Báo cáo tuần 4

Linq và Entity Framework

1. **Entity Framework**

Là một ORM(Object Relational Mapping)framework giúp dễ dàng truy xuất và lưu trữ dữ liệu thông qua việc ánh xạ database thành các objects tương ứng trong code

1. **Các thành phần trong EF**

* EDM(Entity Data Model)

Là thành phần dùng để lưu trữ các thông tin ánh xạ,nối các lớp model với CSDL,mang lại sự tách biệt giữa chương trình và CSDL

* EDM bao gồm 3 thành phần chính
* Mô hình khái niệm(Concept schema definition language)

Chứa các lớp mô hình và mối quan hệ giữa các lớp,độc lập với mô hình quan hệ các bảng trong database

* Mô hình lưu trữ (store schema definition language)

Là mô hình thiết kế bao gồm các table,view,stored procedure,mối quan hệ giữa chúng các bảng và khóa.Mô hình này thể hiện gần giống mô hình quan hệ các bảng trong database

* Mô hình ánh xạ(Mapping specification language)

Gồm thông tin về các mô hình khái niệm được ánh xạ đến mô hình lưu trữ

* LINQ to Entities

Là ngôn ngữ để viết truy vấn đến các object model,trả về các thực thể đã định nghĩa trong mô hình khái niệm

* Entity SQL

Là một ngôn ngữ truy vấn khác tương tự như LINQ to Entities

* Object service

Entity Client data provider

Chịu trách nhiệm tương tác với Data provider của ADO.Net,chuyển đổi truy vấn LINQ to entities hoặc entity sql về truy vấn SQL.Sau đó Data provider sẽ thực thi truy vấn SQL

Một nhiệm vụ khác của Entity Client là chuyển đổi kết quả thu được từ truy vấn SQL về dạng trung gian,kết quả này chuyển lên cho Object Services xử lý

* Object Service

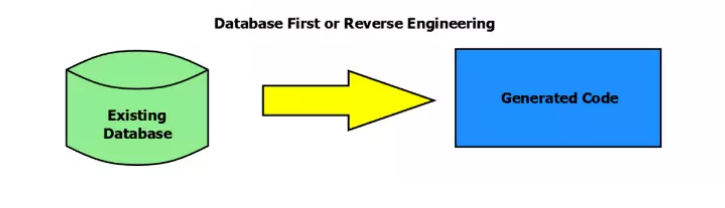
Là cánh cổng truy xuất dữ liệu,thực hiện quá trình gọi là chuyển đổi dữ liệu từ Entity Client về dạng Object.Lưu ý dữ liệu dạng bảng từ Entity Client không phải là dữ liệu bảng thu trực tiếp từ CSDL mà là dạng trung gian gọi là Model tabular structure

Ngoài ra còn đảm nhận nhiệm vụ quản lý và theo dõi trạng thái của object

1. **Các hướng tiếp cận của Entity Framework**

* Database First

Đây là hướng tiếp cận đầu tiên trong Entity Framework,có thể sử dụng hướng tiếp cận này khi database đã được thiết kế và sẵn sàng sử dụng,giúp giảm số lượng code cần phải viết



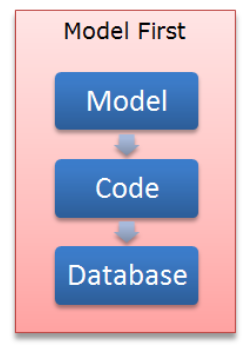
Ưu điểm

* Cách này rất phổ biến khi ta đã có sẵn CSDL
* EDM tự động tạo các thực thể và mối quan hệ của nó
* Viết code ít hơn vào việc tạo model

Nhược điểm

* Rất khó để duy trì một mô hình phức tạp
* Tùy biến model rất khó.Vd khi ta không có khóa ngoại trong DB và cần tạo liên kết trong mô hình khái niệm thì ta phải tạo theo cách thủ công và khó duy trì khi ta cập nhật model từ DB
* Model First

Hướng tiếp cận này sử dụng ORM designer để thiết kế model trước,sau đó trình thiết kế này sẽ tạo ra các lớp model tương ứng và mã SQL để tạo CSDL.Có nghĩa là tạo sơ đồ của thực thể và mối quan hệ sẽ tự động chuyển đổi thành code



Ưu điểm

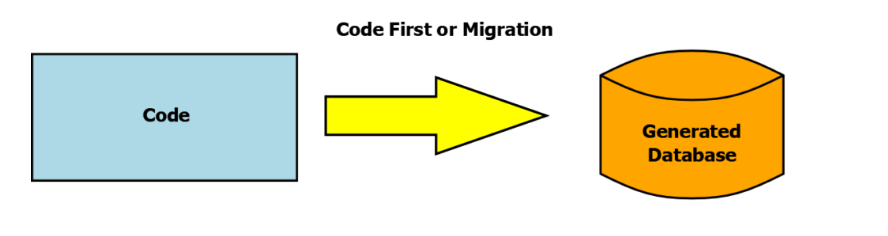
* Trình thiết kế trực quan để tạo lược đồ cơ sở dữ liệu
* Sơ đồ mô hình có thể cập nhật dễ dàng khi CSDL thay đổi
* EF tạo code và database script

Nhược điểm

* Khi thay đổi mô hình và tạo SQL để đồng bộ CSDL thì luôn mất dữ liệu vì phải drop table trước
* Không có nhiều khả năng kiểm soát ở thực thể và CSDL
* Cần có kiến thức tốt về EF để cập nhật mô hình và CSDL
* Code First

Đây là hướng tiếp cận được sử dụng phổ biến nhất

Với vai trò là lập trình viên,ta không cần quan tâm đến SQL hay trình thiết kế.Tất cả những gì mà ta làm là xây dựng các lớp model theo kiểu hướng đối tượng.EF sẽ làm tất cả những gì còn lại để ta có CSDL phù hợp.Có nghĩa là,ta chỉ cần code mô hình khái niệm,EF sẽ tự sinh ra mô hình ánh xạ và mô hình lưu trữ



Ưu điểm

* Có toàn quyền kiểm soát model từ code
* Hỗ trợ migration,vì thế rất dễ đồng bộ CSDL

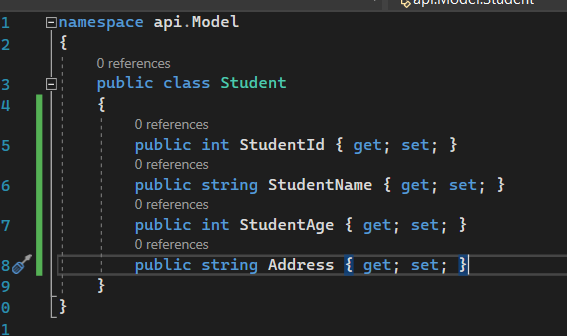
Nhược điểm

* Khó duy trì CSDL so với công cụ thiết kế
* Yêu cầu có kiến thức tốt về C# và data annotation

