

BÀI 4. ENTITY FRAMEWORK TRONG ASP.NET MVC

- Mục đích: Hướng dẫn sinh viên tìm hiểu và sử dụng Entity Framework để kết nối với cơ sở dữ liệu.

- Yêu cầu: Sau bài học sinh viên có khả năng lập trình kết nối với cơ sở dữ liệu sử dụng Entity Framework xây dựng các chức năng xem, thêm, sửa, xóa (CRUD) các dòng trong bảng.

- Hình thức tổ chức dạy học: Lý thuyết, tự học

- Thời gian: Lý thuyết (trên lớp: 3; online: 3); Tự học, tự nghiên cứu: 10

- **Nội dung chính:**

BÀI 4. ENTITY FRAMEWORK TRONG ASP.NET MVC.....	1
1. Entity Framework là gì	1
2. Lịch sử phát triển Entity Framework	2
3. Kiến trúc tổ chức	2
4. Các mô hình lập trình Entity Framework	3
5. Mô hình Code first	4
6. Tạo một ứng dụng ASP .NET MVC với cách tiếp cận code first	6
7. Làm việc với cơ sở dữ liệu đã có bằng Entity Framework Code First.....	7
8. Hướng dẫn thực hành:.....	7
8.1. Tạo một ứng dụng ASP .NET MVC với cách tiếp cận code first.....	7
8.2. Tạo ứng dụng ASP .NET MVC với chức năng CRUD trên bảng theo mô hình Code First	14

1. Entity Framework là gì

- Entity Framework (EF) là một khung ORM (Object Relational Mapping) mã nguồn mở cho các ứng dụng .NET.
- Nó cho phép làm việc với dữ liệu bằng cách sử dụng các đối tượng thuộc các lớp thực thể mà không cần sử dụng trực tiếp các bảng và cột lưu trữ dữ liệu.
- Với Entity Framework, các nhà phát triển có thể làm việc ở mức độ trừu

tượng cao hơn và duy trì các ứng dụng hướng dữ liệu với ít mã hơn so với các ứng dụng truyền thống.

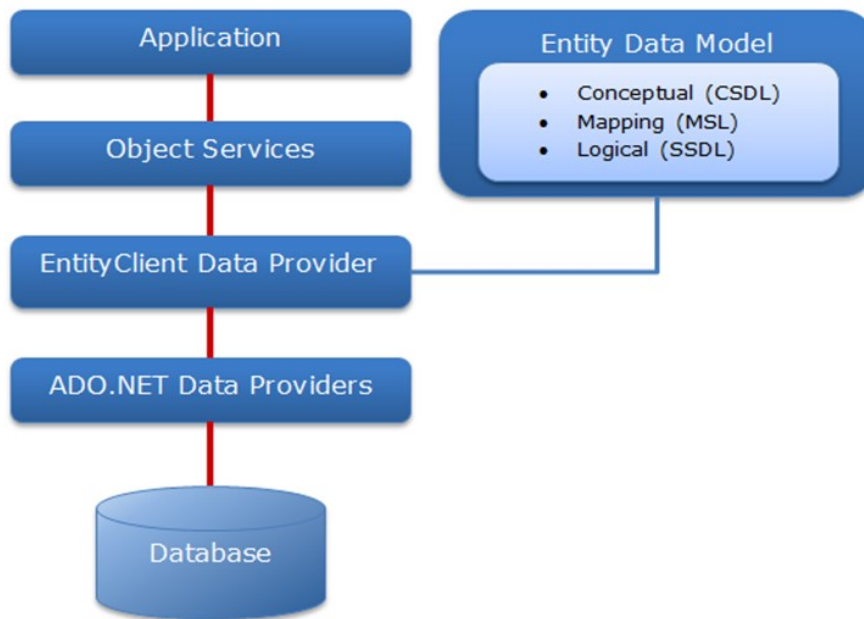
2. Lịch sử phát triển Entity Framework

- Entity Framework ra đời từ 2008 trong phiên bản của .NET 3.5. Phiên bản hiện nay là Entity Framework 6.0.
- Phiên bản đầu tiên chỉ hỗ trợ Database first. Entity Framework chỉ có thể làm việc với một CSDL có sẵn.
- Trong phiên bản 4.0 xuất hiện hướng tiếp cận Model first cho phép thiết kế các lớp thực thể trước bằng cách sử dụng giao diện đồ họa.
- Phiên bản 4.1 đưa thêm hướng tiếp cận Code first, cho phép viết code trước sau đó mới sinh ra cơ sở dữ liệu.

3. Kiến trúc tổ chức

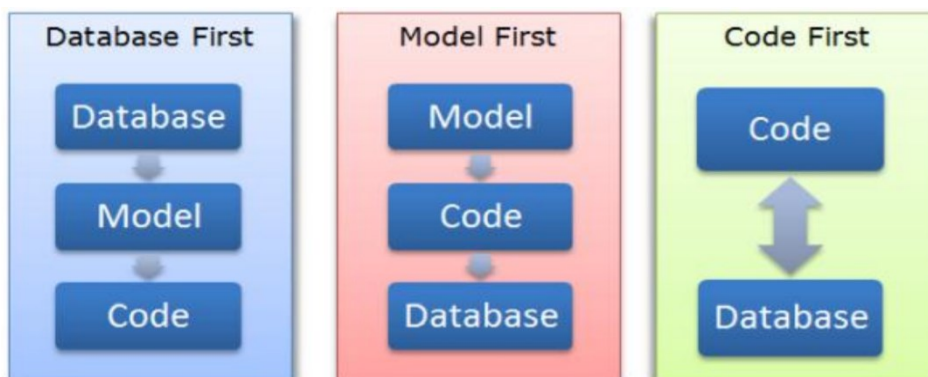
- Application: là tầng chứa giao diện trang Web (HTML, CSS, Javascript, hình ảnh, ...) và các đoạn mã nguồn (C#,...) để tương tác dữ liệu với các tầng khác trong mô hình thông qua Object Services.
- Object Services: là tầng chứa quá trình tương tác giữa ứng dụng và database, hay nói cách khác nó là nơi chủ yếu để truy cập dữ liệu từ database và trả ngược kết quả về giao diện. Object Services cung cấp các tiện ích để truy vết các thay đổi và quản lý nhận dạng, đồng thời là các quan hệ và thay đổi ở database.
- ADO.NET Data Providers: Đây là tầng thấp nhất để dịch các truy vấn L2E (LINQ to Entity) thông qua câu lệnh thành các câu lệnh SQL và thực thi các câu lệnh trong hệ thống DBMS (database management system – hệ quản lý dữ liệu) nào đó. Tầng này kết với database sử dụng ADO.NET.
- EDM (Entity Data Model): chứa 3 phần chính là mô hình khái niệm (CSDL – Conceptual schema definition language), mô hình ánh xạ (MSL – mapping specification language) và mô hình lưu trữ (SSDL – store schema definition language). EDM khác với EntityClient Data Provider ở chỗ EDM sử dụng LINQ là ngôn ngữ truy vấn tương tác với database.
- ADO.NET Data Providers: Đây là tầng thấp nhất để dịch các truy vấn L2E (LINQ to Entity) thông qua câu lệnh thành các câu lệnh SQL và thực thi

các câu lệnh trong hệ thống DBMS (database management system – hệ quản lý dữ liệu) nào đó. Tầng này kết với database sử dụng ADO.NET.



- **Mô hình khái niệm (CSDL – Conceptual schema definition language):** Chứa các lớp mô hình và mối quan hệ giữa các lớp này.
- **Mô hình lưu trữ (SSDL – store schema definition language):** Gồm các bảng, view, stored procedure và mối quan hệ giữa chúng. Mô hình này thể hiện gần giống mô hình quan hệ các bảng trong CSDL.
- **Mô hình ánh xạ (MSL – mapping specification language)**
Mô hình ánh xạ gồm thông tin về cách mô hình khái niệm được ánh xạ đến mô hình lưu trữ.
- **L2E (LINQ to Entities):** là 1 ngôn ngữ truy vấn được dùng để viết các truy vấn trái với mô hình đối tượng. L2E trả về các thực thể, được định nghĩa bởi mô hình khái niệm. Chúng ta có thể dùng LINQ trong ngôn ngữ này.

