



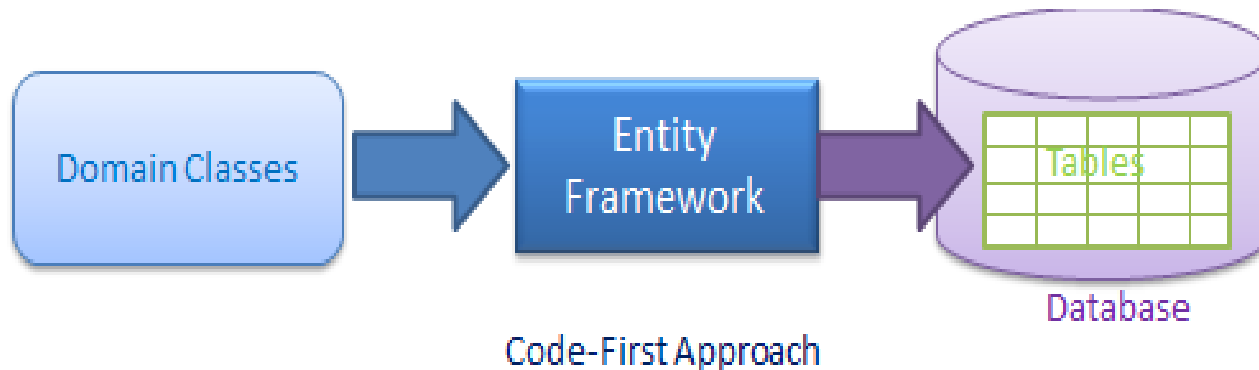
LẬP TRÌNH C# 3

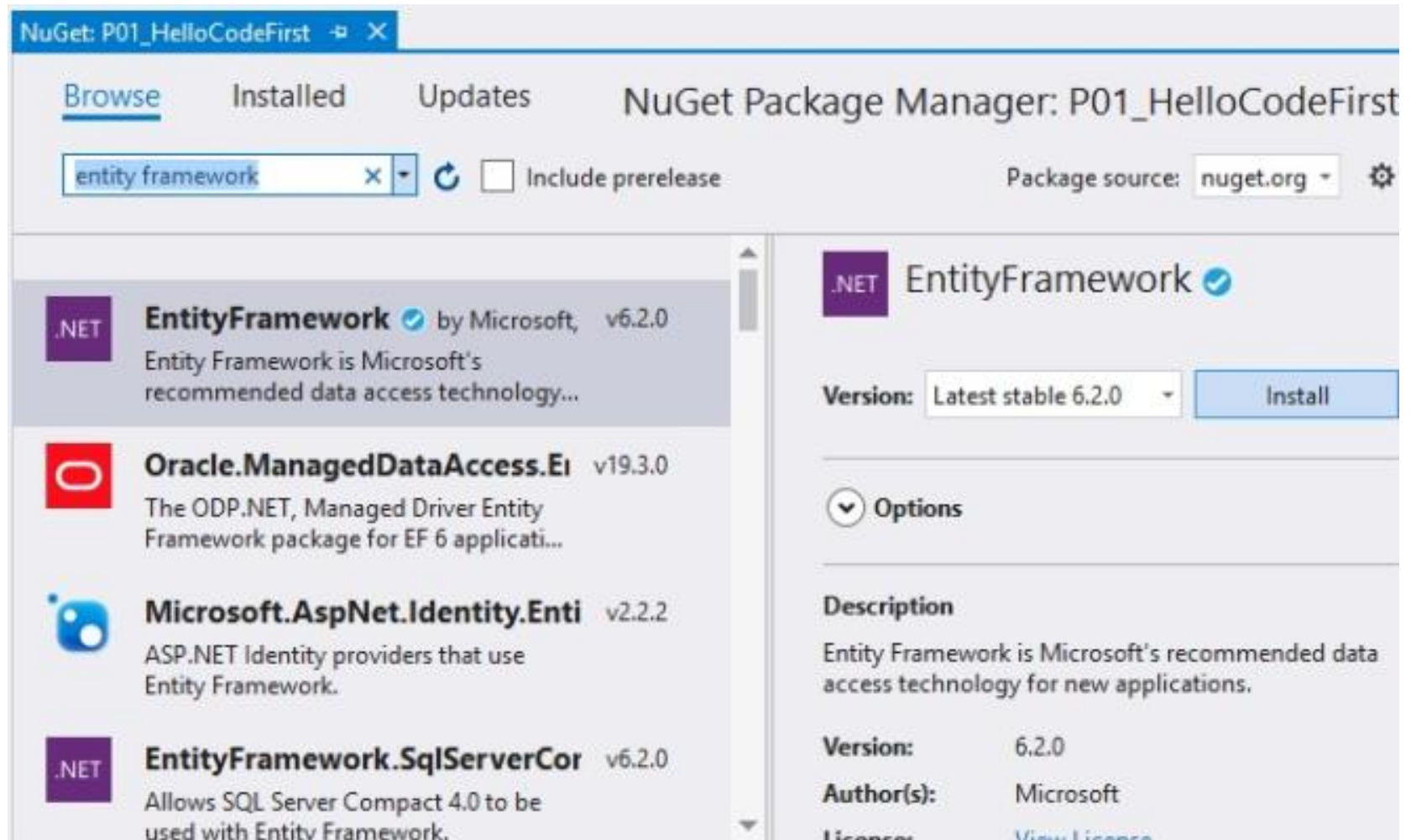
BÀI 6: ENTITY FRAMEWORK CODE FIRST

- ⊙ Code first
- ⊙ Migrations, Data Annotations, Fluent API



- ❑ Code First cho phép lập trình viên định nghĩa domain model với code thay vì phải sử dụng một tập tin EDMX.
- ❑ Code First API tạo ra cơ sở dữ liệu mới hoặc ánh xạ các lớp với cơ sở dữ liệu đã tồn tại
- ❑ Thiết kế mã nguồn chương trình trước, sau đó gieo mới cơ sở dữ liệu sau





