



## Bài 6

# *Truy cập dữ liệu sử dụng Entity Framework*



- ◆ Giới thiệu về Entity Framework
- ◆ Sử dụng phương pháp code first
- ◆ Sử dụng phương pháp database first
- ◆ Sử dụng LINQ để truy vấn dữ liệu
- ◆ Tích hợp CKEditor và CKFinder vào ứng dụng

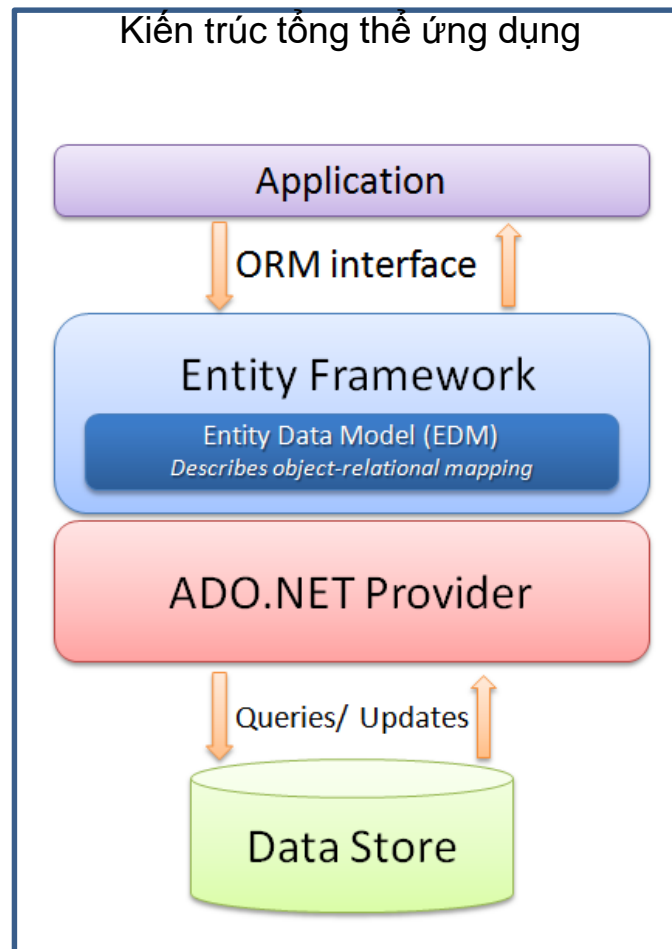


- ◆ Để chỉ ra những yêu cầu truy cập dữ liệu của ứng dụng ASP.NET MVC bạn có thể sử dụng một ORM framework.
- ◆ ORM framework giúp
  - ◆ Truy cập dữ liệu dễ dàng hơn từ các ứng dụng
  - ◆ Thực thi những chuyển đổi cần thiết giữa các loại hệ thống cơ sở dữ liệu quan hệ và các ngôn ngữ lập trình hướng đối tượng
- ◆ Entity Framework là một ORM framework mà cá ứng dụng ASP.NET MVC có thể sử dụng
- ◆ Entity Framework là sự thực thi của Entity Data Model (EDM), mà là một mô hình khái niệm mô tả các thực thể và mối liên kết mà chúng tham gia trong một ứng dụng
- ◆ EDM cho phép bạn xử lý logic truy cập dữ liệu bằng cách lập trình với các thực thể mà không cần phải lo lắng về cấu trúc của tầng lưu trữ dữ liệu và làm thế nào để kết nối với nó.

# Các phiên bản của EF

Phiên bản	Tính năng
EF 3.5	Hỗ trợ ORM cơ bản cho việc tiếp cận với Database
EF 4.0	Hỗ trợ POCO, Lazy, cải thiện khả năng kiểm tra, tùy biến mã và tiếp cận với Model First
EF 4.1	Gói NuGet, DbContext API, mã Code First và những bảng vá lỗi của EF 4.1.1
EF 4.3	Code First cho phép tạo Database bởi những dòng code cơ bản. EF 4.3.1 được phát hành với những bản vá lỗi EF 4.3.2n
EF 5.0	Cung cấp EF dạng mã nguồn mở (open source). Hỗ trợ Enum, table-valued function, kiểu dữ liệu không gian spatial data types, multiple-diagrams der model, ....
EF 6.0 – Hiện tại	EF 6.0 đang chuẩn bị phát hành. Chức năng sẽ gồm Task-based async, Store Procedure, Function trong Code First...

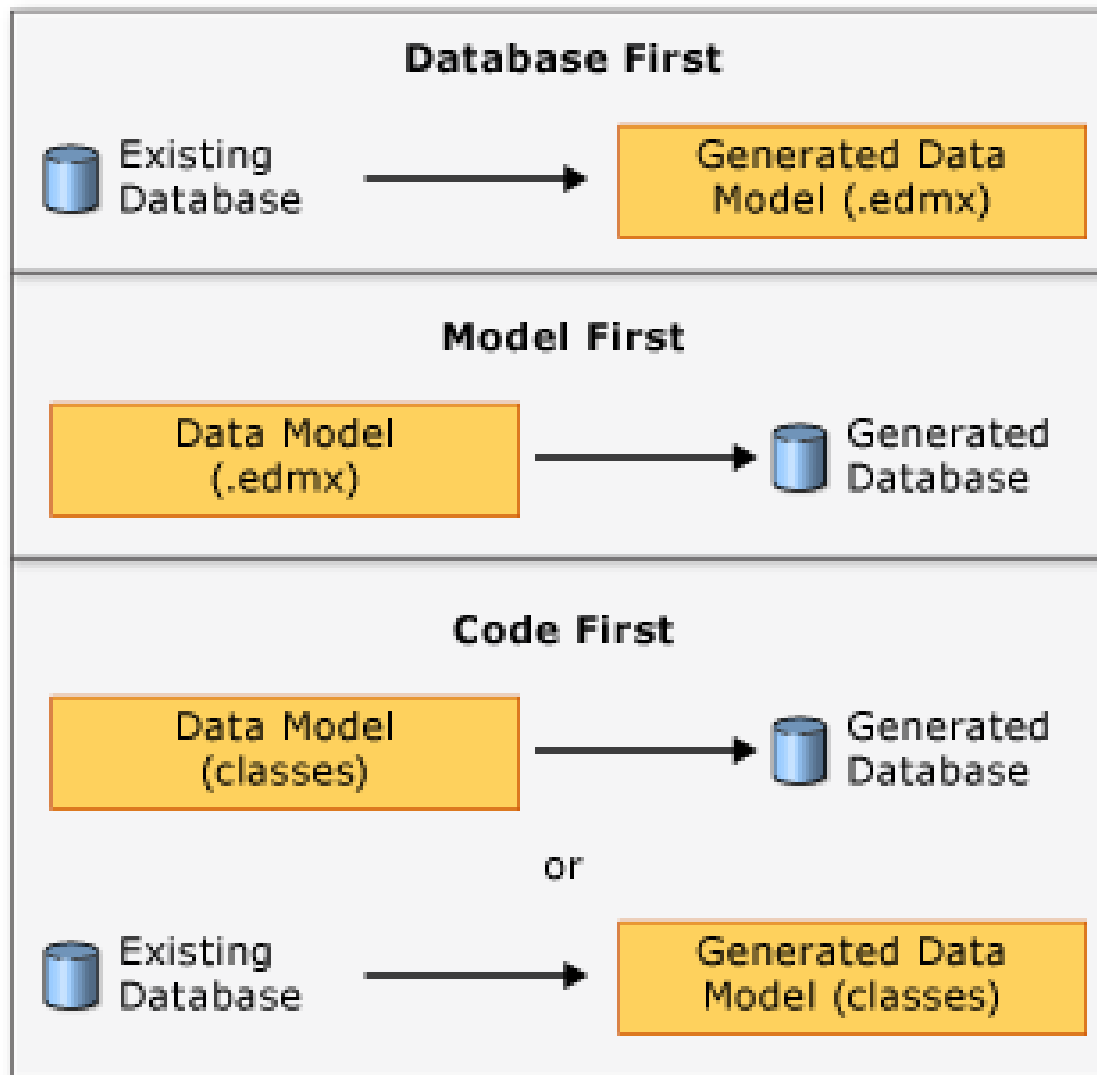
# Kiến trúc Entity Framework



- ◆ **Database First:** Trong trường hợp bạn muốn làm việc với database đã có sẵn. Dùng công cụ thiết kế có sẵn trong Visual Studio để generated từ database ra bản thiết kế model.
- ◆ **Model First:** Trong trường hợp này, chúng ta sẽ làm việc trên visual studio và tạo ra database mới tại đây. Dùng công cụ để generated code từ bản thiết kế model ra database
- ◆ **Code First:** Riêng với cách này bạn có 2 lựa chọn, làm việc với database có sẵn hoặc sẽ tạo mới. Nhưng dù làm với cách nào, chúng ta cũng dùng codebehind để xử lý là chính. Không dùng các tool, giao diện trực quan giống như 2 cách bên trên.

# Các cách sử dụng EF

- ◆ **Database First:** Trờ sẵn. Dùng công cụ t database ra bản thi
- ◆ **Model First:** Trong t và tạo ra database r thiết kế model ra d
- ◆ **Code First:** Riêng vớ sẵn hoặc sẽ tạo mớ codebehind để xử lý giống như 2 cách b



database đã có  
enerated từ

rên visual studio  
d code từ bản

với database có  
a cũng dùng  
iện trực quan

- ◆ Với cách tiếp cận code-first approach, Entity Framework sẽ tạo các đối tượng cơ sở dữ liệu dựa trên model mà bạn tạo để biểu diễn dữ liệu ứng dụng.
- ◆ Một vài quy ước của code-first cho phép tự động cấu hình của một model như sau:
  - ◆ Quy ước tên bảng: Khi bạn tạo một class mô tả các thực thể User sẽ được lưu trữ trong database thì EF mặc định sẽ tạo ra bảng có tên Users.
  - ◆ Quy ước khóa chính: Khi bạn tạo thuộc tính có tên UserId trong lớp User của model, thuộc tính này được nhận là khóa chính. Hơn nữa, EF sẽ thiết lập một cột khóa auto-incrementing để lưu trữ giá trị thuộc tính.
  - ◆ Quy ước về mối quan hệ: EF cung cấp các quy ước khác nhau để nhận biết một mối quan hệ giữa 2 models.



