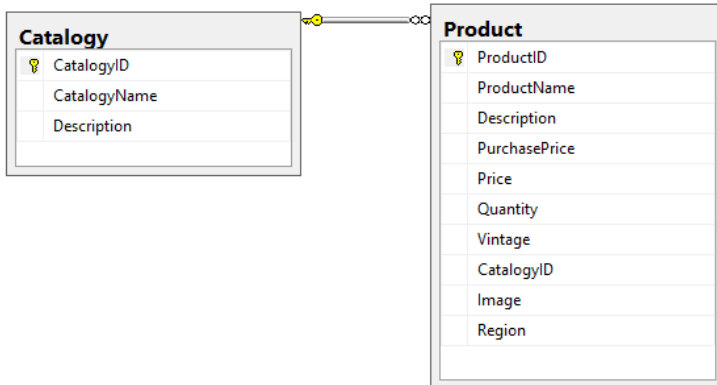


# Bài tập 4

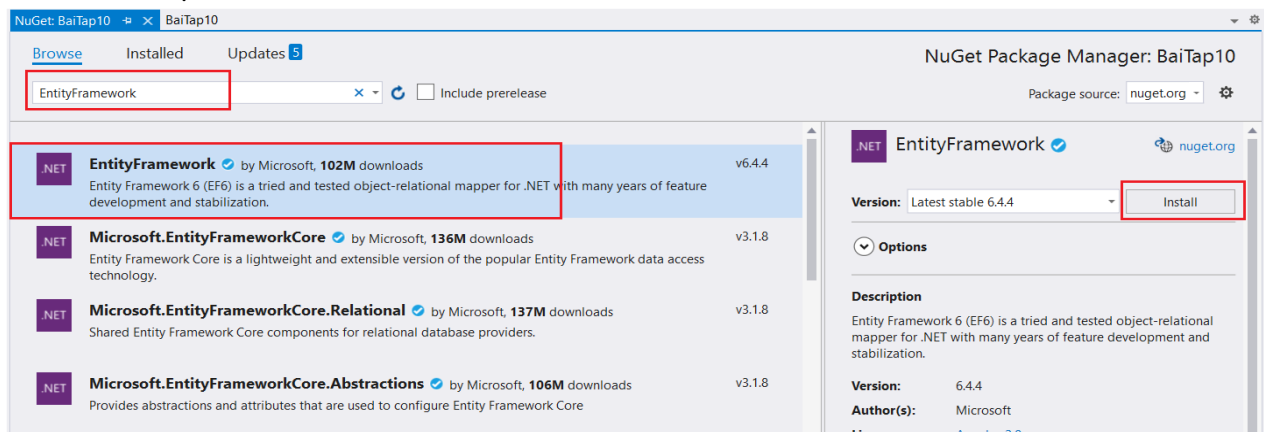
## Lab 10 – EntityFramework (EF)

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

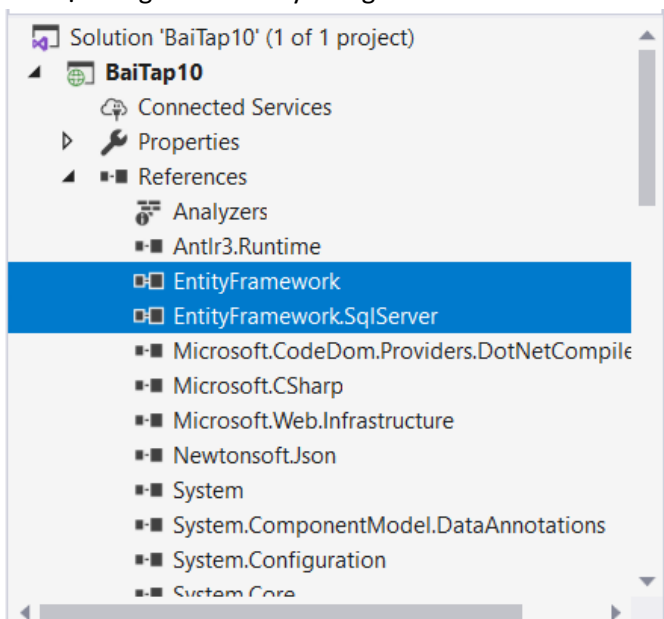


2. Cài đặt EntityFramework sử dụng NuGet Package Manager

- 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.

