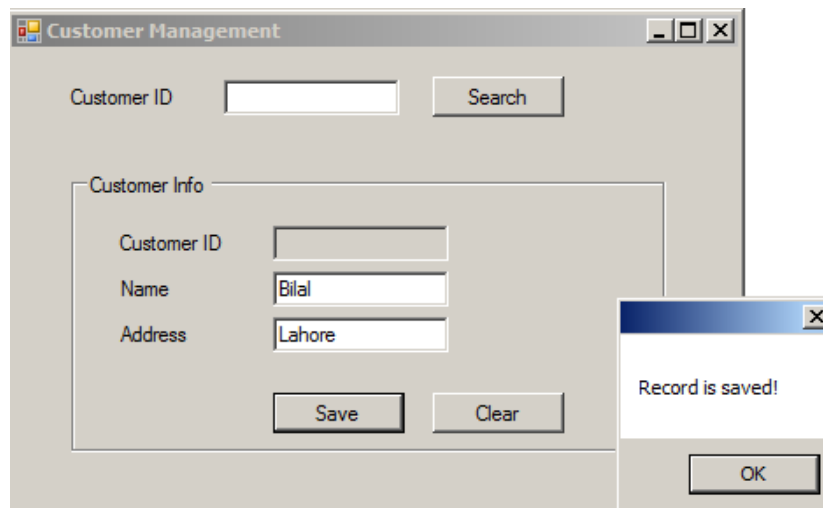
17- In the “Save” button click event, add following code. This code is reading the values from text boxes. If “readyonly” customer id text box is not having anything, it will insert the record otherwise it will update the record.



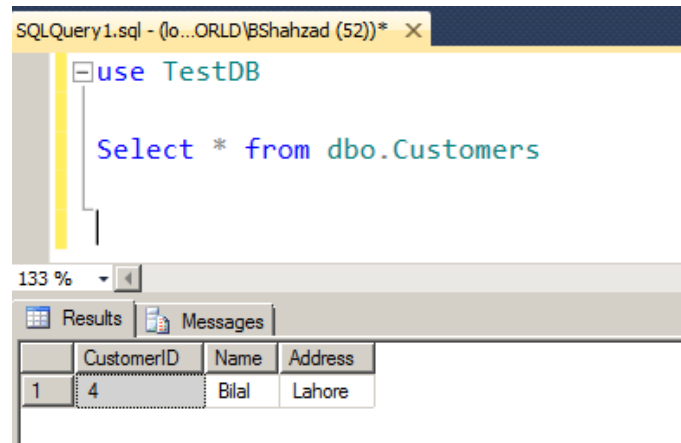
```
19 private void btnSave_Click(object sender, EventArgs e)
20 {
21     CustomerDTO dto = new CustomerDTO();
22     //Get value from ID textbox
23     int id = 0;
24     Int32.TryParse(txtCustomerID.Text, out id);
25
26     dto.CustomerID = id;
27     dto.Name = txtName.Text;
28     dto.Address = txtAddress.Text;
29
30     CustomerDAL dal = new CustomerDAL();
31     if (id == 0) //Insert Case
32     {
33         dal.SaveCustomer(dto);
34     }
35     else //update case
36     {
37         dal.UpdateCustomer(dto);
38     }
39
40     MessageBox.Show("Record is saved!");
```

1 reference
Parse textbox value, if no valid id, it will place 0 in id.

