Truy xuất dữ liệu sử dụng Entity Framework Core

Mục tiêu

- Giới thiệu về Entity Framework Core

Giới thiệu về Entity Framework Core

- **Ententity Framework Core là một phiên bản mới sau X 6.x. No là open-source, nhẹ, có khả năng mở rộng và là công nghệ truy xuất dữ liệu hỗ trợ đa nền tảng.**
- **Entity Framework là một Framework Object/Relational Mapping (O/RM). Nó là bản cải tiến nâng cấp từ ADO.NET, đem đến cho các nhà phát triển cơ chế truy xuất và lưu trữ dữ liệu trong database một cách tự động.**
- **EF Core được thiết kế để sử dụng với các ứng dụng .NET Core. Tuy nhiên, nó cũng có thể được sử dụng với các ứng dụng dựa trên .NET 4.5+ Framework Standard.**

EF Core Database Providers

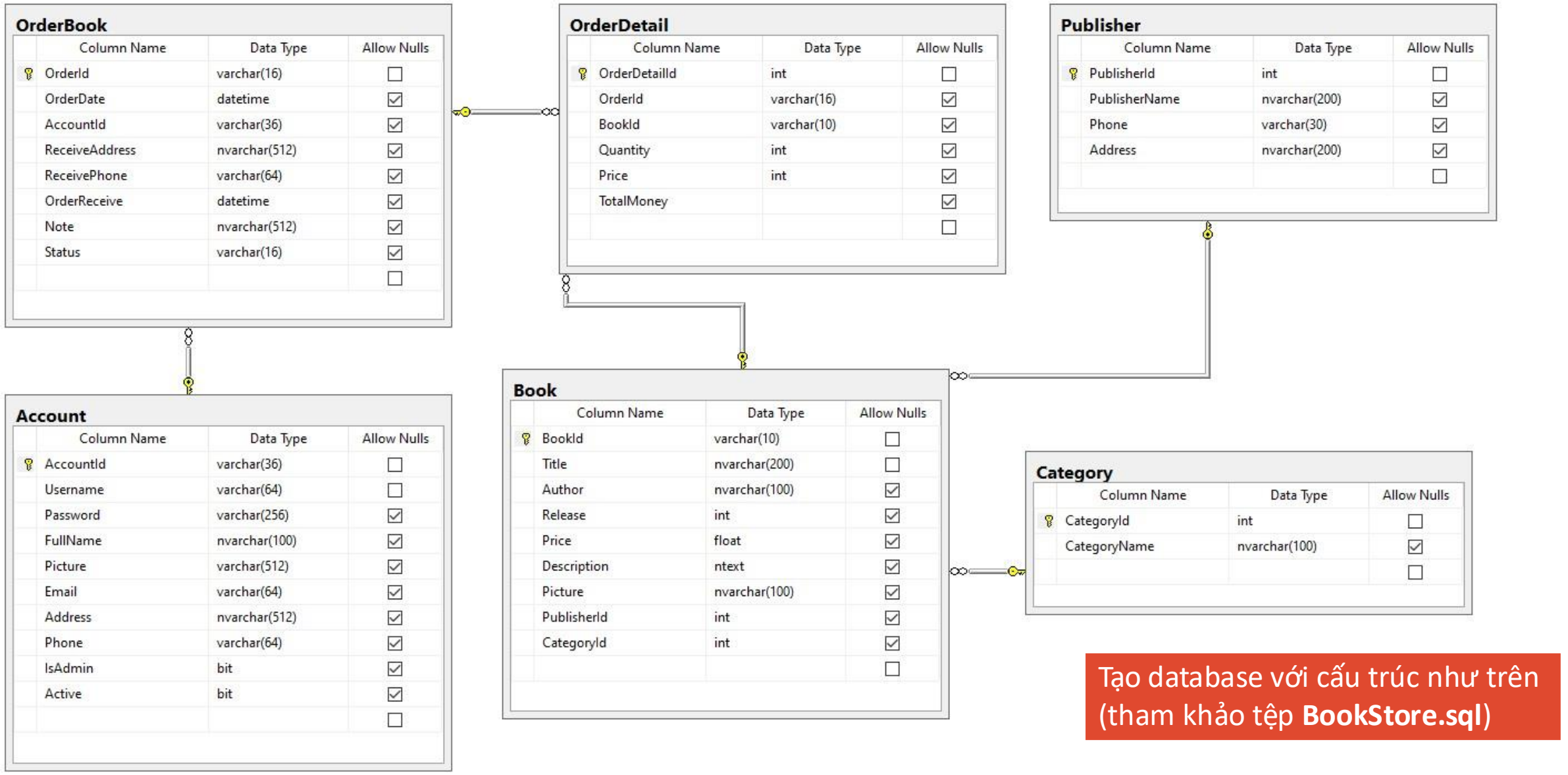
Database	NuGet Package
SQL Server	Microsoft.EntityFrameworkCore.SqlServer
MySQL	MySql.Data.EntityFrameworkCore
PostgreSQL	Npgsql.EntityFrameworkCore.PostgreSQL
SQLite	Microsoft.EntityFrameworkCore.SQLite
SQL Compact	EntityFrameworkCore.SqlServerCompact40
In-memory	Microsoft.EntityFrameworkCore.InMemory

