

## BÀI 10. ENTITY FRAMEWORK TRONG ASP.NET MVC

### PHẦN 2

- Mục đích: Cung cấp cho sinh viên kiến thức về ngôn ngữ truy vấn tích hợp LINQ.
- Yêu cầu: Sinh viên viết được các câu lệnh LINQ để truy vấn dữ liệu.
- Hình thức tổ chức dạy học: Lý thuyết, tự học
- Thời gian: Lý thuyết( trên lớp: 0; online: 3) Tự học, tự nghiên cứu: 6

**- Nội dung chính:**

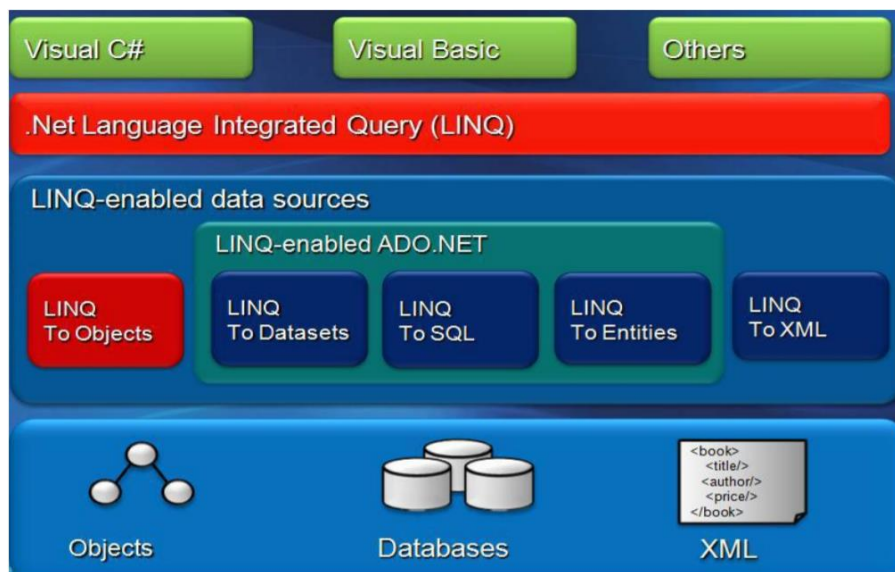
1. Khái niệm LINQ.....	2
1.1. LINQ là gì?.....	2
1.2. Kiến trúc của LINQ .....	2
2. Thành phần cấu thành LINQ.....	2
2.1. Nguồn dữ liệu (Data source): .....	4
2.2. Câu truy vấn (Query).....	4
2.3. Thực thi truy vấn (Query Execution).....	5
2.4. Các ví dụ .....	6
3. Cú pháp truy vấn.....	8
3.1. Truy vấn cơ bản .....	8
3.2. Truy vấn phân trang .....	8
3.3. Truy vấn một phần tử.....	9
3.4. Tổng hợp số liệu .....	9
3.5. Kiểm tra phần tử trong tập .....	9
3.6. Các ví dụ: .....	10
4. Hướng dẫn thực hành.....	11
4.1. Hiển thị dữ liệu .....	12
4.2. Sắp xếp .....	12
4.3. Lọc.....	14
4.4. Phân trang đơn giản.....	15
4.5. Phân trang có cả sắp xếp và tìm kiếm .....	16

## 1. Khái niệm LINQ

### 1.1. LINQ là gì?

- LINQ - Language Integrated Query (ngôn ngữ truy vấn tích hợp) - đưa ra một mô hình bền vững để hoạt động với các dạng nguồn dữ liệu và định dạng dữ liệu khác nhau.
- LINQ cho phép dùng các đoạn code đơn giản để truy vấn và chuyển đổi dữ liệu trong
  - Các tài liệu XML
  - Cơ sở dữ liệu SQL
  - Tập dữ liệu ADO.NET
  - Các tập hợp .NET
  - Và bất kỳ định dạng nào mà LINQ provider hỗ trợ.
- LINQ ra đời và được thêm vào phiên bản .NET 3.5