- ❑ Để sử dụng code-first, bạn cần tạo hai loại class:
 - ❖ Thứ nhất là các class mô tả các thông tin cần quản lý. Mỗi class thuộc loại này sẽ ánh xạ sang một bảng của cơ sở dữ liệu. Nhóm class này có thể được gọi theo nhiều cách khác nhau như model/domain/business/entity class. Đặc điểm chung của nhóm class này là chỉ chứa dữ liệu trong các property.
 - ❖ Thứ hai là một class con của lớp DbContext, ánh xạ sang chính cơ sở dữ liệu. Loại class này cũng thường được gọi tắt là context hoặc entity container

- ❑ Ví dụ tạo hai lớp **Blog** và **Post**. Một Blog chứa nhiều Post (bài viết) và 1 Post (bài viết) chỉ thuộc về 1 Blog duy nhất.
- ❑ Lớp Post chứa 3 thuộc tính PostId, Title, Content và thuộc tính khóa ngoại BlogId, **thuộc tính điều hướng Blog** để cho biết 1 Post thuộc về duy nhất 1 Blog nào.
- ❑ Lớp Blog chứa 2 thuộc tính BlogId và Name, và **thuộc tính điều hướng Posts** liệt kê các bài viết thuộc 1 Blog

```
public class Post
{
    0 references
    public int PostId { get; set; }
    0 references
    public string Title { get; set; }
    0 references
    public string Content { get; set; }
    //thuộc tính khóa ngoại BlogId
    0 references
    public int BlogId { get; set; }
    //thuộc tính điều hướng Blog để cho biết 1 Post thuộc về duy nhất 1 Blog nào
    0 references
    public Blog Blog { get; set; }
}

public class Blog
{
    0 references
    public int BlogId { get; set; }
    3 references
    public string Name { get; set; }
    //thuộc tính điều hướng Posts liệt kê các bài viết thuộc 1 Blog
    0 references
    public List<Post> Posts { get; set; }
}
```

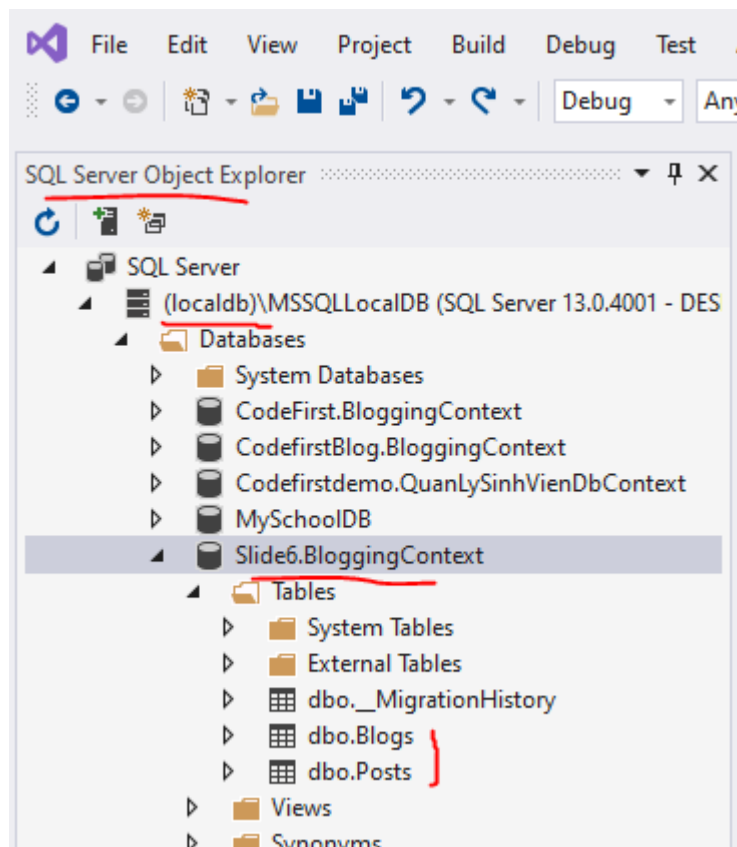
- ❑ context class thể hiện 1 phiên làm việc với database, cho phép thực thi truy vấn và lưu trữ dữ liệu
- ❑ Lớp này kế thừa lớp DbContext
- ❑ Bên trong lớp này dùng DbSet đại diện cho các entity class
- ❑ constructor của Context gọi tới constructor của lớp cha DbContext thiết lập tên database, tên db mặc định là "projectname.context class". Ví dụ tên project:Slide6 và tên context class: BloggingContext thì tên db mặc định: Slide6.BloggingContext


```
public class BloggingContext: DbContext
{
    //Xác định tên database, mặc định là Slide6.BloggingContext
    1 reference
    public BloggingContext() : base()
    {
    }
    //DbSet<Blog> biểu diễn lớp Blogs trong mô hình
    2 references
    public DbSet<Blog> Blogs { get; set; }
    //DbSet<Post> biểu diễn lớp Posts trong mô hình
    0 references
    public DbSet<Post> Posts { get; set; }
}
```