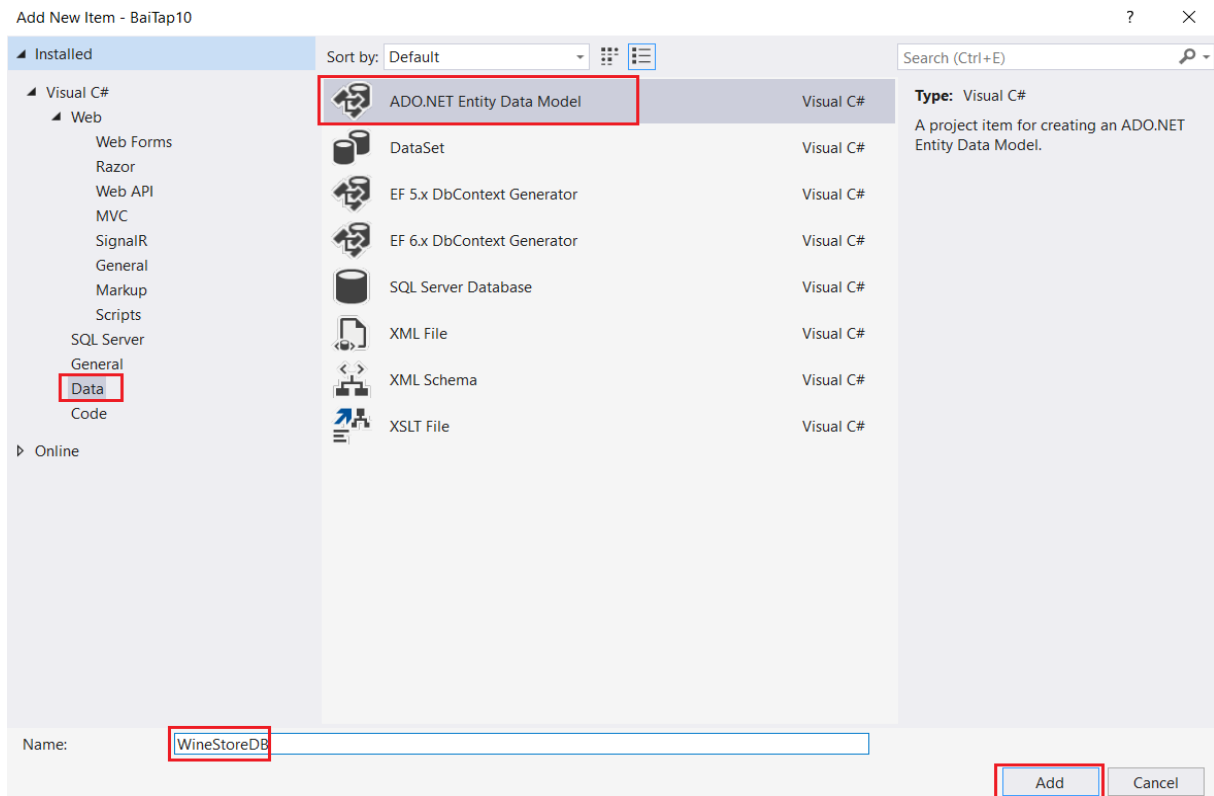


- Cài đặt xong sẽ nhìn thấy trong References

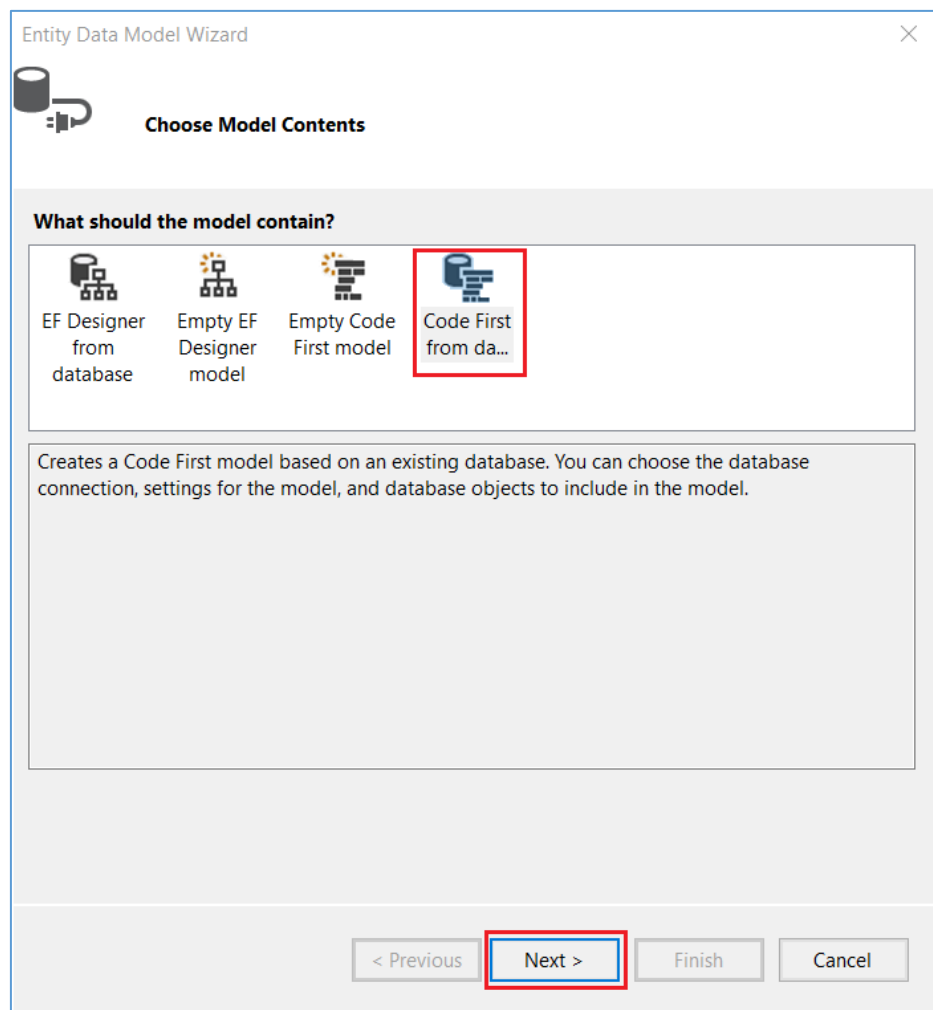


### 3. Tạo kết nối với Database

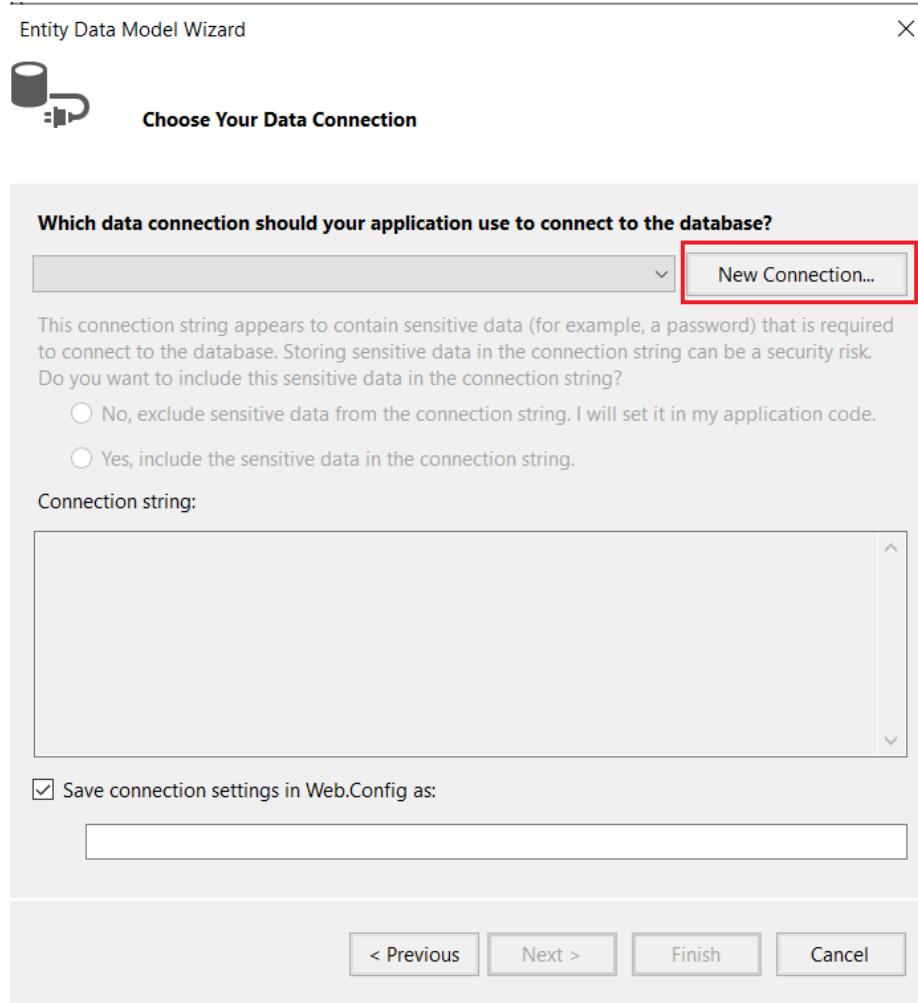
- Kích chuột phải vào folder Models chọn Add => New item => Data => ADO.NET Entity Model như sau:



- Chọn Code First from database rồi kích vào nút Next



- Kích vào New Connection...