# Ví dụ sử dụng Code-First

- ◆ Giả sử ứng dụng cần lưu trữ thông tin về các quyển sách với các chủ đề khác nhau. Các bước cần thực hiện như sau:
- ◆ **Bước 1** tạo lớp Category trong thư mục Models

```
namespace Session06.Models
{
    0 references
    public class Category
    {
        0 references | 0 exceptions
        public int CategoryId { get; set; }
        0 references | 0 exceptions
        public string CategoryName { get; set; }
        //Thuộc tính điều hướng
        0 references | 0 exceptions
        public virtual ICollection<Book> Books { get; set; }
    }
}
```

# Ví dụ sử dụng Code-First

- ◆ Bước 2: Tạo lớp **Book** trong thư mục Models

```
namespace Session06.Models
{
    1 reference
    public class Book
    {
        0 references | 0 exceptions
        public string BookId { get; set; }
        0 references | 0 exceptions
        public string Title { get; set; }
        0 references | 0 exceptions
        public string Author { get; set; }
        0 references | 0 exceptions
        public int Year { get; set; }
        0 references | 0 exceptions
        public string Publisher { get; set; }
        0 references | 0 exceptions
        public string Picture { get; set; }
        0 references | 0 exceptions
        public int CategoryId { get; set; }
        //Thuộc tính điều hướng
        0 references | 0 exceptions
        public virtual Category Category { get; set; }
    }
}
```

- ◆ Bước 3: Tạo lớp BookStore kế thừa từ lớp DbContext.
- ◆ Lớp **DbContext** nằm trong **System.Data.Entity** của ASP.NET MVC Framework.
  - ◆ Nó được sử dụng để định nghĩa lớp Database Context sau khi tạo model class,
  - ◆ Nó phối hợp với Entity Framework và cho phép bạn truy vấn cũng như lưu trữ dữ liệu vào database.
  - ◆ Mỗi thuộc tính của lớp DbContext có kiểu DbSet <T> sẽ tạo ra một bảng trong Database, T biểu diễn kiểu của đối tượng sẽ lưu trong database.

# Ví dụ sử dụng Code-First

```
namespace Session06.Models
```

```
{
```

1 reference

```
public class BookStore:DbContext
```

```
{
```

0 references | 0 exceptions

```
public BookStore() : base() { }
```

//Khai báo các thuộc tính tương ứng với bảng trong csdl

0 references | 0 exceptions

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

0 references | 0 exceptions

```
public DbSet<Book> Books { get; set; }
```

```
}
```

```
}
```

- ◆ Bước 4: Tạo controller có tên “**CategoryController**” trong thư mục controllers, code cho action Index như sau:



# Ví dụ sử dụng Code-First

- ◆ Bước 4: 1  
controlle

```
namespace Session06.Controllers
{
    0 references
    public class CategoryController : Controller
    {
        // GET: Category
        0 references | 0 requests | 0 exceptions
        public ActionResult Index()
        {
            /* Khởi tạo DataContext - lúc này EF sẽ tìm thông tin kết nối
             * trong file machine.config của .NET Framework cài trên máy bạn và
             * sau đó tạo csdl với tên Session06.Models.BookStore
             */

            BookStore store = new BookStore();
            return View(store.Categories.AsEnumerable());
        }
    }
}
```

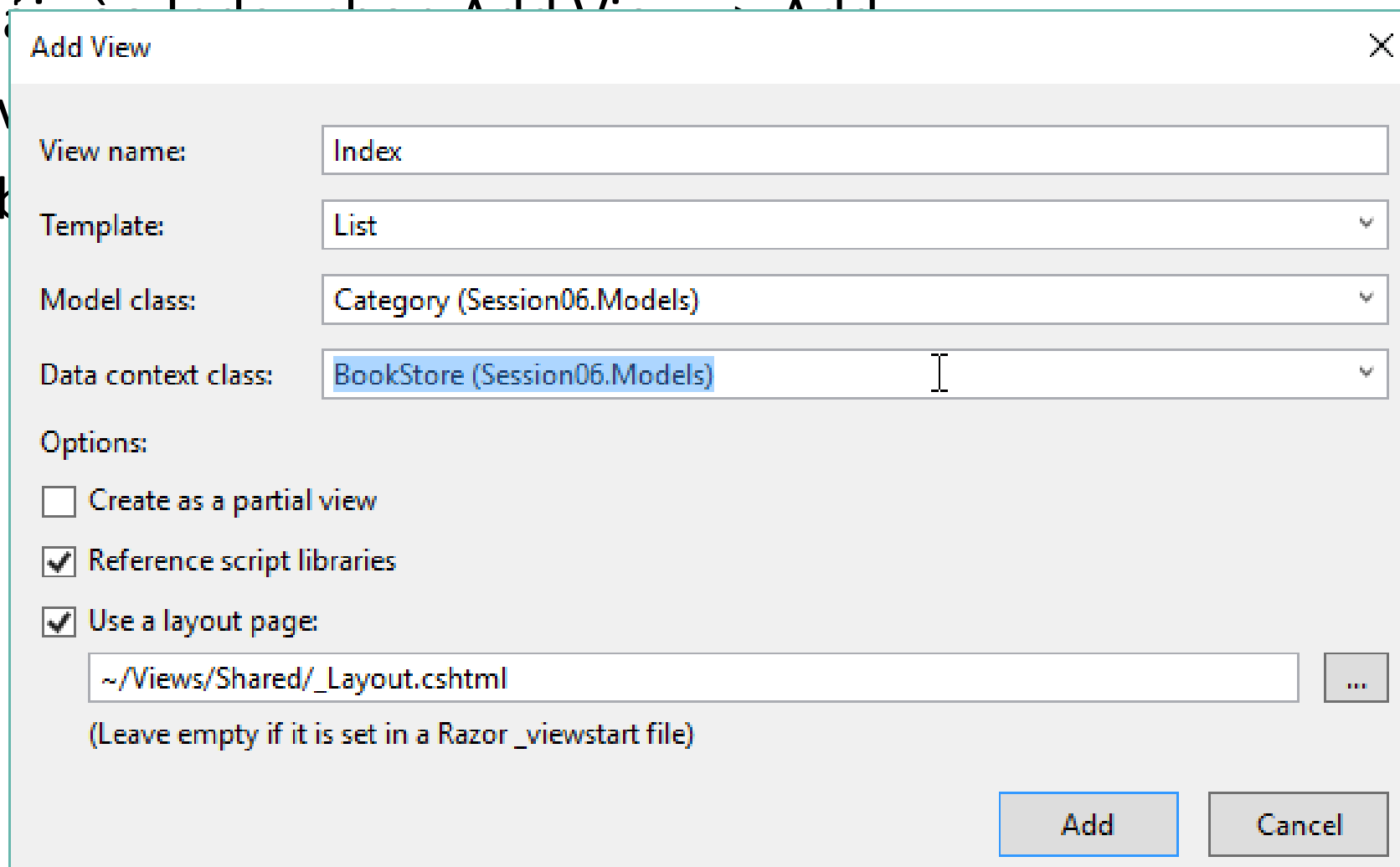
# Ví dụ sử dụng Code-First

- ◆ Bước 5: Tạo view cho action **Index** trong **CategoryController**
- ◆ Kích chuột phải vào **Index** chọn **Add View -> Add**
- ◆ Browse **View** vừa tạo
- ◆ Kiểm tra database trong SQL Server.



# Ví dụ sử dụng Code-First

- ◆ Bước 5: Tạo view cho action Index trong CategoryController
- ◆ Kích chuột phải vào Controller và chọn Add View...
- ◆ Browse View và chọn List
- ◆ Kiểm tra datakey



Add View

View name: Index

Template: List

Model class: Category (Session06.Models)

Data context class: BookStore (Session06.Models)

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

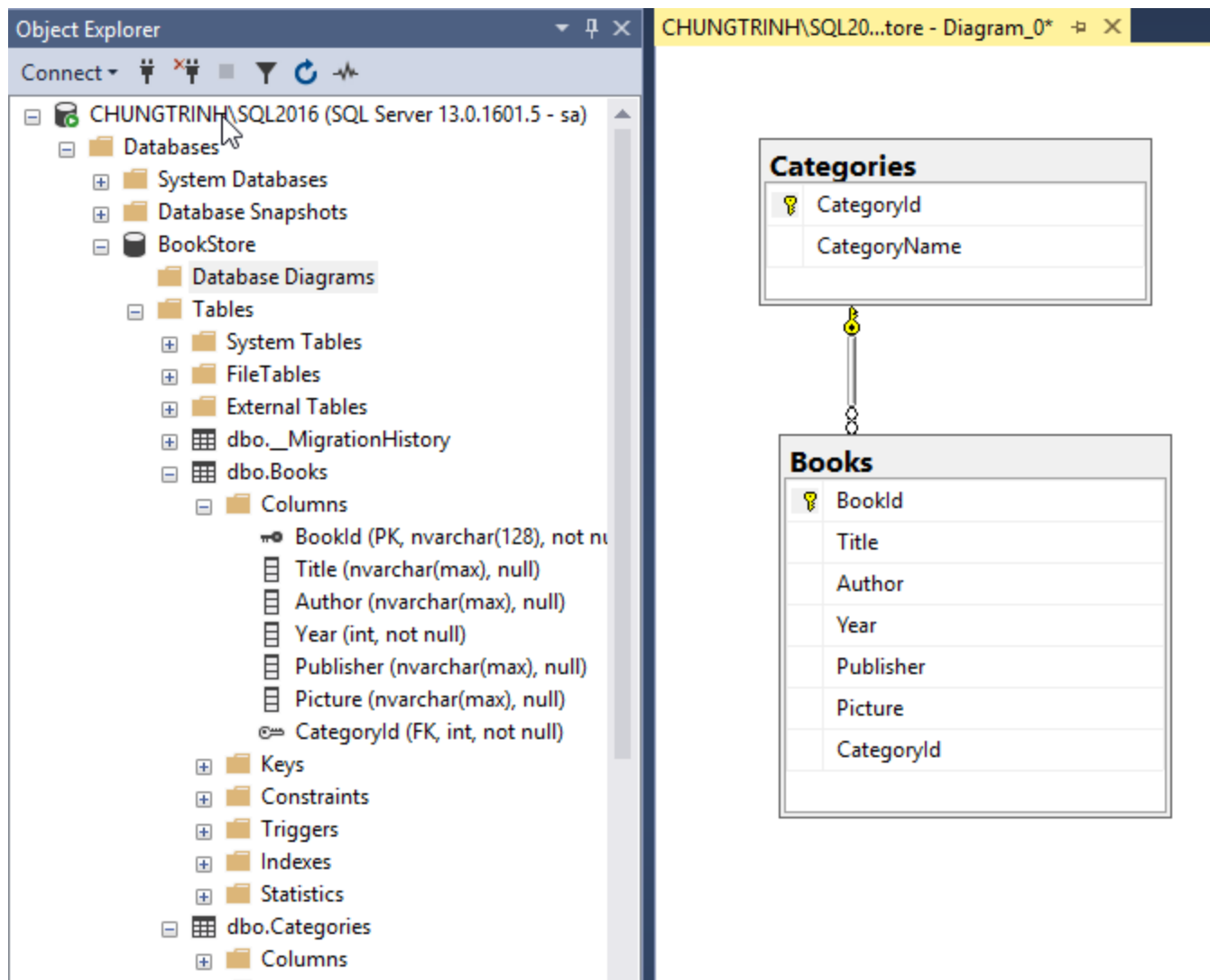
~/Views/Shared/\_Layout.cshtml

(Leave empty if it is set in a Razor \_viewstart file)