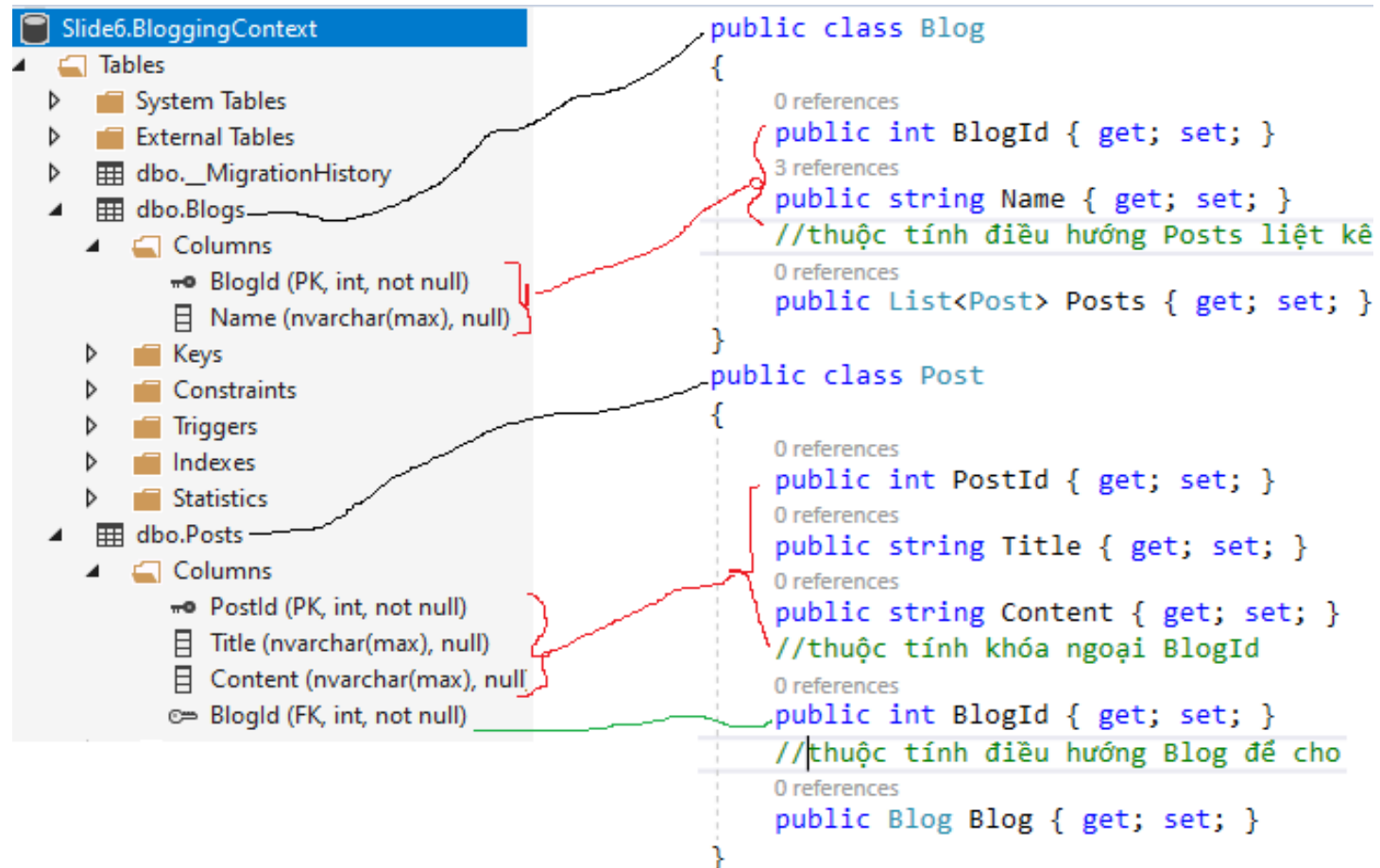
- ❑ Để code-first giúp tạo cơ sở dữ liệu từ các class vừa xây dựng ta dùng phương thức `CreateIfNotExists`, sẽ kiểm tra xem cơ sở dữ liệu chỉ định trong connection string đã tồn tại hay chưa. Nếu chưa, code-first sẽ tạo cơ sở dữ liệu cùng tên với giá trị tham số Initial Catalog.

```
static void Main(string[] args)
{
    using (var db = new BloggingContext())
    {
        db.Database.CreateIfNotExists();
    }
}
```

- ❑ Biên dịch và chạy ứng dụng bạn sẽ nhận được một csdl mới chứa trong LocalDb của bộ công cụ visual studio

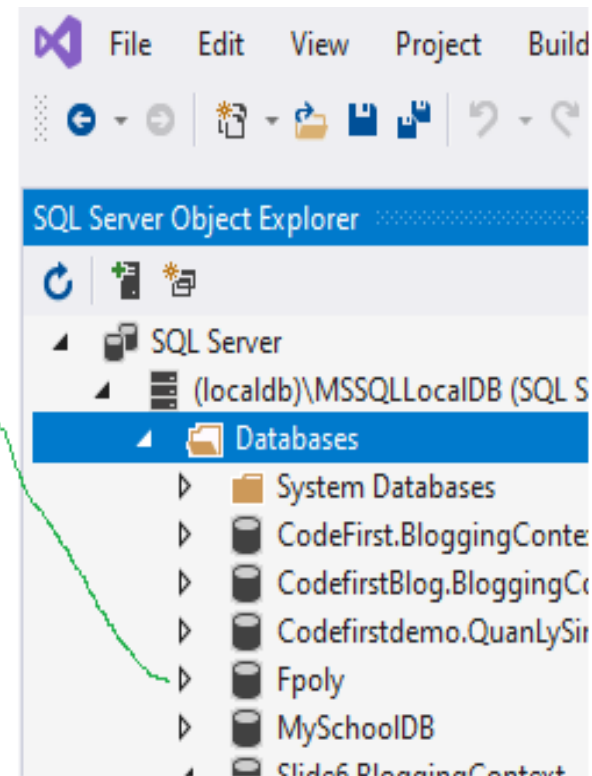


- ❑ DbContext dùng tập mặc định các quy ước Code First để xác định bảng và tên các cột, xác định kiểu dữ liệu, tìm các khóa chính, ...



