**Entity Framework Code First:**

1. **Một-nhiều (one-to-many)**

**Products** và **Categories** có mối quan hệ một-nhiều (one-to-many). Cụ thể:

• Một **Category** có thể chứa nhiều **Product**.

• Một **Product** chỉ thuộc về một **Category**.

Chi Tiết Quan Hệ Một-Nhiều

• Category là một thực thể cha (parent entity).

• Product là một thực thể con (child entity).

Mỗi Product sẽ có một khóa ngoại (foreign key) trỏ đến Category mà nó thuộc về, trong khi Category có thể chứa một tập hợp (collection) các Product.

Cách Định Nghĩa Mối Quan Hệ Trong Mô Hình

Dưới đây là cách mối quan hệ được định nghĩa trong các lớp Product và Category:

**Lớp Product**

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

// Foreign key

public int CategoryId { get; set; }

// Navigation property

public Category Category { get; set; }

}

Lớp Category

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

// Navigation property

public ICollection<Product> Products { get; set; }

}

DbContext

Định Nghĩa Mối Quan Hệ trong DbContext

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Product>()

.HasOne(p => p.Category)

.WithMany(c => c.Products)

.HasForeignKey(p => p.CategoryId);

}

1. **Một-một (one-to-one)**

Hai thực thể User và UserProfile, trong đó mỗi User có một UserProfile và ngược lại.

Lớp User

public class User

{

public int UserId { get; set; }

public string Username { get; set; }

public string Email { get; set; }

// Navigation property

public UserProfile UserProfile { get; set; }

}

Lớp UserProfile

public class UserProfile

{

public int UserProfileId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime DateOfBirth { get; set; }

// Foreign key

public int UserId { get; set; }

// Navigation property

public User User { get; set; }

}

DbContext

public class AppDbContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<UserProfile> UserProfiles { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<User>()

.HasOne(u => u.UserProfile)

.WithOne(up => up.User)

.HasForeignKey<UserProfile>(up => up.UserId);

}

}

1. Nhiều-nhiều (many-to-many)

Student và Course, trong đó mỗi Student có thể đăng ký nhiều Course và mỗi Course có thể có nhiều Student.

Lớp Student

public class Student

{

public int StudentId { get; set; }

public string Name { get; set; }

// Navigation property

public ICollection<Course> Courses { get; set; }

}

Lớp Course

public class Course

{

public int CourseId { get; set; }

public string Title { get; set; }

// Navigation property

public ICollection<Student> Students { get; set; }

}

DbContext

public class AppDbContext : DbContext

{

public DbSet<Student> Students { get; set; }

public DbSet<Course> Courses { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Student>()

.HasMany(s => s.Courses)

.WithMany(c => c.Students)

.UsingEntity<Dictionary<string, object>>(

"StudentCourse",

j => j.HasOne<Course>().WithMany().HasForeignKey("CourseId"),

j => j.HasOne<Student>().WithMany().HasForeignKey("StudentId"));

}

}

Các phương thức của fluent api

Các phương thức tác dụng trên lớp entity

