**LINQ To Entity:**

1. Create Console Application
2. Create Database Table
3. Add Required Packages
4. Define the connection String in the App.Config

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="DbConnection" connectionString="Server=LIN59015995;Database=MyDatabase\_Demo;User Id=rajesh;password=Nagpur@123;Encrypt=False"/>

</connectionStrings>

</configuration>

**Conext File:**

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

string ConnectionString = ConfigurationManager.ConnectionStrings["DbConnection"].ConnectionString;

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.

//=> optionsBuilder.UseSqlServer("Server=LIN59015995;Database=MyDatabase\_Demo;User Id=rajesh;password=Nagpur@123;Encrypt=False");

optionsBuilder.UseSqlServer(ConnectionString);

}

1. Execute Scaffold-DbConetxt Comand to create Model & Context.
2. Write LINQ Code in the Program.cs

using LINQ\_TO\_Entiity\_Demo.Models;

using System;

using Microsoft.Extensions.Logging.Console;

using Microsoft.Extensions.Logging;

namespace LINQ\_TO\_Entiity\_Demo

{

class Program

{

public static void Main(string[] args)

{

using (var context=new MyDatabaseDemoContext())

{

//var Emp=context.Employees.First(e => e.EmpName == "MyEmployee1");

//Console.WriteLine("Emp Name:{0}, Emp Salary:{1}",Emp.EmpName,Emp.EmpSalary);

var Emp = from e in context.Employees

where e.EmpSalary>5000

orderby e.EmpSalary descending

select e;

foreach(var data in Emp)

{

Console.WriteLine("Emp Name:{0}, Emp Salary:{1}",data.EmpName, data.EmpSalary);

}

}

var loggerFactory = LoggerFactory.Create(builder => builder.AddConsole());

var logger = loggerFactory.CreateLogger<Program>();

logger.LogWarning("Warning log message");

logger.LogError("Error log message");

logger.LogInformation("Info log message");

}

}

}

**Logging in .Net Core:**

**Install Packages**

Microsoft.Extensions.Logging

Microsoft.Extensions.Logging.Console;

**Program.cs**

using LINQ\_TO\_Entiity\_Demo.Models;

using System;

using Microsoft.Extensions.Logging.Console;

using Microsoft.Extensions.Logging;

namespace LINQ\_TO\_Entiity\_Demo

{

class Program

{

public static void Main(string[] args)

{

var loggerFactory = LoggerFactory.Create(builder => builder.AddConsole());

var logger = loggerFactory.CreateLogger<Program>();

logger.LogWarning("Warning log message");

logger.LogError("Error log message");

logger.LogInformation("Info log message");

}

}

}

**For File Logging:**

* Install Serilog from NuGet Package to the project.
* Install NetEscapades.Extensions.Logging.RollingFile package
* In The Program.cs add ConfigureLogging to setup the file log

builder.Host.ConfigureLogging(logging =>

{

logging.ClearProviders();

logging.AddConsole();

logging.AddFile();

});

* Add Logger message in any Controller:

\_logger.LogInformation("Info: Index Action Method Called");

\_logger.LogError("Error Found");

**Raw Sql Query :**

1. Create Console Application
2. Create Database Table
3. Add Required Packages
4. Define the connection String in the App.Config

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="DbConnection" connectionString="Server=LIN59015995;Database=MyDatabase\_Demo;User Id=rajesh;password=Nagpur@123;Encrypt=False"/>

</connectionStrings>

</configuration>

**Conext File:**

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

string ConnectionString = ConfigurationManager.ConnectionStrings["DbConnection"].ConnectionString;

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.

//=> optionsBuilder.UseSqlServer("Server=LIN59015995;Database=MyDatabase\_Demo;User Id=rajesh;password=Nagpur@123;Encrypt=False");

optionsBuilder.UseSqlServer(ConnectionString);

}

1. Execute Scaffold-DbConetxt Comand to create Model & Context.
2. Write LINQ Code in the Program.cs

using LINQ\_TO\_Entiity\_Demo.Models;

using System;

using Microsoft.Extensions.Logging.Console;

using Microsoft.Extensions.Logging;

using Microsoft.EntityFrameworkCore;

namespace LINQ\_TO\_Entiity\_Demo

{

class Program

{

public static void Main(string[] args)

{

using (var context = new MyDatabaseDemoContext())

{

var data=context.Employees.FromSqlRaw("select emp\_id,emp\_name,emp\_salary,address from Employees where emp\_id=1").ToList();

foreach(var item in data)

{

Console.WriteLine("Emp Name:{0},Emp Salary:{1}",item.EmpName,item.EmpSalary);

}

}

}

}

}

**Using Stored Procedure:**

using LINQ\_TO\_Entiity\_Demo.Models;

using System;

using Microsoft.Extensions.Logging.Console;

using Microsoft.Extensions.Logging;

using Microsoft.EntityFrameworkCore;

namespace LINQ\_TO\_Entiity\_Demo

{

class Program

{

public static void Main(string[] args)

{

using (var context = new MyDatabaseDemoContext())

{

var data=context.Employees.FromSqlRaw("p\_GetEmployees").ToList();

foreach(var item in data)

{

Console.WriteLine("Emp Name:{0},Emp Salary:{1},Address:{2}",item.EmpName,item.EmpSalary,item.Address);

}

}

}

}

}