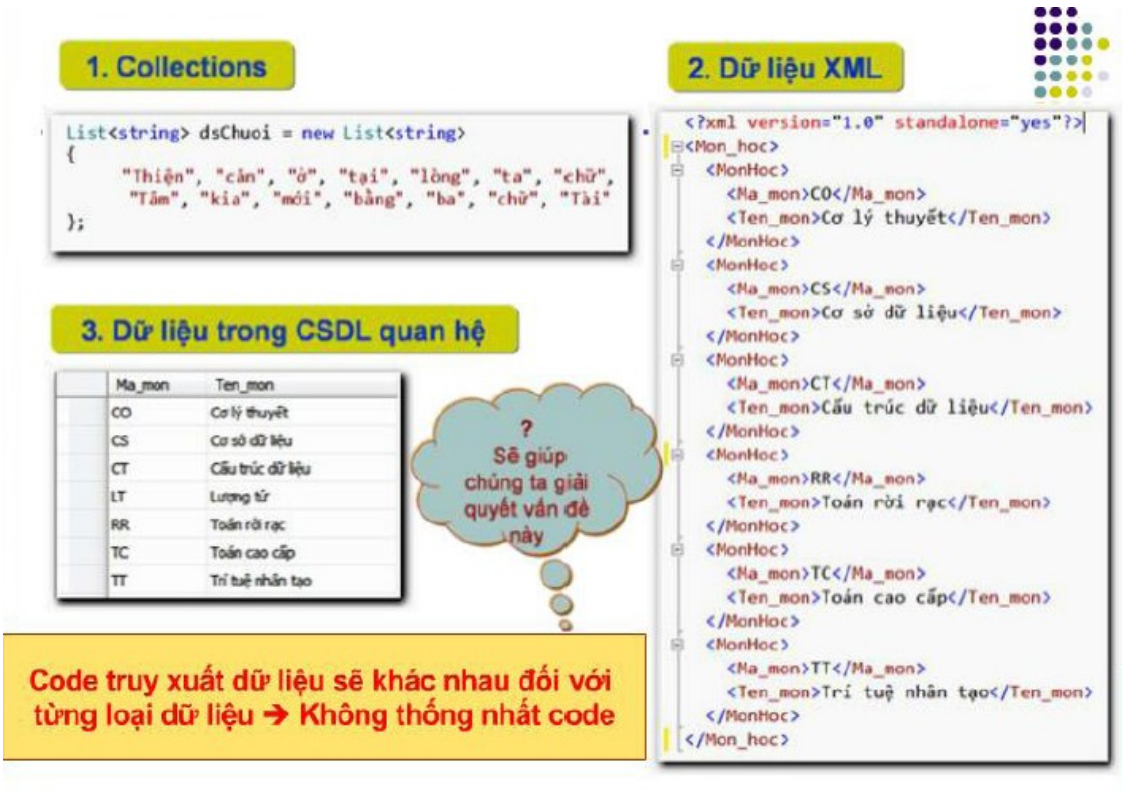
Truy vấn và xử lý dữ liệu với Entity Framework Core

1. Tổng quan về LINQ

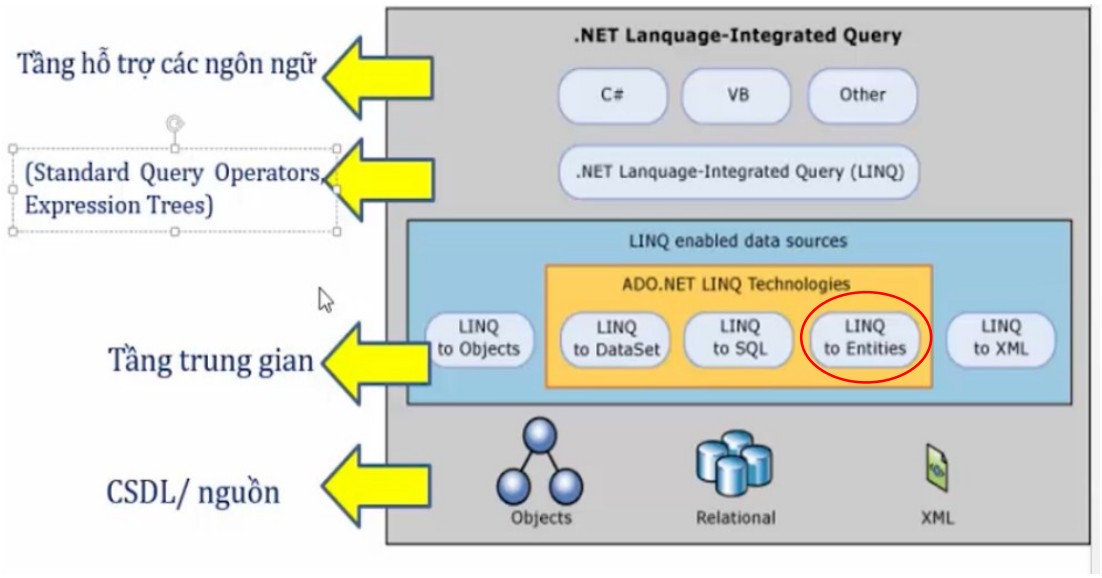
1.1. Giới thiệu LINQ

Trong thực tế 1 dự án có thể phải làm việc với nhiều nguồn dữ liệu khác nhau, mỗi loại nguồn dữ liệu sẽ có công nghệ tương tác tương ứng: CSDL SQLServer-ADO.NET; XML/JSON-DOM;…=> Code không thống nhất.