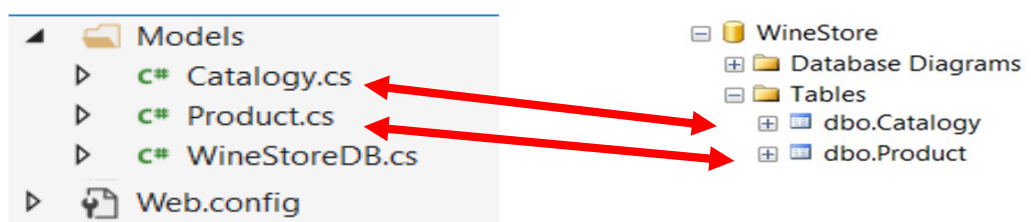
4. Các mô hình lập trình Entity Framework



- Database First: Đã có các bảng trong CSDL, EF sinh ra các domain class
 - Database First là: xây dựng cơ sở dữ liệu trước rồi mới xây dựng mã nguồn ứng dụng.
 - Database First cho phép tạo một mô hình đảo ngược của database và lưu trong tập tin EDMX (.edmx). Có thể xem và chỉnh sửa tập tin này trong Entity Framework Designer. Các lớp tương ứng với các bảng sẽ được tự động sinh từ tập tin EDMX.
 - Đây là 1 cách tiếp cận được sử dụng rất phổ biến trong các hệ thống lớn khi mà 1 dự án có 1 bộ phận chuyên dụng thiết kế database
- Model First: Từ một biểu đồ UML, EF sinh ra các domain class và bảng trong CSDL
 - Model First: cho phép tạo 1 mô hình dùng Entity Framework Designer và sau đó tạo lược đồ cơ sở dữ liệu từ mô hình.
 - Mô hình được lưu trữ ở tập tin EDMX (.edmx) và có thể xem, chỉnh sửa ở Entity Framework Designer. Các lớp tương tác với ứng dụng được tự động gieo từ tập tin EDMX.
 - Phương pháp này giúp hình dung được mô hình phần mềm ở mức độ tổng quan, từ đó mới phân chia phần mềm thành các phần con và triển khai xây dựng ứng dụng.
 - Thông thường, các ứng dụng có quy mô lớn đến rất lớn sẽ ưu tiên sử dụng phương pháp này.
- Code First: Từ các domain class, EF sinh ra các bảng trong CSDL
 - Với cách tiếp cận code-first, Entity Framework sẽ tạo các đối tượng bảng cơ sở dữ liệu dựa trên model đã tạo để biểu diễn dữ liệu ứng dụng.
 - Mô hình này rất phổ biến kiểm soát hoàn toàn code model, thêm xóa sửa thuộc tính vô cùng dễ dàng.
 - Tuy nhiên Các thay đổi cấu trúc trực tiếp trên CSDL sẽ mất. Khó kiểm soát những column sẽ tạo trên CSDL.

5. Mô hình Code first

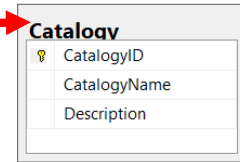
- Ví dụ mô hình Code first:



Entity class

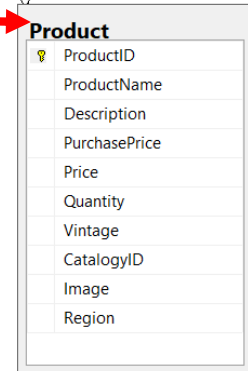
```
[Table("Catalogy")]
public partial class Catalogy
{
    public string CatalogyID { get; set; }
    public string CatalogyName { get; set; }
    public string Description { get; set; }

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

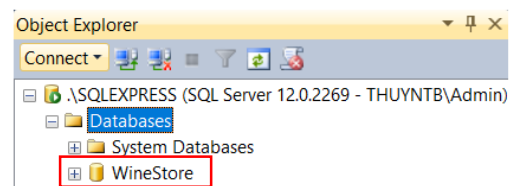


```
[Table("Product")]
public partial class Product
{
    public int ProductID { get; set; }
    public string ProductName { get; set; }
    public string Description { get; set; }
    public decimal PurchasePrice { get; set; }
    public decimal Price { get; set; }
    public int Quantity { get; set; }
    public string Vintage { get; set; }
    public string CatalogyID { get; set; }
    public string Image { get; set; }
    public string Region { get; set; }

    public virtual Catalogy Catalogy { get; set; }
}
```



Database context



```
public partial class WineStoreDB : DbContext
{
    public WineStoreDB(): base("name=WineStoreDB"){ }

    public virtual DbSet<Catalogy> Catalogies { get; set; }
    public virtual DbSet<Product> Products { get; set; }
}
```

```
<connectionStrings>
  <add name="WineStoreDB"
    connectionString="data source=.; initial catalog = WineStore; integrated
    security=True; MultipleActiveResultSets=True;App=EntityFramework"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Quy ước ánh xạ thực thể

- Tên thực thể **số ít** sẽ ánh xạ với bảng cùng tên **số nhiều**
- Tùy biến với [Table("Tên bảng")]
 - Tên thuộc tính **cùng tên** với tên cột
 - Tùy biến với [Column ("Tên cột")]
- Tên thuộc tính phải là ID hoặc EntityID
 - Tùy biến với [Key]
- Khóa int được hiểu là tự tăng

- Tùy biến với
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
- Các phương thức thao tác dữ liệu
 - Tạo đối tượng db context
var db = new DatabaseContext();
 - Thao tác & truy vấn thực thể

<i>Thêm mới thực thể</i>	db.Courses.Add(course);
<i>Cập nhật thông tin của thực thể</i>	db.Entry(course).State = EntityState.Modified;
<i>Xóa thực thể</i>	db.Courses.Remove(course);
<i>Truy vấn một thực thể theo mã</i>	var course = db.Courses.Find(id);
<i>Truy vấn tất cả các thực thể</i>	var list = db.Courses

- Lưu sự thay đổi
 - db.SaveChanges()

6. Tạo một ứng dụng ASP .NET MVC với cách tiếp cận code first

- Bước 1. Tạo project, chọn mẫu MVC
- Bước 2. Tạo Model
 - Kích chuột phải vào thư mục model => Add => Class
 - Tạo các Class, tên class không nên đặt tên có ký tự "s" (Số nhiều) vì quá trình generate ở database sẽ tạo ra bảng có thêm 1 ký tự "s" nữa đằng sau.
 - Cấu hình các class thêm các thuộc tính và quan hệ
- Bước 3. Connect database
 - Mở Sql Sever và tiến hành tạo 1 database.
 - Chọn property của sever để lấy sever name.
 - Vào Project => Manager nuget package => Chọn tab browse và tìm Entity Framework.
 - Vào Sever Explorer => Chọn Connect to database => Nhập tên sever và

chọn database vừa tạo.

- Kích chuột phải vào data connection vừa kết nối => Property => Copy connection string.
- Mở file webconfig.cs thêm đoạn connection string vào trong thẻ configuration.
- Bước 4. Tạo Data accept layer chứa các context
 - Vào project tạo thêm 1 thư mục có tên DAL => Tạo một class có tên XXXContext.cs và cấu hình.
 - Tiến hành Rebuild Project (**bắt buộc**)
- Bước 5. Tạo controller và view
 - Kích chuột phải thư mục controller => add => Add new scaffolded Item => MVC 5 Controller with view, using entity framewrok
 - Generate cả controller và view của các model
- Bước 6. Chạy ứng dụng để tạo bảng trong cơ sở dữ liệu

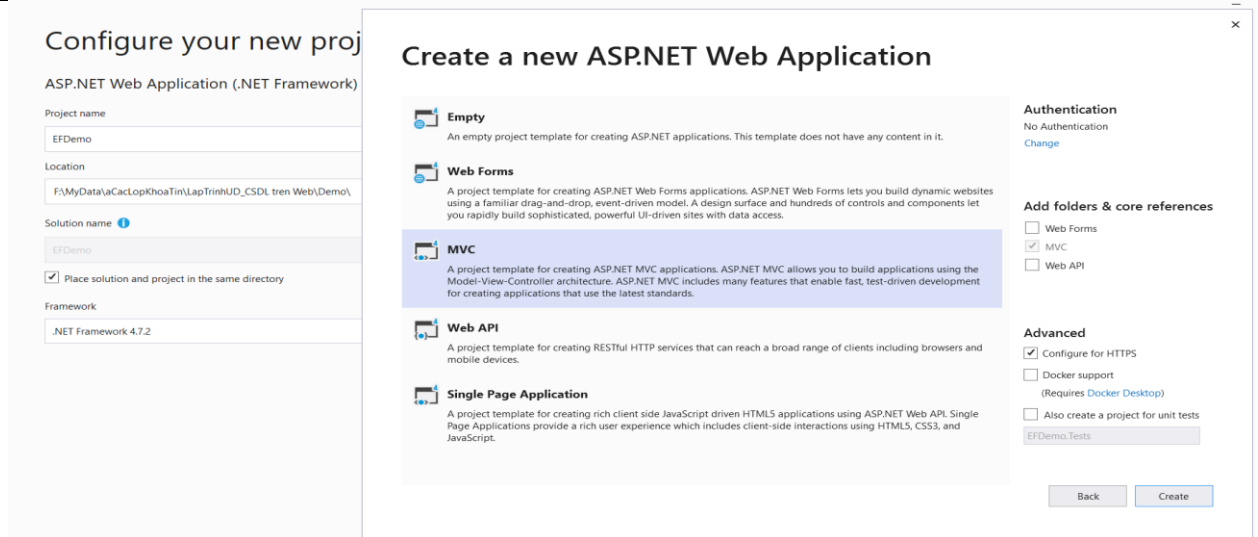
7. Làm việc với cơ sở dữ liệu đã có bằng Entity Framework Code First