Các phương pháp lập trình với EF Core

Database-First Approach
Sinh ra các lớp bao bọc và truy xuất dữ liệu từ database có sẵn

Code-First Approach
Tạo database từ các lớp

Tạo model bằng phương pháp Database-First

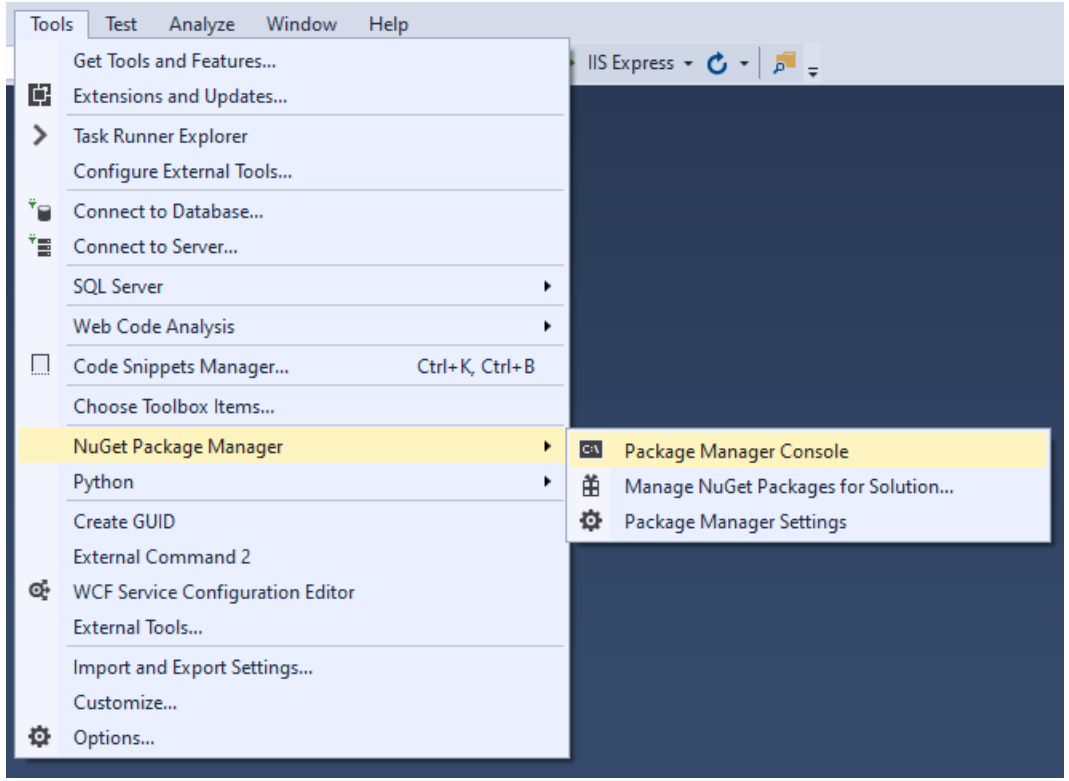


Tạo database với cấu trúc như trên (tham khảo tệp **BookStore.sql**)