- ❑ Có thể thay đổi tên csdl bằng cách truyền tên db vào constructor của lớp kế thừa DbContext

```
public class BloggingContext: DbContext
{
    //Xác định tên database, mặc định là Sli
    1 reference
    public BloggingContext() : base("Fpoly")
    {
    }
    ...
}
```

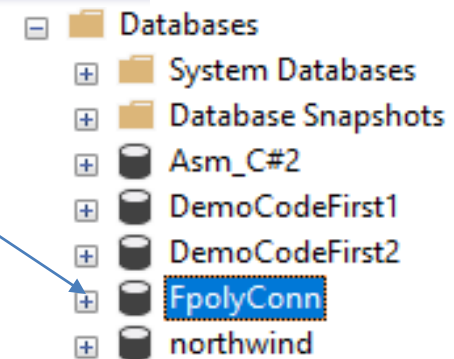


- ❑ Có thể thay đổi tên csdl bằng cách truyền ConnectionString Name vào constructor của lớp kế thừa DbContext

```
public BloggingContext() : base("name = Dbname")  
{  
    ...  
}
```

- ❑ Sử dụng ConnectionString Name trong file App.config

```
<connectionStrings>  
  <add name="Dbname"  
    connectionString="Data Source=.;Initial Catalog=FpolyConn;Integrated Security=true"  
    providerName="System.Data.SqlClient"/>  
</connectionStrings>
```



- ❑ Tạo mới 1 Blog và lấy toàn bộ Blog sắp xếp theo Name trong database bằng LINQ để xuất kết quả ra màn hình

```
static void Main(string[] args)
{
    using (var db = new BloggingContext())
    {
        db.Database.CreateIfNotExists();
        // Nhập tên 1 Blog mới cần tạo
        Console.WriteLine("Ten cua 1 Blog moi: ");
        var name = Console.ReadLine();
        // Thêm 1 Blog mới
        var blog = new Blog { Name = name };
        db.Blogs.Add(blog);
        db.SaveChanges();

        // Hiển thị các Blog trong database, sắp xếp theo Name
        var query = from b in db.Blogs
                    orderby b.Name
                    select b;
        Console.WriteLine("Tat ca blog trong database:");
        foreach (var item in query)
        {
            Console.WriteLine(item.Name);
        }
        Console.ReadKey();
    }
}
```



DEMO

-Hiện thực các ví dụ



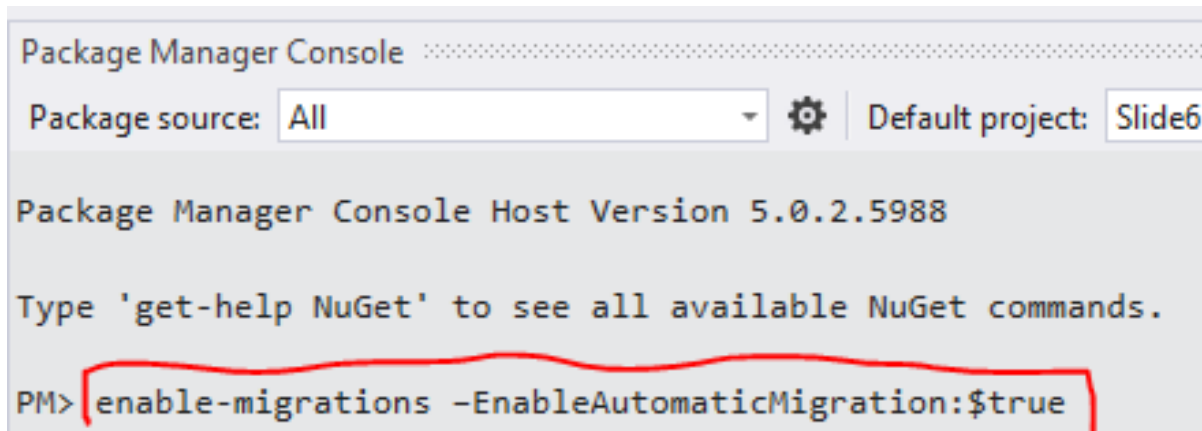


LẬP TRÌNH C# 3

BÀI 6: ENTITY FRAMEWORK CODE FIRST (P2)

- ❑ Code First cho phép thay đổi mô hình code, từ đó thay đổi luôn database tùy ý thông qua tính năng **Code First Migration**
- ❑ Migration là một cơ chế đặc biệt trong Entity Framework cho phép cập nhật CSDL theo cấu trúc model class mà không làm mất dữ liệu hiện có
- ❑ Migration sử dụng một bộ khởi tạo mới có tên là `MigrateDatabaseToLatestVersion`
- ❑ Khi sử dụng migration bạn có thể lựa chọn một trong các phương án: automatic migration và code-based migration

- ❑ Khi sử dụng migration tự động bạn không phải tự mình chạy công cụ migration mỗi lần thay đổi domain class
- ❑ Kích hoạt migration cho một project:
 - ❖ Tool -> NuGet Package Manager-> Package Manager Console
 - ❖ Chạy lệnh : `enable-migrations-EnableAutomaticMigration:$true`



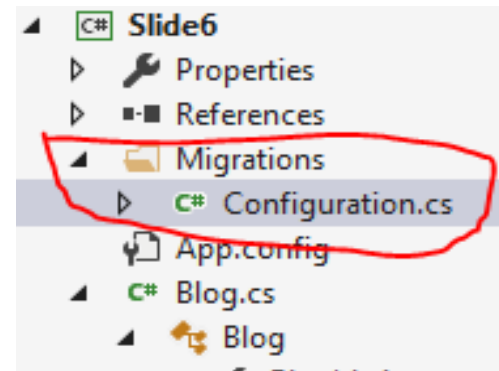
```
Package Manager Console
Package source: All
Default project: Slide6

Package Manager Console Host Version 5.0.2.5988

Type 'get-help NuGet' to see all available NuGet commands.

PM> enable-migrations -EnableAutomaticMigration:$true
```

- ❑ migration được kích hoạt thành công bạn sẽ thu được kết quả



```
internal sealed class Configuration : DbMigrationsConfiguration<Slide6.BloggingContext>
{
    0 references
    public Configuration()
    {
        AutomaticMigrationsEnabled = true;
        ContextKey = "Slide6.BloggingContext";
    }
}
```

- ❑ AutomaticMigrationsEnabled = true báo hiệu chế độ automatic migration đang được sử dụng

❑ Điều chỉnh constructor của lớp Context

```
public BloggingContext() : base("name = Dbname")  
{  
    var initializer = new MigrateDatabaseToLatestVersion<BloggingContext,  
        Migrations.Configuration>();  
    Database.SetInitializer(initializer);  
}
```