Add Cancel



# Ví dụ sử dụng Code-First



- ◆ Trường hợp customize chuỗi kết nối thì bạn bổ sung dòng code vào file Web.config như sau:

```
<connectionStrings>
  <clear/>
  <add name="BookStoreConnect"
        connectionString="Server=CHUNGTRINH\SQL2016; Database=BookStore; uid=sa;pwd=1234#"
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```

- ◆ Khi chạy database tạo ra có tên như hình



- ◆ Trong khi phát triển ứng dụng ASP.NET MVC Web có những lúc bạn cần thay đổi lại Model classes, lúc này cần đảm bảo database cũng phải thay đổi theo. Để duy trì sự đồng bộ này bạn cần tạo lại database.
- ◆ EF cung cấp 2 lớp trong namespace System.Data.Entity cho phép tạo lại database:
  - ◆ DropCreateDatabaseAlways: Cho phép tạo lại database bất cứ khi nào ứng dụng start.
  - ◆ DropCreateDatabaseIfModelChanges: Cho phép tạo lại database bất cứ khi nào Model class thay đổi.
- ◆ Mở tệp tin Global.asax và bổ sung dòng code như sau



- ◆ Trong khi phát triển ứng dụng ASP.NET MVC Web có những lúc bạn cần thay

```
namespace Session06
{
    - references
    public class MvcApplication : System.Web.HttpApplication
    {
        - references | 0 exceptions
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            //Tạo database bất cứ khi nào thay đổi
            Database.SetInitializer(new DropCreateDatabaseIfModelChanges<BookStore>());
        }
    }
}
```

- ◆ Tạo thêm lớp Publisher và thay đổi lại Model class như sau bạn sẽ thấy kết quả ngay khi chạy.

```
namespace Session06.Models
{
    - references
    public class Publisher
    {
        - references | 0 exceptions
        public int PublisherId { get; set; }
        - references | 0 exceptions
        public string PublisherName { get; set; }
        - references | 0 exceptions
        public string Phone { get; set; }
        - references | 0 exceptions
        public string Address { get; set; }
    }
}
```

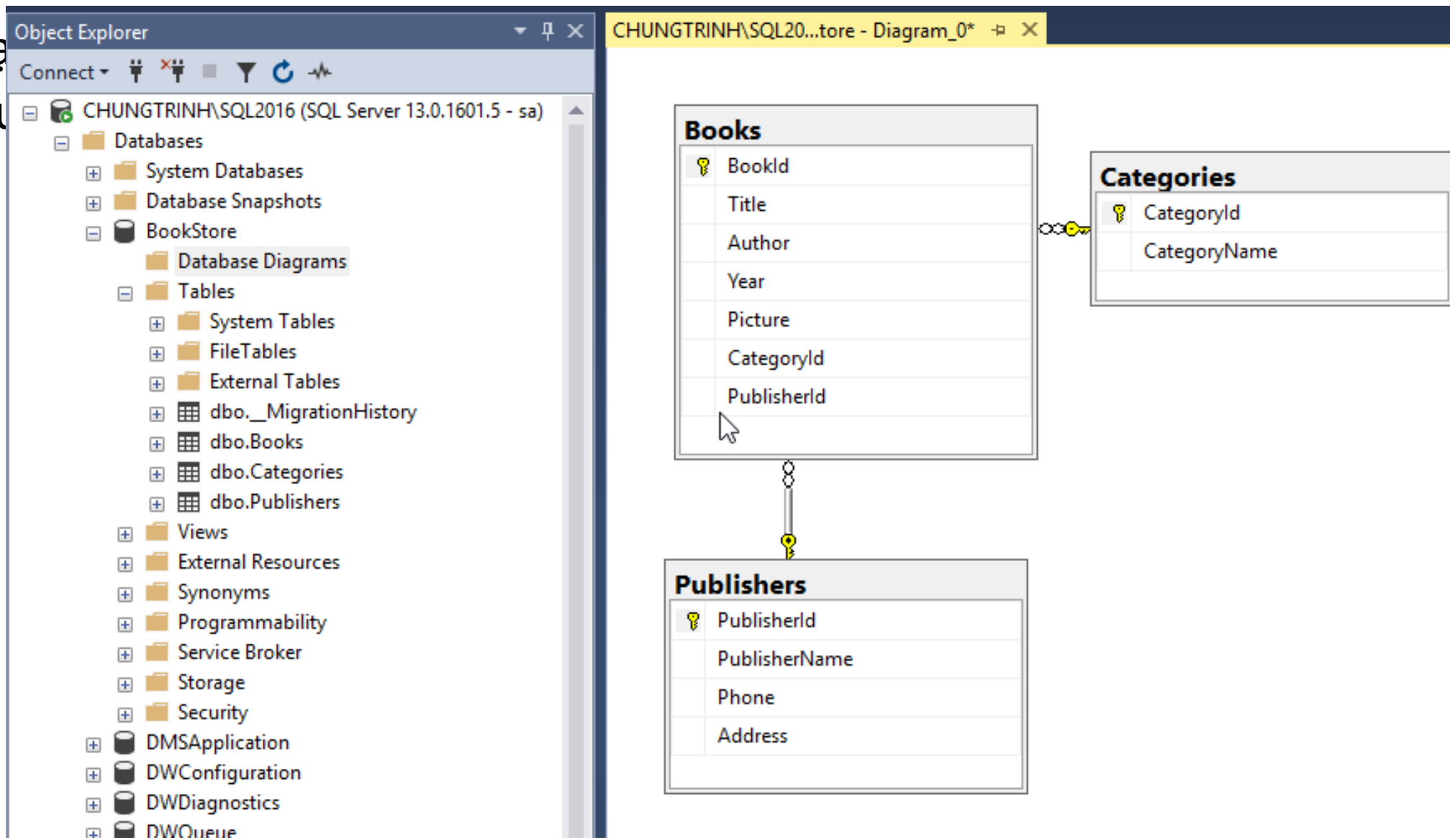
```
namespace Session06.Models
{
    - references
    public class Book
    {
        - references | 0 exceptions
        public string BookId { get; set; }
        - references | 0 exceptions
        public string Title { get; set; }
        - references | 0 exceptions
        public string Author { get; set; }
        - references | 0 exceptions
        public int Year { get; set; }
        - references | 0 exceptions
        public string Picture { get; set; }
        - references | 0 exceptions
        public int CategoryId { get; set; }
        - references | 0 exceptions
        public int PublisherId { get; set; }
        //Thuộc tính điều hướng
        - references | 0 exceptions
        public virtual Category Category { get; set; }
        - references | 0 exceptions
        public virtual Publisher Publisher { get; set; }
    }
}
```

- ◆ Tạo thêm lớp Publisher và thay đổi lại Model class như sau bạn sẽ thấy kết quả ngay khi chạy.