Entity Data Model Wizard

**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?

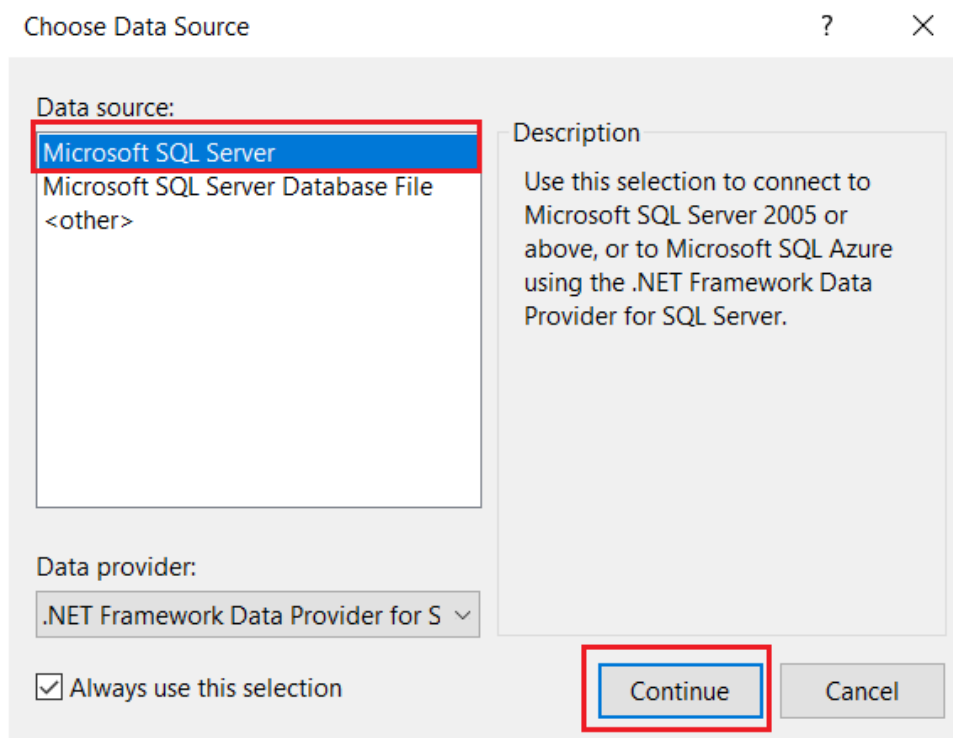
☐ No, exclude sensitive data from the connection string. I will set it in my application code.

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

Connection string:

☒ Save connection settings in Web.Config as:

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



Choose Data Source

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

- 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

- Kịch 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

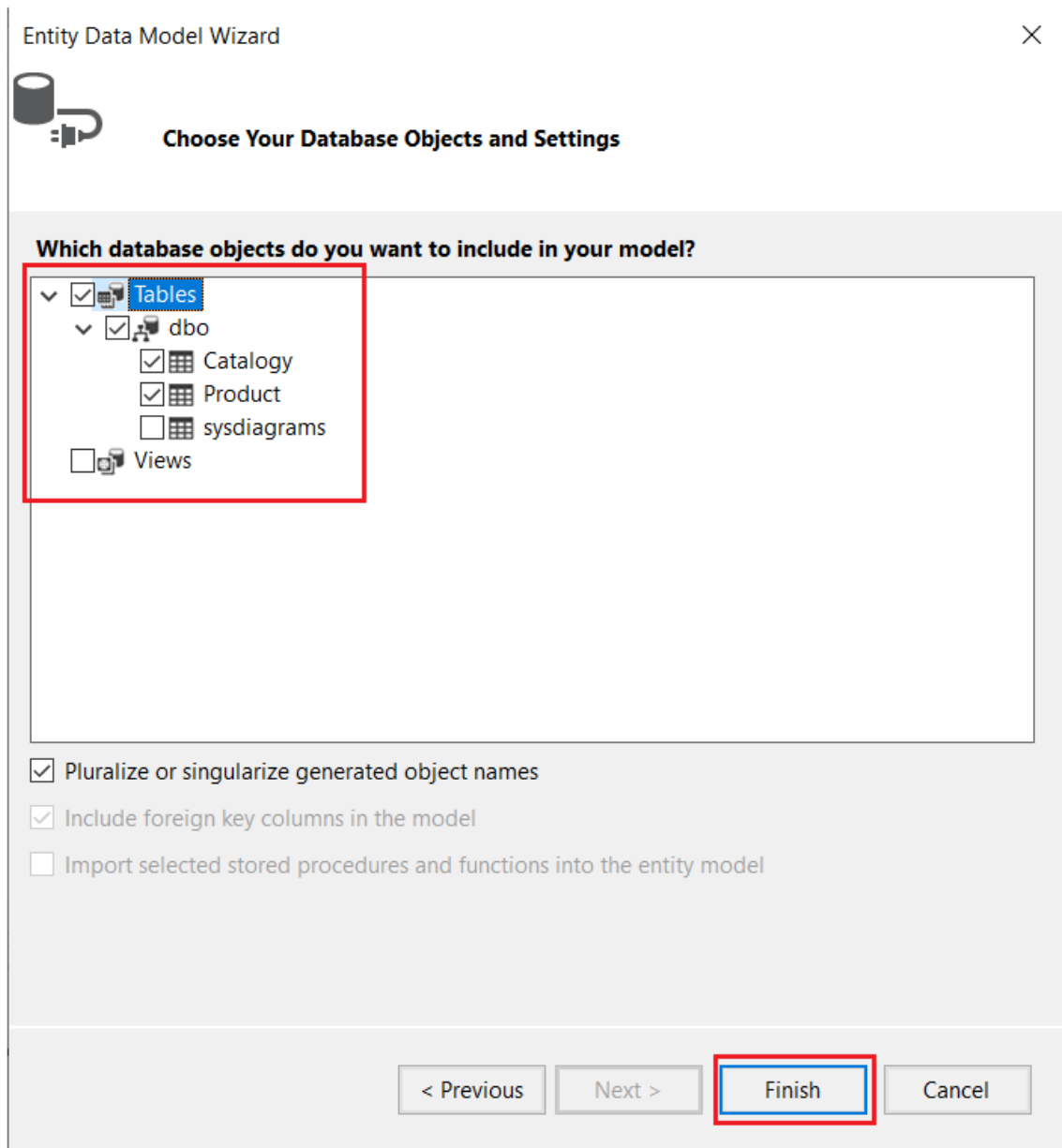
Save connection settings in Web.Config as:

WineStoreDB

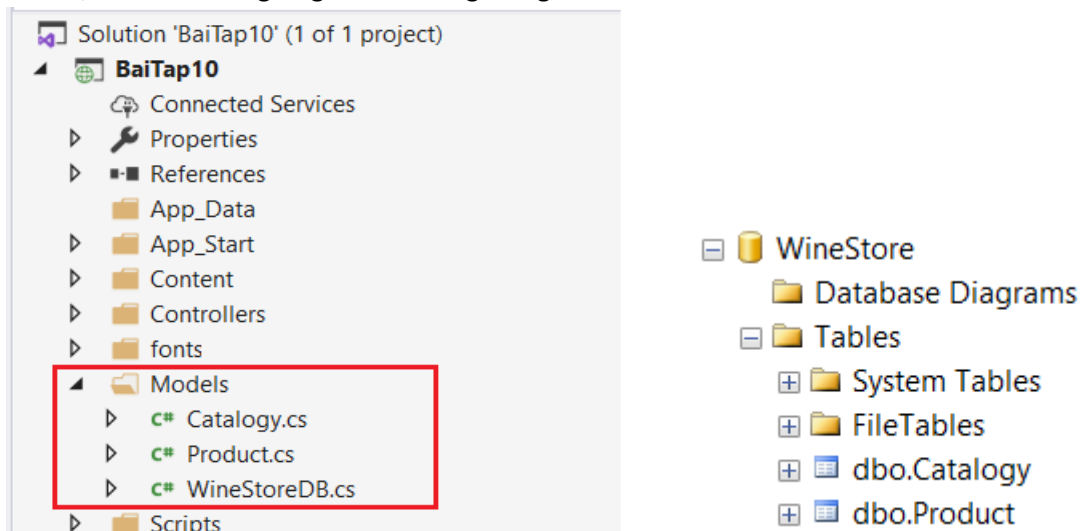
Tên connection

< Previous Next > Finish Cancel

- Chọn Tables và kích Finish



- 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; }
}

```

```

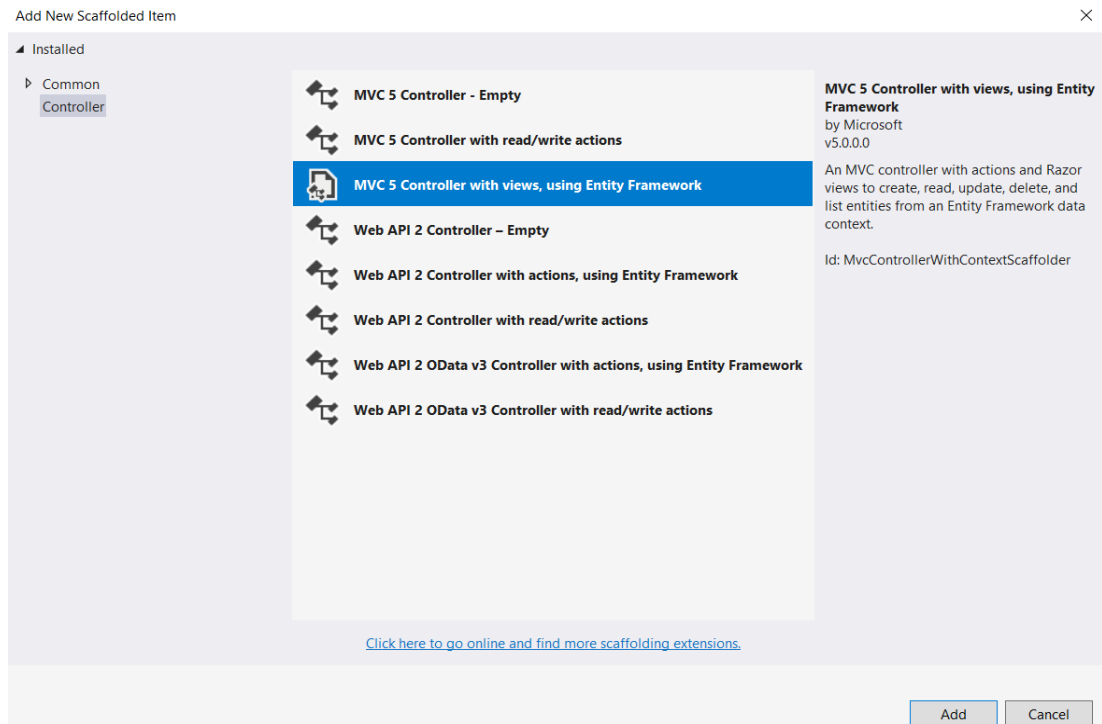
[Required(ErrorMessage = "Vùng không được để trống!")]
[StringLength(100)]
[DisplayName("Vùng")]
public string Region { get; set; }

