

# LINQ - ENTITIES

[http://www.tutorialspoint.com/linq/linq\\_entities.htm](http://www.tutorialspoint.com/linq/linq_entities.htm)

Copyright © tutorialspoint.com

A part of the ADO.NET Entity Framework, LINQ to Entities is more flexible than LINQ to SQL, but is not much popular because of its complexity and lack of key features. However, it does not have the limitations of LINQ to SQL that allows data query only in SQL server database as LINQ to Entities facilitates data query in a large number of data providers like Oracle, MySQL, etc.

Moreover, it has got a major support from ASP.Net in the sense that users can make use of a data source control for executing a query via LINQ to Entities and facilitates binding of the results without any need of extra coding.

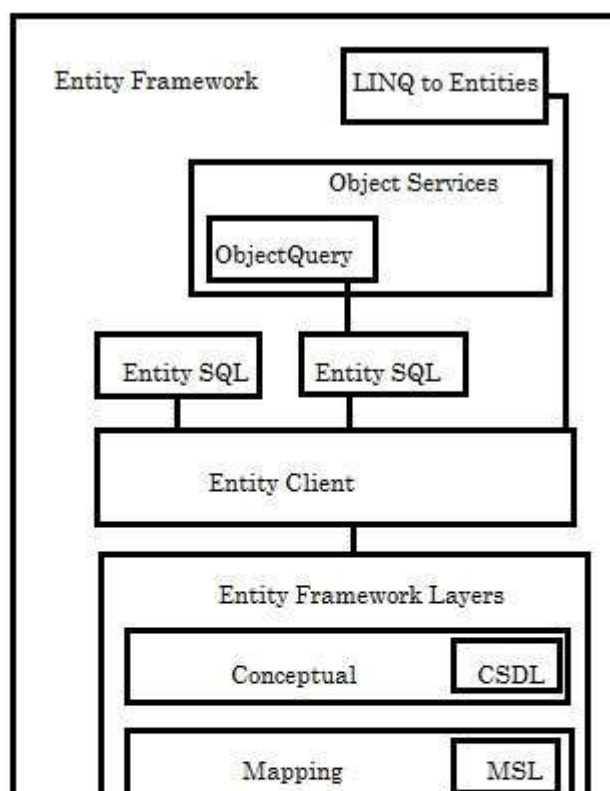
LINQ to Entities has for these advantages become the standard mechanism for the usage of LINQ on databases nowadays. It is also possible with LINQ to Entities to change queried data details and committing a batch update easily. What is the most intriguing fact about LINQ to Entities is that it has same syntax like that of SQL and even has the same group of standard query operators like Join, Select, OrderBy, etc.

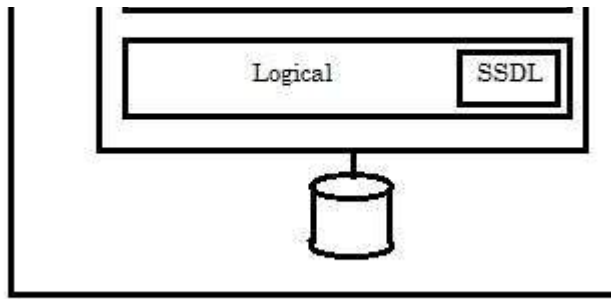
## LINQ to Entities Query Creation and Execution Process

- Construction of an **ObjectQuery** instance out of an **ObjectContext** *EntityConnection*
- Composing a query either in C# or Visual Basic VB by using the newly constructed instance
- Conversion of standard query operators of LINQ as well as LINQ expressions into command trees
- Executing the query passing any exceptions encountered to the client directly
- Returning to the client all the query results

**ObjectContext** is here the primary class that enables interaction with **Entity Data Model** or in other words acts as a bridge that connects LINQ to the database. Command trees are here query representation with compatibility with the Entity framework. The Entity Framework on the other hand is actually **Object Relational Mapper** abbreviated generally as ORM by the developers that does the generation of business objects as well as entities as per the database tables and facilitates various basic operations like create, update, delete and read.

Below is a diagram of the entity framework to have a better understanding of the concept.