### 1.2. Kiến trúc của LINQ



## 2. Thành phần cấu thành LINQ

Ví dụ:

```
List<int> alist = new List<int>();
// Ba thành phần của 1 truy vấn LINQ:
// 1. Nguồn dữ liệu
int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

// 2. Tạo truy vấn
// numQuery là 1 IEnumerable<int>
var numQuery =
```

```

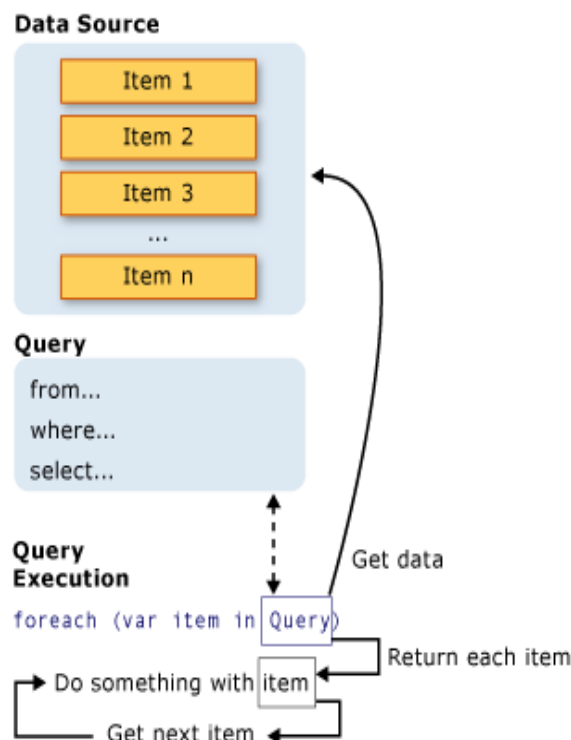
from num in numbers
where (num % 2) == 0
select num;

// 3. Truy vấn dữ liệu
foreach (int num in numQuery)
{
    alist.Add(num);
}

```

- Nguồn dữ liệu trong ví dụ là mảng số nguyên tên là numbers gồm 7 phần tử có giá trị là 0, 1, 2, 3, 4, 5, 6.
- Câu truy vấn được tạo bằng cách sử dụng 1 diễn giải Lambda (lambda expression), gần giống như ngôn ngữ truy vấn SQL. Trong trường hợp này yêu cầu tạo 1 biến numQuery không xác định kiểu dữ liệu. Câu “from num in numbers where (num % 2) == 0 select num;” yêu cầu chọn tất cả các số trong mảng numbers chia hết cho 2. Sau khi khai báo diễn giải Lambda cho biến numQuery thì có thể hiểu numQuery là 1 danh sách các số chẵn (chia hết cho 2), vì vậy kiểu dữ liệu ngầm của numQuery chính là IEnumerable. IEnumerable là 1 lớp giao diện cơ bản cho tất cả các tập hợp cho thể được liệt kê. Cách dùng phổ biến của IEnumerable thường là System.Collections.Generic.IEnumerable<T> với T là kiểu dữ liệu của mỗi đối tượng trong danh sách.
- Vì biến numQuery dùng diễn giải Lambda cho nên kiểu dữ liệu của nó chính là 1 danh sách đối tượng các giá trị chia hết cho 2. Do đó có thể dùng 1 vòng foreach để truy vấn các giá trị phần tử này và in ra màn hình.

Có thể hình dung bằng hình ảnh như sau:

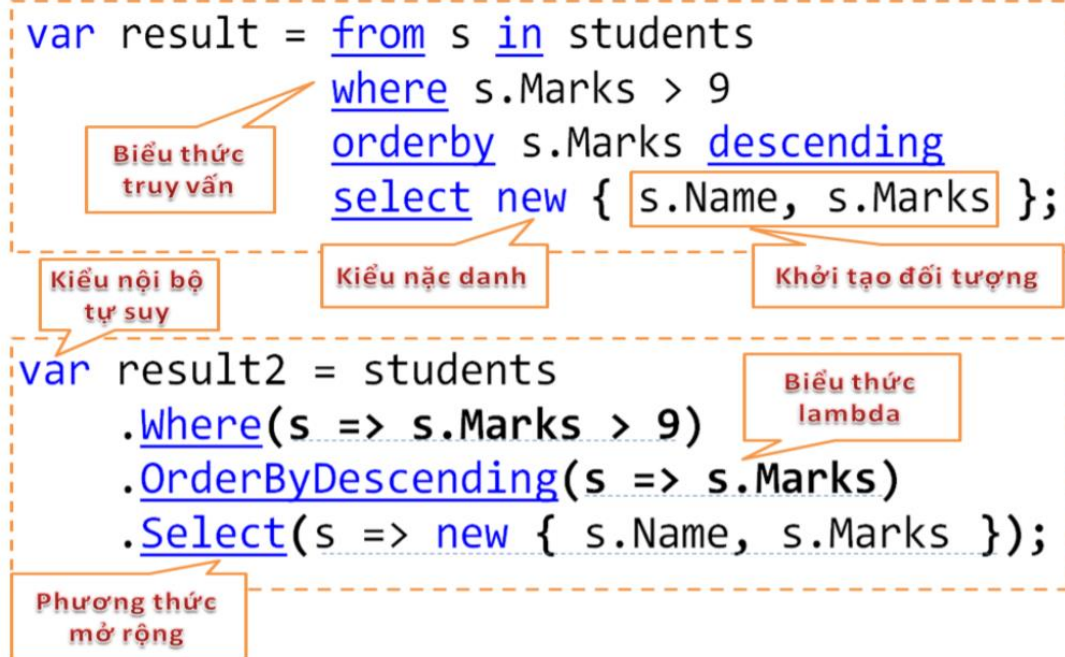


## 2.1. Nguồn dữ liệu (Data source):

Tên provider	Mô tả
LINQ to Objects	Sử dụng LINQ đối với các đối tượng collection mà implement từ IEnumerable hoặc IEnumerable<T> (dữ liệu được lưu trong bộ nhớ). Được sử dụng rộng rãi đặc biệt với những bài toán cần hiệu năng cao.
LINQ to SQL	Thực hiện map các tables, views, store procedures thành các đối tượng. LINQ sẽ thực hiện truy vấn trên các đối tượng đó bằng cách chuyển đổi qua lại giữa đối tượng và câu lệnh sql. Ngoài truy vấn có thể thêm/sửa/xóa dữ liệu dựa vào các đối tượng trên. Hỗ trợ transaction. Ưu điểm: được sử dụng khá nhiều trong thực tế dưới cái tên Entity Framework hoặc Entity Framework Core. Nhược điểm: chỉ làm việc với cơ sở dữ liệu là SQL Server.
LINQ to Entities	Tương tự như LINQ to SQL nhưng hỗ trợ nhiều loại cơ sở dữ liệu. Nhược điểm: sử dụng phức tạp. Nhiều cơ sở dữ liệu không thích hợp sử dụng chung với .Net.
LINQ to DataSets	Sự kết hợp giữa LINQ và ADO.NET.
LINQ to XML	Truy vấn thông tin trong file XML.

## 2.2. Câu truy vấn (Query)

- **Query (câu truy vấn):** Câu truy vấn mô tả cách thông tin được rút trích từ nguồn dữ liệu hay các nguồn. Câu query cũng có thể mô tả cách thông tin được sắp xếp, gom nhóm và thay đổi trước khi trả về. Một câu truy vấn được chứa trong 1 biến truy vấn và được bắt đầu bằng 1 diễn giải truy vấn (query expression).



- Có 2 cách để tạo ra câu truy vấn:
  - Biểu thức truy vấn (query syntax)

```
var result = from s in students
             where s.Marks > 9
             orderby s.Marks descending
             select new { s.Name, s.Marks };
```

- Phương thức mở rộng (method syntax).

```
var result2 = students
    .Where(s => s.Marks > 9)
    .OrderByDescending(s => s.Marks)
    .Select(s => new { s.Name, s.Marks });
```

- Để viết được câu truy vấn cần using thư viện System.Linq.

### 2.3. Thực thi truy vấn (Query Execution)

- **Thực thi trì hoãn (Deferred Execution):** Biến truy vấn chỉ lưu trữ chính các lệnh truy vấn. Các thực thi thực sự của truy vấn được trì hoãn cho đến khi lặp qua các biến truy vấn trong mệnh đề foreach. Ví dụ:

```
// numQuery là 1 IEnumerable<int>
var numQuery = from num in numbers
               where (num % 2) == 0
               select num;

// 3. Truy vấn dữ liệu
foreach (int num in numQuery)
```

```
{
    alist.Add(num);
}
```

Mệnh đề `foreach` cũng là nơi kết quả truy vấn được rút trích. Trong ví dụ, biến `num` giữ giá trị mỗi phần tử (tại từng thời điểm) trong chuỗi trả về. Bởi vì biến truy vấn không giữ các kết quả truy vấn, nên có thể thực thi chúng nhiều lần nếu muốn.