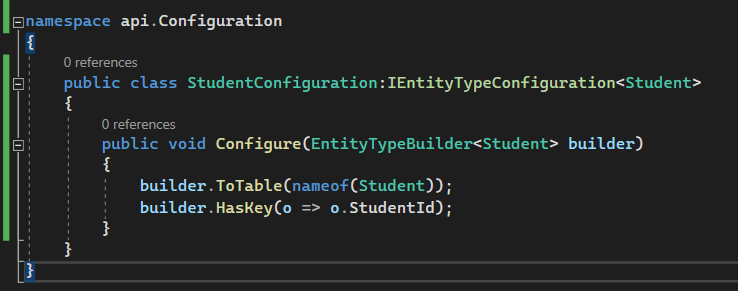
1. **Demo sử dụng C# kết nối SQL sử dụng Entity Framework**

* Hướng tiếp cận :Code First

B1:tạo model



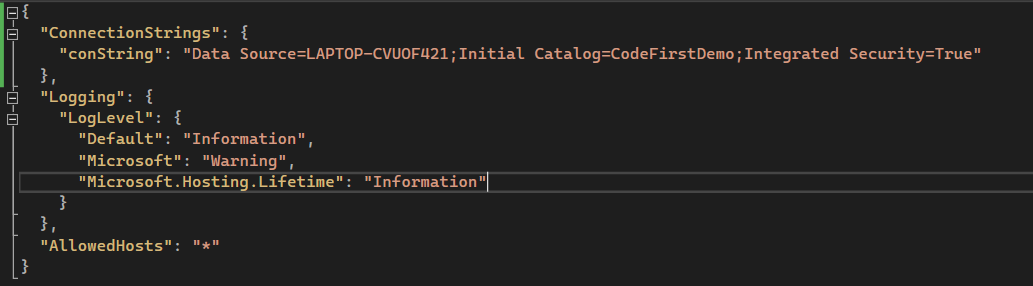
B2:config model



B3:tạo Context



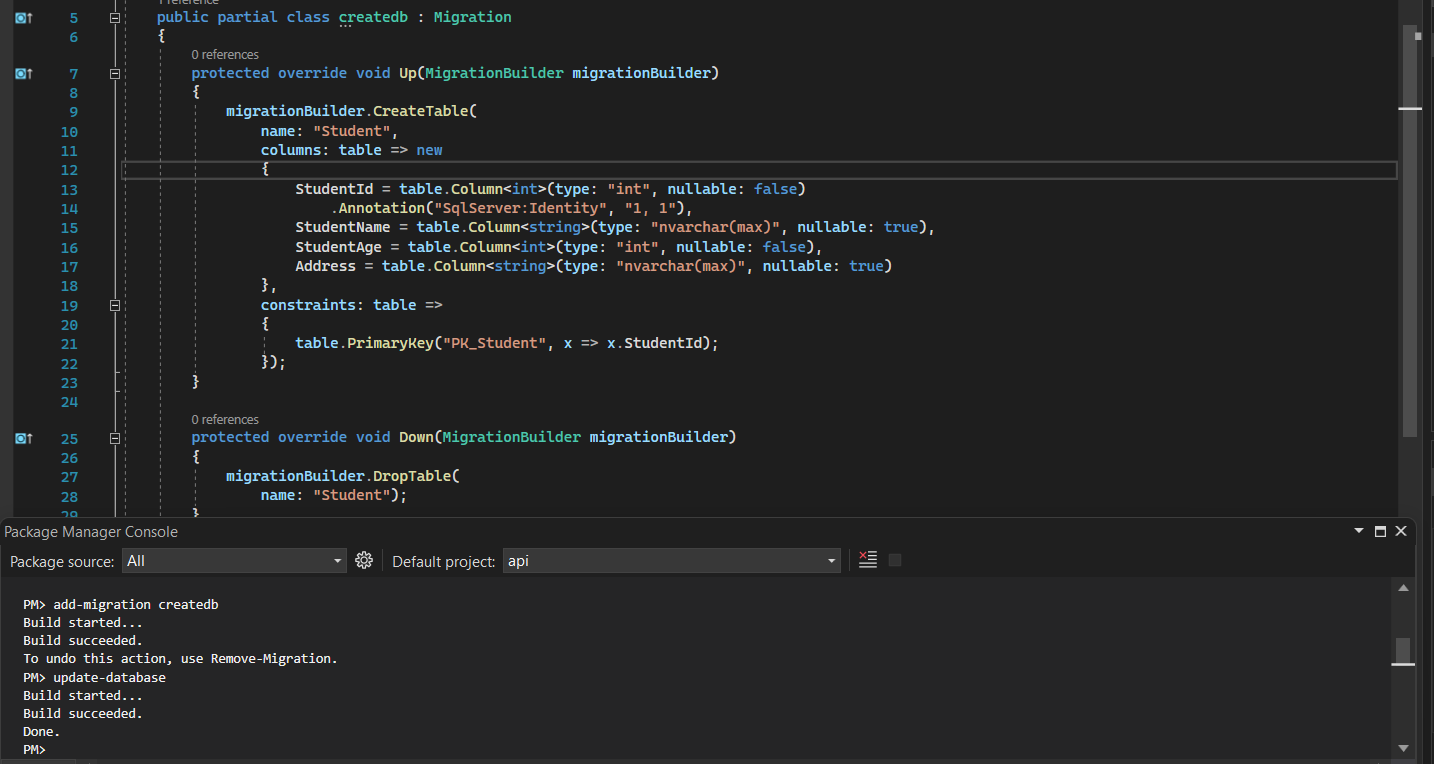
B4:Tạo connectionstring



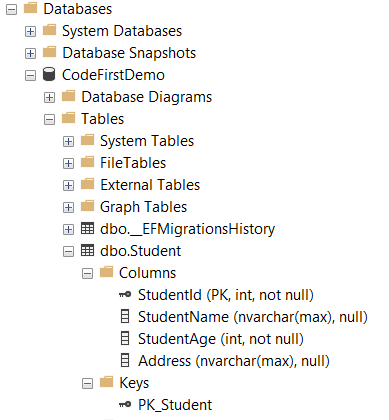
B5:add DBContext



B6:add migration và update database

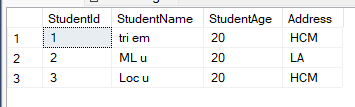


Kết quả

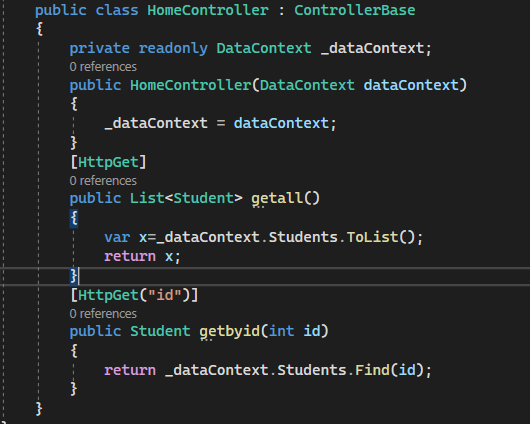


Tương tác dữ liệu và truy vấn với EF

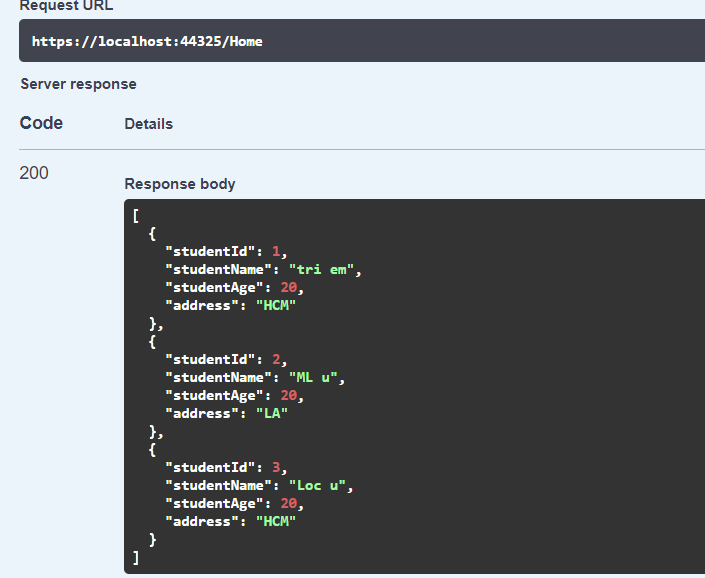
* Database

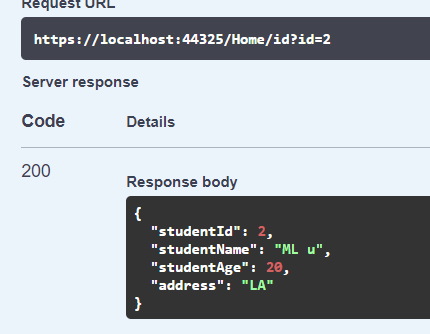


Tạo những phương thức để lấy và truy vấn dữ liệu



Kết quả





* **Linq**

LINQ (Language Integrated Query) - ngôn ngữ truy vấn tích hợp - nó tích hợp cú pháp truy vấn (gần giống các câu lệnh SQL) vào bên trong ngôn ngữ lập trình C#, cho nó C# khả năng truy cập các nguồn dữ liệu khác nhau (SQL Db, XML, List ...) với cùng cú pháp.

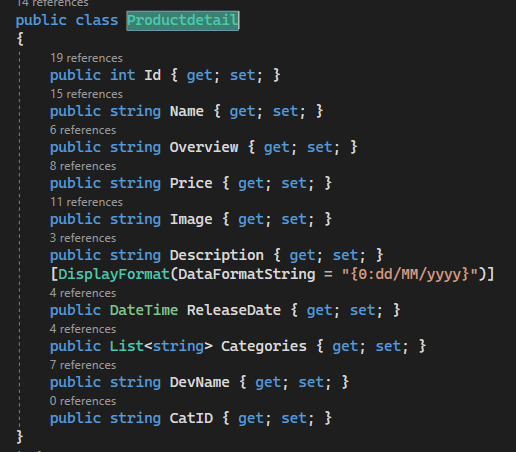
Projection

Select:lấy giá trị từ collection

Select many:lấy các giá trị từ collection trong collection,nghĩa là tập hợp lồng nhau

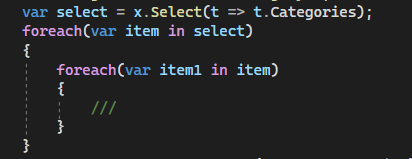
VD

model

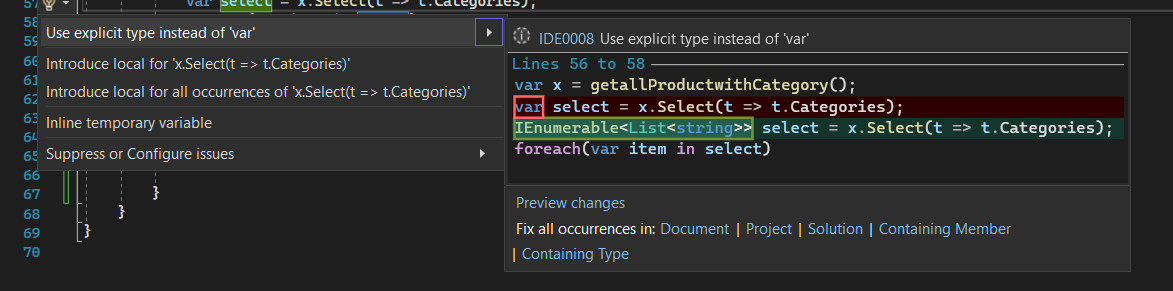


Giả sử lấy từng giá trị của Categories trong list Productdetail