- Bước 1. Cần có một cơ sở dữ liệu trong hệ quản trị cơ sở dữ liệu
- Bước 2. Tạo project MVC
- Bước 3. Tạo kết nối với Database và ánh xạ các bảng thành model
 - Kích chuột phải vào folder Models chọn Add => New item => Data => ADO.NET Entity Model
 - Chọn Code First from database
 - Kích vào New Connection...
 - Nhập Server name và chọn Database
 - Chọn Tables
- Bước 4. Mở file webconfig.cs để kiểm tra connection string
- Bước 5. Tạo controller và view
 - Kích chuột phải thư mục controller => add => Add new scaffolded Item => MVC 5 Controller with view, using entity framewrok
 - Generate cả controller và view của các model

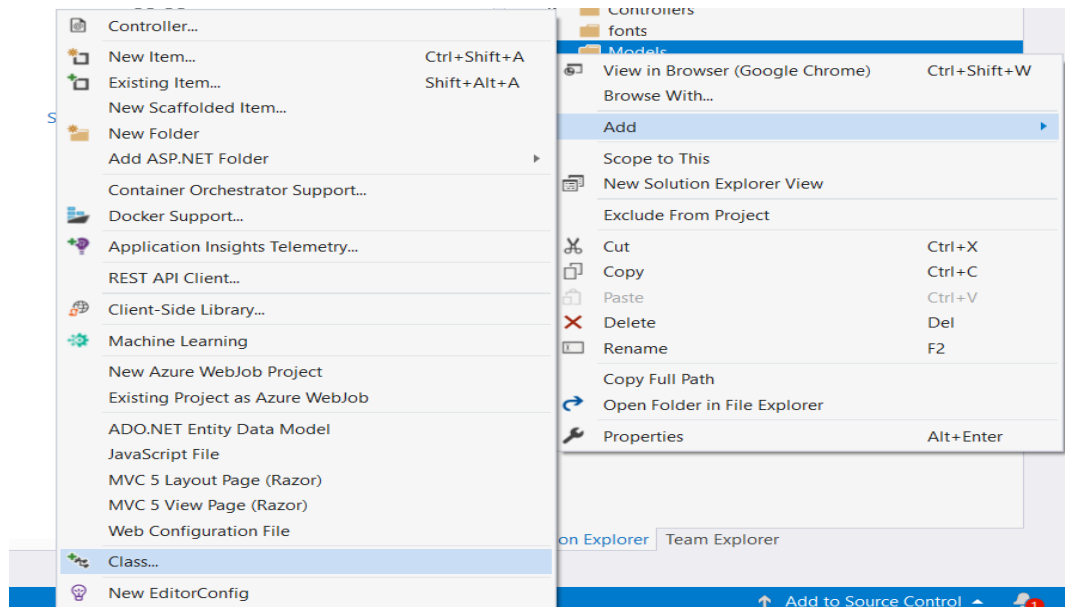
8. Hướng dẫn thực hành:

8.1. Tạo một ứng dụng ASP .NET MVC với cách tiếp cận code first

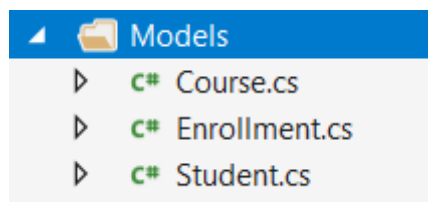
- Bước 1. Tạo project, chọn mẫu MVC



- Bước 2. Tạo Model
 - Kích chuột phải vào thư mục model => Add => Class



- Tạo các Class: Student, Course, Enrollment



- Cấu hình các class thêm các thuộc tính và quan hệ: Bảng Student sẽ liên kết với bảng Enrollment, để định nghĩa dùng từ khóa **virtual** và một kiểu dữ liệu là `Iconnection<Enrollment>` (tham số là Enrollment entity ta định nghĩa sau) => định nghĩa được bảng Student được mapping sang bảng Enrollment và bảng này sẽ có khóa ngoài StudentID. Tương tự với bảng Course.

```
public class Student {
    public int Id { get; set; }
    public string LastName { get; set; }
    public string FirstName { get; set; }
}
```



```

public DateTime EnrollmentDate { get; set; }

public virtual ICollection<Enrollment>Enrollments { get; set; }
}

public class Enrollment {
    public int EnrollmentID { get; set; }
    public int Grade { get; set; }
    public int CourseID { get; set; }
    public int StudentID {get; set;}

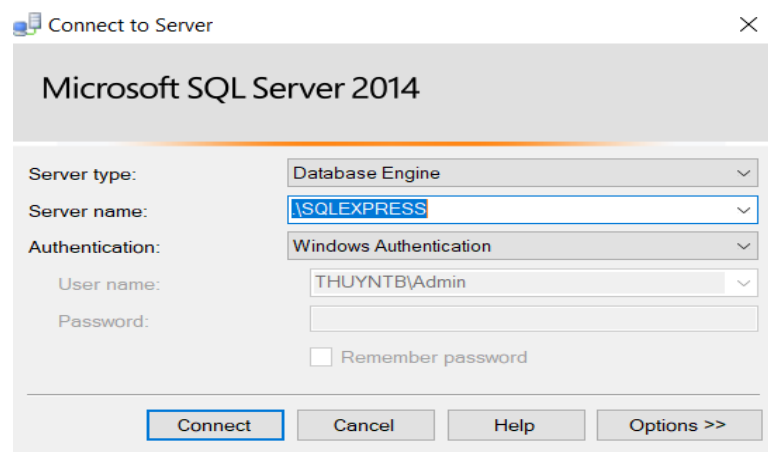
    public virtual Student Student { get; set; }
    public Course Course { get; set; }
}

public class Course {
    [DatabaseGenerated(DatabaseGeneratedOption.None)] //Sử dụng
    anotation để tự sinh các mã tự động trong database
    public int CourseID { get; set; }
    public string Title { get; set; }
    public int Credits { get; set; }

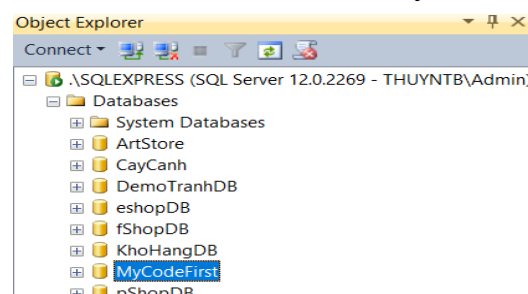
    public virtual ICollection<Enrollment>Enrollments { get; set; }
}

```

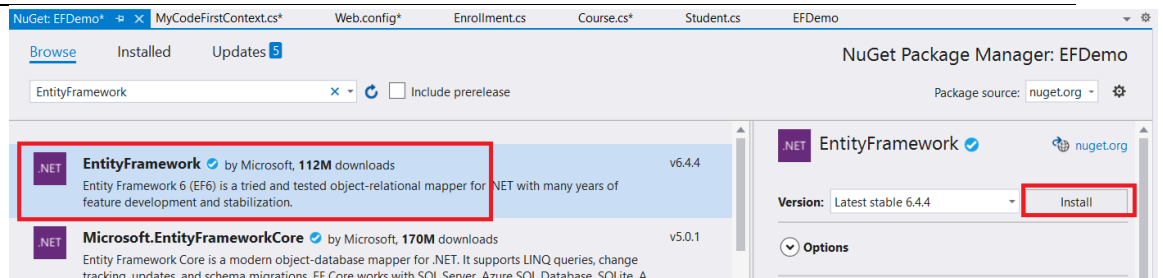
- Bước 3. Connect database
 - Mở Sql Sever lấy Server Name