Tạo model bằng phương pháp Database-First

1. Tạo project ASP.NET Core

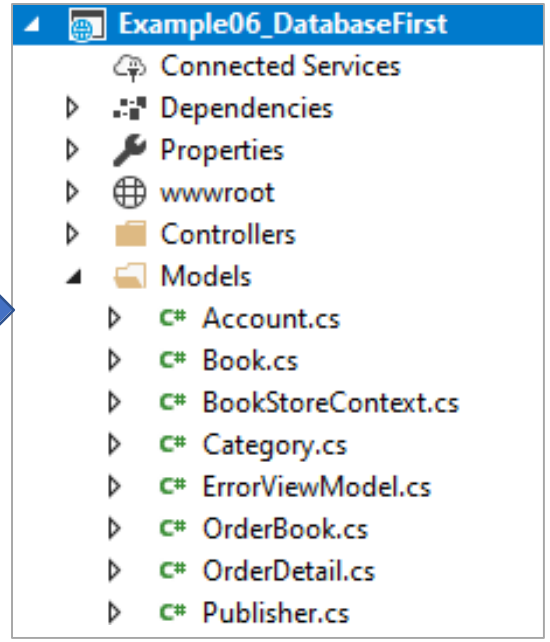
2. Mở cửa sổ Package Manager Console



3. Gõ lệnh màu đỏ sau

```
PM> Scaffold-DbContext
"Server=localhost;Database=BookStore;uid=sa;p
wd=123465;MultipleActiveResultSets=True;"
Microsoft.EntityFrameworkCore.SqlServer -
OutputDir Models
```

Kết quả nhận được



Tạo model bằng phương pháp Database-First

- Mặc định chuỗi kết nối được đặt cứng trong phương thức **OnConfiguring** của **lớp BookStoreContext**.

  Tuy nhiên đây không phải là cách phù hợp.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseSqlServer(@"Server=localhost;Database=BookStore;uid=sa;pwd=123465;MultipleActiveResultSets=True;");
    }
}
```

- Bạn nên đặt nó trong tệp appsettings.json và cấu hình Depedency Injection cho BookStoreContext
- trong phương thức ConfigureServices của lớp Startup. Đây là phương pháp phù hợp nhất, thuận tiện cho việc sử dụng BookStoreContext ở các controller cũng như việc sinh code tự động trong Visual Studio sau này.

Tạo model bằng phương pháp Database-First

File: appsettings.json

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
  ,
  "ConnectionStrings": {
    "BookStoreString": "Server=localhost;Database=BookStore;uid=sa;pwd=123465;MultipleActiveResultSets=True;"
  }
}
```

Class: BookStoreContext.cs

```
public partial class BookStoreContext : DbContext
{
    public BookStoreContext(DbContextOptions<BookStoreContext> options):base(options)
    {

    }
```

Tạo model bằng phương pháp Database-First

Class: Startup.cs

```
using Microsoft.EntityFrameworkCore;
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
        services.AddDbContext<BookStoreContext>(options => options.UseSqlServer(Configuration.GetConnectionString("BookStoreString")));
    }
```

Add Dependency Injection cho BookStoreContext

Làm việc với Database

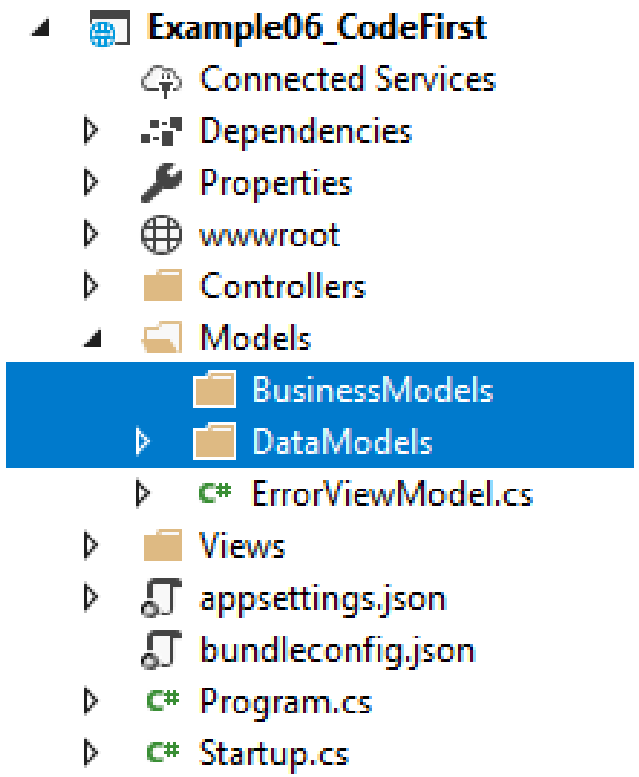
- Lớp DbContext sẽ đảm nhiệm các công việc với database bao gồm
  - Manage database connection
  - Configure model & relationship
  - Querying database
  - Saving data to the database
  - Configure change tracking
  - Caching
  - Transaction management