- Language Integrated Query (LINQ) là ngôn ngữ truy vấn tích hợp, nó tích hợp cú pháp truy vấn (gần giống các câu lệnh SQL) vào bên trong ngôn ngữ lập trình C#.
- Với LINQ, ta có thể truy vấn/cập nhật nhiều nguồn dữ liệu khác nhau trong C#: đối tượng (object: array, List); cơ sở dữ liệu quan hệ: SQL Server, MySQL; tài liệu XML/JSON, mô hình dữ liệu thực thể (entity data model).
- Đưa ra khả năng lập trình mới trong .NET - **Giải pháp lập trình hợp nhất.**

1.2. . Kiến trúc và các thành phần của LINQ (4 tầng)



1.3. Các thư viện hỗ trợ LINQ

- System.Linq: hỗ trợ sử dụng các Objects
- System.Data.Linq: hỗ trợ sử dụng các CSDL quan hệ
- System.Data.Objects: hỗ trợ sử dụng các Entities
- System.XML.Linq: hỗ trợ sử dụng XML

1.4. Truy vấn LINQ theo biểu thức

- **Cú pháp:**

var *Tên-biến* = **from** *id* **in** *source*
[Các-mệnh-đề-truy-vấn-chuẩn**: where, orderby,…**]
select *id/expr*

- **Trong đó:**
  + Từ khóa **var** để chỉ định *Tên-biến* có kiểu bất kỳ
  + **source** có thể là: mảng, tập hợp, bảng, xml…
  + **id** là định danh từng phần tử trong **source**
- **Ba thành phần hoạt động của truy vấn:**
  - Nhập nguồn dữ liêu
  - Tạo truy vấn
  - Thi hành truy vấn

- **Ví dụ:** Đưa ra các số chia hết cho 2 từ mảng các số nguyên

```
// 1. Data source.
      int[] numbers = new int[7] {1, 6, 9, 2, 3, 4,5 };
// 2. Query creation.
      var Query = from x in numbers
            where (x % 2) == 0
            orderby x descending
            select x;
 // 3. Query execution.
```

```
foreach (int x in Query)
  { Console.Write("{0} ", x); }
```

1.5. Truy vấn LINQ theo phương thức mở rộng

a) Biểu thức Lambda – Lambda Expression

- **Cú pháp:**
  Tham_số_đầu_vào => *Biểu_thức* ***hoặc*** *Khối_lệnh*
- **Trong đó:**
  - =>: Gọi là toán tử **Lambda**
  - *Tham_số_đầu_vào*: Có thể có 1 hoặc nhiều tham số (có kiểu tường minh hoặc không), cách nhau bằng dấu phẩy
- **Ví dụ:**

```
x=>x*x+2  //nếu tham số x=1 thì biểu thức Lambda=3
x=> x==6//nếu tham số x=6 thì biểu thức Lambda=True
(x,y)=>x+y  //nếu tham số x=2, y=4 thì biểu thức Lambda=6
(x,y)=> x==y  //nếu tham số x=2, y=4 thì biểu thức Lambda=False
```

b) Cú pháp LINQ theo phương thức mở rộng

- **Cú pháp:**

var ***Tên-biến= Source.Tên-phương-thức-mở-rộng(Biểu-thức-Lambda)***

- **Trong đó:**
  + **Source** có thể là: mảng, tập hợp, bảng, xml…
  + *Tên-phương-thức-mở-rộng:* có thể là: *Select, Where, Orderby*,…
- **Ví dụ:** Đưa ra các số chia hết cho 2 từ mảng các số nguyên

```
// 1. Data source.
   int[] numbers = new int[7] { 1, 6, 9, 2, 3, 4, 5 };
// 2. Query creation.
    var Query = numbers.Where (x=>(x % 2)== 0)
                              .OrderBy(x =>x)
                             //.OrderByDescending(x => x)
                              .Select (x=>x);
// 3. Query execution.
       foreach (int x in Query)
            { Console.Write("{0} ", x); }
             Console.ReadLine();
```

c) Danh sách các phương thức mở rộng

- **Phương thức truy vấn cơ bản**

Phương thức	Mô tả	Ví dụ
.Where(e=>điều kiện)	Lọc	Students.Where(s=>s.Marks>9)
.GroupBy(e=>biểu thức)	Nhóm	Students.GroupBy(s=>s.Clazz)
.OrderBy(e=>biểu thức) .OrderByDescending(e=>biểu thức)	Sắp xếp	Students.OrderBy(s=>s.Name)
.Select(e=>đối tượng)	Chọn	Students.Select(s=>new{s.Name, s.Marks}
.Distinct()	Chọn các thành phần khác nhau	Numbers.Distinct()