- **Thực thi ngay lập tức (Immediate Execution):** Các truy vấn thực thi các hàm kết hợp trên 1 dãy các yếu tố nguồn phải lập trước trên từng yếu tố đó.
  - Các truy vấn như **Count, Max, Average và First**. Những thực thi này không cần mệnh đề `foreach` minh bạch vì truy vấn chính nó phải dùng `foreach` để trả về kết quả. Lưu ý các dạng truy vấn trả về 1 biến đơn, không phải 1 tập hợp `IEnumerable`. Ví dụ: Truy vấn sau trả về biến đếm tổng số các số trong 1 mảng nguồn.

```
var evenNumQuery = from num in numbers
                   where (num % 2) == 0
                   select num;

// Đếm tổng các số chia hết cho 2 được
//trả về biến không xác định kiểu
int evenNumCount = evenNumQuery.Count();
```

- Để thực thi ngay lập tức bất kỳ truy vấn nào và lấy dữ liệu của nó, có thể dùng 2 phương thức `ToList<TSource>` hoặc `ToArray<TSource>`. Ví dụ:

```
List<int> numQuery2 = (from num in numbers
                     where (num % 2) == 0
                     select num).ToList();

// hoặc như cách
// numQuery3 vẫn là 1 mảng int[]
var numQuery3 = (from num in numbers
                 where (num % 2) == 0
                 select num).ToArray();
```

## 2.4. Các ví dụ

- Ví dụ 1: Biểu thức truy vấn

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers
            where n % 2 == 0
            select n;
```

```
foreach(int n in numbers){
    if(n % 2 == 0){
        tích lũy số chẵn
    }
}
```

- ❑ **from**: chỉ ra phần tử được lấy từ tập hợp cần truy vấn
- ❑ **where**: chỉ ra điều kiện lọc
- ❑ **select**: chỉ ra đối tượng nhận được

- Ví dụ 2: Tạo đối tượng mới

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers
            where n % 2 == 0
            let rate = n / numbers.Sum()
            orderby n descending
            select new { number = n, rate = rate };
```

Đối tượng

```
foreach (var e in evens)
{
    int n = e.N
}
```

N	
Equals	
GetHashCode	
GetType	
<b>number</b>	int 'a.number
rate	
ToString	

Anonymous Types:  
'a' is new { int number, int rate }

- Ví dụ 3: Tổng hợp – Thống kê

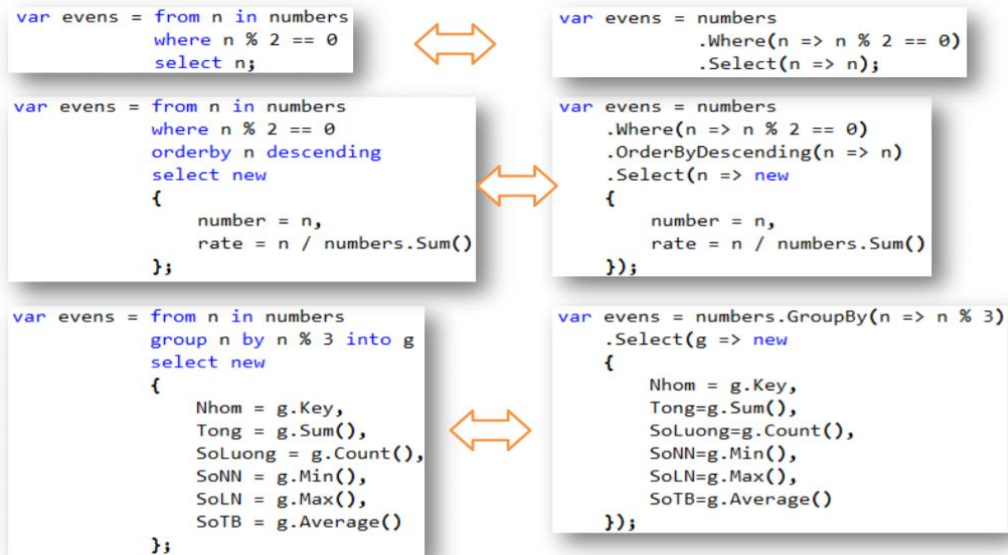
```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers
            group n by n % 3 into g
            select new
            {
                Nhom = g.Key,
                Tong = g.Sum(),
                SoLuong = g.Count(),
                SoNN = g.Min(),
                SoLN = g.Max(),
                SoTB = g.Average()
            };
```