Các thao tác cơ bản CRUD

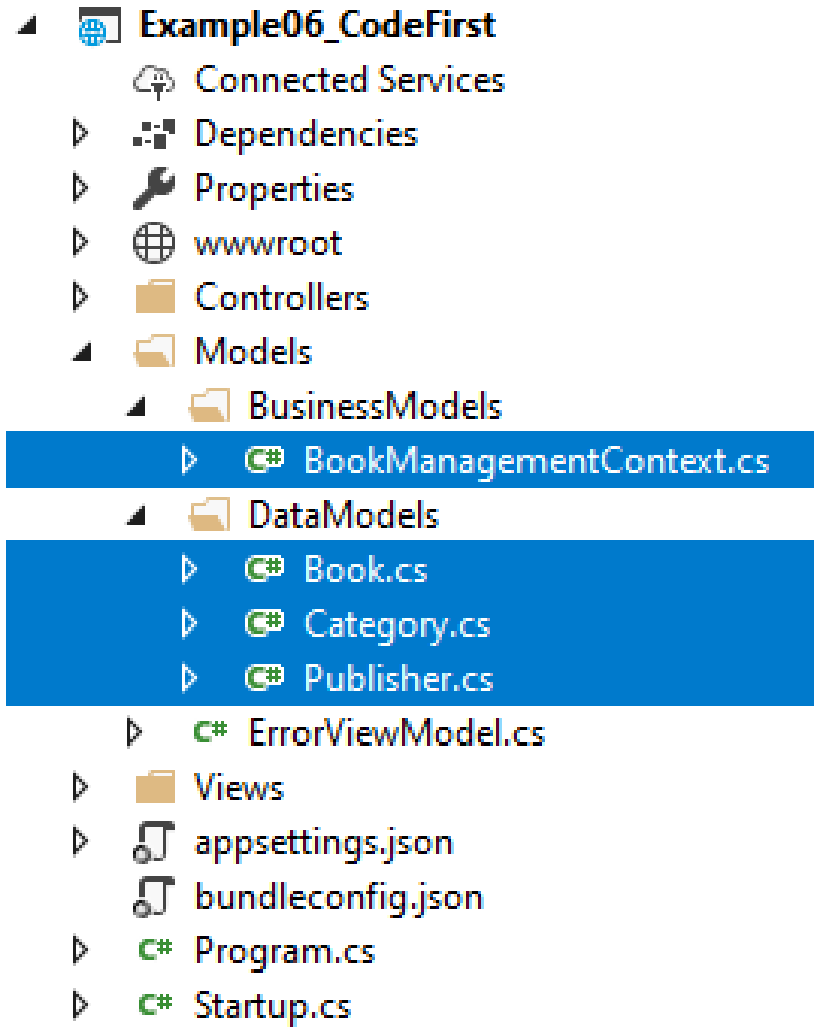
Tạo model bằng phương pháp Database-

1. Tạo project ASP.NET Core

2. Tạo các thư mục trong Models như hình dưới



3. Tạo các lớp Data Model và Context như hình dưới



Tạo model bằng phương pháp Database-

```
[Table("Books")] //Books là tên bảng sinh ra trong database
public class Book
{
    [DisplayName("Mã sách")]
    [StringLength(10)]
    public string BookId { get; set; }
    [DisplayName("Tên sách sách")]
    [StringLength(200)]
    public string Title { get; set; }
    [DisplayName("Tác giả")]
    [StringLength(100)]
    public string Author { get; set; }
    [DisplayName("Năm xuấtt bản")]
    public int? Release { get; set; }
    [DisplayName("Giá")]
    public double? Price { get; set; }
    [DisplayName("Mô tả")]
    public string Description { get; set; }
    [DisplayName("Hình ảnh")]
    [StringLength(500)]
    public string Picture { get; set; }
    [DisplayName("Mã xuất bản")]
    public int? PublisherId { get; set; }
    [DisplayName("Mã loại")]
    public int? CategoryId { get; set; }
    //Khai báo thuộc tính Navigation tạo quan hệ giữa thực thể Book và Category, Publihser
    public Category Category { get; set; }
    public Publisher Publisher { get; set; }
}
```

Định nghĩa các thực thể và các thuộc tính của chúng

```
[Table("Publishers")]
public class Publisher
{
    [DisplayName("Mã NXB")]
    public int PublisherId { get; set; }
    [DisplayName("Tên nhà xuất bản")]
    [StringLength(100)]
    public string PublisherName { get; set; }
    [DisplayName("Điện thoại")]