- **Phương thức truy vấn phân trang**

Phương thức	Mô tả	Ví dụ
.Take(số lượng)	Lấy các phần tử đầu	Students.Take(5)
.Skip(số lượng)	Bỏ qua các phần tử đầu	Students.Skip(3).Take(6)
.TakeWhile(e=>đ kiện)	Lấy các phần tử đầu thỏa điều kiện	Students.TakeWhile(s=>s.Marks<4)
.SkipWhile(e=>đ kiện)	Bỏ qua các phần tử đầu thỏa điều kiện	Students.SkipWhile(s=> s.Marks<0}

- **Phương thức truy vấn 1 phần tử**

Phương thức	Mô tả	Ví dụ
.Single(e=>đ kiện)	Lấy một phần tử thỏa điều kiện. Ngoại lệ nếu không tìm thấy hoặc nhiều hơn 1	Students.Single (s=>s.Id=="Hoa")
.First()	Lấy phần tử đầu	Students.First()
.Last()	Lấy phần tử cuối	Students.Last()

- **Phương thức truy vấn thống kê**

Phương thức	Mô tả	Ví dụ
.Sum(e=>biểu thức số học)	Tính tổng	Students.Sum(s=>s.Marks)
.Count(e=>biểu thức số học)	Đếm số lượng	Students.Count(s=>s.Id)
.Min(e=>biểu thức số học)	Giá trị nhỏ nhất	Students.Min(s=>s.Marks)
.Max(e=>biểu thức số học)	Giá trị lớn nhất	Students.Max(s=>s.Marks)
.Average(e=>biểu thức số học)	Giá trị trung bình	Students.Average(s=>s.Marks)

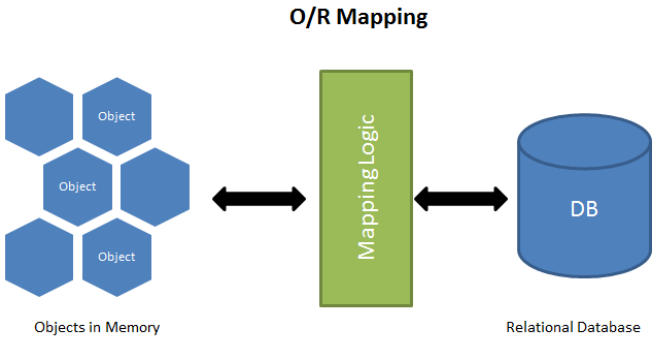
- **Phương thức kiểm tra phần tử trong tập**

Phương thức	Mô tả	Ví dụ
.Contains(phần tử)	Tập có chứa phần tử không	Students.Contains(hoa)
.Any(e=>điều kiện)	Ít nhất một phần tử trong tập thỏa điều kiện	Students.Any(s=>s.Marks<3)
.All(e=>điều kiện)	Tất cả các phần tử trong tập thỏa điều kiện	Students.All(s=>s.Marks>5)

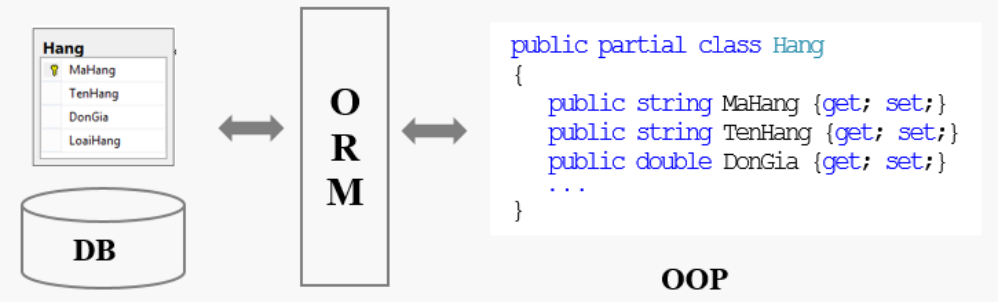
2. ORM và EF

2.1. ORM (Object Relational Mapping)

ORM là một kỹ thuật thực hiện ánh xạ CSDL sang các đối tượng trong các ngôn ngữ lập trình hướng đối tượng



Trong ORM các bảng tương ứng các class, mối ràng buộc giữa các bảng tương ứng quan hệ giữa các class