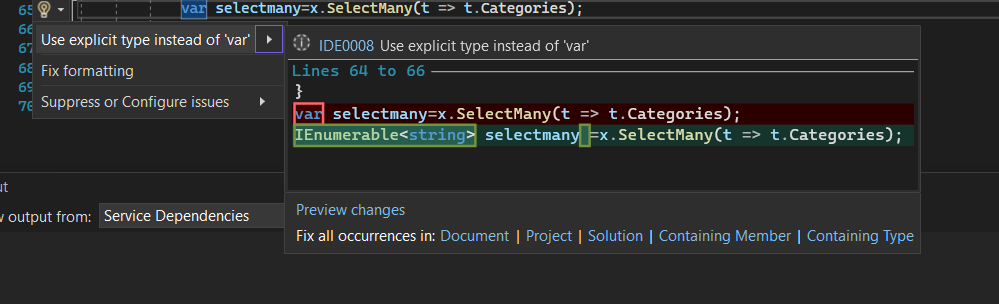
Với select



do select chỉ lấy ra được list nên ta phải foreach thêm để lấy từng giá trị



Đối với selectmany thì không cần thiết



Restriction

Dùng để lọc dữ liệu theo điều kiện

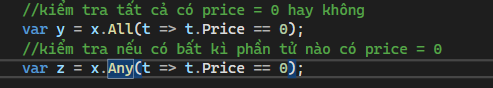
Vd



Quantifiers

Trả về kiểu dữ liệu Boolean cho biết 1 số hoặc toàn bộ các phần tử có thỏa mãn 1 điều kiện nào đó hay không

VD



Ordering

Dùng sắp xếp thứ tự các phần tử dựa trên 1 hoặc nhiều thuộc tính

OrderBy:sắp xếp tăng dần

OrderbyDescending:sắp xếp giảm dần

Thenby:sắp xếp tăng dần theo trường tiếp theo nếu trường phía trước có giá trị giống nhau

ThenbyDescending:sử dụng để sắp xếp giảm dần theo trường tiếp theo nếu trường phía trước có giá trị giống nhau

Reverse:đảo ngược thứ tự các phần tử

Grouping

Trả về nhóm dữ liệu từ 1 tập hợp dựa trên key đã cung cấp

Nhóm theo 1 trường



Nhóm theo nhiều trường

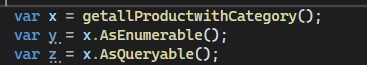


Conversion(chuyển đổi)

Sử dụng trong việc chuyển đổi kiểu dữ liệu của các phần tử trong 1 tập hợp

AsEnumerable & AsQueryable

Dùng để cast hoặc covert đối tượng nguồn thành Ienumerable<T> or Iqueryable<T>



Cast

Giống như AsEnumerable<T>,chuyển đối tượng nguồn thành Ienumerable<T>



ToArray(),ToList(),ToDictionary()