❑ tạo một bộ khởi tạo từ lớp

MigrateDatabaseToLatestVersion sử dụng lớp Configuration ở trên và truyền bộ khởi tạo này cho phương thức SetInitializer

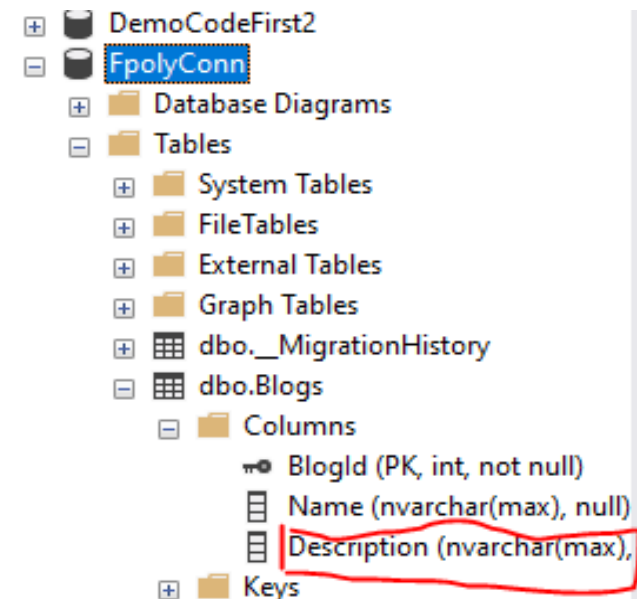
❑ Migration sẽ cập nhật CSDL theo cấu trúc model

- ❑ Ví dụ thay đổi model class bằng cách thêm thuộc tính Description vào class Blog

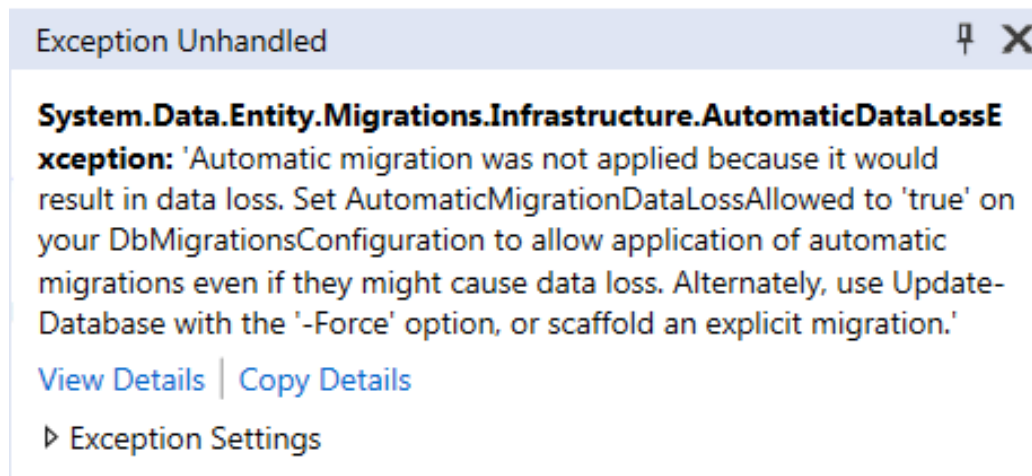
```
public class Blog
{
    0 references
    public int BlogId { get; set; }
    3 references
    public string Name { get; set; }

    0 references
    public string Description { get; set; }
    //thuộc tính điều hướng Posts liệt kê các
    0 references
    public List<Post> Posts { get; set; }
}
```

- ❑ Run ứng dụng và kiểm tra csdl



- ❑ Vấn đề xóa/thay đổi thuộc tính hoặc xóa bỏ domain class sẽ dẫn đến mất dữ liệu. Migration mặc định sẽ cấm tất cả các hoạt động cập nhật nếu gây mất dữ liệu
- ❑ Ví dụ xóa bớt thuộc tính Name trong class Blog sẽ nhận thông báo



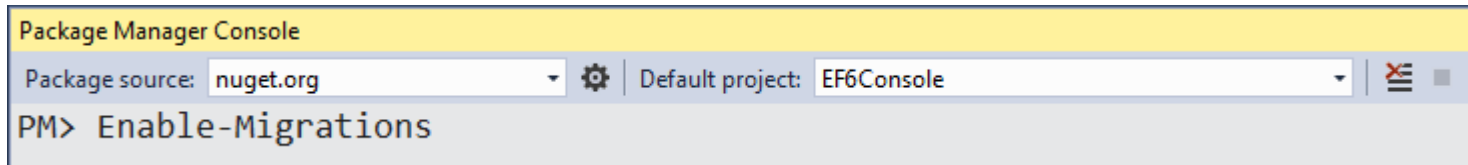
- ❑ Bạn có thể thay đổi cách làm mặc định của migration bằng cách điều chỉnh constructor của lớp Configuration như sau:

```
public Configuration()  
{  
    AutomaticMigrationsEnabled = true;  
    AutomaticMigrationDataLossAllowed = true;  
}
```

- ❑ Lệnh điều chỉnh property AutomaticMigrationDataLossAllowed thành giá trị true. Thông tin này báo cho migration rằng bạn chấp nhận mất một phần dữ liệu khi thực hiện thay đổi CSDL. Bạn sẽ tự chịu trách nhiệm cho việc mất một phần dữ liệu liên quan.