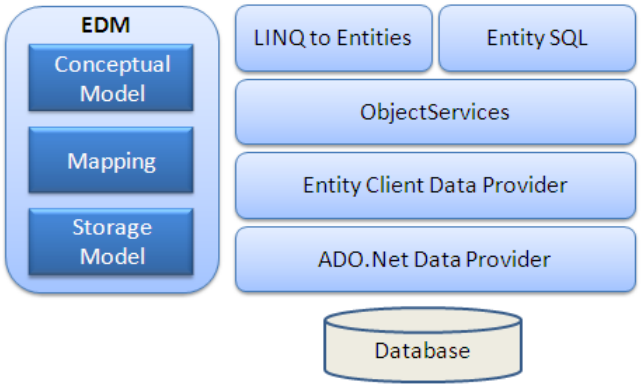


ORM Là khái niệm phổ biến, được cài đặt trong tất cả các ngôn ngữ hiện đại như: c#, java, php, node.js, …

2.2. EF – ENTITY FRAMEWORK

- EF là cách thực hiện của ORM trong .NET. Đây là một cải tiến của ADO.NET, cung cấp cho các nhà phát triển một cơ chế tự động để truy cập và lưu trữ dữ liệu trong cơ sở dữ liệu.
- Các tính năng:
  + Cung cấp cơ chế để truy cập và lưu trữ dữ liệu vào CSDL
  + Cung cấp các dịch vụ như theo dõi sự thay đổi của dữ liệu, dịch truy vấn . . .
  + Sử dụng LINQ to Entities để truy vấn, thêm, sửa, xóa dữ liệu
  + Tăng khả năng bảo trì và mở rộng cho ứng dụng

Kiến trúc của EF



3. Entity Framework Core (EF Core)

EF Core là phiên bản mới của EF. Nó là phiên bản nhẹ, có thể mở rộng, mã nguồn mở và đa nền tảng của EF.

EF Core làm việc như một ánh xạ ORM, nó cho phép :

+ Các nhà phát triển .NET làm việc với cơ sở dữ liệu sử dụng các đối tượng .NET
+ Loại bỏ hầu hết các lệnh truy cập dữ liệu thường phải viết.

Có thể truy cập nhiều csdl khác nhau như SQL Server, Sqllite, MySQL … thông qua các thư viện plug-in được gọi là Database Providers

4. Truy vấn dữ liệu sử dụng EF Core

Theo cách tiếp cận Database First, EF Core API tạo các lớp thực thể và Context dựa trên cơ sở dữ liệu hiện có bằng cách sử dụng các lệnh của EF Core