Như tên gọi,các phương thức này sẽ chuyển đổi đối tượng nguồn thành Array,List,Dictionary

Element

Trả về phần tử cụ thể,được xác định từ 1 chuỗi

ElementAt:trả về phần tử tại 1 chỉ mục được chỉ định trong collection

ElementAtOrDefault:tương tự như ElementAt nhưng sẽ trả về giá trị mặc định khi chỉ mục nằm ngoài phạm vi

First:trả về phần tử đầu tiên hoặc phần tử đầu tiên thỏa mãn điều kiện

FirstOrDefault:tương tự như First nhưng sẽ trả về kết quả mặc định nếu không có phần tử nào tồn tại

Last:trả về phần tử cuối hoặc phần tử cuối thỏa mãn điều kiện

LastOrDefault:tương tự như FirstOrDefault nhưng thay vì phần tử đầu thì đây là phần tử cuối

Single:trả về phần tử duy nhất hoặc phần tử duy nhất thỏa mãn điều kiện

SingleOrDefault:tương tự như Single nhưng sẽ trả về kết quả mặc định khi không có phần tử trả về.Throw an [InvalidOperationException](https://learn.microsoft.com/en-us/dotnet/api/system.invalidoperationexception) nếu có hơn 1 phần tử trả về

Generation

Dùng để tạo ra 1 chuỗi giá trị

DefaultIfEmpty:thay thế collection trống thành collection có giá trị mặc định

Empty:trả về collection trống

Range:tạo collection chứa 1 chuỗi số

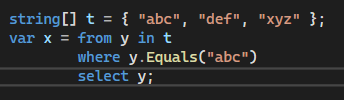
Repeat:tạo collection chứa 1 giá trị lặp lại

Các loại LINQ trong truy vấn

LINQ to Objects

Thường trả về các biển kiểu Ienumerable<T>,cung cấp 1 hướng tiếp cận mới thay vì như trước,viết các dòng lặp foreach dài dòng để lấy dữ liệu.Thì nay đã thay thế bằng cách khai báo các dữ liệu mong muốn được truy xuất

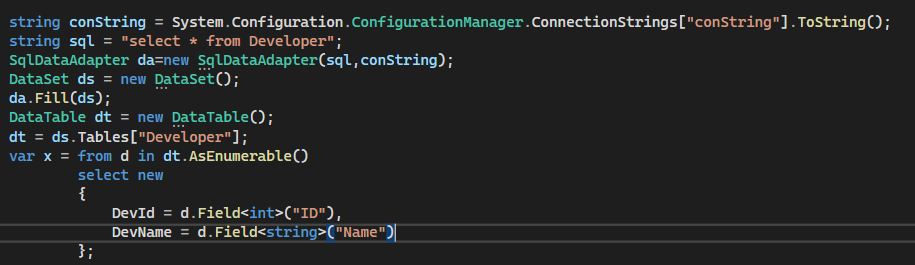
Vd



LINQ to Dataset

Giúp thực hiện truy vấn trực tiếp trên Dataset

Vd:

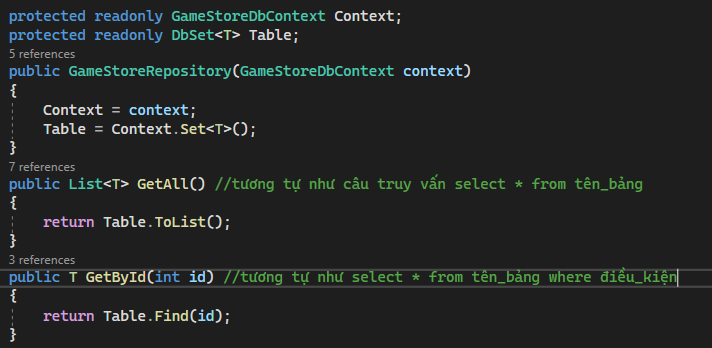


Các loại LINQ truy vấn

LINQ to Entities

Dùng để truy vấn dữ liệu tới cơ sở dữ liệu thông qua hỗ trợ của Entity Framework,nó chuyển đổi truy vấn viết bằng LINQ sang truy vấn SQL để thực thi trên CSDL.Do LINQ là thống nhất,những gì ta biết khi học LINQ to Object có thể áp dụng được trên LINQ to Entities

Vd:



LINQ to SQL

Cho phép truy vấn dữ liệu trong SQL Server bằng cách sử dụng LINQ thông thường

Vd:

LINQToSQLDataContext context=new LINQToSQLDataContext();

Var x= from d context.Developer

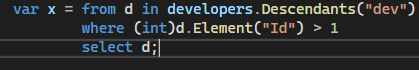
Where d.ID==1

Select d;

LINQ to XML

Cho phép truy vấn LINQ trên các XML document để lấy dữ liệu cần thiết

Vd



Báo cáo

**Mô hình MVC là gì?**

MVC là viết tắt của “Model-View-Controller”,là mô hình thiết kế được sử dụng trong kỹ thuật phần mềm.MVC là mẫu kiến trúc phần mềm chia thành 3 thành phần được kết nối với nhau và mỗi thành phần có nhiệm vụ riêng và độc lập với thành phần khác

1. **Các thành phần trong MVC**
2. Model

Đây là một bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là cầu nối giữa 2 thành phần View và Controller.

Model thể hiện dưới hình thức là một cơ sở dữ liệu. Nó hiện rõ các thao tác với cơ sở dữ liệu như cho phép xem, truy xuất, xử lý dữ liệu,…

1. View

Đây là phần giao diện (theme) dành cho người dùng. Nơi mà người dùng có thể lấy được các thông tin dữ liệu của MVC thông qua thao tác truy vấn như tìm kiếm hoặc thông qua các website.

Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của người dùng để tương tác với phần Controller.

Tuy nhiên, View lại không có mối quan hệ trực tiếp với Controller. Đồng thời, nó cũng không lấy được dữ liệu từ Controller mà chỉ có chức năng hiển thị yêu cầu và chuyển cho Controller.

Vd:nút “Create” được tạo bởi View khi người dùng nhấn vào sẽ có 1 hành động trong controller

1. Controller

là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

Sự tương tác giữa các thành phần

* **Controller** tương tác với qua lại với **View**
* **Controller** tương tác qua lại với **Model**
* **Model** và **View** không có sự tương tác với nhau mà nó tương tác với nhau thông qua **Controller**.

1. **Luồng xử lý mô hình MVC**

Client gửi yêu cầu đến server thông qua Controller, Controller sẽ tiếp nhận yêu cầu.

Client gửi yêu cầu đến server thông qua Controller, Controller sẽ tiếp nhận yêu cầu.  
Model tương tác với Database để truy xuất dữ liệu phù hợp với yêu cầu.

Database trả về cho Model dữ liệu theo yêu cầu của Model.

Model trả về dữ liệu cho Controller xử lý.

Controller sẽ gọi đến View phù hợp với yêu cầu và kèm theo dữ liệu cho View. View chịu trách nhiệm hiển thị dữ liệu phù hợp với yêu cầu.

Sau khi xử lý hiển thị dữ liệu, View trả về cho Controller kết quả (HTML, XML hoặc JSON…).