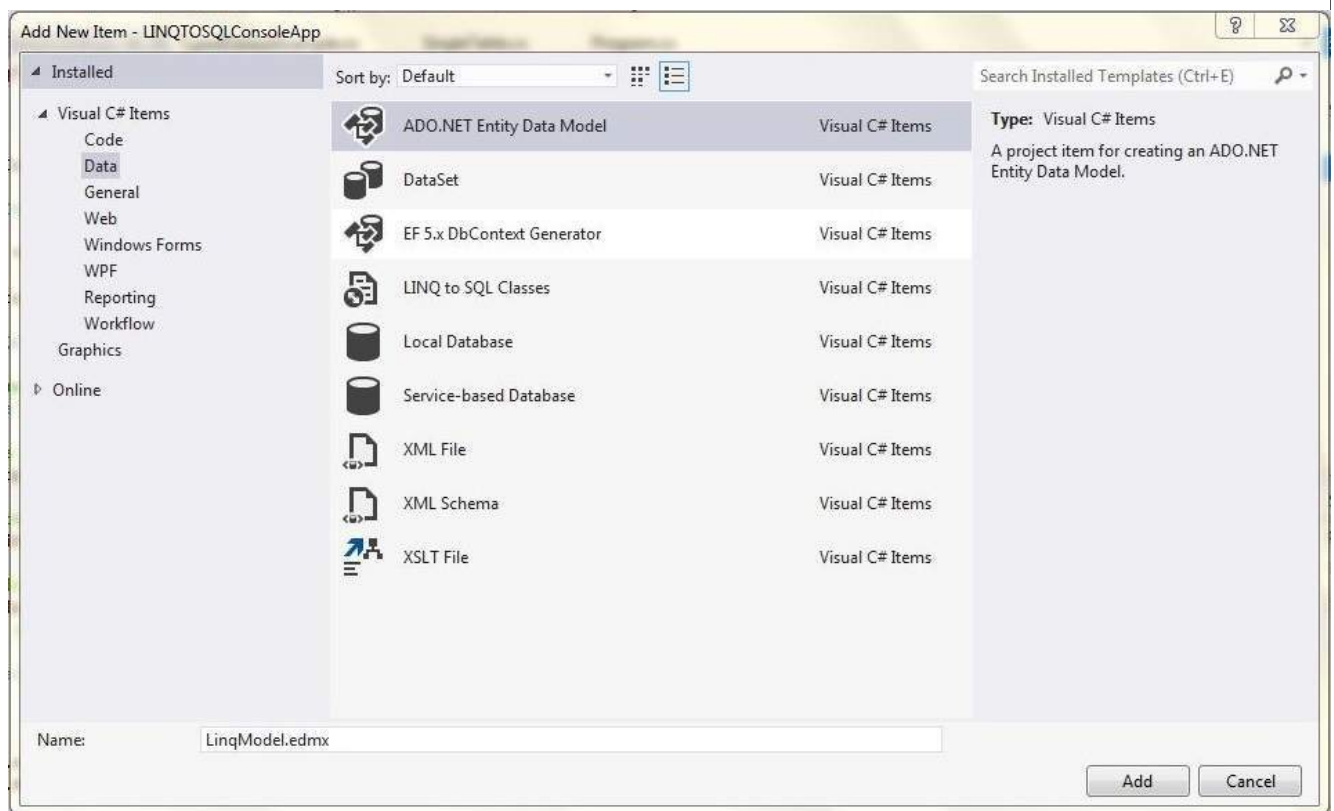




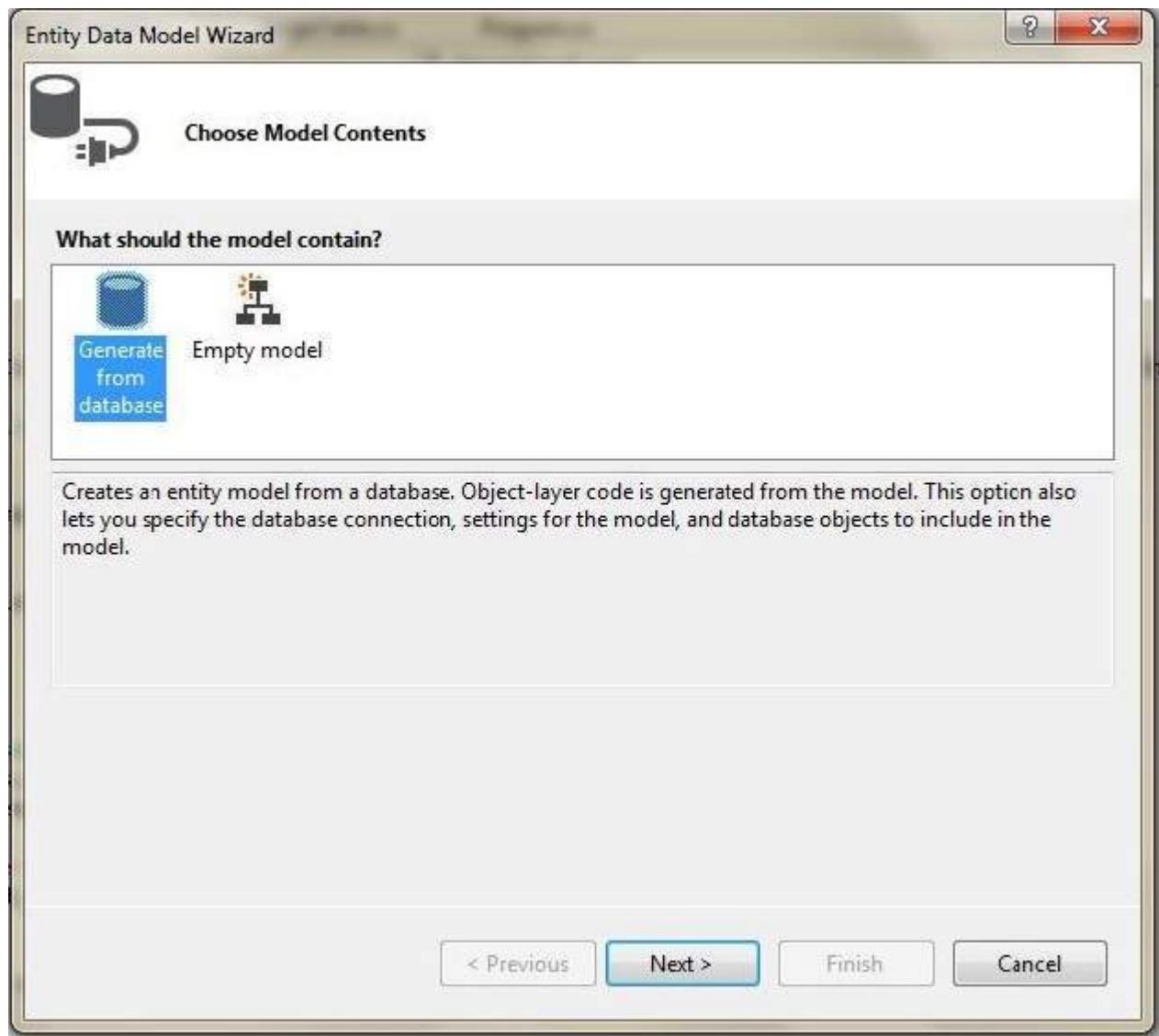
## Example of ADD, UPDATE and DELETE using LINQ with Entity Model

First add Entity Model by following below steps.

- **Step 1:** Right click on project and click add new item will open window as per below. Select ADO.NET Entity Data Model and specify name and click on Add.




- **Step 2:** Select Generate from database.



- **Step 3:** Choose Database Connection.

Entity Data Model Wizard

 Choose Your Data Connection

Which data connection should your application use to connect to the database?

LinqToSQLDBConnectionString (Settings) New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