```
namespace Session06.Models
{
    - references
    public class BookStore:DbContext
    {
        - references | 0 exceptions
        public BookStore() : base("BookStoreConnect") { }
        //Khai báo các thuộc tính tương ứng với bảng trong csdl
        - references | 0 exceptions
        public DbSet<Category> Categories { get; set; }
        - references | 0 exceptions
        public DbSet<Book> Books { get; set; }
        - references | 0 exceptions
        public DbSet<Publisher> Publishers { get; set; }
    }
}
```

# Tạo lại database khi thay đổi Model class

- ◆ Tạo lại database khi thay đổi Model class

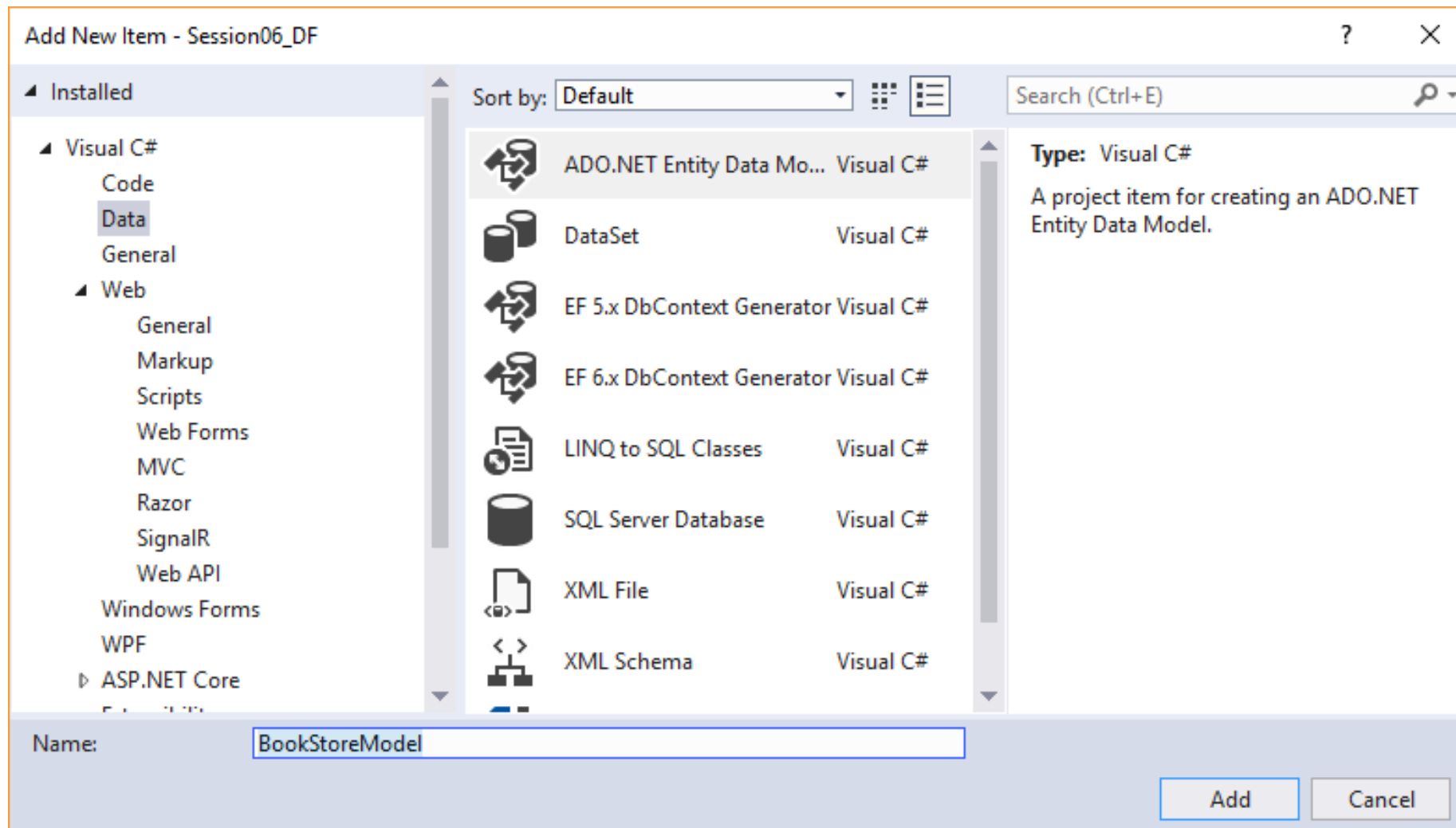


- ◆ Trong trường hợp bạn muốn làm việc với database đã có sẵn, dùng công cụ thiết kế có sẵn trong Visual Studio để sinh ra ra bản thiết kế model từ database bạn sẽ sử dụng phương pháp **Database First**.
- ◆ Model sinh ra có thể được cập nhật lược đồ bất cứ khi nào cơ sở dữ liệu thay đổi. Ngoài ra, phương pháp tiếp cận Database First hỗ trợ thủ tục lưu trữ, view, ...
- ◆ Thông thường thì một **developer** thường chọn Database First để triển khai ứng dụng, vì nó tương đối đơn giản và cách làm gần tương tự LINQ to SQL; Model First thì cũng tương đối khó hơn Database First, và Code First lại càng khó khăn hơn cho những bạn lập trình viên mới bắt đầu làm việc với ASP.NET



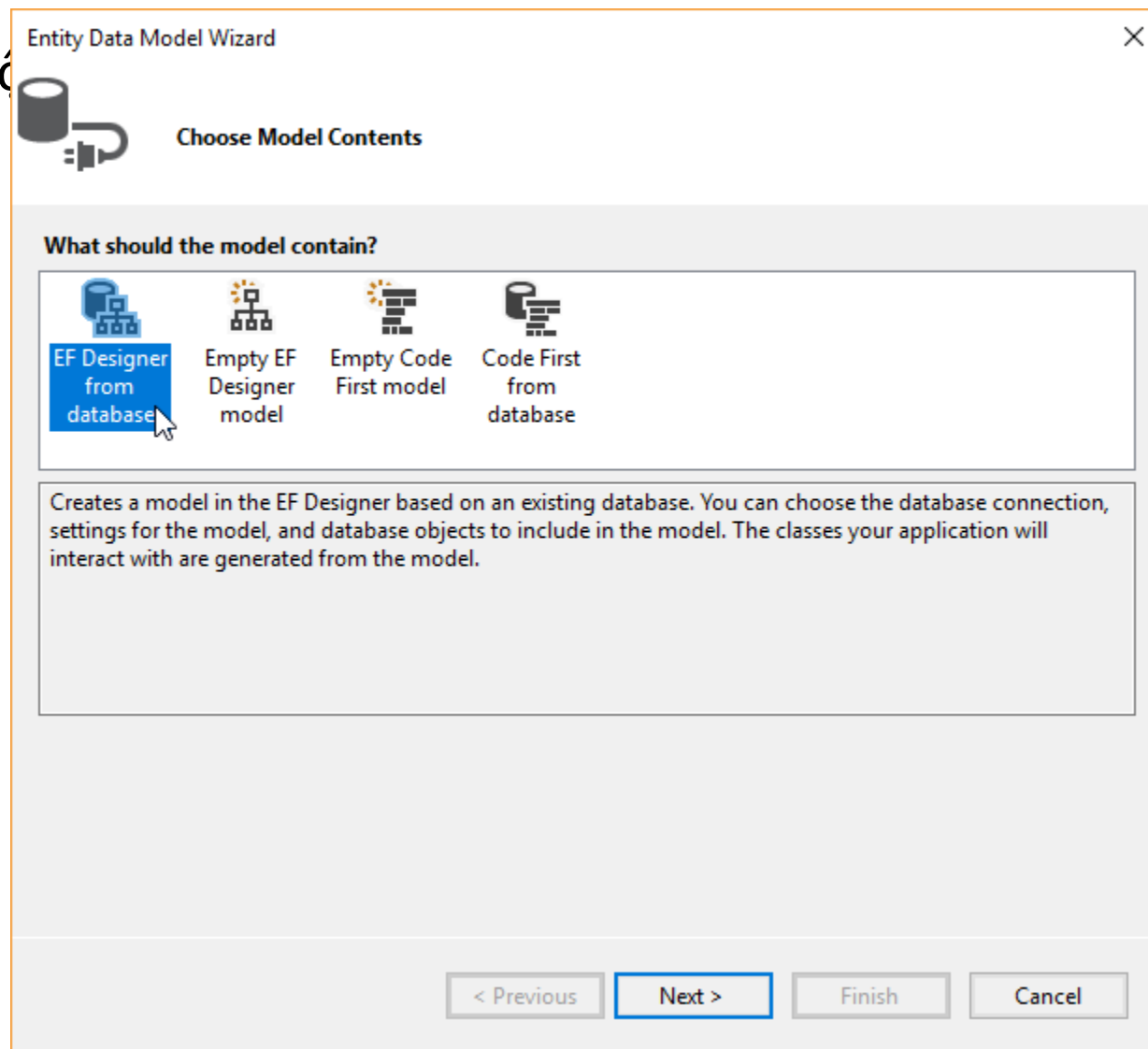
# Ví dụ sử dụng Database-First

- ◆ Bước 1: Kích chuột phải vào thư mục Models -> Add -> New Item



# Ví dụ sử dụng Database-First

- ◆ Bước 1: Kích chuột



# Ví dụ sử dụng Database-First

## ◆ Bước 1: Kích chuột

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:  
Microsoft SQL Server (SqlClient) Change...

Server name:  
CHUNGTRINH\SQL2016 Refresh

Log on to the server

Authentication: SQL Server Authentication

User name: sa

Password: ●●●●●

☐ Save my password

Connect to a database

☒ Select or enter a database name  
BookStore

☐ Attach a database file:  
Browse...

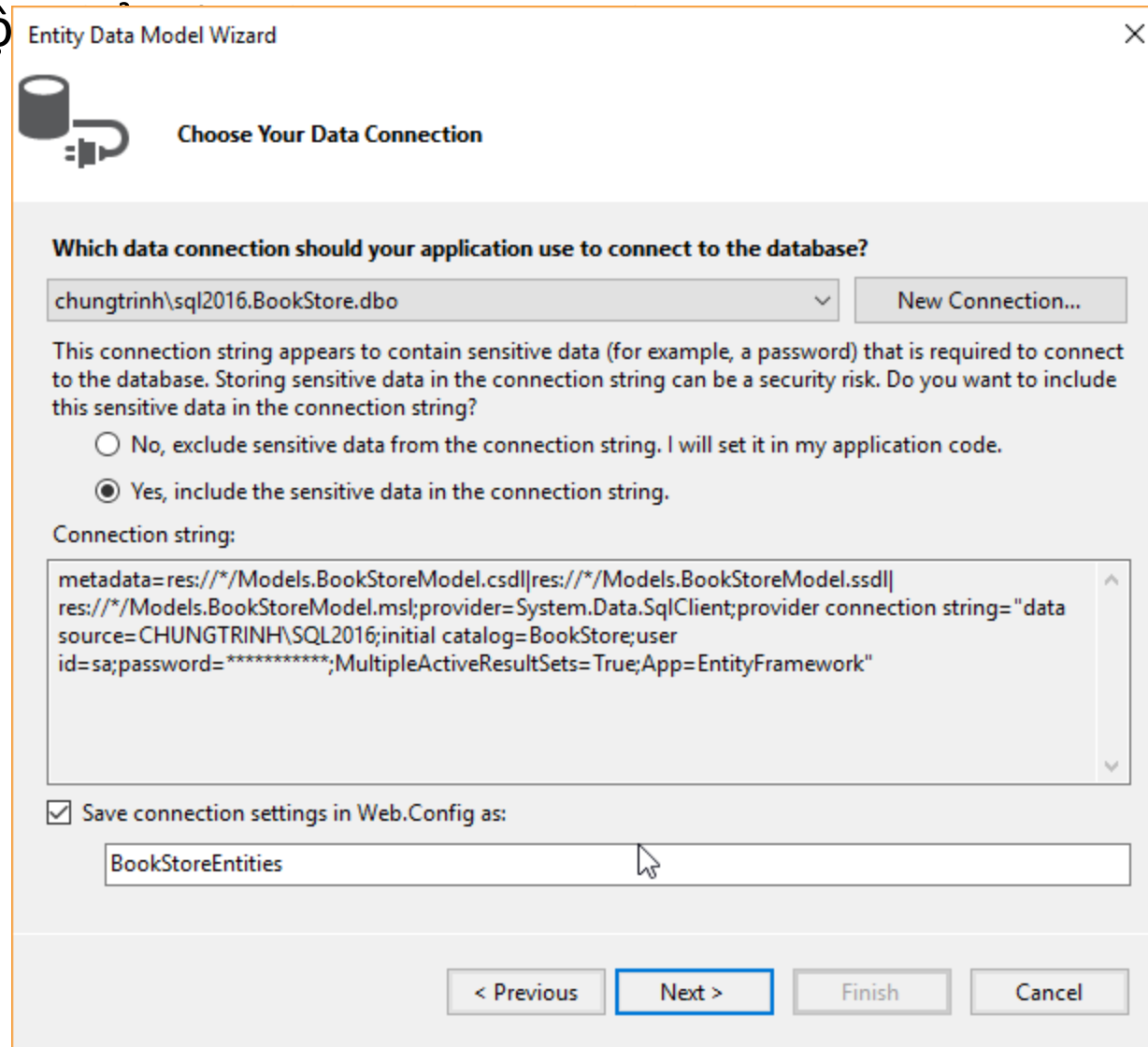
Logical name:

Advanced...