Sau khi hoàn tất, Controller sẽ trả về kết quả cho Client.

Đây là luồng xử lý cơ bản và phổ biến nhất

Các trường hợp xử lý khác trong mô hình MVC

Trường hợp 1: Người dùng chỉ gửi yêu cầu chuyển từ trang hiện tại sang một trang khác của web không có yêu cầu về dữ liệu.

**Request** được gửi từ **Browser** (trình duyệt) đến **Route** (nơi định tuyến hay phân định các request sẽ được xử lý ở đâu là trực tiếp tại **Route** hay là tại **Controller** nào đó).

Nếu để việc **redirect** trang ở trực tiếp tại **Route** thì luồng hoạt động của nó là yêu cầu chuyển trang sẽ được xử lý ngay tại **Route** mà không cần gọi đến **Controller**

**Trường hợp 2:** Người dùng gửi 1 **request** **redirect** về 1 trang khác của web có trả về dữ liệu (ví dụ như xem danh sách các Tour đã đặt).

Sau khi **request** gửi về **Route** được **Route** chuyển về xử lý tại **Controller**, lúc này các yêu cầu liên quan đến dữ liệu sẽ được xử lý và **Controller** sẽ có sự tương tác với **Model** để lấy dữ liệu, **Controller** sẽ sử dụng các **lớp/hàm** trong **Model** cần thiết để lấy ra những dữ liệu chính xác. **Model** tương tác với **Database** để lấy dữ liệu, dữ liệu trả về được gửi về **Model** từ **Model** gửi lại về **Controller**, **Controller** gọi đến **View** phù hợp với **request** kèm theo dữ liệu cho **View**, **View** sẽ lắp dữ liệu tương ứng vào **HTML** và gửi lại một **HTML** cho **Controller** sau khi thực hiện xong nhiệm vụ của mình. Hoàn tất các công đoạn trên **Controller** sẽ trả kết quả về **Browser**.

Trường hợp Người dùng chỉ yêu cầu dữ liệu nhưng không chuyển trang

Thứ tự xử lý tương tự ở trường hợp 2 nhưng đến khi **Controller** nhận được dữ liệu trả về thì không gọi đến **View** mà trả **respone** lại cho trình duyệt thông qua **API**, dữ liệu trả về thường sẽ là dạng **JSON**.

1. **Ưu và nhược điểm của MVC**

Kiểm tra dễ dàng hơn:Với MVC, bạn có thể dễ dàng thực hiện các công việc như kiểm tra, rà soát lỗi phần mềm trước khi đưa tới tay người tiêu dùng. Việc này đảm bảo chất lượng và độ uy tín sản phẩm cao hơn.

Tạo mô hình chuẩn cho dự án, giúp cho việc tiếp cận với ứng dụng dễ dàng hơn

Trình tự xử lý rõ ràng, nhiệm vụ riêng biệt, độc lập với các thành phần khác và các thành phần có thể tương tác được với nhau.

Đơn giản: Như đã đề cập ở trên, kết cấu của MVC khá đơn giản nên mang đến sự dễ dàng cho người sử dụng, ngay cả khi bạn không có nhiều am hiểu về lập trình.

1. **Nhược điểm**

Phù hợp với dự án lớn,dự án quy mô nhỏ thì MVC dường như trở nên cồng kềnh,tốn thời gian phát triển

1. **ASP.Net MVC là gì?**

Là framework dựa trên mô hình MVC dành cho ASP.Net

1. **Cấu trúc thành phần của của ASP.Net**

Ngôn ngữ: ASP.NET sở hữu rất nhiều ngôn ngữ lập trình như C#, VB.net, PHP, [JavaScript](https://mona.media/thiet-ke-website-bang-ngon-ngu-javascript/)….. Trong đó, C# và VB.net phổ biến và được sử dụng nhiều nhất trong phát triển ứng dụng web.

Thư viện**:** .NET Framework gồm có một bộ các lớp library chuẩn. Library phổ biến nhất được sử dụng cho các ứng dụng web trong .net là Web library. Web library bao gồm tất cả các thành phần cần thiết sử dụng trong phát triển các ứng dụng web-based.

CLR(Common Language Runtime): CLR sẽ thực hiện các tác vụ chính bao gồm xử lý các trường hợp cá biệt và thu gom rác VÀ CLI(Common Language Infrastructure) là một nền tảng dùng để chạy các chương trình .Net

Một số đặc điểm của ASP.Net framework

* Trạng thái code rời

Là khái niệm tách biệt giữa thiết kế và code,giúp việc duy trì ứng dụng ASP.Net dễ dàng hơn.Loại file thông dụng của ASP.Net là aspx. Giả sử chúng ta có một trang web có tên là MyPage.aspx, sẽ có một tệp khác có tên là MyPage.aspx.cs biểu thị cho phần mã code của trang. Bởi vậy, Visual Studio mới tạo ra các tập tin riêng biệt cho mỗi trang web, một cho phần thiết kế và một dành cho mã code.

* Khả năng quản lý các trạng thái

Đặc tính này giúp các ứng dụng, dịch vụ web ghi nhớ các trạng thái bằng các phương tiện quản lý, kiểm soát trạng thái có sẵn trên ASP.NET. Vì vậy, nó thường được sử dụng trong thiết kế tính năng giỏ hàng trong các trang thương mại điện tử.

* Bộ nhớ Cache

Nhằm cải thiện hiệu suất làm việc của ứng dụng cũng như hỗ trợ lưu trữ thông tin, hỗ trợ truy xuất, phản hồi trong thời gian ngắn, mà ASP.NET lựa chọn lưu trữ trên bộ nhớ Cache.

Báo cáo

**Routing**

* Là quá trình ánh xạ(mapping) truy vấn HTTP với 1 đối tượng xử lý truy vấn(handler). Trong MVC, handler chính là action của controller. Có nghĩa là, routing trong ASP.Net MVC là cơ chế ánh xạ giữa truy vấn HTTP và controller action
* Với khái niệm như trên,routing trong asp.Net MVC buộc phải chỉ định rõ mỗi URL tương ứng với action nào.ASP.Net MVC sử dụng 2 cách thức khác nhau để chỉ định:sử dụng template chung cho nhiều controller action(conventional routing) hoặc chỉ định URL cụ thể cho từng controller action riêng biệt(attribute routing)
* **Route template**

Là chuỗi kí tự quy định hình thức của URL hợp lệ

Template mặc định có dạng

{controller}/{action} hoặc {controller}/{action}/{id}

Trong đó

{controller}/{action}/{id} gọi là placeholder hay route parameter.Place holder {controller} và {action} được sử dụng để xác định xem URL này tương ứng với controller với action nào

Ký tự / phân chia chuỗi template

{id} là không bắt buộc

Template này sử dụng cơ chế conventional routing

* **Conventional routing**

Là routing định nghĩa cho nhiều controller action dựa trên quy ước .Bất kì controller và action nào tuân theo quy ước đều tự động tham gia vào routing tương ứng