Các bước để truy vấn dữ liệu sử dụng EF Core

B1. Cài thêm các thư viện

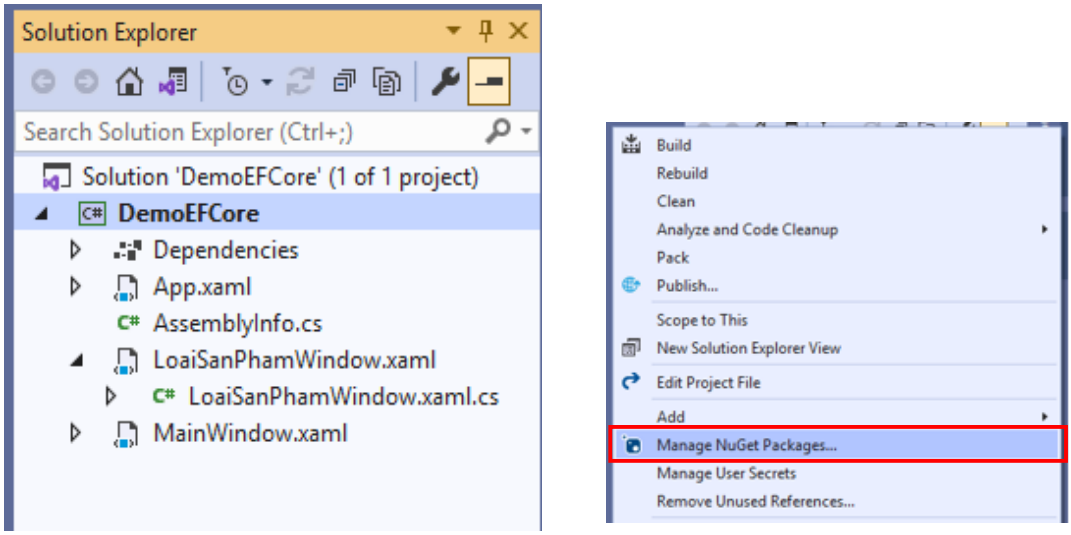
B2. Tạo model

B3. Truy vấn dữ liệu sử dụng LINQ

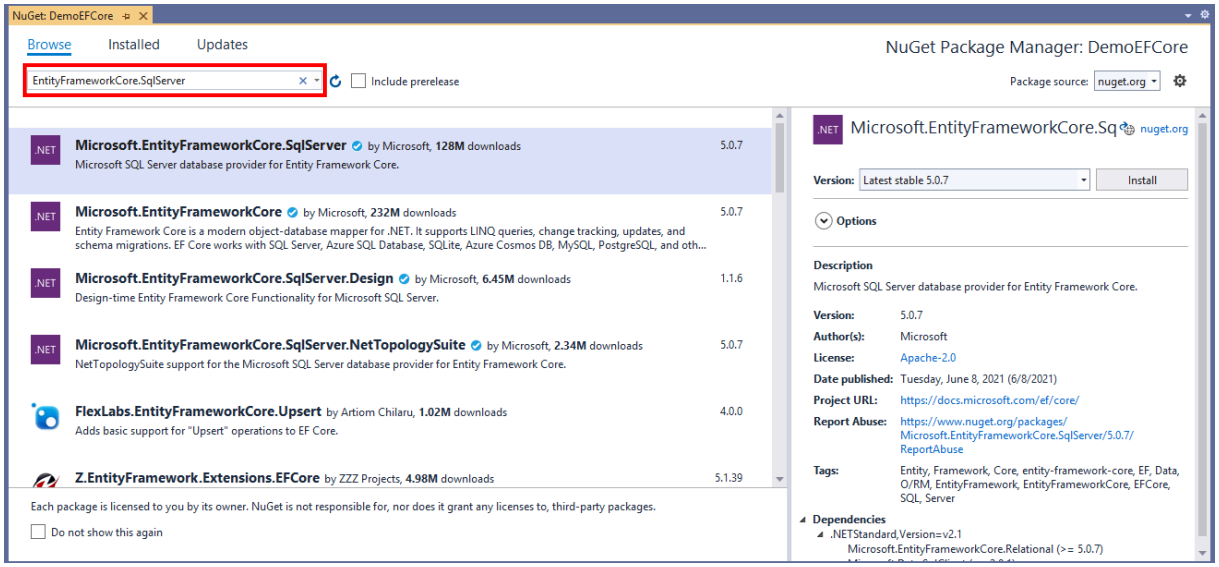
4.1. Cài đặt thêm 2 thư viện

+ Microsoft.EntityFrameworkCore.SqlServer (phiên bản 5.0.7)
+ Microsoft.EntityFrameworkCore.Tools (phiên bản 5.0.7)

Cách cài: ta nhấp phải chuột vào **project** trong cửa sổ **Solution Explorer** và chọn → **Manage NuGet Packages . . .**

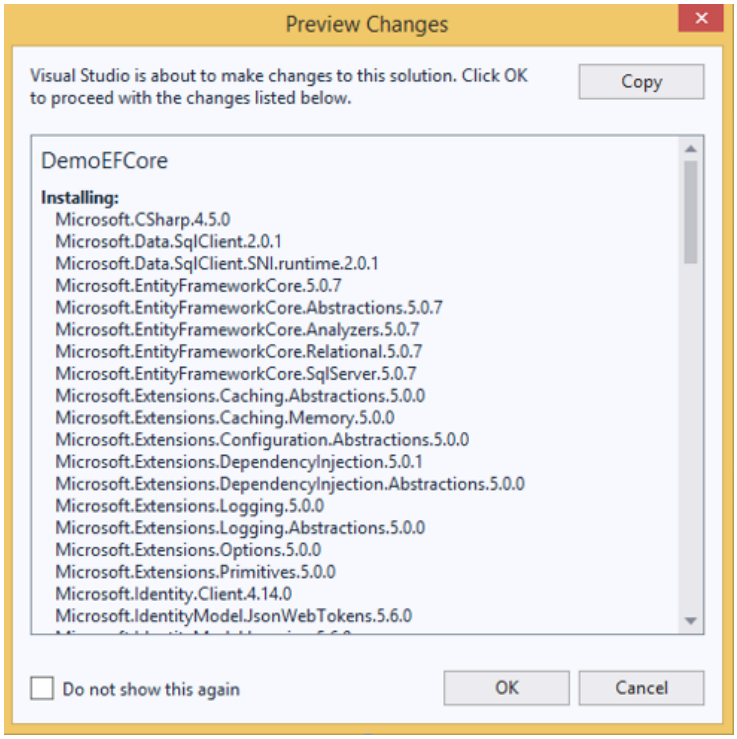


Giao diện trình quản lý gói NuGet được mở. Trong cửa sổ NuGet –Solution → chọn tab Browse và tìm kiếm gói muốn cài đặt như hình sau:

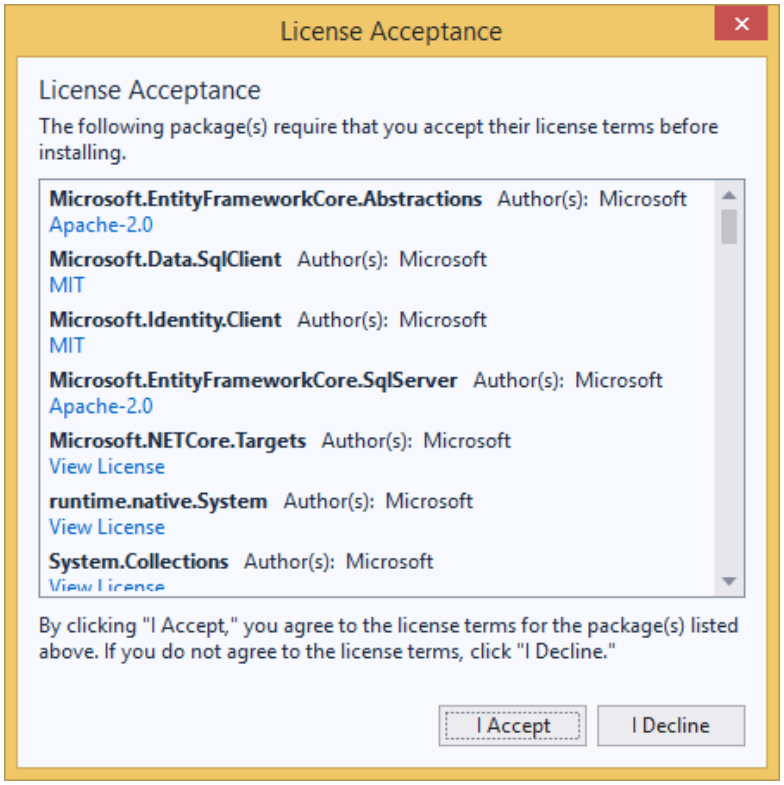


→ nhấn **Install** để cài đặt.

Cửa sổ xem trước hiển thị danh sách các gói sẽ được cài đặt trong ứng dụng, xem lại các thay đổi và nhấn OK

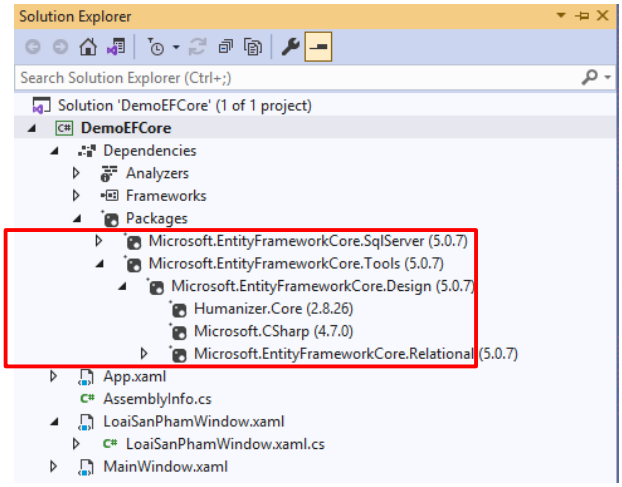


Cuối cùng chấp nhận các điều khoản cấp phép liên quan đến các gói được cài đặt



(Cài tương tự với thư viện **Microsoft.EntityFrameworkCore.Tools)**

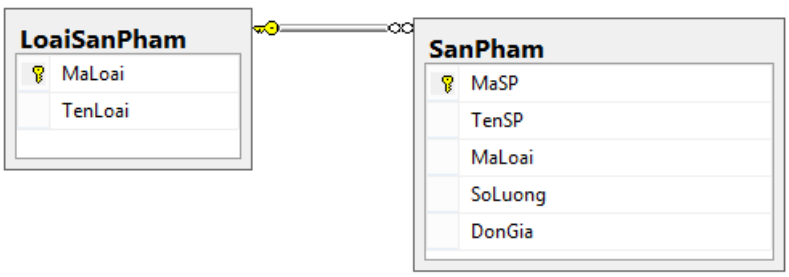
Kết quả sau khi cài:



4.2. Tạo model

4.2.1. Chuẩn bị cơ sở dữ liệu

Tạo cơ sở dữ liệu **QLBanHang** gồm 2 bảng LoaiSanPham và SanPham có lược đồ cơ sở dữ liệu như sau:



Thiết kế của 2 bảng:

DESKTOP-2KB389E\...dbo.LoaiSanPham*

Column Name	Data Type	Allow Nulls
MaLoai	char(10)	☐
TenLoai	nvarchar(50)	☑

DESKTOP-2KB389E\...ng - dbo.SanPham

Column Name	Data Type	Allow Nulls
MaSP	char(10)	☐
TenSP	nvarchar(50)	☑
MaLoai	char(10)	☑
DonGia	float	☑
SoLuong	int	☑

4.2.2. Tạo model

EF Core không hỗ trợ thiết kế trực quan cho mô hình cơ sở dữ liệu, tạo các lớp thực thể và lớp Context như các phiên bản trước. Thay vào đó ta sử dụng lệnh Scaffold-DbContext. Lệnh này tạo các lớp thực thể và Context (bằng cách dẫn xuất từ lớp DbContext) dựa trên lược đồ của cơ sở dữ liệu hiện có.