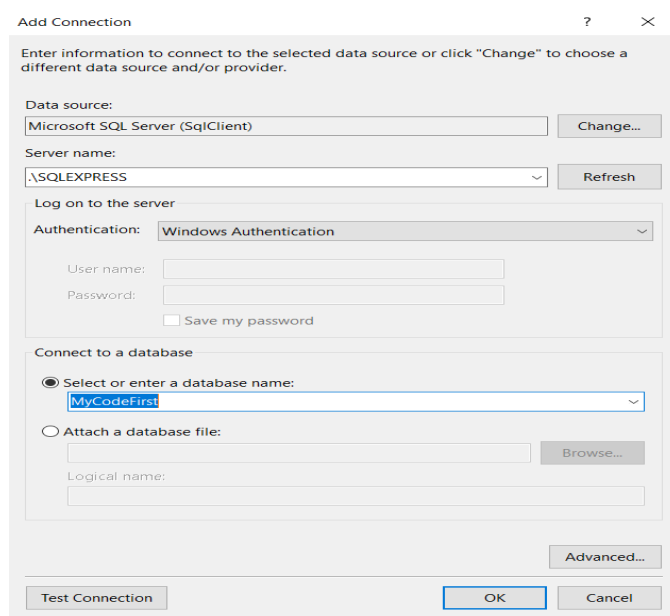
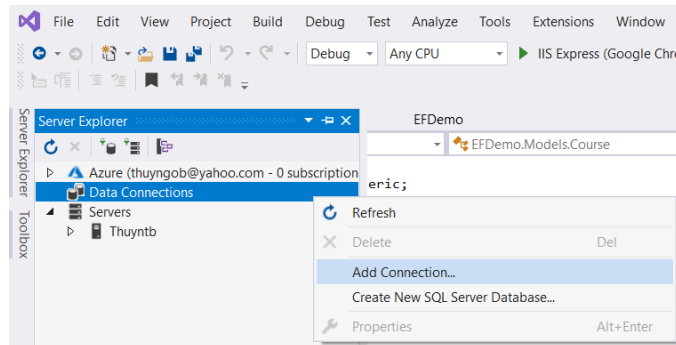
- Tạo một database có tên MyCodeFirst



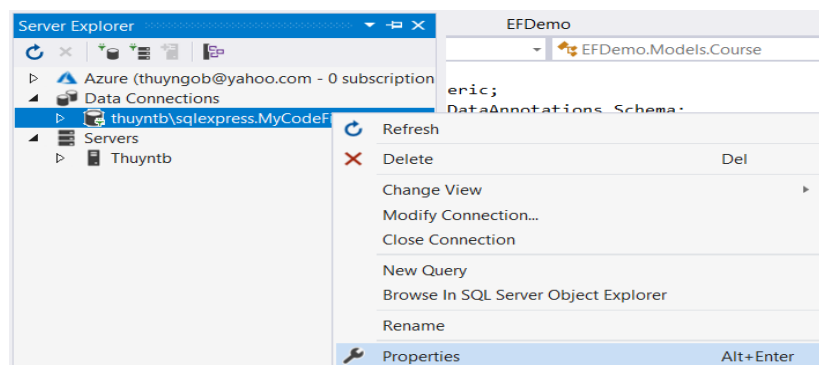
- Vào Project => Manager nuget package => Chọn tab browse và tìm Entity Framework.

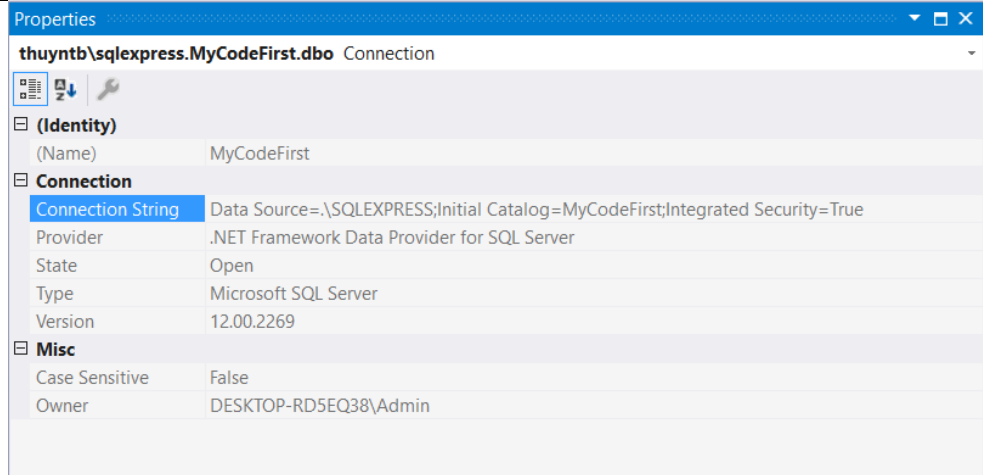


- Vào Sever Explorer => Data Connections=> Add Connection...



- Kích chuột phải vào data connection vừa kết nối => Property => Copy connection string.

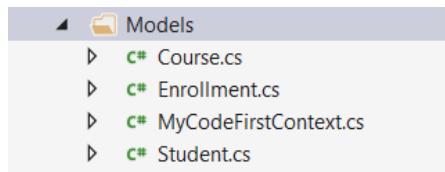




- Mở file webconfig.cs thêm đoạn connection string vào trong thẻ configuration

```
<connectionStrings>
  <add name="MyCodeFirstContext" connectionString="Data
Source=.\SQLEXPRESS;Initial Catalog=MyCodeFirst;Integrated Security=True"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

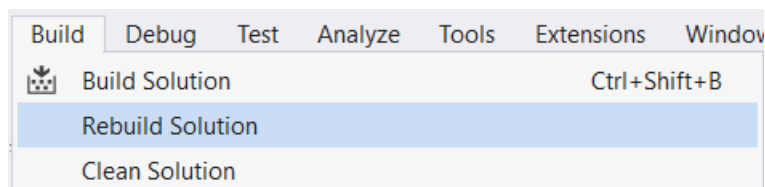
- Bước 4. Tạo Data accept layer chứa các context
 - Vào project trong folder Models tạo thêm một class có tên MyCodeFirstContext.cs và cấu hình như sau



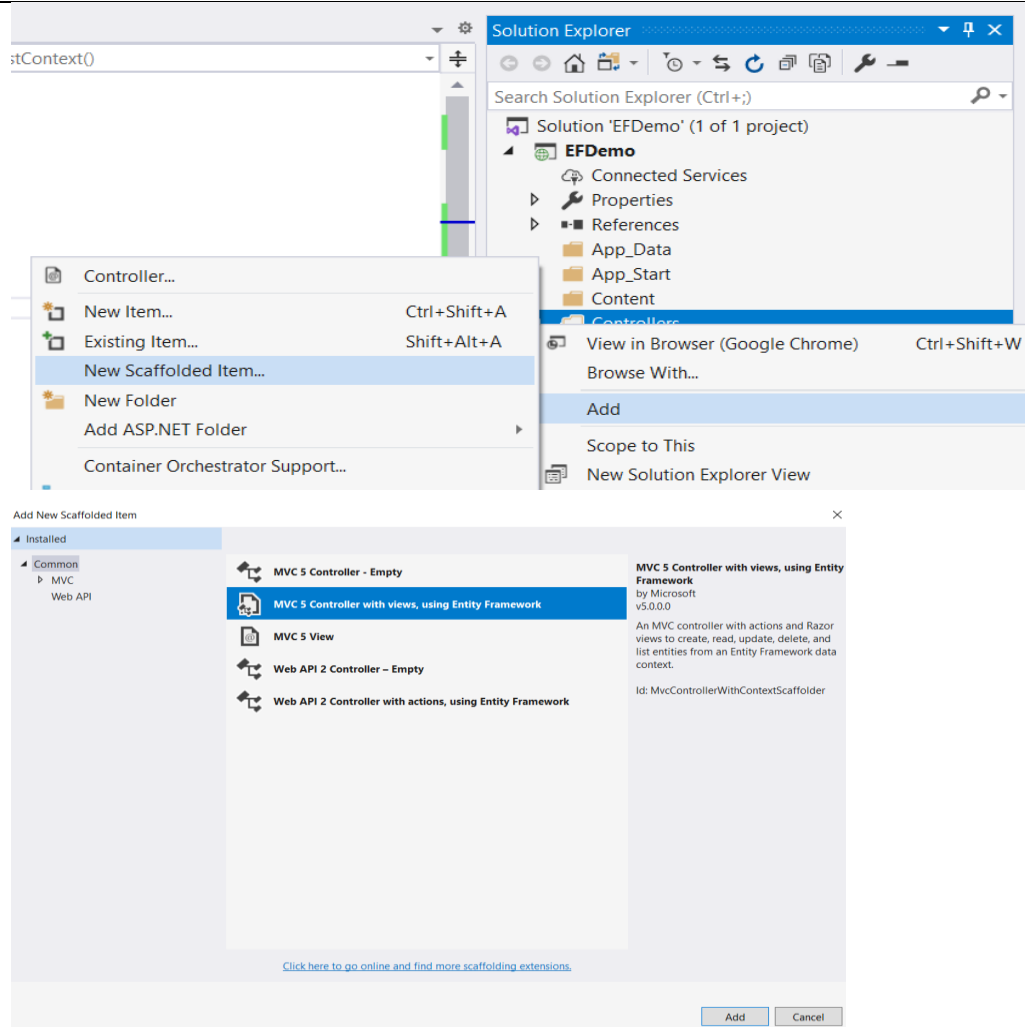
```
public class MyCodeFirstContext : DbContext{
    public MyCodeFirstContext() : base("MyCodeFirstContext") {
    }