* Quy trình hoạt động

1. Khi truy vấn tới,ASP.Net MVC sẽ trích Ra URL từ truy vấn
2. URL sẽ so khớp lần lượt với các template để xem phù hợp với template nào
3. Nếu có nhiều route template,sẽ so khớp lần lượt với từng template theo thứ tự khai báo cho đến khi phù hợp
4. Nếu tìm thấy template phù hợp thì sẽ dừng lại dù vẫn tiếp theo vẫn còn template
5. Tên controller và action được trích ra từ URL
6. Thực thi action tương ứng

* **Attribute routing**

Là loại routing được định nghĩa riêng cho từng action bằng cách [Route] trước mỗi action

* Quy trình hoạt động

1. Khi chạy,ASP.Net MVC sẽ scan qua tất cả các action,nếu action được đánh dấu bằng [Route(…)] thì template tương ứng được trích ra và làm khóa cho từ điển URL,action được sử dụng làm giá trị của từ điển tương ứng,sau quá trình scan được các cặp URL=>action
2. Khi truy vấn tới,URL được trích ra và so sánh với các trong từ điển.Nếu trùng khóa nào thì action sẽ được thực thi
3. Nếu 1 action được thực thi,việc xử lý URL kết thúc.Tất cả conventional template bỏ qua
4. Nếu không thấy khóa trong từ điển,ASP.Net MVC tiếp tục so với conventional template
5. **Controller**

Xử lý mọi URL yêu cầu đến,chứa các phương thức public gọi là các action method.Controller và các action method xử lý các yêu từ trình duyệt,lấy dữ liệu và trả về kết quả tương ứng

Trong ASP.Net MVC mọi tên lớp của Controller phải kết thúc bằng từ Controller-bắt buộc,ngoài ra mỗi Controller phải đặt trong thư mục Controllers của project

VD: StudentController là Controller của Student

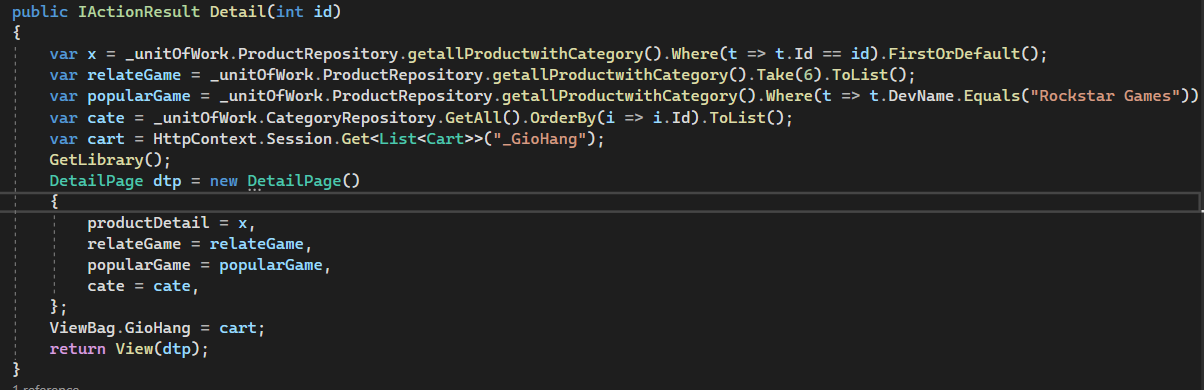
1. **Action**

là phương thức xử lý truy vấn trong lớp controller.Bất kì phương thức public nào của lớp controller đều có thể trở thành action method,lớp controller không giới hạn các action

các action method cũng giống như method bình thường nhưng có vài hạn chế sau

* Action method phải là public,không thể private hay protected
* Action method không thể overload(nạp chồng)
* Action method không thể là static method

Vd:



Detail():tên action method

IActionResult:kiểu trả về

Id:tham số đầu vào

View():được định nghĩa trong base controller class trả về ActionResult

Phương thức ActionResult

Các phương thức thường gặp nhất

View():trả về kết quả chứa HTML

Content():trả về kết quả chứa chuỗi ký tự

Redirect():chuyển hướng người dùng

Json():trả về dữ liệu kiểu Json

File():trả về nội dung của file

NotFound():trả về kết quả 404 Not found

1. **Filter**

Là một lớp tùy chỉnh,nơi ta viết logic tùy chỉnh để thực thi trước hoặc sau khi 1 action method được thực thi

* Các loại filter trong MVC
* Authorization filter:thực hiện xác thực và ủy quyền trước khi thực hiện action method
* Action filter:thực hiện thao tác trước và sau khi action method thực thi
* Result filter:thực hiện thao tác trước và sau khi thực thi kết quả trả về của action method
* Exception filter:thực hiện thao tác khi có ngoại lệ chưa được xử lý trong quá trình thực thi
* Các cấp độ của filter
* Cấp độ toàn cục:áp dụng cho tất cả controller và action method
* Cấp độ controller:áp dụng cho tất cả action method của controller chỉ định,không áp dụng cho controller khác
* Cấp độ action method:áp dụng cho 1 action method riêng lẻ

1. **Action Selector**

Là 1 attribute áp dụng cho controller action,giúp Route Engine chọn đúng action method để xử lý request,gồm 3 kiểu action selector

1. **Action Name**

Định nghĩa tên action,Route Engine sẽ xử dụng tên này thay vì tên action method để match với action name trong routing

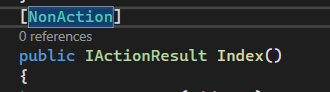
Vd:



Trong đoạn code phía trên,action name cho hành động Detail được đổi thành ProductDetails,ta không gọi phương thức tên Detail nữa mà là ProductDetails trong URL

1. **Non Action**

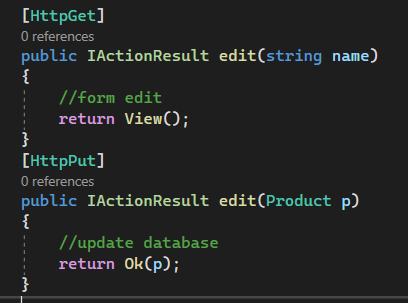
Chỉ định 1 phương thức không phải là action method



1. **Action Verbs**

Dùng để xử lý các loại yêu cầu HTTP khác nhau,cho phép ta định nghĩa 2 phương thức cùng tên để xử lý các yêu cầu HTTP khác nhau

Vd:

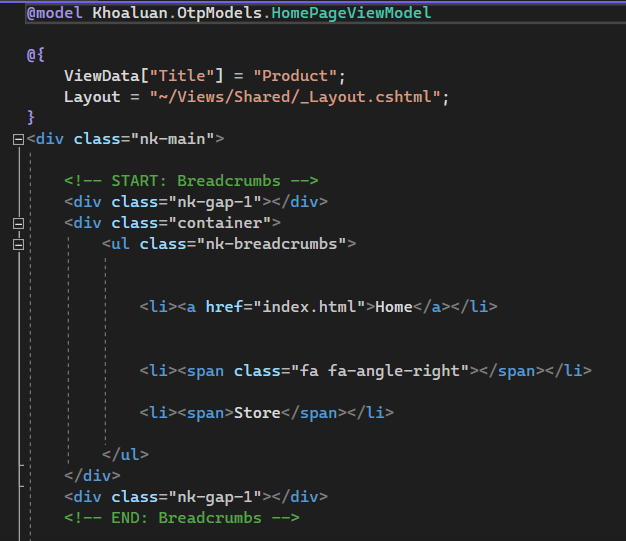


1. **View,Layout View,Partial View**
2. **View**

Là giao diện người dùng,hiển thị dữ liệu từ model cho người dùng và cũng cho phép sửa đổi dữ liệu