- ❑ Sử dụng migration qua code một cách thủ công cho phép bạn kiểm soát quá trình cập nhật CSDL cũng như thực hiện một số cấu hình bổ sung (như giá trị mặc định của cột, cột tính giá trị tự động, v.v.)
- ❑ Có 3 thao tác chính với code-based migration
 - ❖ Enable-Migrations: kích hoạt tạo Configuration class
 - ❖ Add-Migration: yêu cầu tạo ra một class mới để giúp thực hiện cập nhật CSDL
 - ❖ Update-Database: yêu cầu thực thi cập nhật CSDL. Lệnh này chỉ được gọi sau khi thực hiện thành công Add-Migration

□ Enable-Migrations

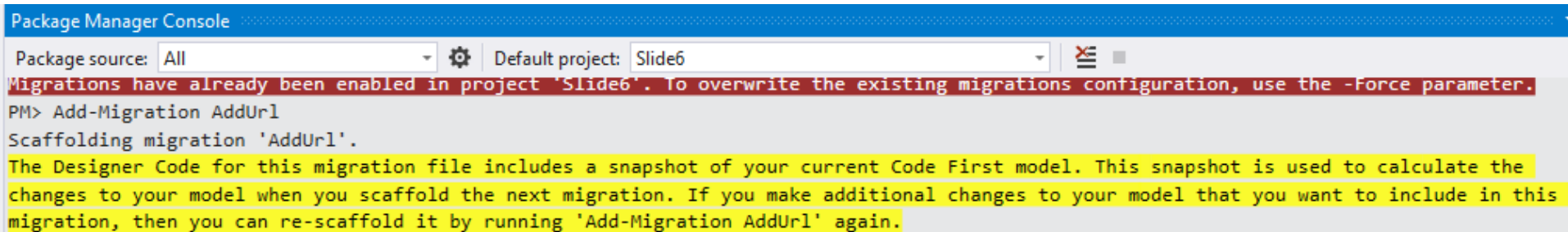


□ Tiếp theo chúng ta thay đổi mô hình, thêm thuộc tính Url vào lớp Blog như sau:

```
public class Blog
{
    public int BlogId { get; set; }
    public string Name { get; set; }
    public string Url { get; set; } // thêm thuộc tính mới

    public virtual List<Post> Posts { get; set; }
}
```

❑ Thực hiện lệnh Add-Migration



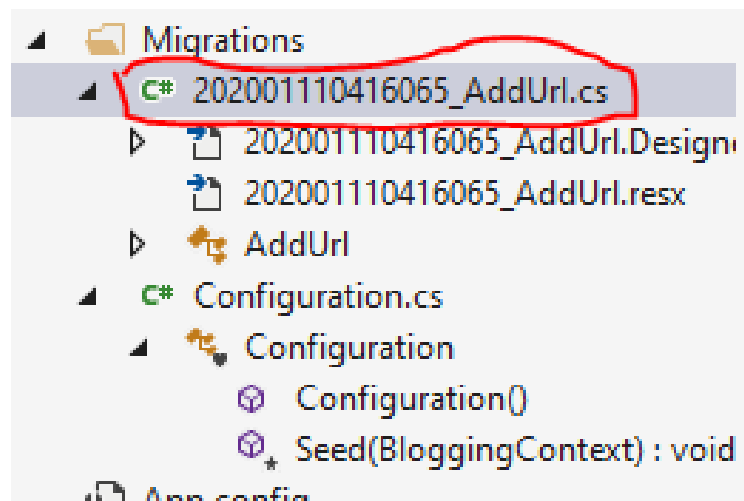
```
Package Manager Console
Package source: All Default project: Slide6
Migrations have already been enabled in project 'Slide6'. To overwrite the existing migrations configuration, use the -Force parameter.
PM> Add-Migration AddUrl
Scaffolding migration 'AddUrl'.
The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your model when you scaffold the next migration. If you make additional changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration AddUrl' again.
```

- ❑ Lệnh này sẽ tạo ra một class mới có tên mặc định phần đầu là thời gian tạo file và phần sau là tên tham số mà bạn truyền trong lệnh Add-Migration Param (trường hợp này là AddUrl)

❑ Lệnh Add-Migration

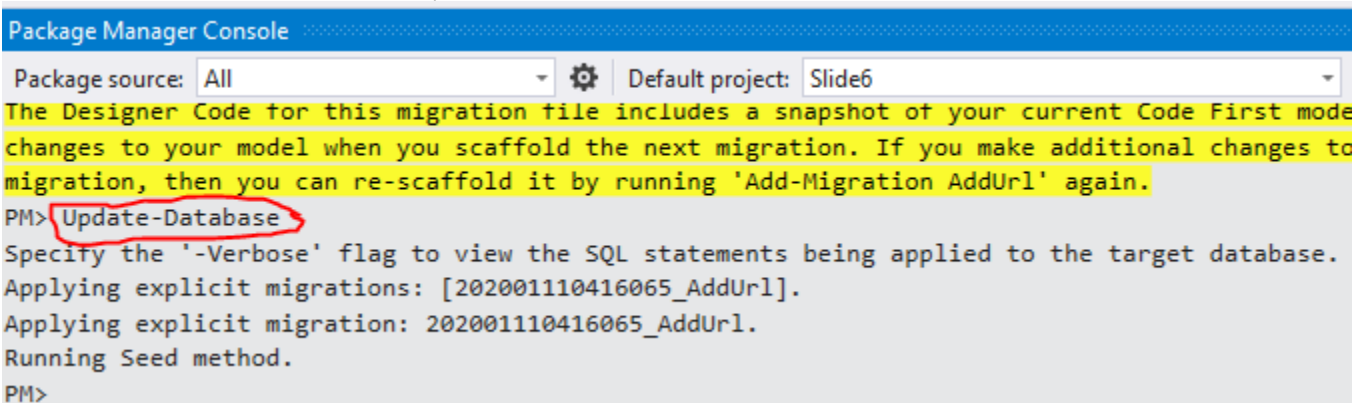
```
public partial class AddUrl : DbMigration
{
    0 references
    public override void Up()
    {
        AddColumn("dbo.Blogs", "Url", c => c.String());
        DropColumn("dbo.Blogs", "Description");
    }

    0 references
    public override void Down()
    {
        AddColumn("dbo.Blogs", "Description", c => c.String());
        DropColumn("dbo.Blogs", "Url");
    }
}
```