    [StringLength(50)]
    public string Phone { get; set; }
    [DisplayName("Địa chỉ")]
    [StringLength(200)]
    public string Address { get; set; }
    //Khai báo thuộc tính Navigation tới thực thể Book
    public ICollection<Book> Book { get; set; }
}

[Table("Categories")]
public class Category
{
    [DisplayName("Mã loại")]
    public int CategoryId { get; set; }
    [DisplayName("Tên danh mục")]
    [StringLength(100)]
    public string CategoryName { get; set; }
    //Khai báo thuộc tính Navigation tới thực thể Book
    public ICollection<Book> Books { get; set; }
}
```

Tạo model bằng phương pháp Database-

Class: BookManagementContext

```
public class BookManagementContext: DbContext
{
    public BookManagementContext(DbContextOptions<BookManagementContext> options)
        :base(options)
    {

    }
    //Khai báo các thuộc tính biểu diễn các tập thực thể
    public DbSet<Book> Books { get; set; }

    public DbSet<Category> Categories { get; set; }

    public DbSet<Publisher> Publishers { get; set; }
}
```

Tạo model bằng phương pháp Database-

File: appsettings.json

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
  ,
  "ConnectionStrings": {
    "BookManagementString": "Server=localhost;database=BookManagement;uid=sa;pwd=123465;MultipleActiveResultSets=True"
  }
}
```

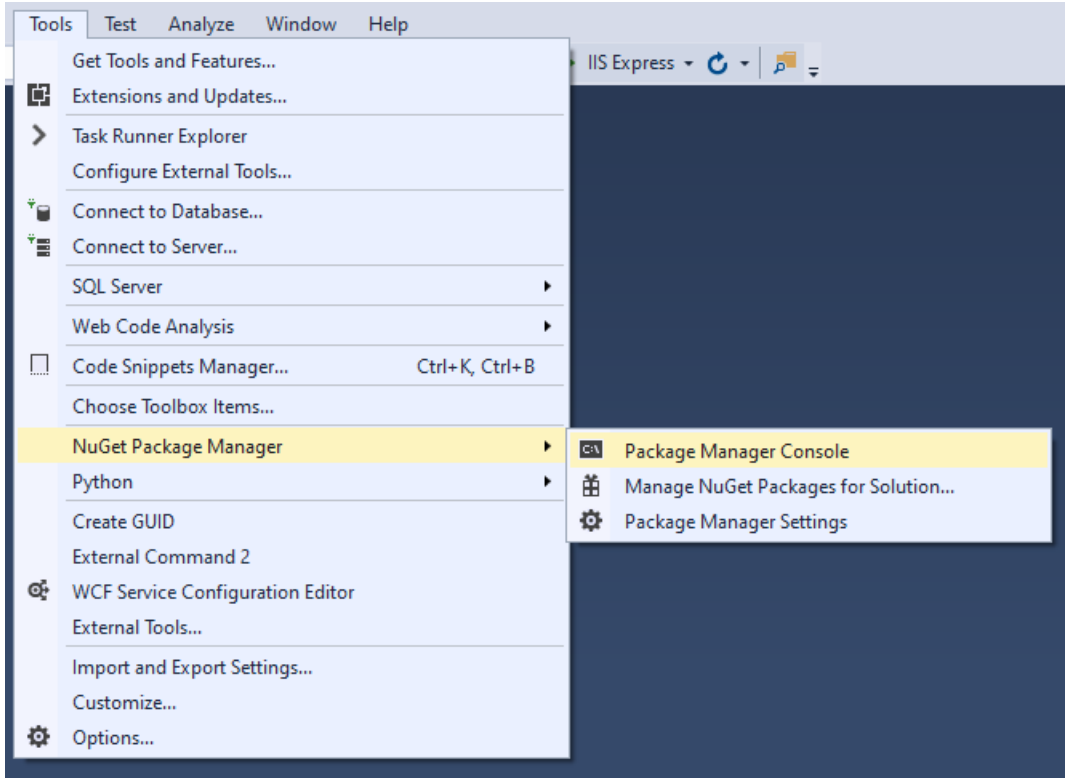
Class: Startup

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddDbContext<BookManagementContext>(
        options => options.UseSqlServer(Configuration.GetConnectionString("BookManagementString")));
}
```

