Hướng Dẫn Sử Dụng LINQ Trong ASP.NET Để Truy Vấn Cơ Sở Dữ Liệu

XAI

July 19, 2025

Mục lục

1 Giới thiệu về LINQ trong ASP.NET

LINQ (Language Integrated Query) là một tính năng mạnh mẽ của C# trong .NET Framework, cho phép thực hiện các truy vấn dữ liệu tích hợp trực tiếp trong mã nguồn. Trong ASP.NET, LINQ thường được sử dụng với Entity Framework (LINQ to Entities) để truy vấn cơ sở dữ liệu một cách hiệu quả, an toàn kiểu dữ liệu (type-safe) và dễ đọc.

Bài hướng dẫn này sẽ trình bày các ví dụ từ cơ bản đến nâng cao về cách sử dụng LINQ để truy vấn cơ sở dữ liệu, kèm giải thích chi tiết về cú pháp và ứng dụng thực tế.

2 Cơ bản về LINQ to Entities

Giả sử chúng ta có một cơ sở dữ liệu với các bảng `Products`, `Categories`, và `Orders`, với cấu trúc như sau:

- `Products`: `ProductID`, `ProductName`, `CategoryID`, `Price`
- `Categories`: `CategoryID`, `CategoryName`
- `Orders`: `OrderID`, `ProductID`, `Quantity`, `OrderDate`

Dưới đây là các ví dụ sử dụng LINQ to Entities trong ASP.NET với Entity Framework.

2.1 Truy vấn cơ bản: Lấy tất cả sản phẩm

```
using (var context = new MyDbContext())
{
    var products = from p in context.Products
                   select p;
    foreach (var product in products)
    {
        Console.WriteLine($"Product: {product.ProductName}, Price: {
   product.Price}");
    }
}
```

Giải thích:

- `from p in context.Products`: Chỉ định nguồn dữ liệu (bảng `Products`).
- `select p`: Chọn toàn bộ đối tượng `Product`.
- Kết quả trả về một tập hợp các sản phẩm, có thể duyệt qua bằng vòng lặp.

2.2 Lọc dữ liệu: Lấy sản phẩm có giá lớn hơn 100

```
using (var context = new MyDbContext())
{
```

```
    var expensiveProducts = from p in context.Products
                            where p.Price > 100
                            select p;
    foreach (var product in expensiveProducts)
    {
        Console.WriteLine($"Product: {product.ProductName}, Price: {product.Price}");
    }
}
```

Giải thích:

- `where p.Price > 100`: Lọc các sản phẩm có giá lớn hơn 100.
- LINQ tự động chuyển câu truy vấn này thành SQL để thực thi trên cơ sở dữ liệu.

3 Truy vấn trung cấp

3.1 Kết hợp bảng (Join): Lấy sản phẩm và danh mục

```
using (var context = new MyDbContext())
{
    var productCategories = from p in context.Products
                            join c in context.Categories
                            on p.CategoryID equals c.CategoryID
                            select new { p.ProductName, c.CategoryName };
    foreach (var item in productCategories)
    {
        Console.WriteLine($"Product: {item.ProductName}, Category: {item.CategoryName}");
    }
}
```

Giải thích:

- `join ... on ... equals`: Kết hợp bảng `Products` và `Categories` dựa trên `CategoryID`.
- `select new`: Tạo một đối tượng vô danh (anonymous type) chỉ chứa các thuộc tính cần thiết.

3.2 Nhóm dữ liệu (Group By): Thống kê số lượng sản phẩm theo danh mục

```
using (var context = new MyDbContext())
{
    var productCountByCategory = from p in context.Products
                                 join c in context.Categories
                                 on p.CategoryID equals c.CategoryID
```

```
                                    group p by c.CategoryName into g
                                    select new { CategoryName = g.Key,
    ProductCount = g.Count() };
     foreach (var group in productCountByCategory)
     {
         Console.WriteLine($"Category: {group.CategoryName}, Count: {
    group.ProductCount}");
     }
}
```

Giải thích:

- `group p by c.CategoryName into g`: Nhóm các sản phẩm theo tên danh mục.
- `g.Key`: Là giá trị của `CategoryName` được dùng để nhóm.
- `g.Count()`: Đếm số lượng sản phẩm trong mỗi nhóm.