    public DbSet<Student> Students { get; set; }
    public DbSet<Enrollment> Enrollments { get; set; }
    public DbSet<Course> Courses { get; set; }
}
```

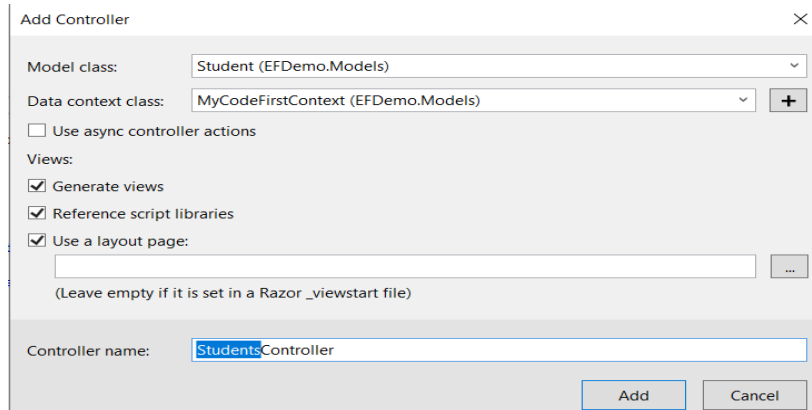
- Tiến hành Rebuild Project (**bắt buộc**)



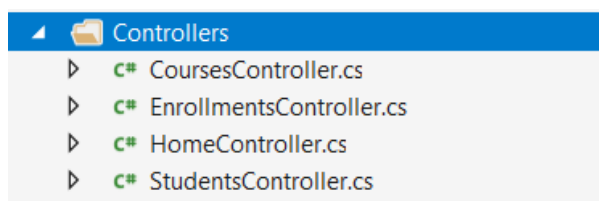
- Bước 5. Tạo controller và view
 - Kích chuột phải thư mục **Controllers** => **Add** => **Add new scaffolded Item** => **MVC 5 Controller with view, using entity framework**



- Generate cả controller và view của các model

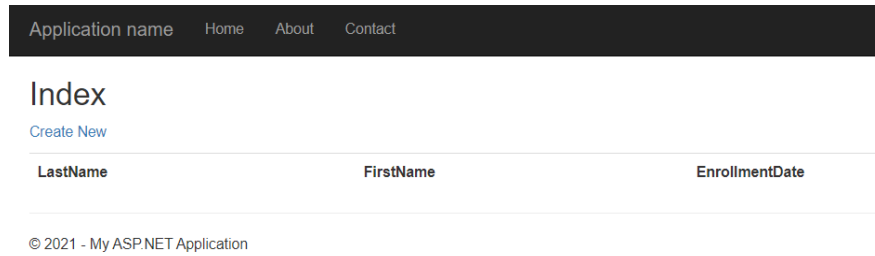


Tương tự với Enrollment và Course

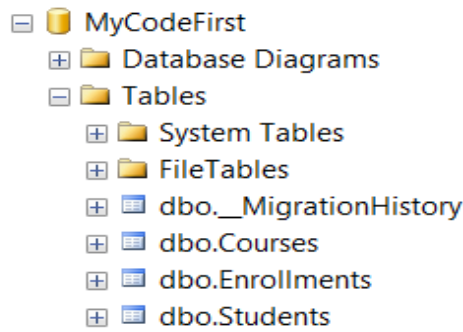


- Chạy ứng dụng ấn **F5** hoặc **Ctrl+F5** và gọi

“https://localhost:44328/students” được kết quả



Trong SQL Server cơ sở dữ liệu MyCodeFirst có thêm các bảng:

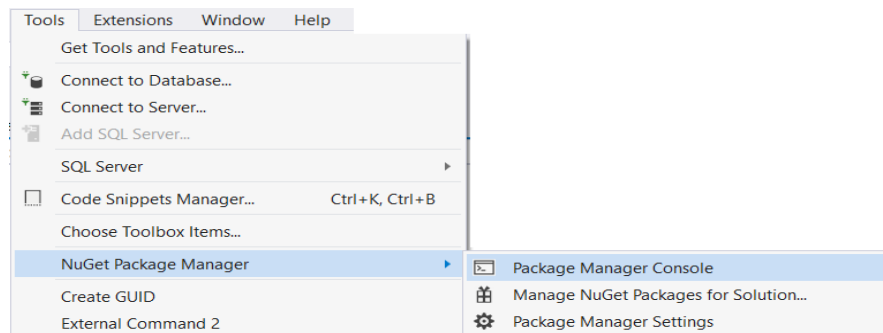


- Thay đổi Model trong Code First

```
public class Student {
    public int Id { get; set; }
    public string LastName { get; set; }
    public string FirstName { get; set; }
    public string Address { get; set; } //thêm trường address
    public DateTime EnrollmentDate { get; set; }

    public virtual ICollection<Enrollment> Enrollments { get; set; }
}
```

- Vào Tool => Nuget package manager => Package manager console:



- Thực hiện 3 câu lệnh

```
enable-migrations
Add-Migration AddAddressToStudent
Update-Database
```

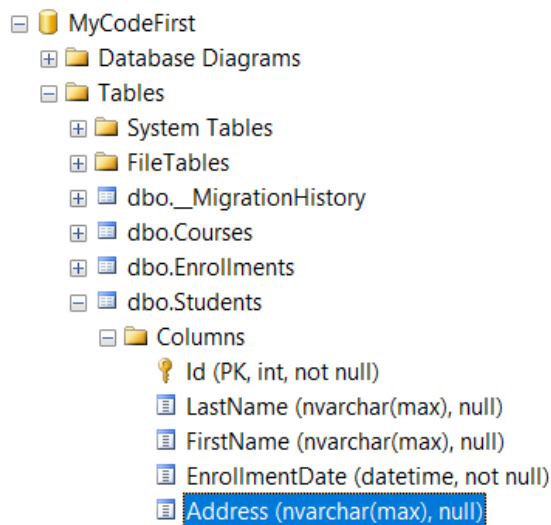
```

Package Manager Console
Package source: All Default project: EFDemo

PM> enable-migrations
Checking if the context targets an existing database...
Detected database created with a database initializer. Scaffolded migration '202101090638158_InitialCreate' corresponding to existing database. To use an automatic migration instead, delete the Migrations folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.
PM> Add-Migration AddAddressToStudent
Scaffolding migration 'AddAddressToStudent'.
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 AddAddressToStudent' again.
PM> Update-Database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [202101090647598_AddAddressToStudent].
Applying explicit migration: 202101090647598_AddAddressToStudent.
Running Seed method.
PM>

```

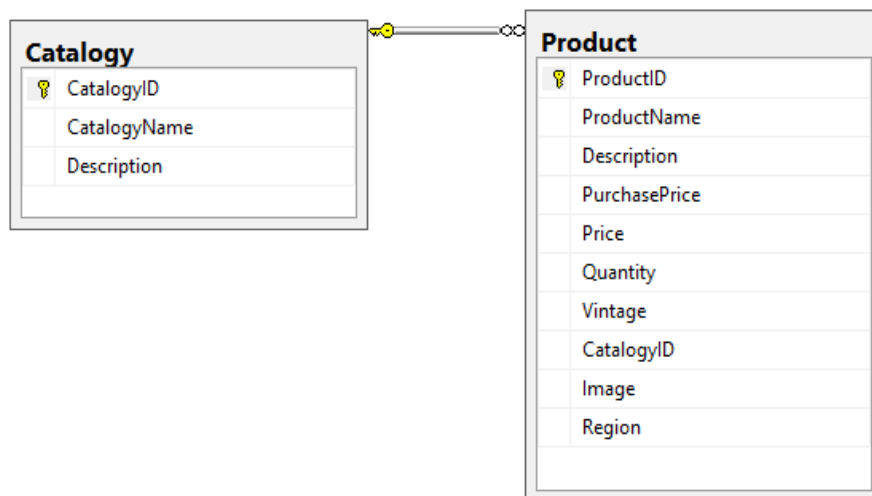
Kết quả trong cơ sở dữ liệu MyCodeFirst



- Do bảng student đã thay đổi nên cần xóa StudentsController và các View tương ứng và tạo lại như bước 5.

8.2. Tạo ứng dụng ASP .NET MVC với chức năng CRUD trên bảng theo mô hình Code First

1. Chạy file script WineDB.sql trong SQLServer để tạo cơ sở dữ liệu WineStore