Razor view:cho phép viết HTML và code C# trong Razor View,sử dụng cú pháp Razor bắt đầu bằng ký tự @ cho code C# thay vì <%...%> như truyền thống

Vd:



@model,@{}:cú pháp Razor

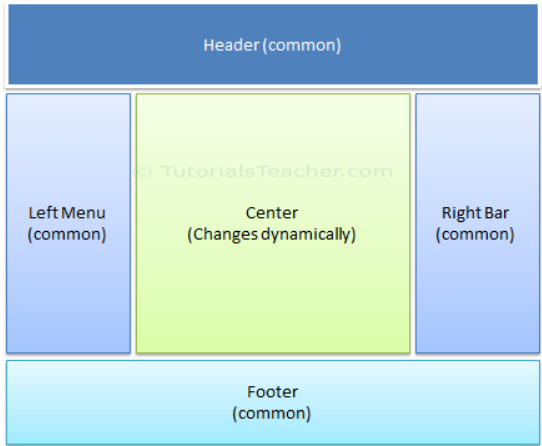
<div></div>:HTML

1. **Layout View**

Là giao diện dùng để dùng chung cho mọi trang,tránh bị trùng lặp code.Layout View tương tự như master page của ASP.Net Web Forms

Cho phép xác định 1 mẫu web chung,có thể kế thừa trong nhiều view để tạo ra giao diện nhất quán trong nhiều trang của ứng dụng

Vd:giao diện người dùng chứa phần đầu trang(header),thanh menu bên trái,bên phải và phần chân trang(footer) và chỉ thay đổi phần trung tâm



Lúc này thì center sẽ là placeholder chứa các view muốn hiển thị trên web

@RenderBody():do các trang giống nhau chỉ khác nhau ở placeholder nên @RenderBody() sẽ trả về view mà ta muốn hiển thị

1. **Partial View**

Là thành phần có thể sử dụng lại trong web,có thể sử dụng trong nhiều View hoặc Layout View,giúp loại bỏ trùng lặp code

1. **ViewBag,ViewData,TempData**
2. **ViewBag**

Dùng để chuyển dữ liệu tạm thời từ controller sang view,là thuộc tính kiểu Dynamic

Vd:

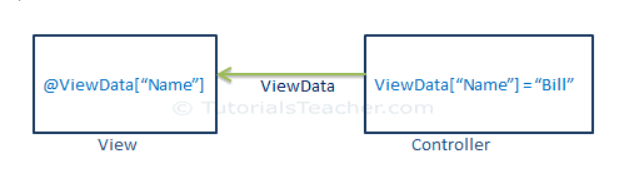


Từ vd trên ta thấy thuộc tính Name được gán giá trị là “Bill” thì bên View ta truy cập giá trị này thông qua @ViewBag.Name và sẽ hiển thị giá trị “Bill” trên giao diện người dùng

1. **ViewData**

Tương tự như ViewBag,truyền dữ liệu từ Controller sang View.ViewData có kiểu Dictionary,chứa các cặp key-value trong đó mỗi key phải là một chuỗi

VD:



1. **TempData**

TempData được sử dụng để chuyển dữ liệu từ Controller sang View và ngược lại từ action method này sang action method khác của cùng Controller hoặc 1 Controller khác

TempData lưu trữ dữ liệu tạm thời và tự động xóa dữ liệu đó sau khi xuất giá trị,ta có thể giữ nó cho yêu cầu tiếp theo bằng phương thức TempData.Keep()

1. **Model**

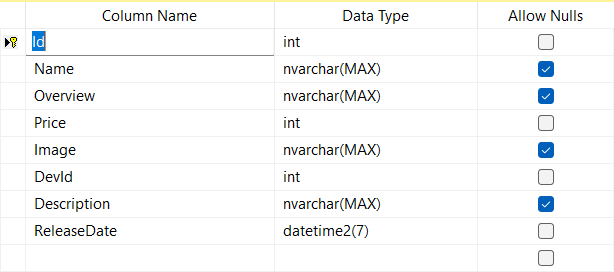
Là tập hợp các đối tượng chứa dữ liệu và có thể chứa thêm logic,model chia thành một số loại dựa trên công dụng và nơi chúng sử dụng,gồm 3 loại

* **Domain Model**

Thể hiện 1 đối tượng trong Database,thường có quan hệ 1-1 với 1 bảng trong CSDL.Domain Model liên quan đến tầng truy cập dữ liệu (DAL) trong ứng dụng.Domain Model cũng được hiểu như Entity Model hay Data Model

VD

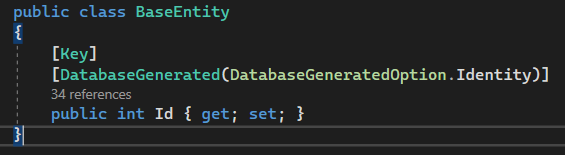
Ta có bảng Product như sau



Nó được đại diện bởi Product Model



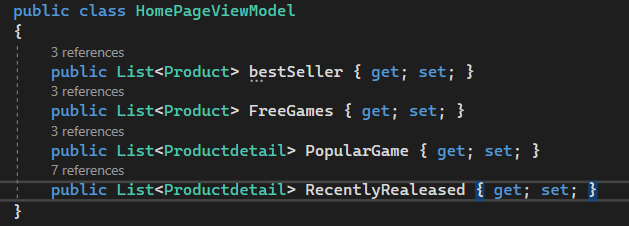
Được kế thừa bởi lớp Base Entity(có thể chứa trong Product Model nếu muốn)



* **View Model**

View Model được tham chiếu đến các đối tượng chứa dữ liệu cần cho việc hiển thị cho người dùng.View Model liên quan đến View của ứng dụng,được định nghĩa dựa trên cách thức dữ liệu hiển thị cho người dùng hơn là cách chúng lưu trữ.View sẽ cần dùng đến Model này

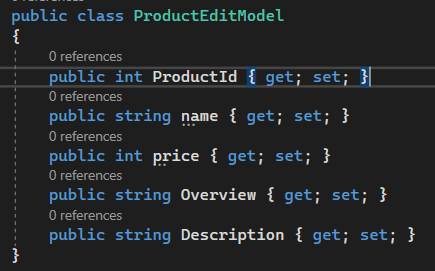
Vd:Home Page cần hiển thị các loại sản phẩm thì View Model trở thành



* **Edit Model(hoặc Input Model)**

Là đại diện dữ liệu cần để người dùng thay đổi hoặc thêm mới.Yêu cầu của Model cần chỉnh sửa khác với yêu cầu hiển thị

Vd:Khi người dùng chỉnh sửa sản phẩm có một số thuộc tính cần thay đổi thì Model trở thành



* **Model Binding**

Là cơ chế map dữ liệu gửi qua HTTP Request vào các tham số của action method trong Controller.HTTP Request có thể chứa dữ liệu nhiều định dạng.Dữ liệu có thể chứa trong HTML Form,có thể 1 phần trên route value hoặc trên query string hoặc có thể là 1 phần của body Request

Cơ chế ASP.NET MVC model binding cho phép chúng ta dễ dàng bind các giá trị này vào các tham số của action method. Các tham số này có thể là kiểu nguyên thủy hoặc kiểu đối tượng phức tạp.