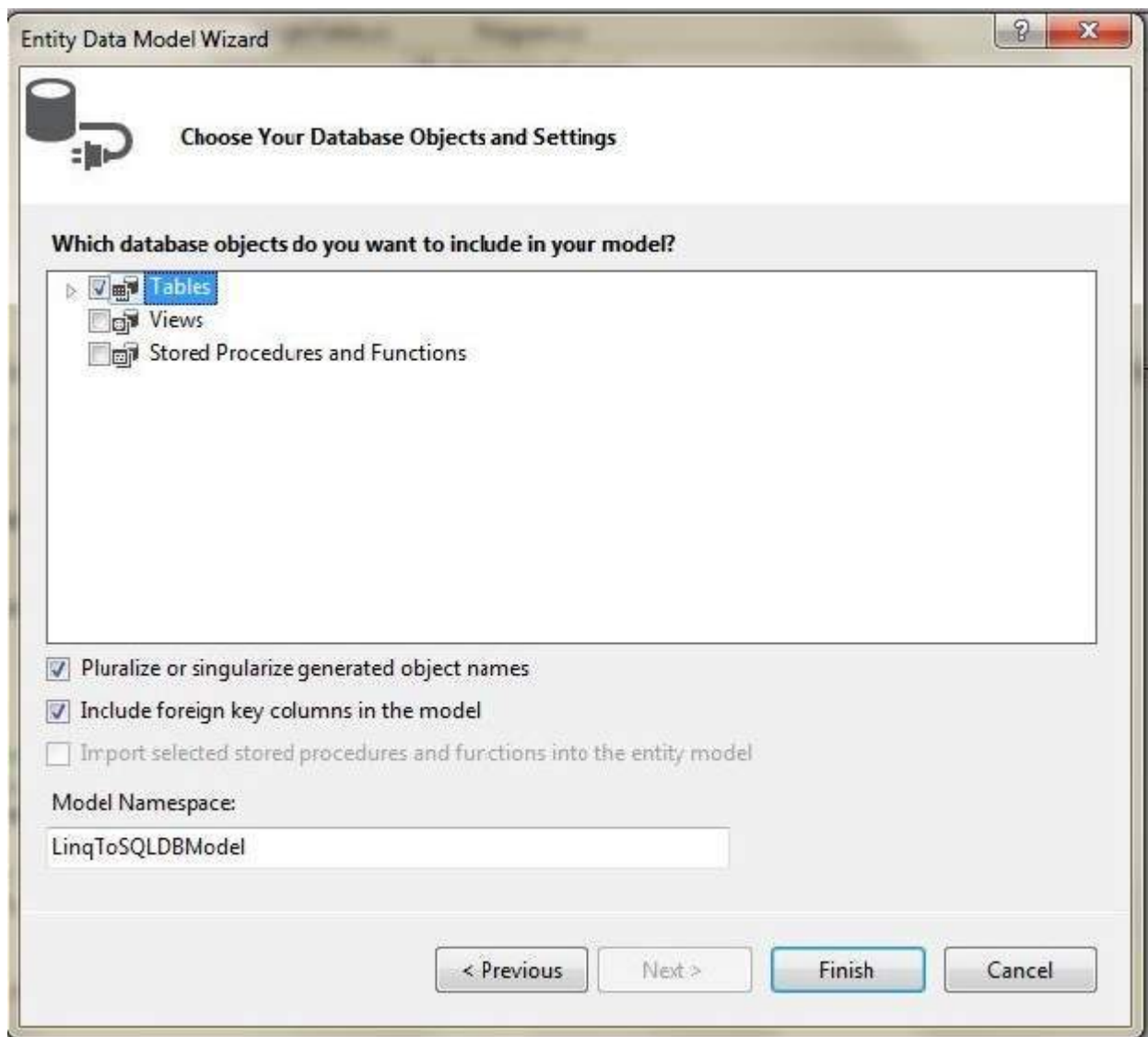
```
metadata=res://*/LinqModel.csdl|res://*/LinqModel.ssdl|
res://*/LinqModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=HITESH-PC\MSSQLSERVER2008;initial catalog=LinqToSQLDB;integrated
security= True;MultipleActiveResultSets= True;App=EntityFramework"
```

☒ Save entity connection settings in App.Config as:

LinqToSQLDBEntities

< Previous **Next >** Finish Cancel

- **Step 4:** Select all the tables.