```
--Tạo cơ sở dữ liệu
CREATE DATABASE [WineStore]
GO

USE [WineStore]
GO

--Tạo bảng Catalogy
CREATE TABLE [dbo].[Catalogy](
[CatalogyID] [nchar](10) NOT NULL,
[CatalogyName] [nvarchar](50) NOT NULL,
[Description] [nvarchar](100) NULL,
CONSTRAINT [PK_Catalogies] PRIMARY KEY CLUSTERED
(
[CatalogyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

--Tạo bảng Product
CREATE TABLE [dbo].[Product](
[ProductID] [int] NOT NULL,
[ProductName] [nvarchar](50) NOT NULL,
[Description] [text] NULL,
[PurchasePrice] [numeric](8, 2) NOT NULL,
[Price] [numeric](8, 2) NOT NULL,
[Quantity] [int] NOT NULL,
[Vintage] [nchar](20) NULL,
[CatalogyID] [nchar](10) NOT NULL,
[Image] [text] NULL,
[Region] [nvarchar](100) NOT NULL,
CONSTRAINT [PK_Products] PRIMARY KEY CLUSTERED
(
[ProductID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

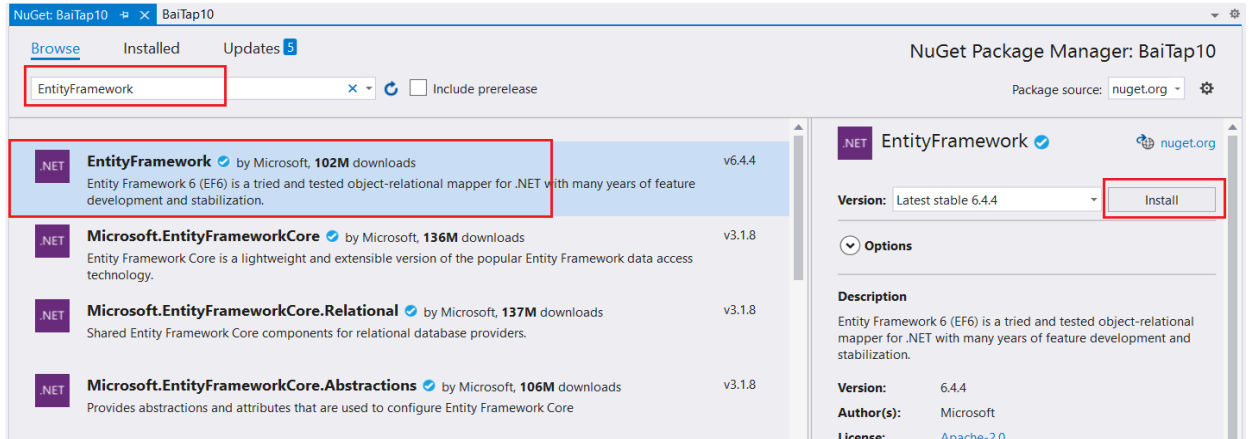
--Tạo các khóa ngoài
ALTER TABLE [dbo].[Product] WITH CHECK ADD CONSTRAINT [FK_Products_Catalogy] FOREIGN
KEY([CatalogyID])
REFERENCES [dbo].[Catalogy] ([CatalogyID])
GO