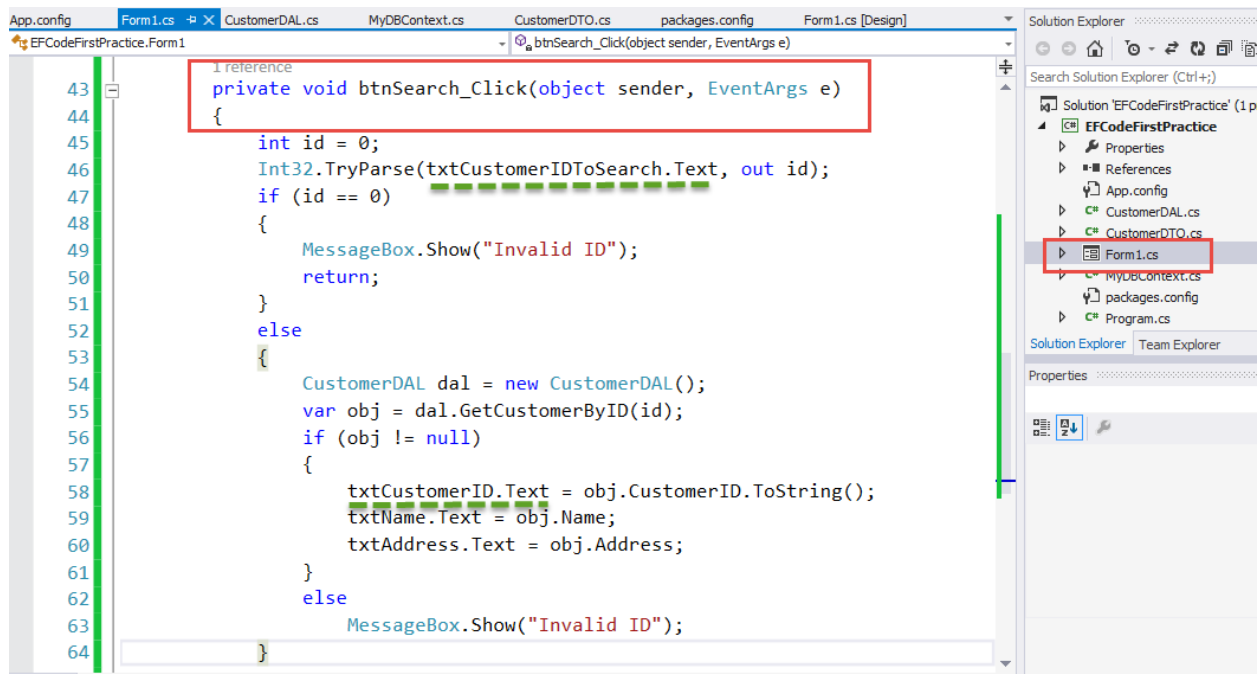
18 – Run the application; provide some data and click “Save”. If you face any error here, check if all steps are followed correctly.



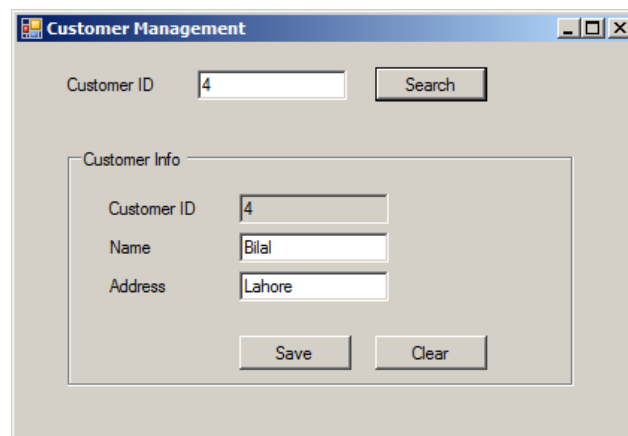
18- Run query in SQL Server to verify if data is saved or not.



19- Now let's write logic of Search button click event handler. Here we are getting record against the id and showing in the fields.



20- Run the application, provide the ID and search it.



21- Now make any changes in the data and Save.

The image shows a 'Customer Management' application window. It has a 'Customer ID' field with the value '4' and a 'Search' button. Below this is a 'Customer Info' dialog box with three input fields: 'Customer ID' (4), 'Name' (Bilal 2), and 'Address' (Lahore 2). There are 'Save' and 'Clear' buttons at the bottom of the dialog. To the right of the dialog, a small message box says 'Record is saved!' with an 'OK' button.

22- Verify in SQL Server if changes are made or not.

The image shows a SQL Server Enterprise Manager window with a query executed. The query is 'use TestDB' followed by 'Select * from dbo.Customers'. The results are displayed in a table with columns 'CustomerID', 'Name', and 'Address'. The first row shows '4', 'Bilal 2', and 'Lahore 2'.

	CustomerID	Name	Address
1	4	Bilal 2	Lahore 2