4 Truy vấn nâng cao

4.1 Sắp xếp và phân trang: Lấy 5 sản phẩm đắt nhất

```
using (var context = new MyDbContext())
{
    var topExpensiveProducts = (from p in context.Products
                                orderby p.Price descending
                                select p)
                                .Take(5);
    foreach (var product in topExpensiveProducts)
    {
        Console.WriteLine($"Product: {product.ProductName}, Price: {
    product.Price}");
    }
}
```

Giải thích:

- `orderby p.Price descending`: Sắp xếp sản phẩm theo giá giảm dần.
- `Take(5)`: Chỉ lấy 5 bản ghi đầu tiên, phù hợp cho phân trang.

4.2 Truy vấn lồng nhau: Lấy danh mục có sản phẩm giá trên 500

```
using (var context = new MyDbContext())
{
    var categoriesWithExpensiveProducts = from c in context.
    Categories
                                          where (from p in context.
    Products
```

```
                                                    where p.CategoryID
    == c.CategoryID
                                                    select p.Price).Any(
    price => price > 500)
                                            select c;
     foreach (var category in categoriesWithExpensiveProducts)
     {
         Console.WriteLine($"Category: {category.CategoryName}");
     }
}
```

Giải thích:

- Truy vấn lồng: Truy vấn con kiểm tra xem có sản phẩm nào trong danh mục có giá lớn hơn 500 hay không.
- `Any()`: Kiểm tra sự tồn tại của bản ghi thỏa mãn điều kiện.

5 Tích hợp LINQ trong ASP.NET

Dưới đây là ví dụ sử dụng LINQ trong một API Controller của ASP.NET Core để trả về danh sách sản phẩm:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.Threading.Tasks;

[Route("api/[controller]")]
[ApiController]
public class ProductsController : ControllerBase
{
    private readonly MyDbContext _context;

    public ProductsController(MyDbContext context)
    {
        _context = context;
    }

    [HttpGet]
    public async Task<IActionResult> GetProducts()
    {
        var products = await (from p in _context.Products
                              where p.Price > 100
                              orderby p.Price descending
                              select new { p.ProductName, p.Price })
                              .ToListAsync();
        return Ok(products);
    }
}
```

Giải thích:

- `async Task<IActionResult>`: Sử dụng async/await để xử lý bất đồng bộ.
- `ToListAsync()`: Đảm bảo truy vấn được thực thi bất đồng bộ, tối ưu cho ứng dụng web.
- Trả về JSON chứa danh sách sản phẩm có giá lớn hơn 100, sắp xếp theo giá giảm dần.

6 Lưu ý khi sử dụng LINQ trong ASP.NET

- **Hiệu suất**: Tránh tải toàn bộ dữ liệu vào bộ nhớ trước khi lọc (sử dụng `Take`, `Skip` cho phân trang).
- **Bảo mật**: LINQ to Entities tự động ngăn chặn SQL Injection.
- **Tối ưu hóa truy vấn**: Sử dụng `AsNoTracking()` cho các truy vấn chỉ đọc để tăng hiệu suất.
- **Debug**: Sử dụng công cụ như SQL Server Profiler để kiểm tra câu lệnh SQL được sinh ra từ LINQ.

7 Kết luận

LINQ là một công cụ mạnh mẽ để truy vấn cơ sở dữ liệu trong ASP.NET, giúp mã dễ đọc, bảo trì và an toàn kiểu dữ liệu. Từ các truy vấn cơ bản như lọc và sắp xếp đến các truy vấn nâng cao như nhóm, kết hợp bảng, hay truy vấn lồng, LINQ cung cấp sự linh hoạt và hiệu quả cho các ứng dụng web.