ALTER TABLE [dbo].[Product] CHECK CONSTRAINT [FK_Products_Catalogy]
GO

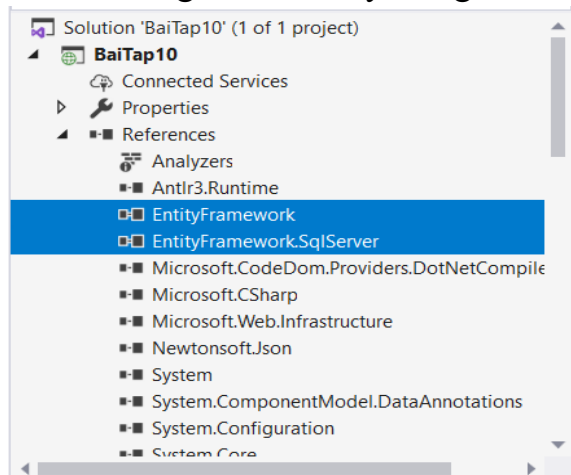
--Chèn dữ liệu cho các bảng
...
```


2. Cài đặt EntityFramework sử dụng NuGet Package Manager

- Tạo một project đặt tên là BaiTap10, chọn mẫu MVC.
- Kích chuột phải vào tên project và chọn Manage NuGet Packages để mở cửa sổ NuGet Package Manager. (**Chú ý máy tính phải nối mạng Internet**)
- Chọn tab Browse, gõ EntityFramework vào thanh tìm kiếm để tìm kiếm EntityFramework sau đó kích vào nút Install để cài đặt.

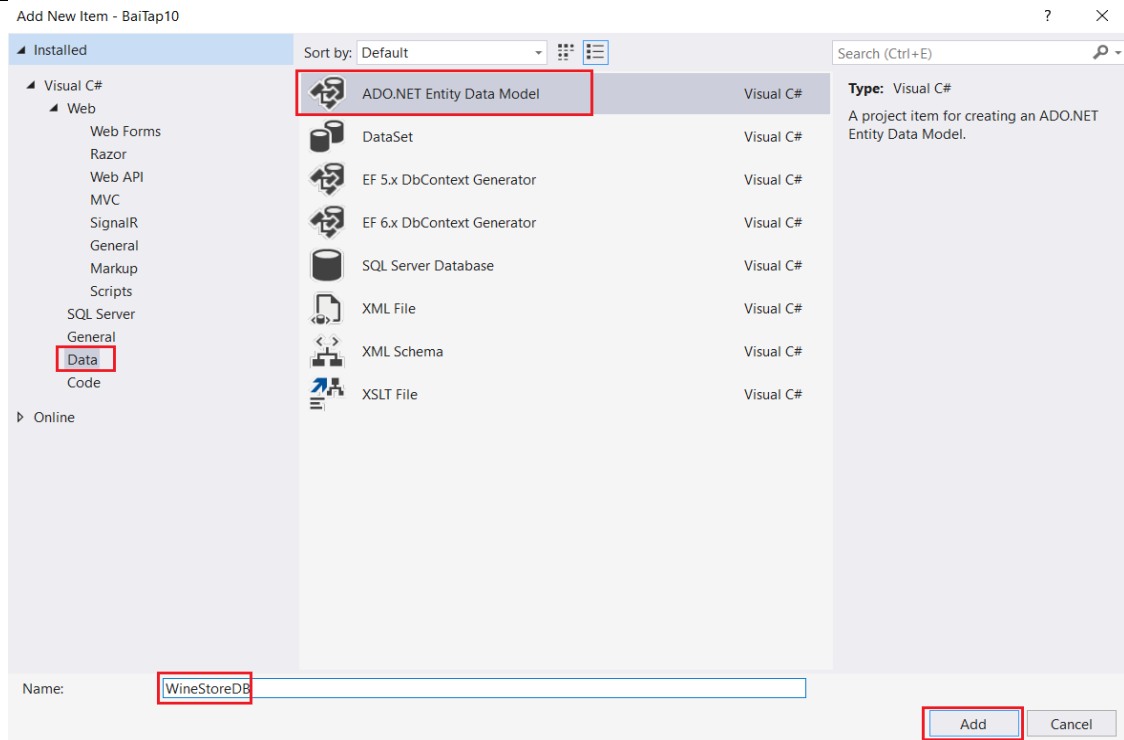


- Cài đặt xong sẽ nhìn thấy trong References

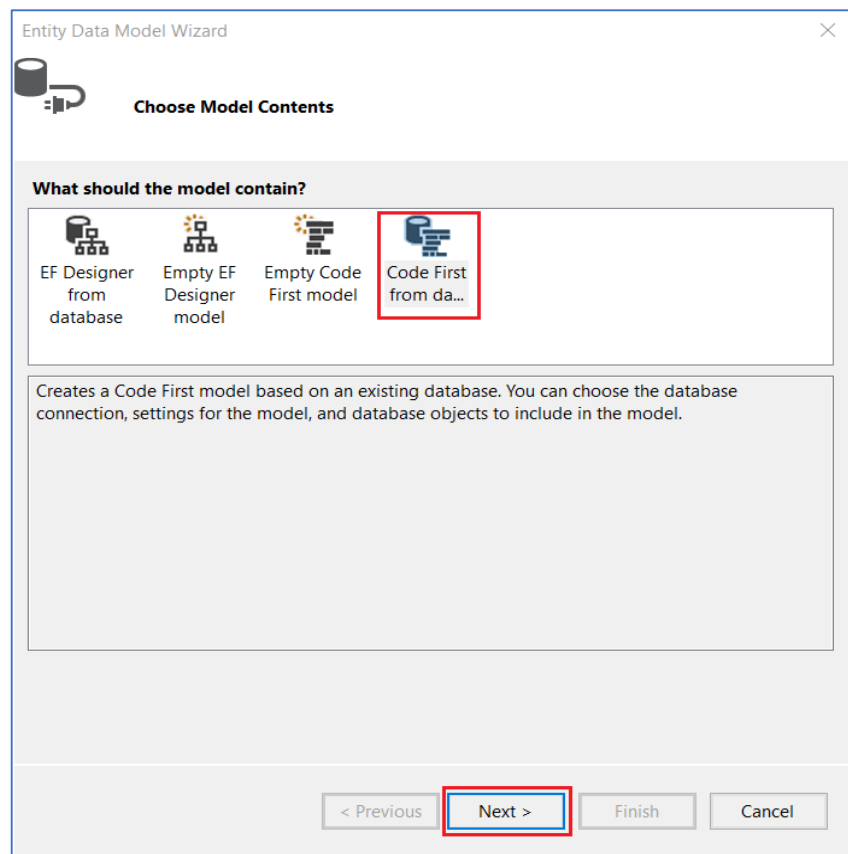


3. Tạo kết nối với Database

- Kích chuột phải vào folder Models chọn Add => New item => Data
=> ADO.NET Entity Model như sau:



- Chọn Code First from database rồi kích vào nút Next



- Kích vào New Connection...



Choose Your Data Connection

Which data connection should your application use to connect to the database?

New Connection...

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?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☐ Yes, include the sensitive data in the connection string.

Connection string:

☒ Save connection settings in Web.Config as:

< Previous

Next >

Finish

Cancel

- Nếu thấy cửa sổ này thì chọn Microsoft SQL Server và kích vào nút Continue

Choose Data Source

?

X

Data source:

Microsoft SQL Server
Microsoft SQL Server Database File
<other>

Description

Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

Data provider:

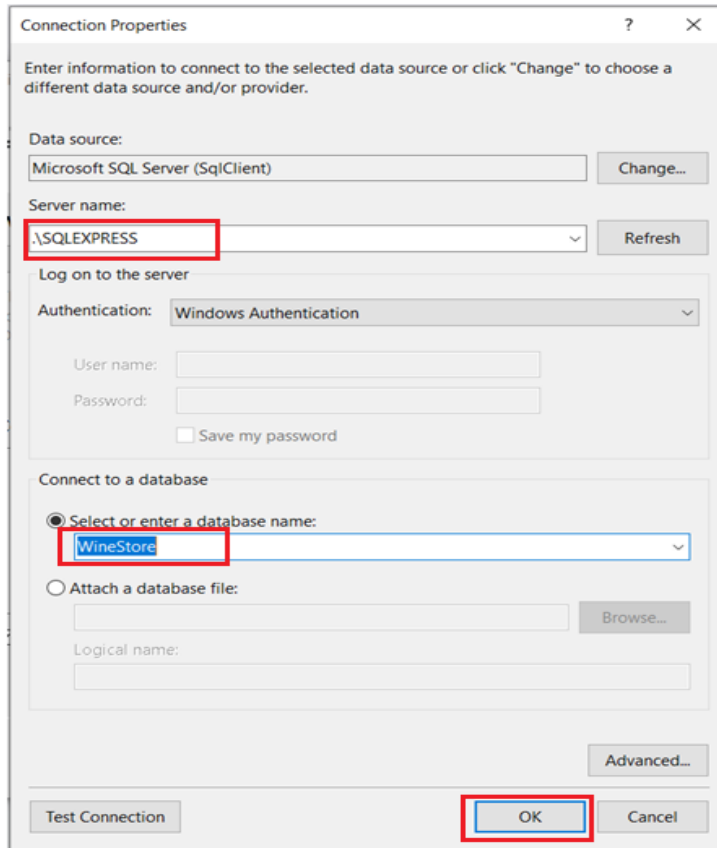
.NET Framework Data Provider for S

☒ Always use this selection

Continue

Cancel

- Chạy SQL Server để lấy server name. Nhập Server name và chọn Database WineStore rồi kích OK



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: .\SQLEXPRESS Refresh

Log on to the server

Authentication: Windows Authentication

User name: Password: Save my password

Connect to a database

☒ Select or enter a database name: WineStore

☐ Attach a database file: Browse...

Logical name:

Advanced...

Test Connection OK Cancel

- Kích Next

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

thuyntb\sqlexpress.WineStore.dbo New Connection...

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?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

data source=.\SQLEXPRESS;initial catalog=WineStore;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework

☒ Save connection settings in Web.Config as:

Tên connection

WineStoreDB

< Previous **Next >** Finish Cancel

- Chọn Tables và kích Finish

Entity Data Model Wizard

Choose Your Database Objects and Settings

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

☒ Tables

☒ dbo

☒ Catalog

☒ Product

☐ sysdiagrams

☐ Views

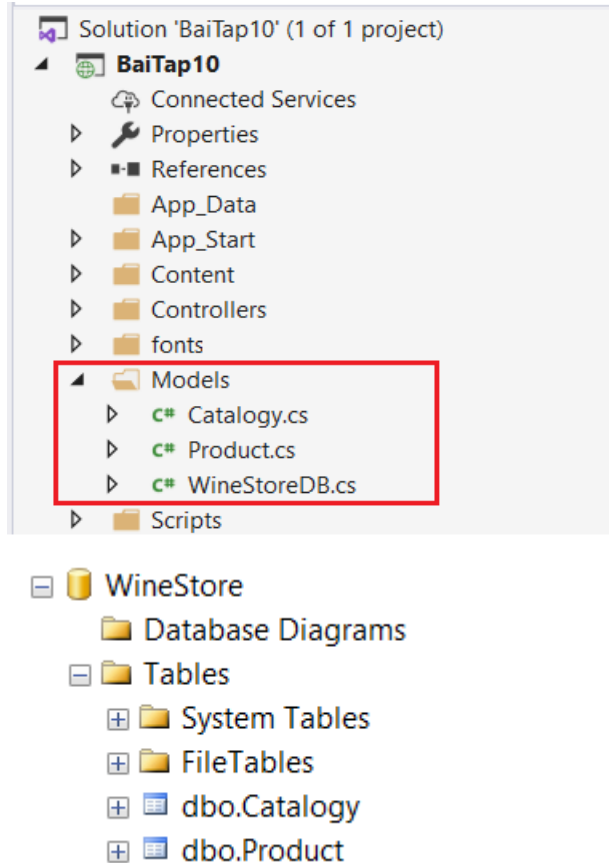
☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☐ Import selected stored procedures and functions into the entity model

< Previous **Next >** **Finish** Cancel

- Các model được sinh ra tương ứng với các bảng trong Database



- Mở các file WineStoreDB.cs là lớp DbContext, Catalogy.cs và Product.cs là các lớp entity để xem code.
- Mở file Web.config để xem `<connectionStrings>`

4. Tùy biến hiển thị tên các property trong các lớp Model và đưa vào các thông báo lỗi.