Test Connection OK Cancel

# Ví dụ sử dụng Database-First

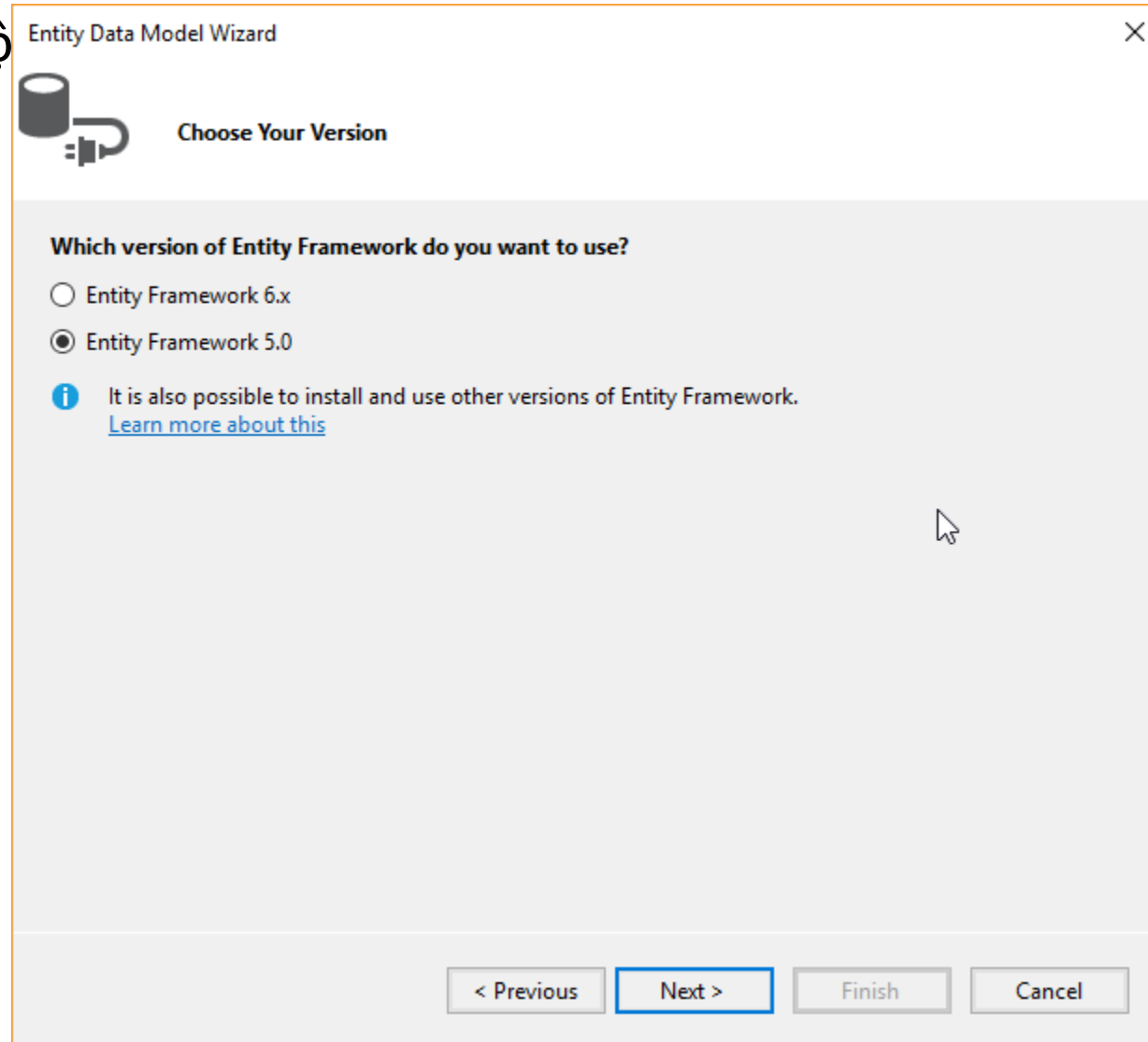
## ◆ Bước 1: Kích chuỗi



The image shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The title bar reads 'Entity Data Model Wizard' with a close button. Below the title bar is a database icon and the text 'Choose Your Data Connection'. The main content area asks 'Which data connection should your application use to connect to the database?'. There is a dropdown menu showing 'chungtrinh\sql2016.BookStore.dbo' and a 'New Connection...' button. Below this, a warning message states: 'This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?'. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' (unselected) and 'Yes, include the sensitive data in the connection string.' (selected). Below the radio buttons is a text box labeled 'Connection string:' containing the following text: `metadata=res://*/Models.BookStoreModel.csdl|res://*/Models.BookStoreModel.ssdl|res://*/Models.BookStoreModel.msl;provider=System.Data.SqlClient;provider connection string="data source=CHUNGTRINH\SQL2016;initial catalog=BookStore;user id=sa;password=*****;MultipleActiveResultSets=True;App=EntityFramework"`. At the bottom, there is a checkbox 'Save connection settings in Web.Config as:' which is checked. Below it is a text box containing 'BookStoreEntities'. At the very bottom are four buttons: '< Previous', 'Next >' (highlighted with a blue border), 'Finish', and 'Cancel'.

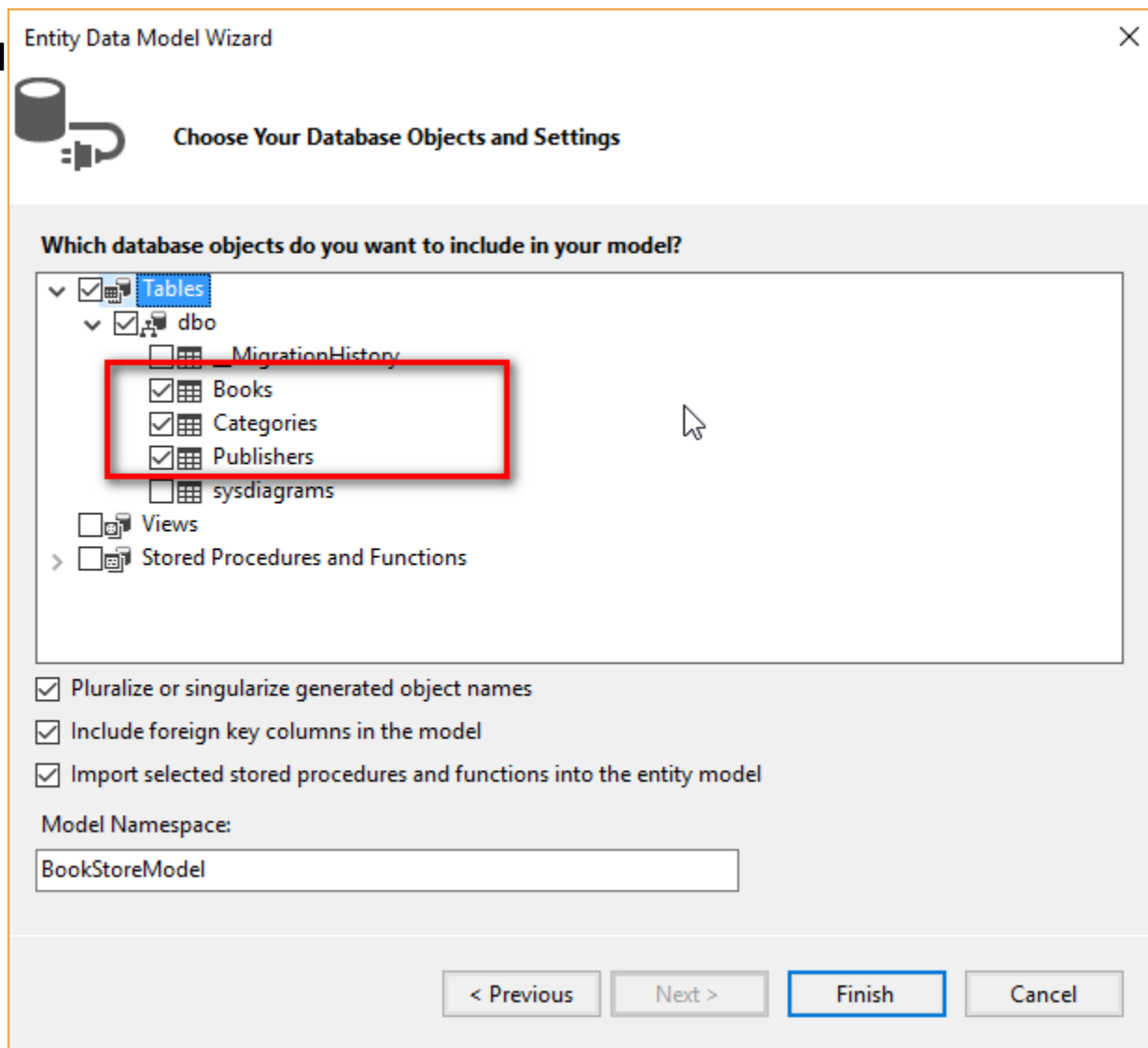
# Ví dụ sử dụng Database-First

## ◆ Bước 1: Kích chuột



# Ví dụ sử dụng Database-First

## ◆ Bước 1: Kích chu



The image shows the 'Entity Data Model Wizard' dialog box, titled 'Choose Your Database Objects and Settings'. It contains a tree view of database objects. The 'Tables' folder is expanded, and the 'dbo' folder is also expanded. A red rectangle highlights the 'Books', 'Categories', and 'Publishers' tables, which are all checked. Other objects like 'MigrationHistory', 'sysdiagrams', 'Views', and 'Stored Procedures