- ❑ Nhóm chia 3 dư 0: gồm 6, 45, 87
- ❑ Nhóm chia 3 dư 1: gồm 19, 13
- ❑ Nhóm chia 3 dư 2: gồm 23, 56, 5, 8

- Ví dụ 4: Phương thức mở rộng





### 3. Cú pháp truy vấn

#### 3.1. 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()

Ví dụ: var studs = Students

```

    .Where(s=>s.Marks>9)
    .OrderBy(s=>s.Marks)
    .Select(s=>s)
  
```

#### 3.2. 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}
.Distinct()	Chọn các thành phần khác nhau	Numbers.Distinct()

```

Ví dụ: var result = db.Products
        .Skip(10).Take(20)

```

### 3.3. Truy vấn một 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()

```
Ví dụ: var result = db.Customers
        .Single(c=>c.Id=="A"&& c.Password=="B")
```

### 3.4. Tổng hợp số liệu

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)

```
Ví dụ: var result = db.Products
        .GroupBy(p=>p.Category)
        .select(g=>new{g.Key.Name,g.Count})
```

### 3.5. 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)

Ví dụ:

```
int[] numbers = { 18, 12, 3, 4, 34, 19, 65, 17 };
if (numbers.All(n => n % 2 == 0))
{
    //Tất cả các số trong numbers đều là số chẵn
}
if (numbers.Any(n => n % 3 == 0))
{
    //Ít nhất một số trong numbers chia hết cho 3
}
if (numbers.Contains(8))
{
    //Tập numbers có chứa số 8
}
```