public virtual Catalogy Catalogy { get; set; }
}

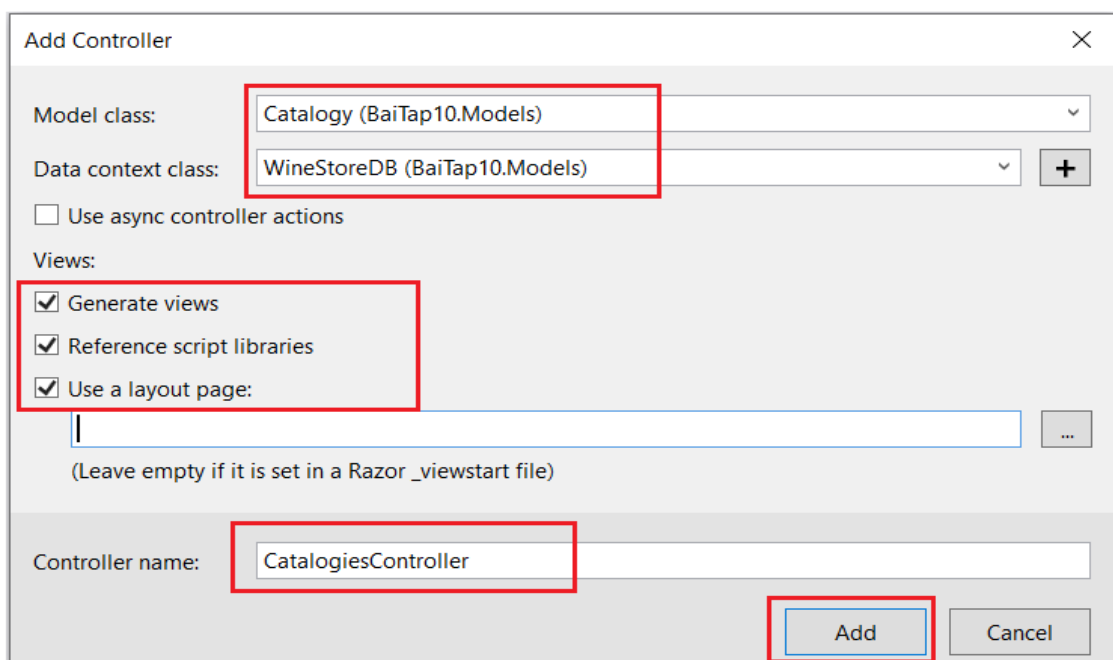
```

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
6. 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.
7. Tạo chức năng CRUD (Thêm, xem, sửa, xóa) cho bảng Product
- Thêm Controllers có mẫu gắn với model class là Product
  - Sửa lại phần action link của **\_Layout.cshtml** thêm một Actionlink: `<li>@Html.ActionLink("Sản phẩm", "Index", "Products")</li>`
  - Sửa các nút lệnh thành tiếng Việt
  - Tùy chỉnh các chức năng để xử lý lỗi:
    - [HttpPost]Create**

```

public ActionResult Create([Bind(Include =
    "ProductID,ProductName,Description,PurchasePrice,Price,Quantity,Vintage,CatalogyID,Image,Region")] Product product)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Products.Add(product);
            db.SaveChanges();
        }
        return RedirectToAction("Index");
    }
    catch (Exception ex)
    {
        ViewBag.Error = "Lỗi nhập dữ liệu! " + ex.Message;
        ViewBag.CatalogyID = new SelectList(db.Catalogies, "CatalogyID",
            "CatalogyName", product.CatalogyID);
        return View(product);
    }
}

```

và **Create.cshtml** ...

- [HttpPost]Edit và Edit.cshtml ...
- DeleteConfirmed và Delete.cshtml
- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- Chạy chức năng của Thêm sản phẩm.
  - Không nhập thông tin gì mà kích luôn vào nút Tạo xem hiển thị lỗi.
  - Sửa lại model Product.cs thêm các thông báo lỗi vào các trường số không được NULL

```
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; }

    [Required(ErrorMessage = "Giá nhập không được để trống!")]
    [Column(TypeName = "numeric")]
    [DisplayName("Giá nhập")]
    public decimal PurchasePrice { get; set; }

    [Required(ErrorMessage = "Giá bán không được để trống!")]
    [Column(TypeName = "numeric")]
    [DisplayName("Giá bán")]
    public decimal Price { get; set; }

    [Required(ErrorMessage = "Số lượng không được để trống!")]
    [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; }
}
```

- Xóa bớt phần hiển thị của cột **Mô tả**, **Giá nhập**, **Chất lượng**, **Vùng** trên view Index.cshtml
- Sửa lại Create.cshtml và Edit.cshtml phần nhãn hiển thị từ "CatalogyID" thành "Danh mục"

```
<div class="form-group">
    @Html.LabelFor(model => model.CatalogyID, "Danh mục", htmlAttributes: new {
@class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("CatalogyID", null, htmlAttributes: new { @class =
"form-control" })
    </div>
</div>
```

```

        @Html.ValidationMessageFor(model => model.CatalogyID, "", new { @class =
"text-danger" })
    </div>
</div>

```

## 8. Hiển thị ảnh của sản phẩm trong danh sách sản phẩm

- Tạo folder wwwroot và copy folder chứa ảnh sản phẩm vào folder này.
- Trong view **Index.cshtml** của folder Products sửa đoạn code hiển thị tên file ảnh

```

<td>
    @Html.DisplayFor(modelItem => item.Image)
</td>

```

Thành

```

<td>
    @{
        //Lấy đường dẫn file ảnh
        string ImagePath = "~/wwwroot/WineImages/" + item.Image;
    }
    
    <br/>
    @Html.DisplayFor(modelItem => item.Image)
</td>

```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- Trong view **Details.cshtml** của folder Products sửa đoạn code hiển thị tên file ảnh