Tạo model bằng phương pháp Database-

- Sinh database sử dụng tính năng Migration của EF Core

1. Mở cửa sổ Package Manager Console



Tạo model bằng phương pháp Database-

2. Gõ các lệnh sau theo thứ tự

Package Manager Console

Package source: All | Default project: Example06_CodeFirst

```
PM> Add-Migration
```

Package Manager Console

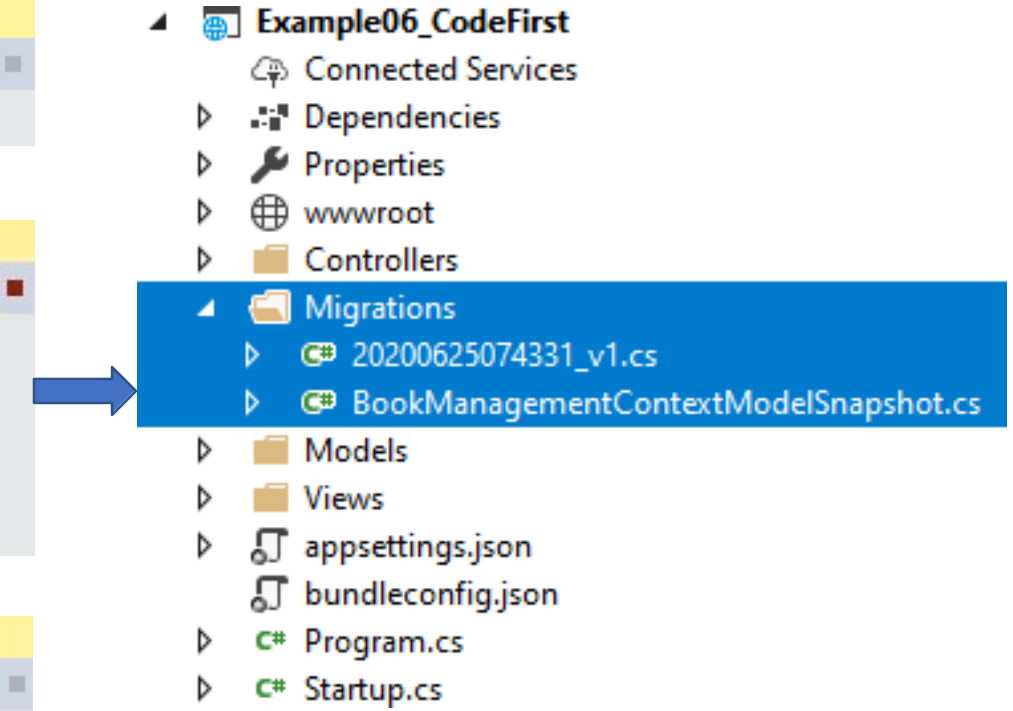
Package source: All | Default project: Example06_CodeFirst

```
PM> Add-Migration
cmdlet Add-Migration at command pipeline position 1
Supply values for the following parameters:
Name: v1
```