### 3.6. Các ví dụ:

- Ví dụ 4. Thống kê doanh số

```
var items7 = db.Products.GroupBy(p => p.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại
        Sum = g.Sum(p=>p.UnitPrice), //--tổng đơn giá hàng hóa của loại
        Count = g.Count(), //--số hàng hóa của loại
        Min = g.Min(p => p.UnitPrice), //--giá hàng hóa thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá hàng hóa cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });

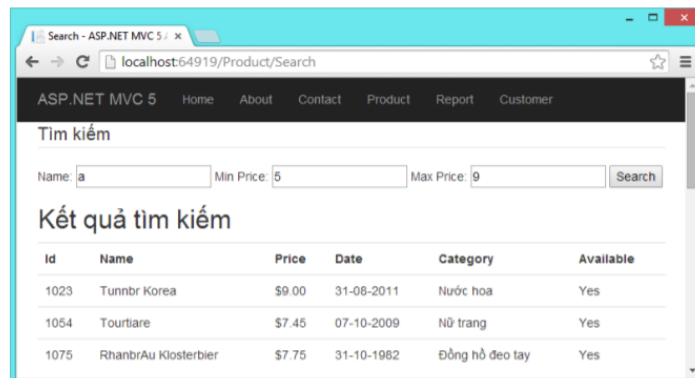
var items8 = db.OrderDetails.GroupBy(d=>d.Product)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên hàng hóa
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị đã bán
        Count = g.Sum(p => p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items9 = db.OrderDetails.GroupBy(d => d.Product.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items10 = db.OrderDetails.GroupBy(d => d.Order.Customer)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Fullname, //--họ và tên khách hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã mua
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã mua
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items11 = db.OrderDetails.GroupBy(d => d.Order.OrderDate.Month)
    .Select(g => new ReportInfo
    {
        Group = g.Key, //--tháng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

#### ○ Ví dụ 5. Lọc dữ liệu



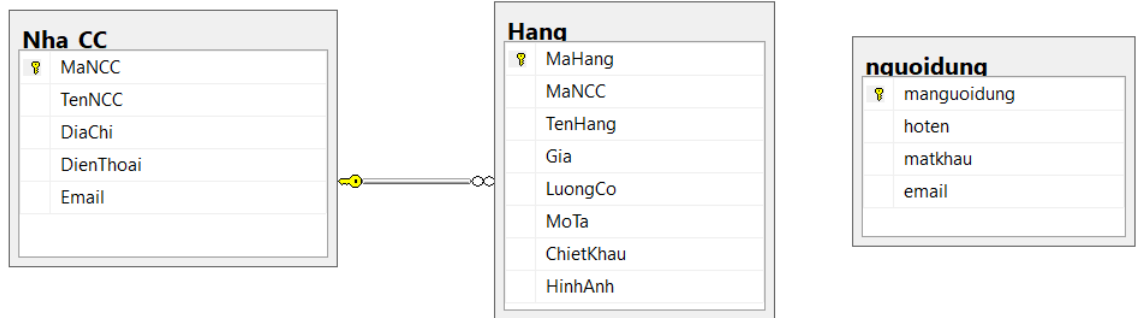
```
public ActionResult Search(String Name = "",
    double Min = double.MinValue, double Max = double.MaxValue)
{
    var list = db.Products
        .Where(p => p.Name.Contains(Name) && p.UnitPrice >= Min
            && p.UnitPrice <= Max).ToList();
    return View(list);
}
```

## 4. Hướng dẫn thực hành

Sử dụng LINQ để hiển thị dữ liệu có sắp xếp, lọc và phân trang

#### 4.1. Hiện thị dữ liệu

- Trong folder HoaQua được cung cấp, chạy file script **fShopDB.sql** trong SQLServer để tạo cơ sở dữ liệu **fShopDB**



- Tạo một project đặt tên là BaiTap12, chọn mẫu **MVC**.
- Cài đặt EntityFramework sử dụng NuGet Package Manager
- Sử dụng EF code first để kết nối với cơ sở dữ liệu **fShopDB** (đặt tên ADO.NET Entity Model là **fShopDB**) chọn tất cả các bảng trong cơ sở dữ liệu.
- Kích **ReBuild Solution** để build lại project (mỗi khi sửa model cần Rebuild lại hệ thống)
- Tạo một controller theo mẫu **MVC 5 Controller with view, using Entity Framework** gắn với model Hang.
- Sửa phần ActionLink trong \_Layout.cshtml thành

```



    ...
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Xem hàng", "Index", "Hangs")</li>
    </ul>
    ...
  
```

#### 4.2. Sắp xếp

- Sửa code trang Index.cshtml của controller Hangs:
  - Bỏ các trường MaNCC, LuongCo, MoTa, ChietKhau.
  - Sửa phần tiêu đề cột <th>...</th> như sau:

```
<table class="table">
  <tr>
    <th>
      Mã hàng
    </th>
    <th>
      Tên hàng
    </th>
    <th>
      Giá
    </th>
    <th>
      Hình ảnh
    </th>
  </tr>
  <tr>
    <td>RYL01</td>
    <td>Bơ Sáp Đắc Lắc</td>
    <td>75000.00</td>
    <td>
      <img alt="Bơ Sáp Đắc Lắc" data-bbox="588 385 650 410"/>
      bo_sap.jpg
    </td>
  </tr>
  <tr>
    <td>RYL02</td>
    <td>Bưởi da xanh</td>
    <td>98000.00</td>
    <td>
      <img alt="Bưởi da xanh" data-bbox="588 425 650 450"/>
      buoi_da_xanh_2.jpg
    </td>
  </tr>
</table>
```

- Xóa nút Create, sửa phần ActionLink của Edit, Details và Delete để khi chạy hiển thị như sau:

Application name    Home    Xem hàng				
Danh sách hàng				
Mã hàng	Tên hàng	Giá	Hình ảnh	
RYL01	Bơ Sáp Đắc Lắc	75000.00	 bo_sap.jpg	<a href="#">Chi tiết</a>   <a href="#">Thêm vào giỏ</a>
RYL02	Bưởi da xanh	98000.00	 buoi_da_xanh_2.jpg	<a href="#">Chi tiết</a>   <a href="#">Thêm vào giỏ</a>

- Sửa action method Index() trong HangsController như sau.