Now write the following code.

```
using DataAccess;
using System;
using System.Linq;

namespace LinqToSQLConsoleApp
{
    public class LinqToEntityModel
    {
        static void Main(string[] args)
        {
            using (LinqToSQLDBEntities context = new LinqToSQLDBEntities())
            {
                //Get the List of Departments from Database
                var departmentList = from d in context.Departments
                                    select d;

                foreach (var dept in departmentList)
                {
                    Console.WriteLine("Department Id = {0} , Department Name = {1}",
                                      dept.DepartmentId, dept.Name);
                }

                //Add new Department
                DataAccess.Department department = new DataAccess.Department();
                department.Name = "Support";

                context.Departments.Add(department);
                context.SaveChanges();
            }
        }
    }
}
```

```

        Console.WriteLine("Department Name = Support is inserted in Database");

        //Update existing Department
        DataAccess.Department updateDepartment = context.Departments.FirstOrDefault(d
=>d.DepartmentId == 1);
        updateDepartment.Name = "Account updated";
        context.SaveChanges();

        Console.WriteLine("Department Name = Account is updated in Database");

        //Delete existing Department
        DataAccess.Department deleteDepartment = context.Departments.FirstOrDefault(d
=>d.DepartmentId == 3);
        context.Departments.Remove(deleteDepartment);
        context.SaveChanges();

        Console.WriteLine("Department Name = Pre-Sales is deleted in Database");

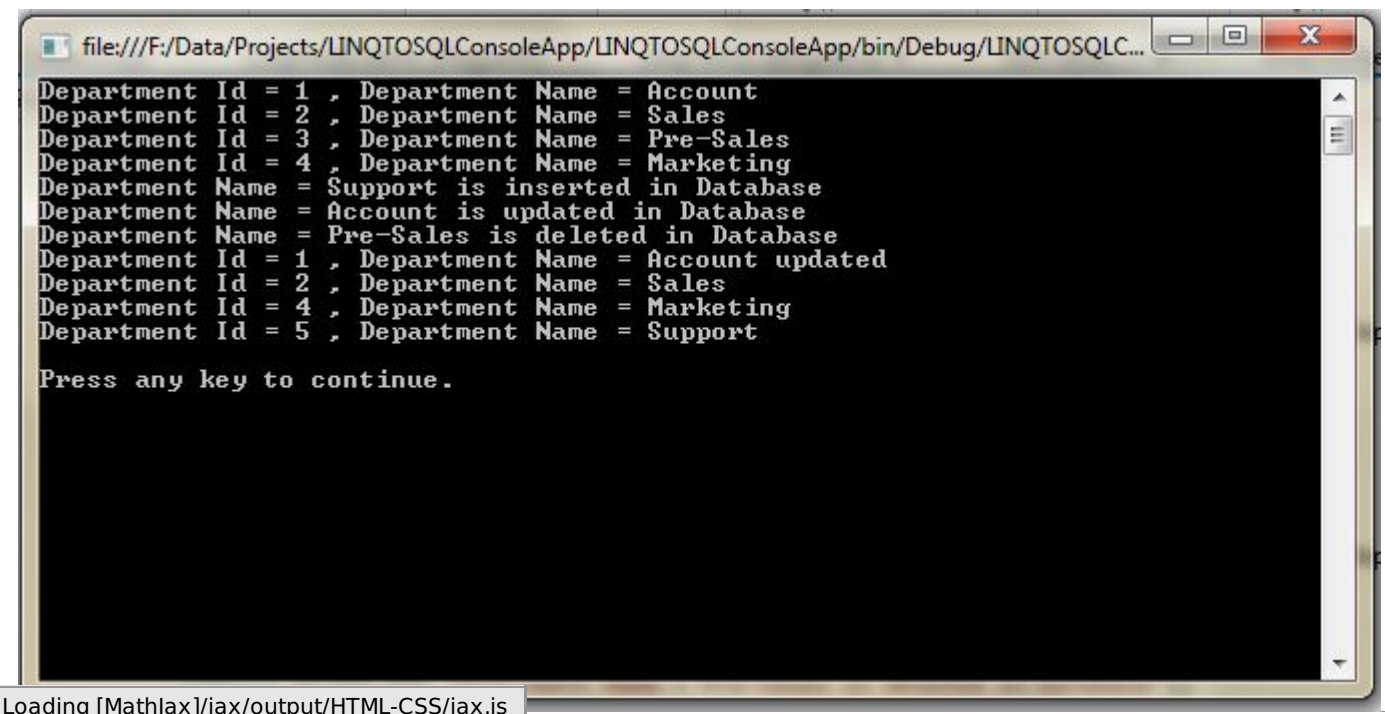
        //Get the Updated List of Departments from Database
        departmentList = from d in context.Departments
        select d;

        foreach (var dept in departmentList)
        {
            Console.WriteLine("Department Id = {0} , Department Name = {1}",
                               dept.DepartmentId, dept.Name);
        }

        Console.WriteLine("\nPress any key to continue.");
        Console.ReadKey();
    }
}

```

When the above code is compiled and executed, it produces the following result:



```

file:///F:/Data/Projects/LINQTOSQLConsoleApp/LINQTOSQLConsoleApp/bin/Debug/LINQTOSQLC...
Department Id = 1 , Department Name = Account
Department Id = 2 , Department Name = Sales
Department Id = 3 , Department Name = Pre-Sales
Department Id = 4 , Department Name = Marketing
Department Name = Support is inserted in Database
Department Name = Account is updated in Database
Department Name = Pre-Sales is deleted in Database
Department Id = 1 , Department Name = Account updated
Department Id = 2 , Department Name = Sales
Department Id = 4 , Department Name = Marketing
Department Id = 5 , Department Name = Support

Press any key to continue.

```

Loading [MathJax]/jax/output/HTML-CSS/jax.js