```

<dt>
    @Html.DisplayNameFor(model => model.Image)
</dt>

```

Thành

```

<dd>
    @{
        //Lấy đường dẫn file ảnh
        string ImagePath = "~/wwwroot/WineImages/" + Model.Image;
    }
    
    <br />
    @Html.DisplayFor(model => model.Image)
</dd>

```

## 9. Upload ảnh trong phần thêm sản phẩm mới

- Trong view **Create.cshtml** của folder Products
  - Sửa **@using** (Html.BeginForm()) thành

```

@using (Html.BeginForm("Create", "Products", FormMethod.Post, new { enctype =
"multipart/form-data" }))

```

- sửa đoạn code lấy tên file ảnh

```

<div class="form-group">
    @Html.LabelFor(model => model.Image, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Image, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Image, "", new { @class = "text-danger" })
    </div>
</div>

```

Thành

```

<div class="form-group">
    @Html.LabelFor(model => model.Image, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        
        <p><label for="ufile" style="cursor: pointer;">Chọn file ảnh</label></p>
        <input name="ImageFile" id="ufile" type="file" style="display: none;"
onchange="loadFile(event)" />
    </div>
</div>

```

- Thêm đoạn code script sau vào cuối file:

```
<script>
    var loadFile = function (event) {
        var image = document.getElementById('output');
        image.src = URL.createObjectURL(event.target.files[0]);
    };
</script>
```

- Trong ProductsController sửa action [HttpPost]Create() như sau:

```
...
try
{
    if (ModelState.IsValid)
    {
        product.Image = "";
        var f = Request.Files["ImageFile"];
        if (f != null && f.ContentLength > 0)
        {
            //Use Namespace called : System.IO
            string FileName = System.IO.Path.GetFileName(f.FileName);
            //Lấy tên file upload
            string UploadPath = Server.MapPath("~/wwwroot/WineImages/" + FileName);
            //Copy Và lưu file vào server.
            f.SaveAs(UploadPath);
            //Lưu tên file vào trường Image
            product.Image = FileName;
        }
        db.Products.Add(product);
        db.SaveChanges();
    }
    return RedirectToAction("Index");
}
...
```

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

## 10. Upload ảnh trong phần sửa sản phẩm

- Trong view Create.cshtml của folder Products
  - Sửa @using (Html.BeginForm()) thành

```
@using (Html.BeginForm("Edit", "Products", FormMethod.Post, new { enctype =
    "multipart/form-data" }))
```

- Khai báo biến lấy đường dẫn đến file ảnh

```
@{
    ViewBag.Title = "Edit";
    var ImagePath = "~/wwwroot/WineImages/" + Model.Image;
}
```

- sửa đoạn code lấy hiển thị tên file ảnh

```
<div class="form-group">
    @Html.LabelFor(model => model.Image, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Image, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Image, "", new { @class = "text-danger" })
    </div>
</div>
```

Thành

```
<div class="form-group">
    @Html.LabelFor(model => model.Image, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        
        <p><label for="ufile" style="cursor: pointer;">Chọn file ảnh</label></p>
        <input name="ImageFile" id="ufile" type="file" style="display: none;"
        onchange="loadFile(event)" />
    </div>
</div>
```

- Thêm đoạn code script sau vào cuối file:

```
<script>
    var loadFile = function (event) {
        var image = document.getElementById('output');
        image.src = URL.createObjectURL(event.target.files[0]);
    };
</script>
```

- Trong ProductsController sửa action [HttpPost]Edit() như sau:

```
...
try
{
    if (ModelState.IsValid)
    {
        var f = Request.Files["ImageFile"];
        if (f != null && f.ContentLength > 0)
        {
            //Use Namespace called : System.IO
            string FileName = System.IO.Path.GetFileName(f.FileName);
            //Lấy tên file upload
            string UploadPath = Server.MapPath("~/wwwroot/WineImages/" + FileName);
            //Copy Và lưu file vào server.
            f.SaveAs(UploadPath);
            //Lưu tên file vào trường Image
            product.Image = FileName;
        }
        db.Entry(product).State = EntityState.Modified;
        db.SaveChanges();
    }
    return RedirectToAction("Index");
}
...
}
```

- Ấn phím F5 (hoặc Ctrl+F5) để chạy thử.
- \*\*\*Sửa code sử dụng **EditorTemplates** và **DisplayTemplates** trong việc hiển thị và upload ảnh