```
public ActionResult Index()
{
    //Lấy danh sách hàng
    var hang = db.Hangs.Select(p=>p);
    return View(hang.ToList());
}
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- Sửa action method Index() trong HangsController để sắp xếp danh sách hàng theo TenHang và Gia như sau:

```
public ActionResult Index(string sortOrder)
{
    //Các biến sắp xếp
    ViewBag.SapTheoTen = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    ViewBag.SapTheoGia = sortOrder == "Gia" ? "gia_desc" : "Gia";
    //Lấy danh sách hàng
    var hang = db.Hangs.Select(p=>p);
    //Sắp xếp
    switch (sortOrder)
    {
        case "name_desc":
            hang = hang.OrderByDescending(s => s.TenHang);
            break;
        case "Gia":
            hang = hang.OrderBy(s => s.Gia);
            break;
        case "gia_desc":
            hang = hang.OrderByDescending(s => s.Gia);
            break;
        default:
            hang = hang.OrderBy(s => s.TenHang);
            break;
    }
    return View(hang.ToList());
}
```

- Sửa code trang Index.cshtml của controller Hangs phân tiêu đề cột

<th>...</th> như sau:

```
<table class="table">
  <tr>
    <th>
      Mã hàng
    </th>
    <th>
      @Html.ActionLink("Tên hàng", "Index", new { sortOrder = ViewBag.SapTheoTen })
    </th>
    <th>
      @Html.ActionLink("Giá", "Index", new { sortOrder = ViewBag.SapTheoGia })
    </th>
    <th>
      Hình ảnh
    </th>
  </tr>
  <tr>
    <td>
      ...
    </td>
  </tr>
</table>
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.

### 4.3. Lọc

- Sửa action method Index() trong HangsController để cho phép tìm kiếm hàng theo tên như sau

```
public ActionResult Index(string sortOrder, string searchString)
{
    //Các biến sắp xếp
    ViewBag.SapTheoTen = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    ViewBag.SapTheoGia = sortOrder == "Gia" ? "gia_desc" : "Gia";
    //Lấy danh sách hàng
    var hang = db.Hangs.Select(p=>p);

    //Lọc theo tên hàng
    if (!String.IsNullOrEmpty(searchString))
    {
        hang = hang.Where(p => p.TenHang.Contains(searchString));
    }

    //Sắp xếp
    switch (sortOrder)
    {
        case "name_desc":
            hang = hang.OrderByDescending(s => s.TenHang);
            break;
        case "Gia":
            hang = hang.OrderBy(s => s.Gia);
            break;
        case "gia_desc":
            hang = hang.OrderByDescending(s => s.Gia);
            break;
        default:
            hang = hang.OrderBy(s => s.TenHang);
            break;
    }

    return View(hang.ToList());
}
```

- Sửa code trang Index.cshtml của controller Hangs thêm textbox tìm kiếm như sau:

```
<h2>Danh sách hàng</h2>
@using (Html.BeginForm())
{
    <p>
        Tìm một hàng: @Html.TextBox("SearchString")
        <input type="submit" value="Tìm" />
    </p>
}
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.

#### 4.4. Phân trang đơn giản

- Thêm ActionLink sau vào menu trong \_Layout: `<li>@Html.ActionLink("Xem nhà cung cấp", "DisplaySupplier", "Home")</li>`
- Sử dụng Nuget cài gói `PagedList.Mvc`
- Tạo một view tên là `DisplaySupplier` theo Template List của model `Nha_CC`.
- Sửa HomeController thêm một action method là `DisplaySupplier`:

```
public class HomeController : Controller
{
    private FShopDB = new FShopDB();
    public ActionResult DisplaySupplier()
    {
        var supplies = db.Nha_CC.Select(s => s);
        return View(supplies.ToList());
    }
}
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- Trong HomeController:
  - Thêm khai báo namespaces

```
using PagedList;
```

- Sửa action method `DisplaySupplier()`

```
public ActionResult DisplaySupplier(int? page)
{
    var supplies = db.Nha_CC.Select(s => s);

    //Cần sắp xếp trước khi phân trang
    supplies = supplies.OrderBy(s => s.MaNCC);

    int pageSize = 3; //Kích thước trang
    int pageNumber = (page ?? 1); //Nếu page bằng null thì trả về 1

    return View(supplies.ToPagedList(pageNumber, pageSize));
}
```

- `DisplaySupplier.cshtml` của controller Home trong folder Home:
  - Sửa khai báo