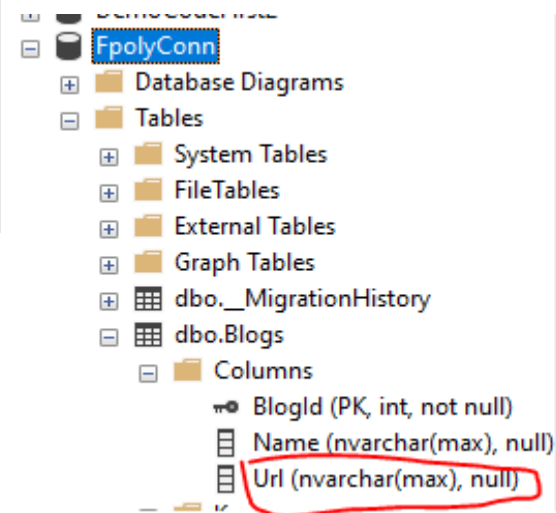


❑ Có thể vào lớp AddUrl vừa tạo để chỉnh tay thêm/xóa các trường cần thiết:

- ❑ Thực hiện lệnh Update-Database: dòng lệnh Update-Database để cập nhật mô hình mới trong database. Trong database, bảng Blog sẽ có thêm cột Url mới



```
Package Manager Console
Package source: All
Default project: Slide6
The Designer Code for this migration file includes a snapshot of your current Code First model
changes to your model when you scaffold the next migration. If you make additional changes to
migration, then you can re-scaffold it by running 'Add-Migration AddUrl' again.
PM> Update-Database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [202001110416065_AddUrl].
Applying explicit migration: 202001110416065_AddUrl.
Running Seed method.
PM>
```



- ❑ Data annotation (chú thích dữ liệu) là tên gọi chung của một nhóm attribute đặc biệt giúp chúng ta chỉ định cấu trúc bảng dữ liệu trong code C#
- ❑ Bộ attribute này có vai trò đặc biệt quan trọng nếu muốn code-first sinh ra các cơ sở dữ liệu phức tạp theo đúng yêu cầu.
- ❑ Ví dụ, (1) làm sao để ràng buộc độ dài của trường văn bản, (2) làm sao để ép một trường không được để trống, (3) làm sao để chỉ định chính xác cấu hình cột dữ liệu trong bảng khi tạo class C# để về sau Entity Framework tạo bảng theo đúng yêu cầu? v.v..

CHÚ THÍCH DỮ LIỆU (DATA ANNOTATIONS)

Attribute	Ý nghĩa
Key	Chỉ định thuộc tính sẽ được ánh xạ sang làm khóa chính của bảng. Sử dụng attribute này nếu tên khóa chính không tuân theo quy ước của Entity Framework.
Timestamp	Chỉ định kiểu dữ liệu của cột (tương ứng với property này) sẽ là <code>rowversion</code> .
ConcurrencyCheck	Cột tương ứng cần đưa vào kiểm tra cạnh tranh (theo kiểu optimistic).
Required	Chỉ định cột tương ứng phải là NotNull.
MinLength	Chỉ định độ dài tối thiểu của cột kiểu văn bản hoặc mảng byte.
MaxLength	Chỉ định độ dài tối đa của cột kiểu văn bản hoặc mảng byte.
StringLength	Chỉ định độ dài tối đa của cột kiểu văn bản.

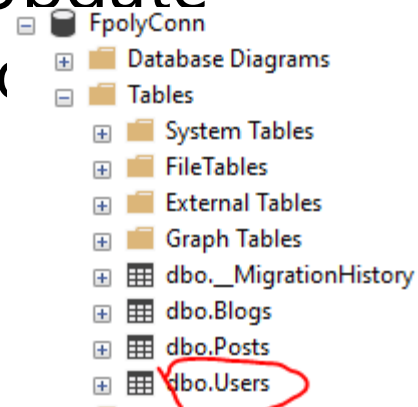
Attribute	Description
Table	Áp dụng cho entity class, chỉ định tên bảng và sơ đồ (schema).
Column	Áp dụng cho property, chỉ định tên cột, thứ tự và kiểu dữ liệu của cột.
Index	Chỉ định cột tương ứng cần có Index (từ EF 6.1)
ForeignKey	Chỉ định thuộc tính này sẽ là khóa ngoài. Bạn sẽ gặp lại attribute này trong bài học về quan hệ một – nhiều.
NotMapped	Nếu áp dụng cho property chỉ định rằng property này sẽ không ánh xạ sang cột của bảng; Nếu áp dụng cho class chỉ định rằng class sẽ không được ánh xạ sang bảng. Nói cách khác, EF không được sinh ra cột hoặc bảng từ property hoặc class tương ứng.
DatabaseGenerated	Chỉ định rằng giá trị của cột tương ứng sẽ do cơ sở dữ liệu sinh ra.
InverseProperty	Sử dụng khi cấu hình nhiều quan hệ tương tự nhau giữa hai class. Ví dụ, hai class có cùng lúc hai quan hệ 1 – nhiều .
ComplexType	Đánh dấu class là kiểu phức tạp.

- ❑ Ví dụ thêm 1 lớp mới User vào mô hình, chú thích thuộc tính Username là khóa chính với cú pháp [Key]

```
public class User
{
    [Key] // đây là chú thích dữ liệu có ý nghĩa thuộc tính Username là khóa chính của lớp
    0 references
    public string Username { get; set; }
    0 references
    public string DisplayName { get; set; }
}

public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
    public DbSet<User> Users { get; set; }
}
```

- ❑ Cần bổ sung DbSet User vào DbContext
- ❑ Dùng lệnh Add-Migration AddUser và Update-Database trong Package Manager Console cập nhật mô hình mới



- ❑ Fluent API là cách chỉ định cấu trúc cơ sở dữ liệu thứ hai trong Entity Framework Code first
- ❑ Cấu hình này sử dụng classDbModelBuilder với chuỗi phương thức (method chaining) ghép nối với nhau
- ❑ Để dùng Fluent API, bạn có thể ghi đè (override) lên phương thức OnModelCreating ở DbContext