```
public partial class Catalogy
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
    "CA2214:DoNotCallOverridableMethodsInConstructors")]
    public Catalogy()
    {
        Products = new HashSet<Product>();
    }

    [Key]
    [StringLength(10)]
    [Required(ErrorMessage = "Mã danh mục không được để trống!")]
    public string CatalogyID { get; set; }

    [Required(ErrorMessage = "Tên danh mục không được để trống!")]
    [StringLength(50)]
    [DisplayName("Tên danh mục")]
    public string CatalogyName { get; set; }

    [StringLength(100)]
    [DisplayName("Mô tả")]
    public string Description { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
    "CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Product> Products { get; set; }
}
```

```

public partial class Product
{
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    [Key]
    [DisplayName("Mã rượu")]
    public int ProductID { get; set; }

    [Required(ErrorMessage = "Tên rượu không được để trống!")]
    [StringLength(50)]
    [DisplayName("Tên rượu")]
    public string ProductName { get; set; }

    [Column(TypeName = "text")]
    [DisplayName("Mô tả")]
    public string Description { get; set; }

    [Column(TypeName = "numeric")]
    [DisplayName("Giá nhập")]
    public decimal PurchasePrice { get; set; }

    [Column(TypeName = "numeric")]
    [DisplayName("Giá bán")]
    public decimal Price { get; set; }
    [DisplayName("Số lượng")]
    public int Quantity { get; set; }

    [StringLength(20)]
    [DisplayName("Năm sản xuất")]
    public string Vintage { get; set; }

    [Required(ErrorMessage = "Danh mục không được để trống!")]
    [StringLength(10)]
    public string CatalogyID { get; set; }

    [Column(TypeName = "text")]
    [DisplayName("Hình ảnh")]
    public string Image { get; set; }

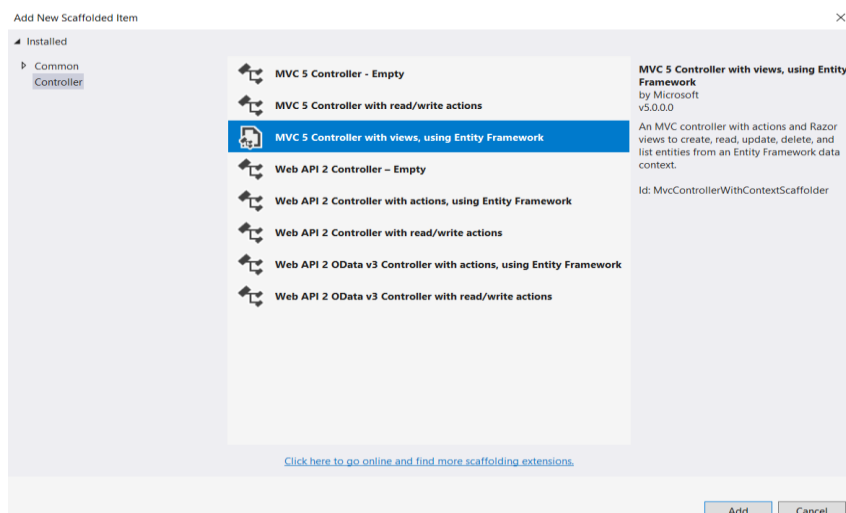
    [Required(ErrorMessage = "Vùng không được để trống!")]
    [StringLength(100)]
    [DisplayName("Vùng")]
    public string Region { get; set; }

    public virtual Catalogy Catalogy { get; set; }
}

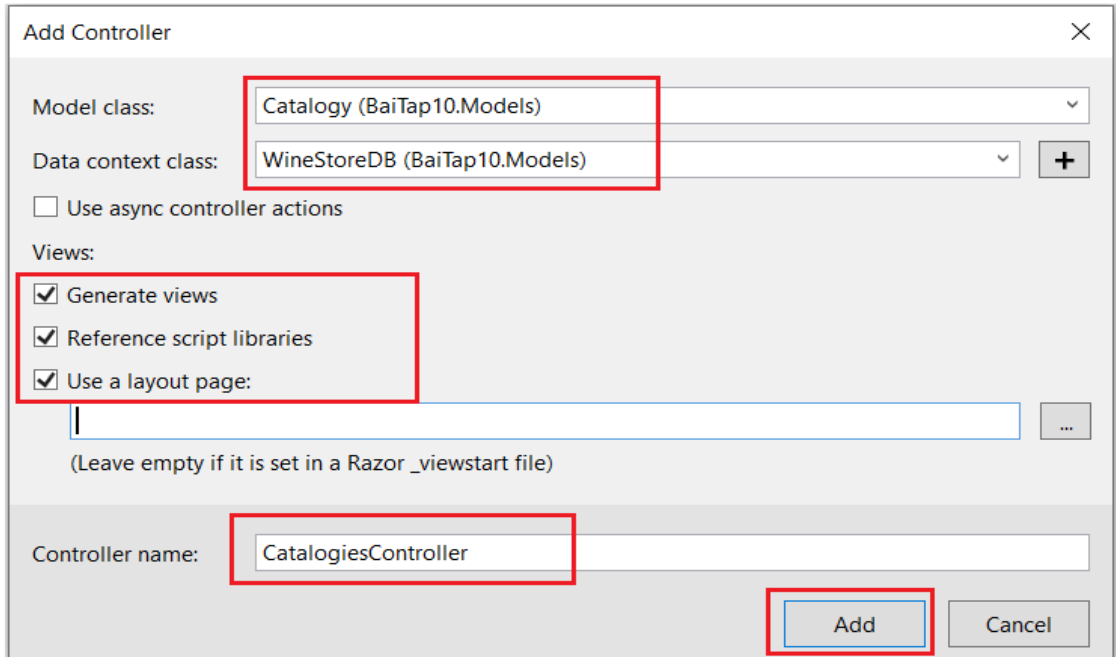
```

5. Tạo chức năng CRUD (Thêm, xem, sửa, xóa) cho bảng Catalogy

- Kích vào Build → ReBuild Solution để build lại project (Phải làm trước khi tạo controller)
- Kích chuột phải vào folder Controllers chọn Add → Controller... Sau đó chọn mẫu **MVC 5 Controller with view, using Entity Framework** như trong hình.



- Chọn như trong hình



- Sửa lại phần action link của _Layout.cshtml

```

...
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("Danh mục", "Index", "Catalogies")</li>
</ul>
...

```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử. Kích vào “Danh mục”
- Sửa lỗi trong file Index.cshtml phần ActionLink

```

...
<td>
    @Html.ActionLink("Edit", "Edit", new { id=item.CatalogyID.Trim() }) |
    @Html.ActionLink("Details", "Details", new { id=item.CatalogyID.Trim() }) |
    @Html.ActionLink("Delete", "Delete", new { id=item.CatalogyID.Trim() })
</td>
...

```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử. Kích vào “Danh mục”. Thử các nút [Create New](#), [Edit](#), [Details](#), [Delete](#)
 - Sửa các nút lệnh thành tiếng Việt
- ❖ **Tùy chỉnh các chức năng để xử lý lỗi**
- Chạy chức năng của Thêm danh mục.
 - Nhập một danh mục mới với không có tên.
 - Nhập một danh mục mới với mã danh mục trùng với một mã đã có.
 - Sửa lại code trong action method **[HttpPost]Create** đưa **try... catch** vào để bắt lỗi như sau:

```
public ActionResult Create([Bind(Include = "CatalogyID,CatalogyName,Description")] Catalogy
catalogy)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Catalogies.Add(catalogy);
            db.SaveChanges();
        }
        return RedirectToAction("Index");
    }
    catch (Exception ex)
    {
        ViewBag.Error = "Lỗi nhập dữ liệu! " + ex.Message;
        return View(catalogy);
    }
}
```

- Trong view **Create.cshtml** thêm đoạn code trên dòng **@section Scripts** để hiển thị thông báo lỗi như sau:

```
*****
@if (ViewBag.Error != null)
{
    <br />
    <div class="alert alert-danger" role="alert">@ViewBag.Error</div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- Làm tương tự với **[HttpPost]Edit**
 - Sửa lại code trong action method **[HttpPost]Edit** đưa **try... catch** vào để bắt lỗi
 - Trong view **Edit.cshtml** thêm đoạn code trên dòng **@section Scripts** để hiển thị thông báo lỗi
- Sửa lại code trong action method **DeleteConfirmed** đưa **try... catch** vào để bắt lỗi xóa bản ghi như sau:

```
public ActionResult DeleteConfirmed(string id)
{
    Catalogy catalogy = db.Catalogies.Find(id);
    try
    {
        db.Catalogies.Remove(catalogy);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    catch (Exception ex)
    {
        ViewBag.Error = "Không xóa được bản ghi này! " + ex.Message;
        return View("Delete", catalogy);
    }
}
```

- Trong view **Delete.cshtml** thêm đoạn code để hiển thị thông báo lỗi như sau:

```

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Xóa" class="btn btn-default" /> |
            @Html.ActionLink("Quay lại", "Index")
        </div>
    }
</div>
@if (ViewBag.Error != null)
{
    <br />
    <div class="alert alert-danger" role="alert">@ViewBag.Error</div>
}

```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.