```
@model IEnumerable<BaiTap12.Models.Nha_CC>
```

Thành

```
@model PagedList.IPagedList<BaiTap12.Models.Nha_CC>
@using PagedList.Mvc;
<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />
```

- Thêm đoạn code chuyển trang sau vào cuối trang

```
<br />
Trang @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) / @Model.PageCount
@Html.PagedListPager(Model, page => Url.Action("DisplaySupplier", new { page }))
```



- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.

#### 4.5. Phân trang có cả sắp xếp và tìm kiếm

- Trong HangsController:
  - Thêm khai báo namespaces

```
using PagedList;
```

- Sửa action method Index()

```
public ActionResult Index(string sortOrder, string searchString, string currentFilter, int?
page)
{
    //Các biến sắp xếp
    ViewBag.CurrentSort = sortOrder; //Biến lấy yêu cầu sắp xếp hiện tại

    ViewBag.SapTheoTen = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
    ViewBag.SapTheoGia = sortOrder == "Gia" ? "gia_desc" : "Gia";

    //Lấy giá trị của bộ lọc dữ liệu hiện tại
    if (searchString != null)
    {
        page = 1; //Trang đầu tiên
    }
    else
    {
        searchString = currentFilter;
    }
    ViewBag.CurrentFilter = searchString;
```

```
    //Lấy danh sách hàng
    var hangs = db.Hangs.Select(p => p);
    //Lọc theo tên hàng
    if (!String.IsNullOrEmpty(searchString))
    {
        hangs = hangs.Where(p => p.TenHang.Contains(searchString));
    }

    //Sắp xếp
    switch (sortOrder)
    {
        case "name_desc":
            hangs = hangs.OrderByDescending(s => s.TenHang);
            break;
        case "Gia":
            hangs = hangs.OrderBy(s => s.Gia);
            break;
        case "gia_desc":
            hangs = hangs.OrderByDescending(s => s.Gia);
            break;
        default:
            hangs = hangs.OrderBy(s => s.TenHang);
            break;
    }

    int pageSize = 3; //Kích thước trang
    int pageNumber = (page ?? 1);

    return View(hangs.ToPagedList(pageNumber, pageSize));
}
```

- Trong trang Index.cshtml của controller Hangs
  - Sửa khai báo

```
@model IEnumerable<BaiTap12.Models.Hang>
```

## Thành

```
@model PagedList.IPagedList<BaiTap12.Model.Hang>
@using PagedList.Mvc;
<link href="~/Content/PagedList.css" rel="stylesheet" type="text/css" />
```

### ○ Sửa

```
@using (Html.BeginForm())
{
    <p>
        Tìm mặt hàng: @Html.TextBox("SearchString")
        <input type="submit" value="Tìm" />
    </p>
}
```

## Thành

```
@using (Html.BeginForm())
{
    <p>
        Tìm mặt hàng: @Html.TextBox("SearchString", ViewBag.CurrentFilter as string)
        <input type="submit" value="Tìm" />
    </p>
}
```

### ○ Sửa

```
<th>
    @Html.ActionLink("Tên hàng", "Index", new { sortOrder = ViewBag.SapTheoTen })
</th>
<th>
    @Html.ActionLink("Giá", "Index", new { sortOrder = ViewBag.SapTheoGia })
</th>
```

## Thành

```
<th>
    @Html.ActionLink("Tên hàng", "Index", new { sortOrder = ViewBag.SapTheoTen,
        currentFilter = ViewBag.CurrentFilter })
</th>
<th>
    @Html.ActionLink("Giá", "Index", new { sortOrder = ViewBag.SapTheoGia,
        currentFilter = ViewBag.CurrentFilter })
</th>
```

### ○ Thêm đoạn code chuyển trang sau vào cuối trang

```
</table>
<br />
Trang @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) / @Model.PageCount
@Html.PagedListPager(Model, page => Url.Action("Index",
    new { page, sortOrder = ViewBag.CurrentSort, currentFilter = ViewBag.CurrentFilter
    }))
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử

**Yêu cầu sinh viên chuẩn bị:**

Đọc slides xem hướng dẫn. Xem đề cương chi tiết bài giảng, xem video.