and Functions' are also visible but not selected. Below the tree view, there are three checked options: 'Pluralize or singularize generated object names', 'Include foreign key columns in the model', and 'Import selected stored procedures and functions into the entity model'. At the bottom, the 'Model Namespace' is set to 'BookStoreModel'. The 'Finish' button is highlighted with a blue border.

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- ☒ Tables
  - ☒ dbo
    - ☐ MigrationHistory
    - ☒ Books
    - ☒ Categories
    - ☒ Publishers
    - ☐ sysdiagrams
  - ☐ Views
  - ☐ Stored Procedures and Functions

☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☒ Import selected stored procedures and functions into the entity model

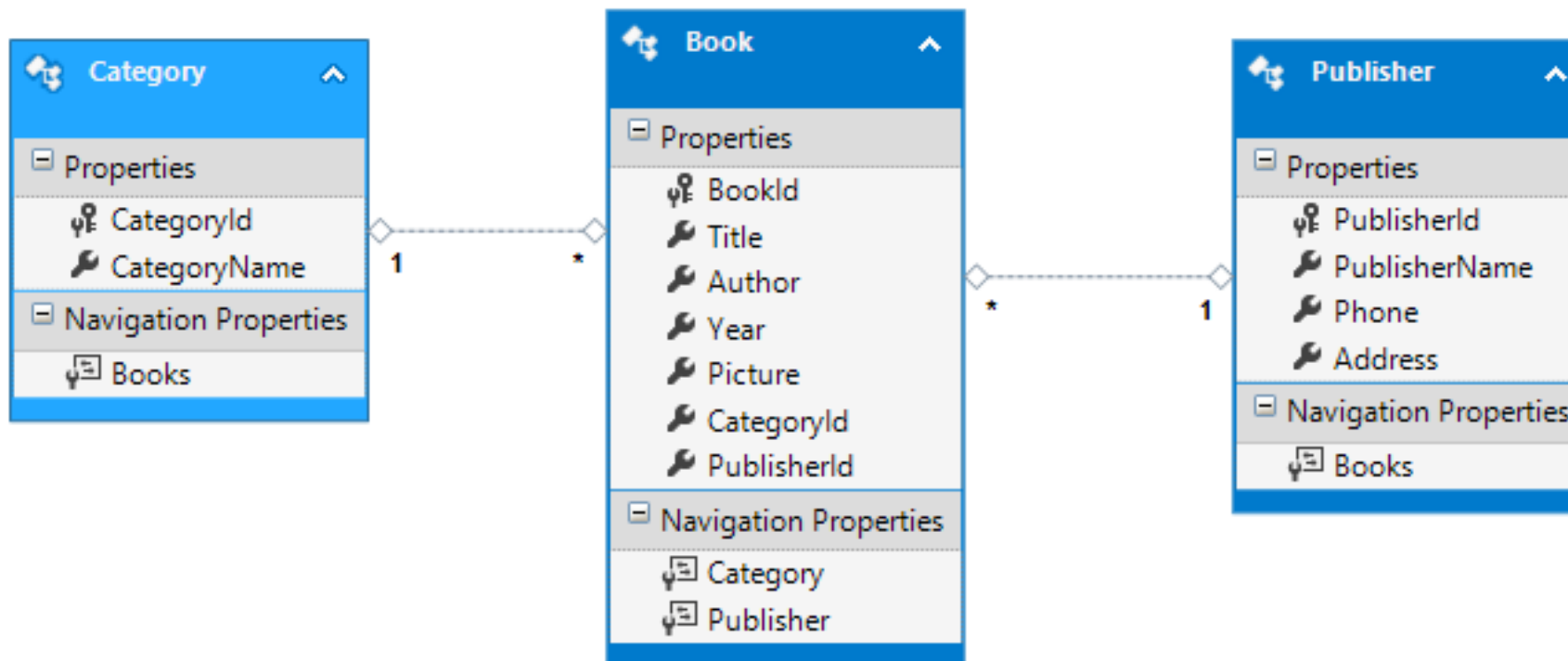
Model Namespace:

BookStoreModel

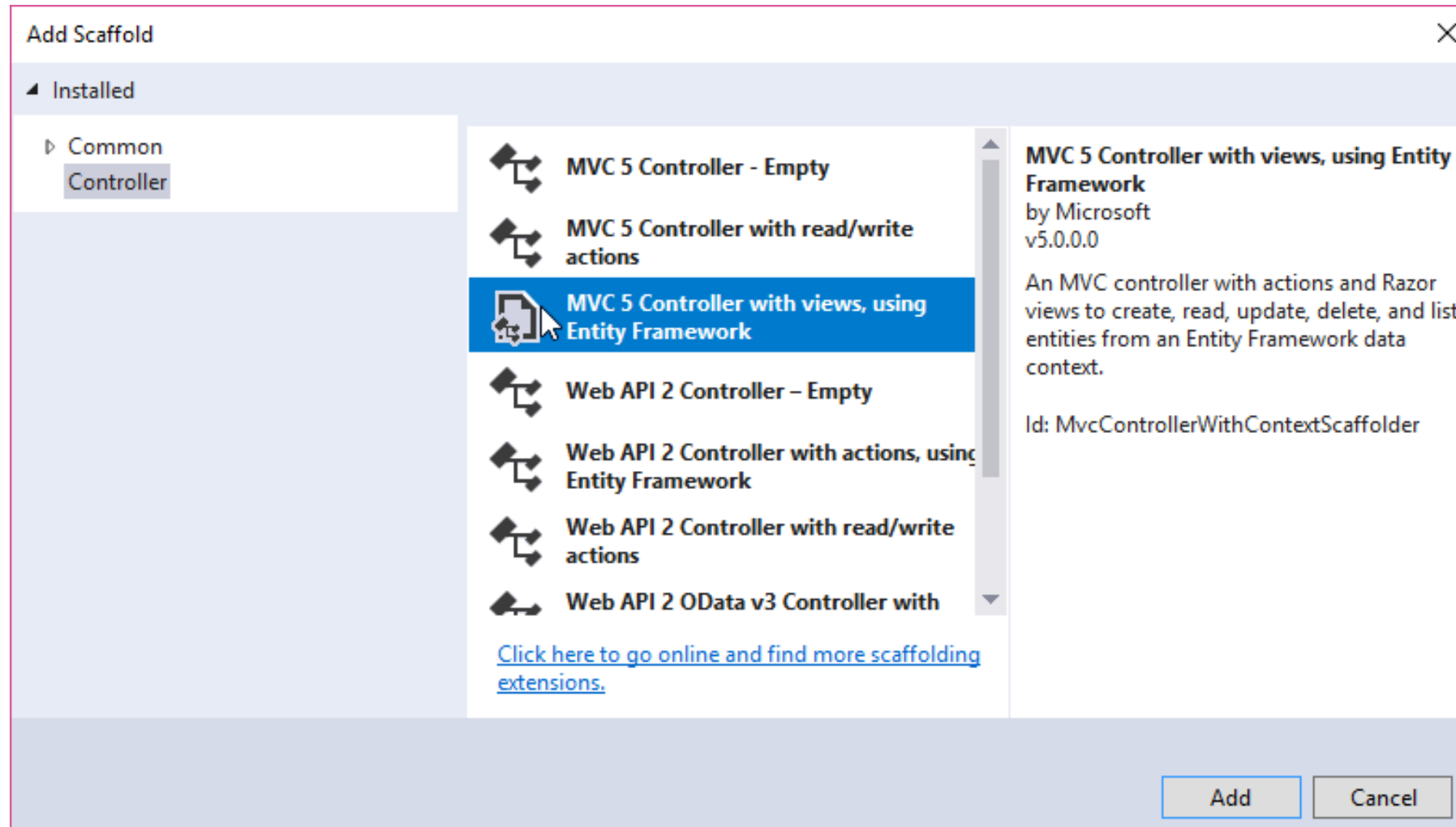
< Previous Next > Finish Cancel

# Ví dụ sử dụng Database-First

- ◆ Bước 1: Kích chuột phải vào thư mục Models -> Add -> New Item



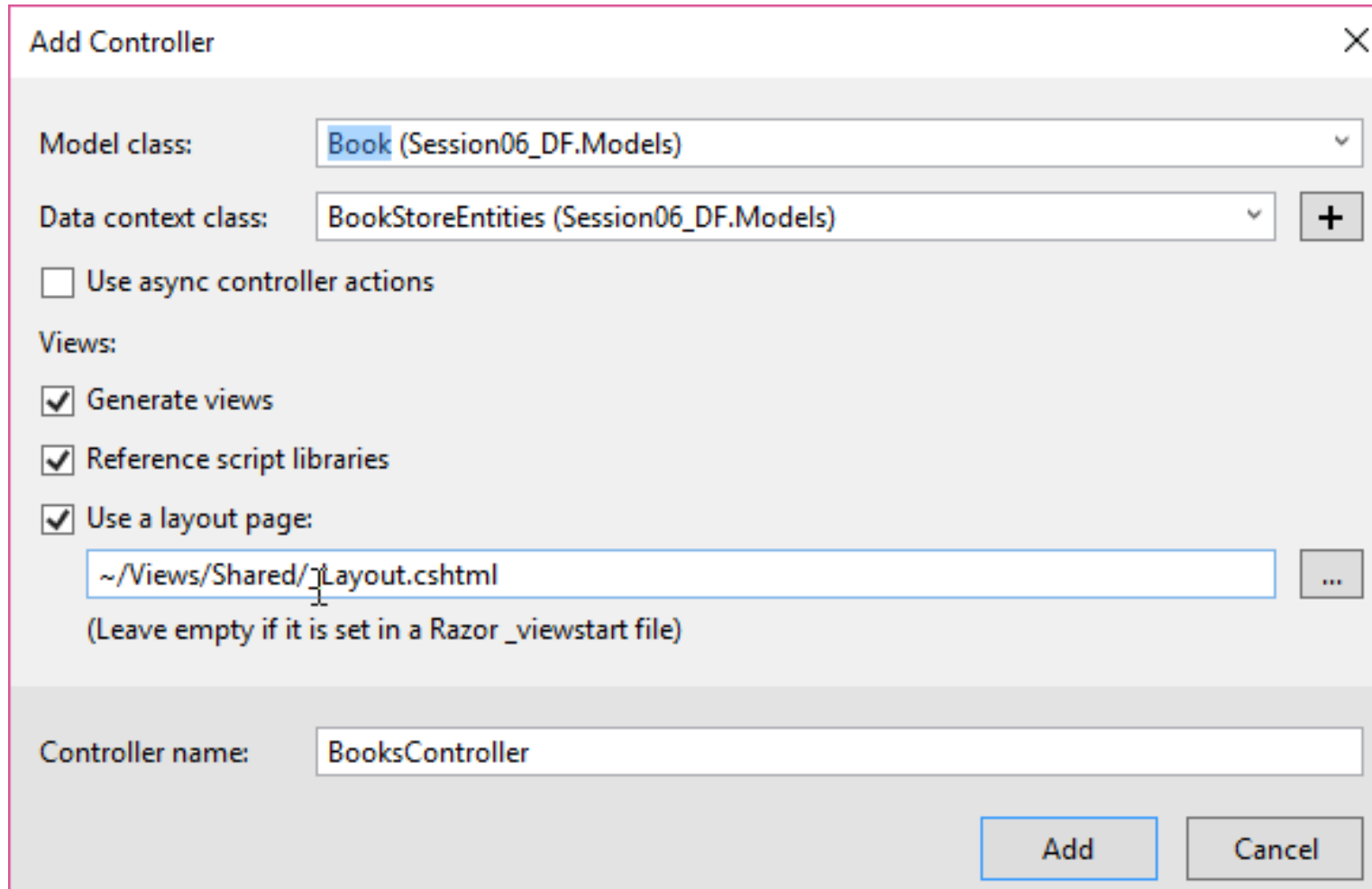
- ◆ Kích chuột phải vào thư mục Controllers -> Add -> Controller -> ...





# Tạo controller với Views sử dụng Entity Framework

- ◆ Kích chuột phải vào thư mục Controllers -> Add -> Controller -> ...



Add Controller

Model class:

Data context class:

☐ Use async controller actions

Views:

☒ Generate views

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

Controller name:

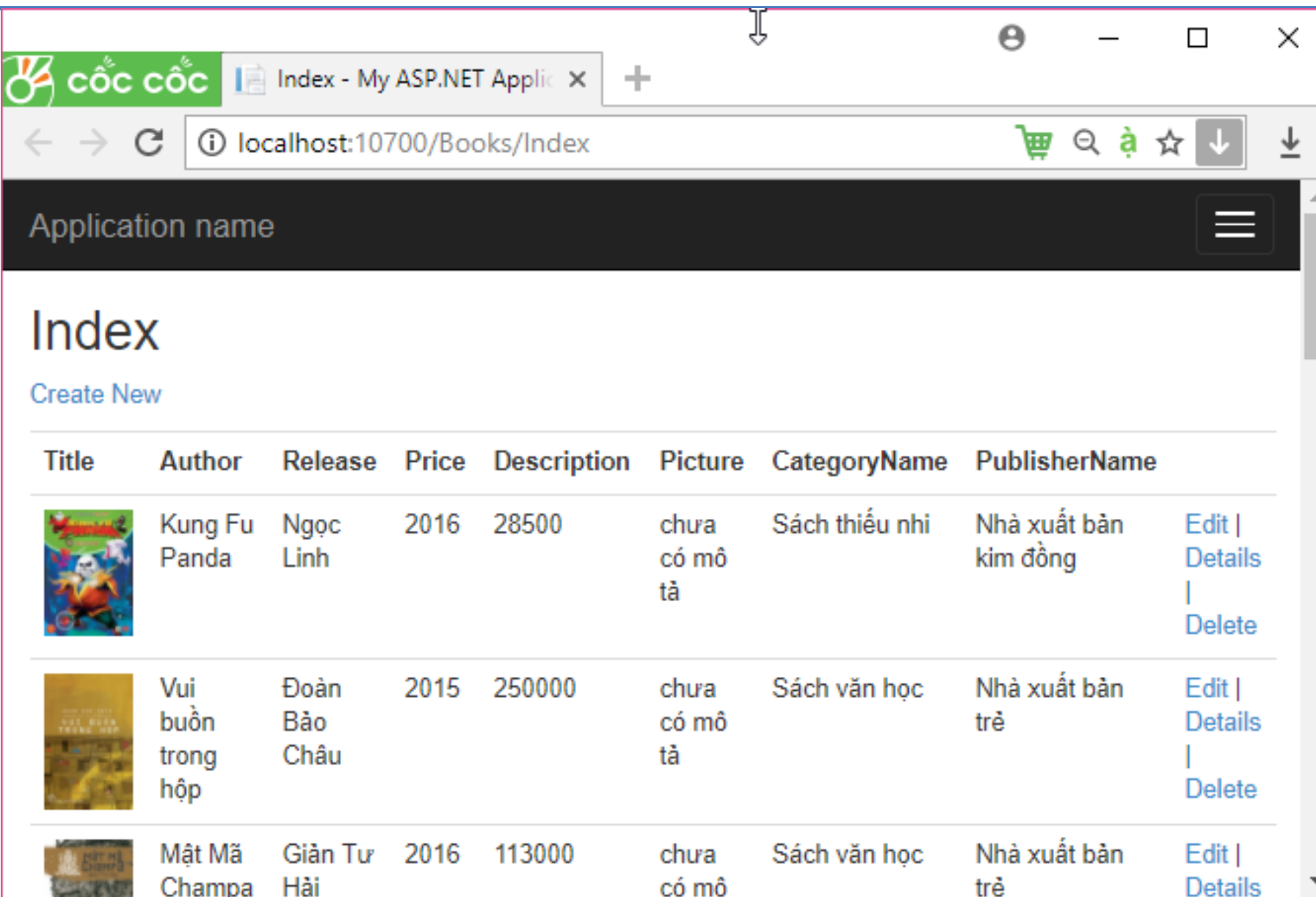
- ◆ Kích chuột phải vào thư mục Controllers -> Add -> Controller -> ...




Kích vào nút Add, -> Với tính năng Scaffolding của Visual Studio, các view và code logic trong controller dạng List, Create, Delete, Detail, Edit tự động được sinh ra, code logic sẽ sử dụng LINQ -> hãy chạy và kiểm tra kết quả



# Tạo controller với Views sử dụng Entity Framework

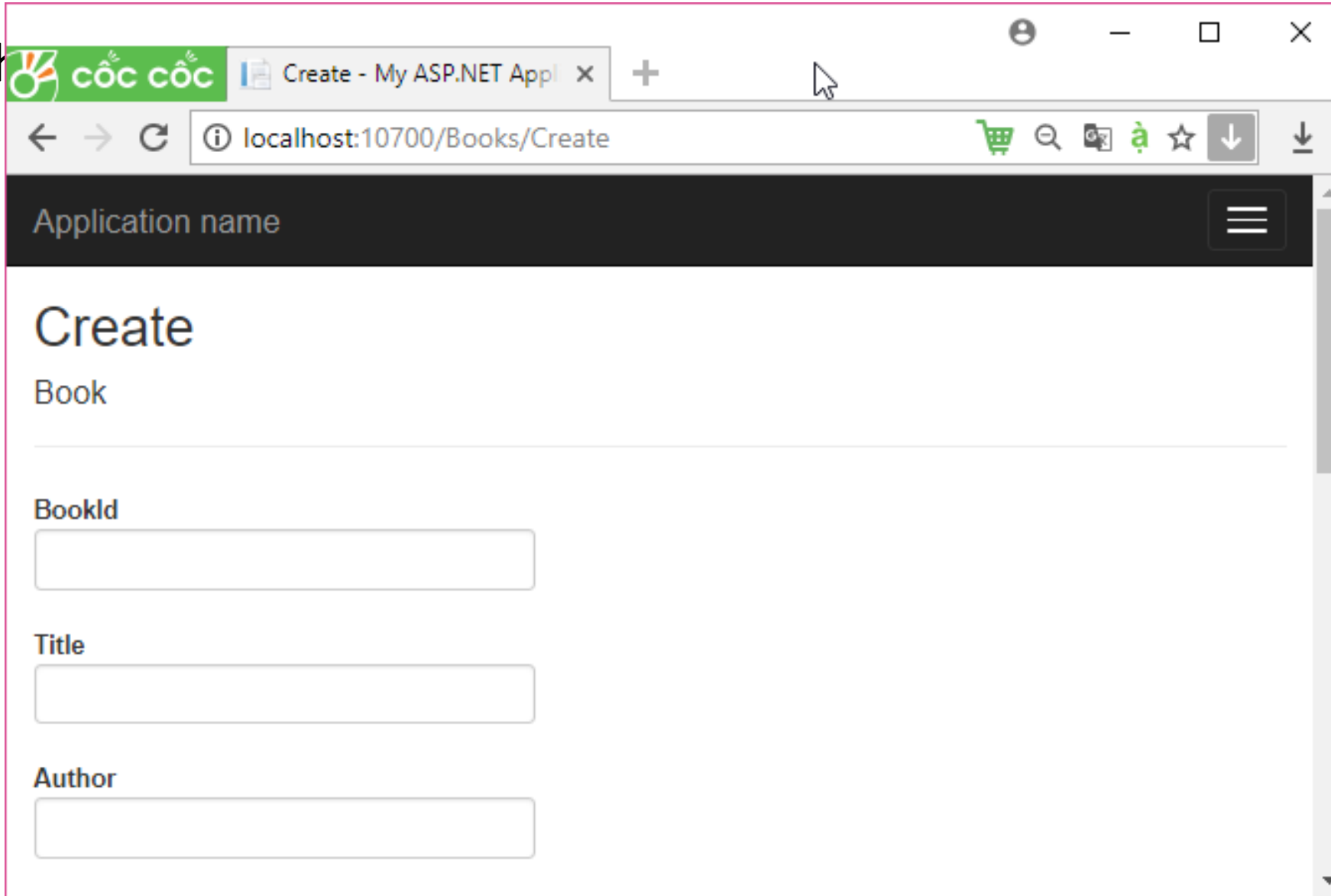
## ◆ Kích



Title	Author	Release	Price	Description	Picture	CategoryName	PublisherName	
	Kung Fu Panda	Ngọc Linh	2016	28500	chưa có mô tả	Sách thiếu nhi	Nhà xuất bản kim đồng	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