- Chọn menu Tools → NuGet Package Manager → Package Manager Console

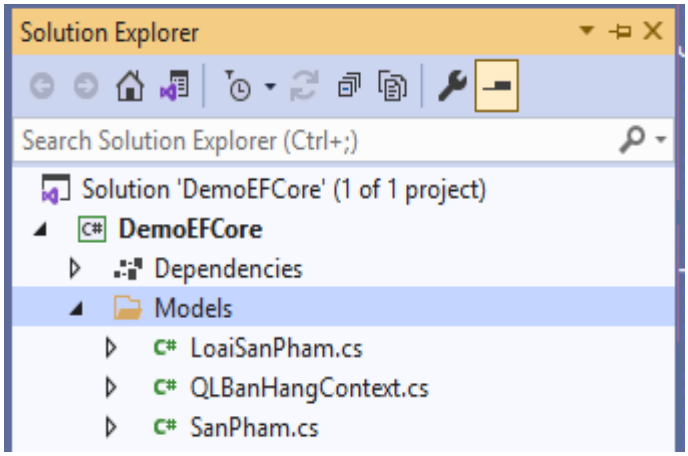
Trong cửa sổ **Package Manager Console** thực hiện lệnh **Scaffold-DbContext** như sau:

Scaffold-DbContext "chuỗi-kết-nối" Microsoft.EntityFrameworkCore.SqlServer –OutputDir tên-thư-mục

Ví dụ:

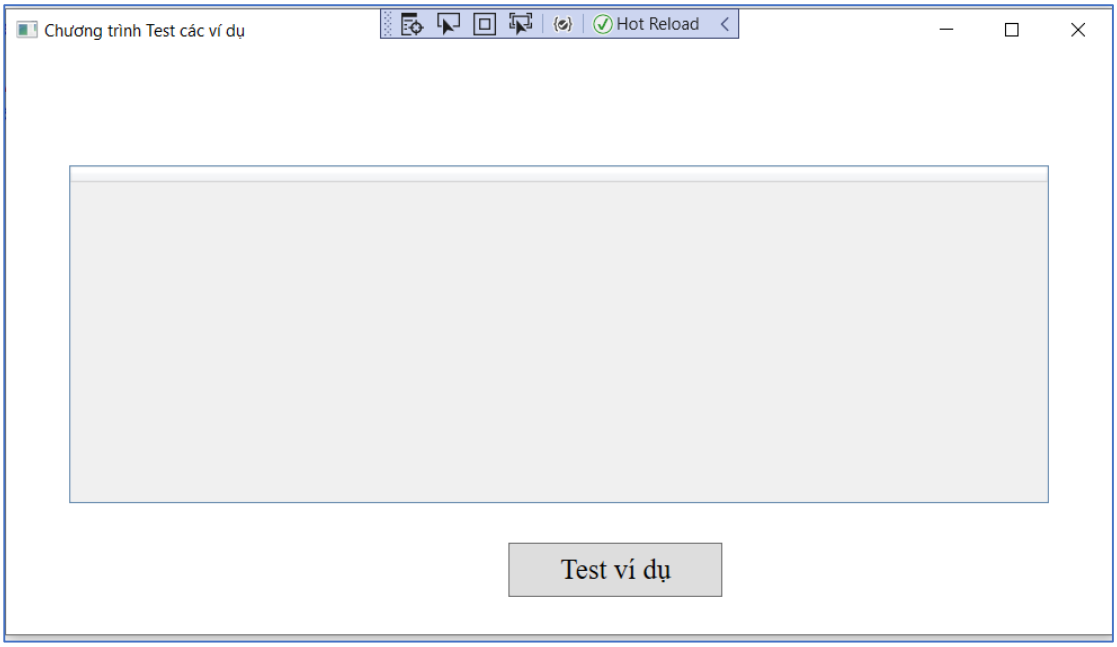
```
Scaffold-DbContext "Data Source= DESKTOP-MTPECJO\MAYAO;Initial
Catalog=QLBanHang;Integrated Security=True"
Microsoft.EntityFrameworkCore.SqlServer –OutputDir Models
```

Lệnh **Scaffold-DbContext** tạo các lớp thực thể cho mỗi bảng trong cơ sở dữ liệu QLBanHang và lớp Context với cấu hình cho tất cả các thực thể trong thư mục Models



4.2.3. Truy vấn dữ liệu sử dụng LINQ

Thiết kế giao diện:



- **Ví dụ 1:** Hiển thị dữ liệu bảng SanPham

```
using DemoEFCore.Models;
namespace DemoEFCore
{
    public partial class Test_Vidu : Window
    {
        public Test_Vidu()
        {
            InitializeComponent();
        }
        //Tạo thể hiện của lớp Context
        QLBanHangContext db = new QLBanHangContext();

        private void btnTest_Click(object sender, RoutedEventArgs e)
        {
              //Cách 1: Truy vấn LINQ theo biểu thức
                        var sp = from p in db.SanPhams
                                 select p;
                    //Hiển thị dữ liệu lên data grid
                        dgvLoaiSanPham.ItemsSource = sp.ToList();

                /* Cách 2: Truy vấn LINQ theo phương thức mở rộng
                        var sp = db.SanPhams
                                   .Select(x => x);
                       dgvLoaiSanPham.ItemsSource = sp.ToList(); */
        }
    }
}
```

❖ **SELECT**

- **Ví dụ 2:** Hiển thị DL bảng SanPham (Lựa chọn cột hiển thị)

```
+ Cách 1:   var sp = from p in db.SanPhams
                     select new { p.MaSp, p.TenSp, p.MaLoai};
+ Cách 2:  var sp = db.SanPhams
                     .Select(p =>new {p.MaSp, p.TenSp, p.MaLoai });
```

❖ **TAKE, SKIP**

- **Ví dụ 3:** Lấy mẫu tin đầu tiên trong bảng (Lấy sản phẩm đầu tiên trong bảng SanPham).

```
+ Cách 1:   var sp = (from p in db.SanPhams
                     select p).Skip(0).Take(1); //Tương đương SELECT TOP 1 trong SQL
+ Cách 2:  var sp = db.SanPhams
                .Select(x => x).Skip(0).Take(1);
```

- **Ví dụ 4:** Tương tự trên, nhưng lấy mẫu tin cuối cùng (kết hợp 2 cách)

```
var sp = (from p in db.SanPhams
          orderby p.MaSp descending
          select p).Skip(0).Take(1);
```

- **Ví dụ 5:** Tương tự trên, nhưng lấy mẫu tin thứ 5 và 6

```
var sp = (from p in db.SanPhams
          select p).Skip(4).Take(2);
```

❖ **ORDERBY**

- **Ví dụ 6:** Sắp xếp giảm dần theo cột đơn giá (mặc định tăng dần)

```
+ Cách 1:  var sp = from p in db.SanPhams
           orderby p.DonGia descending
           select new { p.MaSp, p.TenSp, p.DonGia };
+ Cách 2:  var sp = db.SanPhams
                    .OrderByDescending(x=>x.DonGia)
                    .Select(x=>new {x.MaSp, x.TenSp, x.DonGia });
```

❖ **DISTINCT**

- **Ví dụ 7:** Loại bỏ các phần tử trùng nhau

```
var sp = (from p in db.SanPhams
          select new { p.MaSp, p.TenSp, p.MaLoai}). Distinct();
```

❖ **WHERE**

- **Ví dụ 8:** Lấy điều kiện theo MaLoai

```
+ Cách 1:   var sp = from p in db.SanPhams
                     where p.MaLoai=="L01"
                     select new { p.MaSP, p.TenSP, p.MaLoai };
+ Cách 2:  var sp = db.SanPhams
                    .Where (p=>p.MaLoai=="L01")
                    .Select(x => new { x.MaSP, x.TenSP, x.DonGia });
```

❖ **JOIN**

+ Cách 1:

```
var sp = from p in db.SanPhams
         join k in db.LoaiSanPhams
         on p.MaLoai equals k.MaLoai
         select new { p.MaSp, p.TenSp, k.TenLoai };
```

+ Cách 1:

```
var sp = db.SanPhams
    .Join(db.LoaiSanPhams,p=>p.MaLoai,k=>k.MaLoai,(p,k)=>new{p.MaSp,p.TenSp,k.TenLoai})
    .Select(sp => sp);
```

❖ **GROUP**

• **Ví dụ 10**: Tính tổng tiền từng loại sản phẩm gồm: Mã loại, tổng tiền.

```
var query = from sp in db.SanPhams
            group sp by sp.MaLoai into LspGroup
            select new
            {
                MaLoai = LspGroup.Key,
                TongTien = LspGroup.Sum(t => t.DonGia * t.SoLuong)
            };
```

Kết quả:

MaLoai	TongTien
l1	1000000121
l2	340000000

• **Ví dụ 11**: Tính tổng tiền từng loại sản phẩm gồm: Mã loại, tên loại, tổng tiền.

```
var query1 = from sp in db.SanPhams
        group sp by sp.MaLoai into LspGroup
        select new
        {
            MaLoai = LspGroup.Key,
            TongTien = LspGroup.Sum(t => t.DonGia * t.SoLuong)
        };
var query2 = from t in query1
          join lsp in db.LoaiSanPhams on t.MaLoai equals lsp.MaLoai
          select new
          {
              lsp.MaLoai,
              lsp.TenLoai,
              t.TongTien
          };
```

Kết quả:

MaLoai	TenLoai	TongTien
l1	Điện tử	1000000121
l2	Quần áo	340000000