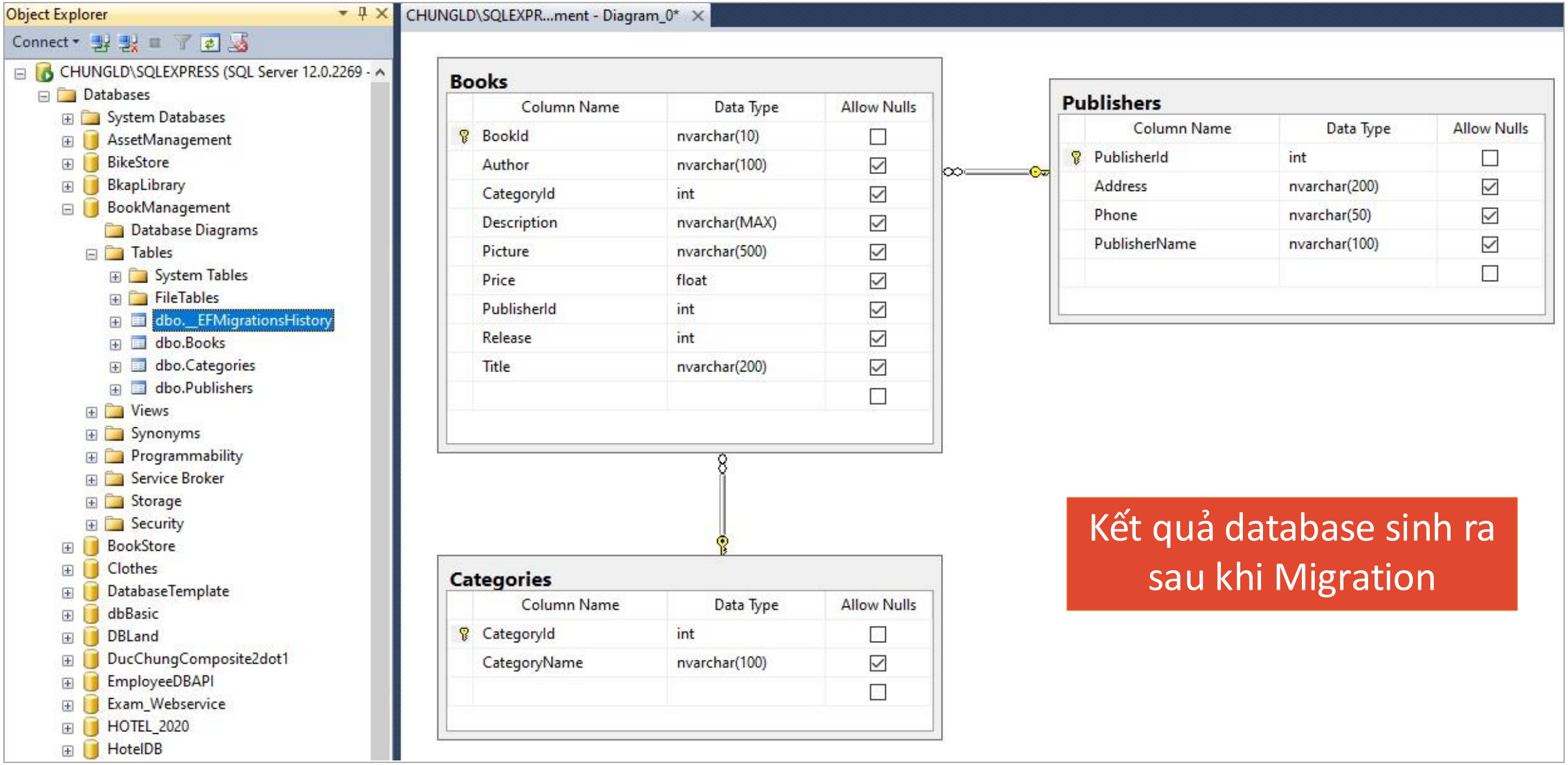
Package Manager Console

Package source: All | Default project: Example06_CodeFirst

```
PM> update-database
```



Tạo model bằng phương pháp Database-



Kết quả database sinh ra sau khi Migration

Các thao tác cơ bản CRUD