	Vui buồn trong hộp	Đoàn Bảo Châu	2015	250000	chưa có mô tả	Sách văn học	Nhà xuất bản trẻ	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
	Mật Mã Champa	Giản Tư Hải	2016	113000	chưa có mô tả	Sách văn học	Nhà xuất bản trẻ	<a href="#">Edit</a>   <a href="#">Details</a>

# Tạo controller với Views sử dụng Entity Framework

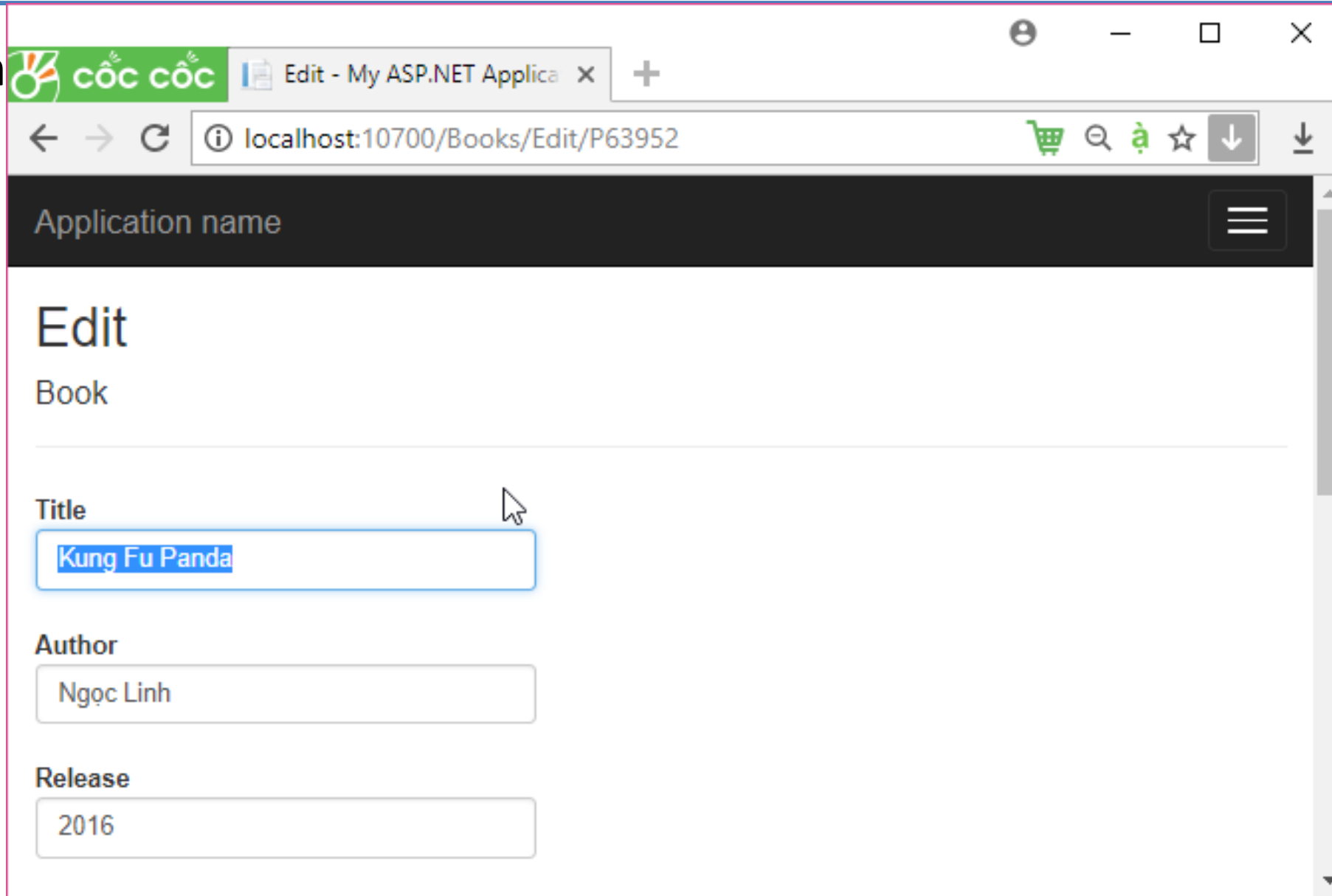
◆ Kích ch



The screenshot shows a web browser window with the address bar displaying 'localhost:10700/Books/Create'. The page has a dark header with the text 'Application name' and a hamburger menu icon. Below the header, the main content area is titled 'Create Book'. There are three input fields: 'BookId', 'Title', and 'Author', each with a label above it. The 'BookId' field is empty, while the 'Title' and 'Author' fields contain placeholder text.

# Tạo controller với Views sử dụng Entity Framework

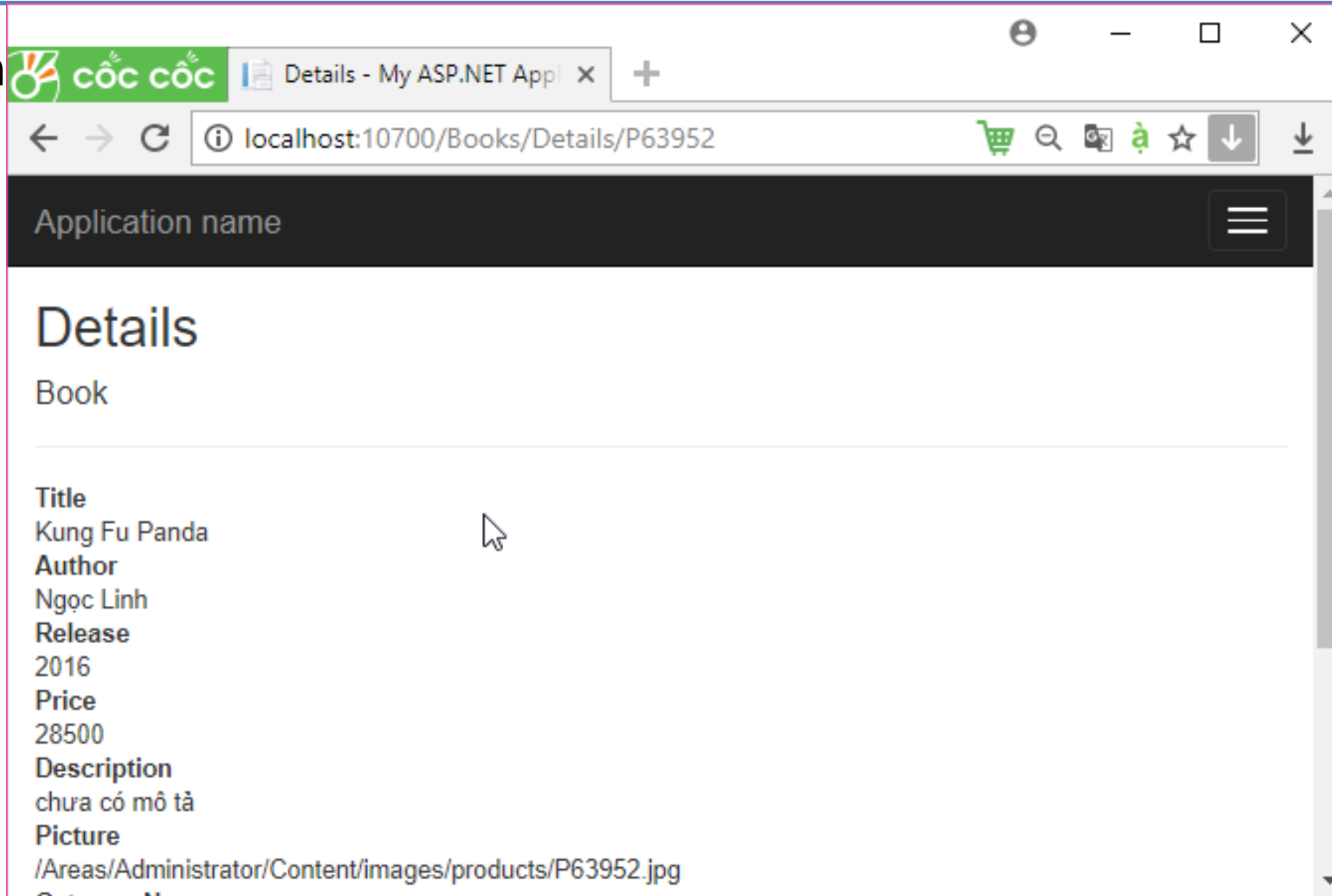
## ◆ Kích



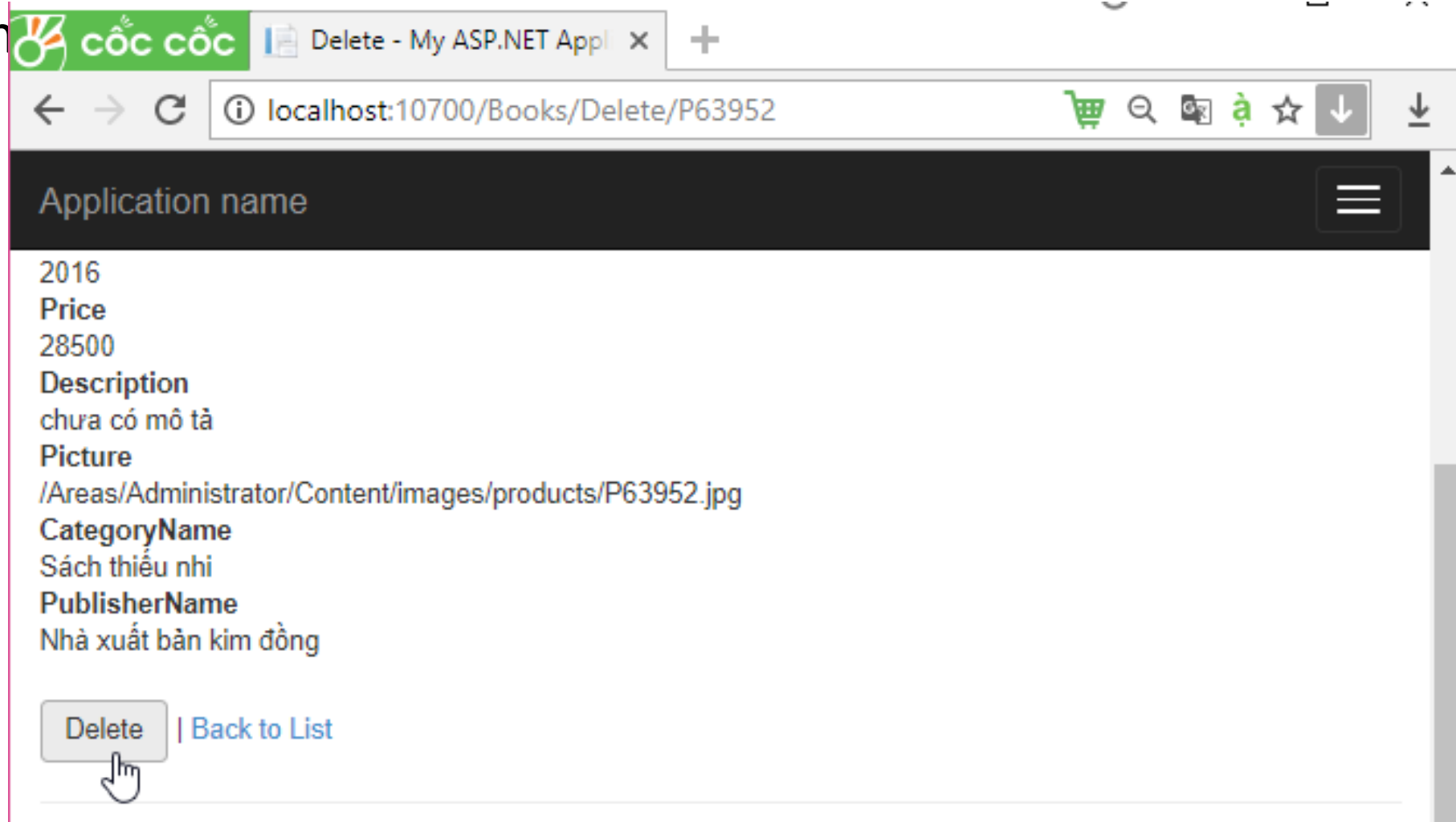
The screenshot shows a web browser window with the address bar displaying 'localhost:10700/Books/Edit/P63952'. The page title is 'Edit Book'. The form contains three input fields: 'Title' with the value 'Kung Fu Panda', 'Author' with the value 'Ngọc Linh', and 'Release' with the value '2016'. A mouse cursor is hovering over the 'Title' field.

# Tạo controller với Views sử dụng Entity Framework

◆ Kích



## ◆ Kích



Cuối cùng bạn chỉ cần Customize lại View học Code logic trong controller nếu muốn





- ◆ LINQ là một tập các API cho phép liệu cho phép bạn viết câu truy vấn ngay trong ngôn ngữ lập trình như C#, VB để truy cập và thao tác với các nguồn dữ liệu một cách độc lập.
- ◆ Truy vấn hoạt động trên strongly-typed collection của các đối tượng với sự hỗ trợ của các từ khóa ngôn ngữ và các toán tử chung.



- ◆ Bước 1 cung cấp nguồn dữ liệu
- ◆ Bước 2 Tạo câu truy vấn, thực thi và nhận kết quả, kết quả nhận về ở dạng `IEnumerable<T>` hoặc `IQueryable<T>`
- ◆ Bước 3 Đọc và xử lý kết quả
- ◆ Ví dụ