- ❑ Ví dụ thay tên cột User.DisplayName thành display_name

```
public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
    public DbSet<User> Users { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<User>()
            .Property(u => u.DisplayName)
            .HasColumnName("display_name"); // thay đổi tên thuộc tính
    }
}
```

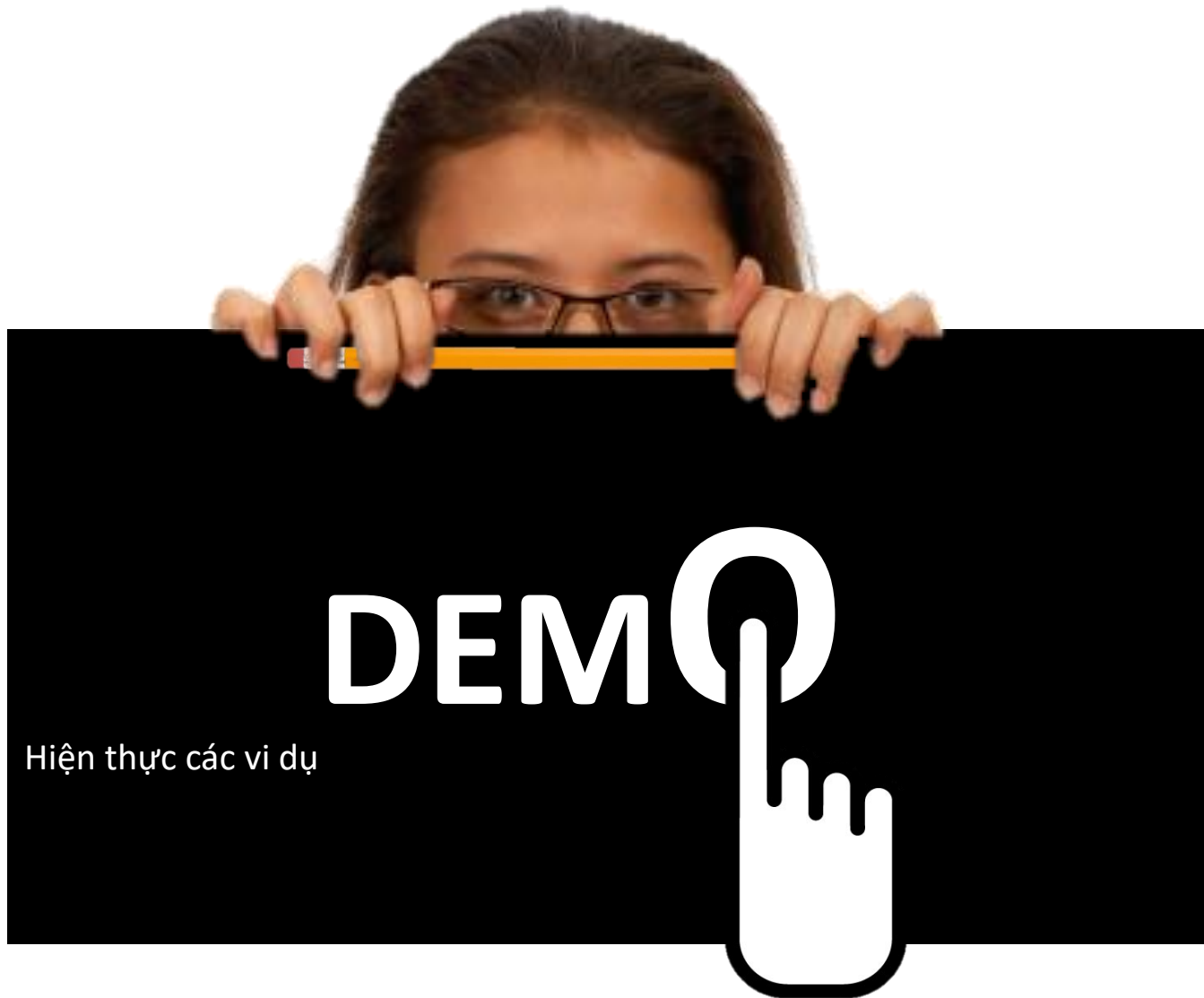
- ❑ chạy lệnh Add-Migration ChangeDisplayName và lệnh Update-Database để thay đổi mô hình.

❑ Các phương thức tác dụng trên lớp entity

HasIndex()	Chỉ định trường index
HasKey()	Chỉ định trường khóa chính
HasMany()	Chỉ định quan hệ 1-n hoặc n-n
HasOptional()	Chỉ định quan hệ 1-0..1
HasRequired()	Chỉ định quan hệ 1-1
Ignore()	Chỉ định class không được ánh xạ thành bảng dữ liệu
Map()	Chỉ định các cấu hình nâng cao
MapToStoredProcedures()	Cấu hình để sử dụng INSERT, UPDATE and DELETE stored procedures
ToTable()	Chỉ định tên bảng dữ liệu tương ứng

❑ Các phương thức tác dụng trên property

HasColumnAnnotation()	
IsRequired()	Chỉ định trường bắt buộc khi gọi SaveChanges()
IsConcurrencyToken()	Configures the property to be used as an optimistic concurrency token
IsOptional()	Trường tương ứng có thể nhận giá trị null
HasParameterName()	Tên tham số sử dụng trong stored procedure tương ứng của property
HasDatabaseGeneratedOption()	Chỉ định cách sinh dữ liệu tự động của cột trong cơ sở dữ liệu (computed, identity, none)
HasColumnOrder()	Chỉ định thứ tự của cột tương ứng trong csdl
HasColumnType()	Configures the data type of the corresponding column of a property in the database.
HasColumnName()	Configures the corresponding column name of a property in the database.
IsConcurrencyToken()	Configures the property to be used as an optimistic concurrency token.



Tổng kết bài học

☉Code first

☉Migrations, Data Annotations, Fluent API





KẾT THÚC