```
IQueryable<Book> books = from b in db.Books select b;  
foreach (var book in books)  
{  
    Console.WriteLine(book.Title);  
}
```



- ◆ Ngoài việc nhận dữ liệu từ các nguồn dữ liệu bạn cũng có thể sử dụng LINQ để thực hiện các hành động khác nhau trên nguồn dữ liệu được lưu trữ, một vài hành động thường dùng như:
  - ◆ Forming Projections
  - ◆ Filtering the data
  - ◆ Sorting the data
  - ◆ Grouping the data

Xem demo

- ◆ Thay vì viết các câu truy vấn LINQ, Microsoft đã bổ sung một loạt các phương thức mở rộng cho phép bạn truy vấn trực tiếp bằng cách gọi các phương thức này và truyền vào tham số là biểu thức lambda.
- ◆ Một số phương thức phổ biến như:
  - ◆ Select, Where, OrderBy, Find, FirstOrDefault, Include, Take, SingleOrDefault, Skip...

Xem demo

# Tích hợp CKEDITOR và CKFINDER

- ◆ CKEditor là một web text editor cung cấp giao diện soạn thảo đơn giản dễ dung trên môi trường web...
- ◆ CKFinder là một component tích hợp trên web và cho phép duyệt tệp tin, upload tệp tin...

Xem demo