* Cơ chế hoạt động

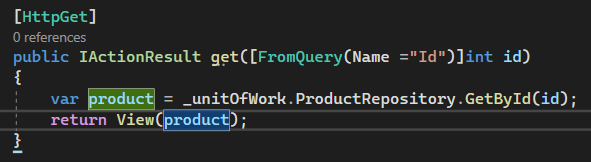
Model Binding là quá trình 2 bước.Đầu tiên,thu thập các giá trị từ yêu cầu HTTP đến và sau đó,liên kết chúng với tham số kiểu nguyên thủy hoặc kiểu phức tạp

Binding kiểu dữ liệu nguyên thủy

Request nhúng dữ liệu trực tiếp vào chuỗi truy vấn,ASP.Net MVC sẽ tự động chuyển đổi chuỗi truy vấn thành các tham số của action method

VD:/Product/?Id=1

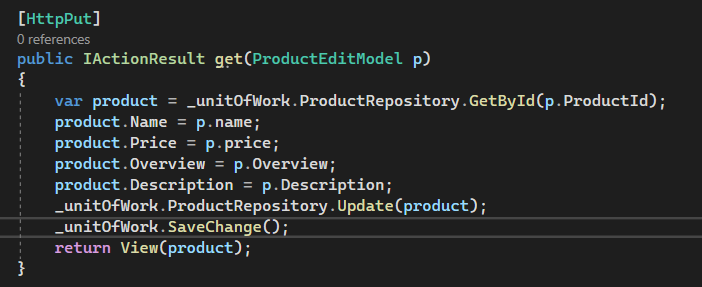
Thì tham số trong action method sẽ là



Binding dữ liệu phức tạp

Model Binding sẽ tự động chuyển đổi dữ liệu từ các trường biểu mẫu của HTTP Request thành các thuộc tính của một tham số kiểu dữ liệu phức tạp của 1 action method

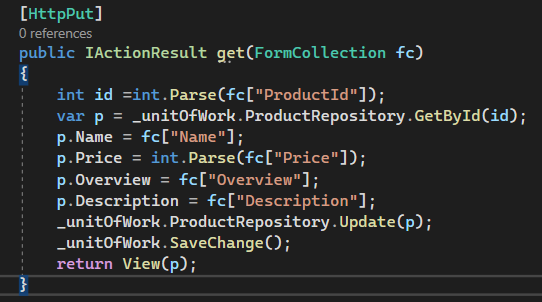
Vd:



**FormCollection**

Ngoài ra cũng có thể có tham số kiểu FormCollection trong action method để truy xuất tất cả giá trị từ các trường

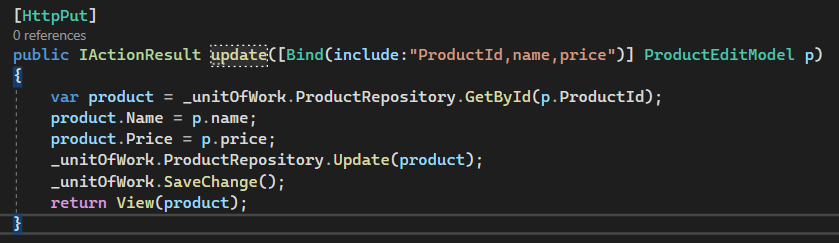
Vd:



Thuộc tính Bind

Cho phép chỉ định các thuộc tính chính xác mà Model Binding liên kết

Vd:phương thức update sẽ chỉ liên kết thuộc tính ProductId,name,price của ProductEditModel



* **Validation**

Dùng để kiểm tra xem dữ liệu đầu vào của người dùng có hợp lệ hay không

Validation bằng cách sử dụng thuộc tính chú thích dữ liệu

Các thuộc tính chú thích dữ liệu dùng để xác thực

Require:chỉ định thuộc tính là bắt buộc

StringLenght:độ dài tối thiểu và tối đa của thuộc tính kiểu chuỗi

Range:ràng buộc phạm vi cho thuộc tính kiểu số

RegularExpression:giá trị thuộc tính phải khớp với biểu thức chỉ định

CreditCard:chỉ định thuộc tính là thẻ tín dụng

CustomValidation:phương pháp xác thực tùy chỉnh để xác thực thuộc tính

EmailAddress:chỉ định thuộc tính là Email

FileExtension:xác thực phần mở rộng của tập tin

MaxLenght:độ dài tối đa được phép của thuộc tính

MinLenght:độ dài tối thiểu của thuộc tính

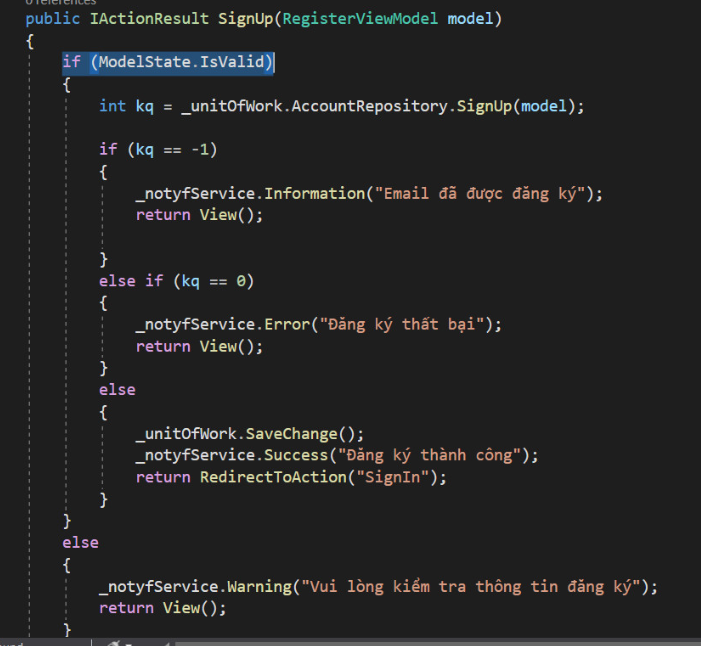
Phone:chỉ định thuộc tính là số điện thoại

Vd:

khi ta muốn validate dữ liệu với những ràng buộc cơ bản



Các ràng buộc trên là chưa đủ cho việc xác thực,ta cần phải kiểm tra xem dữ liệu đã gửi có hợp lệ không trong Controller.Có nghĩa là,ta cần phải kiểm tra trạng thái của Model bằng cách sử dụng ModelState.IsValid có đáp ứng yêu cầu bởi thuộc tính chú thích dữ liệu hay không.Đây là cách kiểm tra trạng thái của Model trước khi tương tác với Database



**Caching:OutputCache**

Là cơ chế cache giúp giảm thời gian load trang bằng cách giữ nội dung thường hay truy cập để tăng hiệu suất của website

Các tiêu chí để cache

Nội dung ít thay đổi(Vd:footer,header,menu,… tùy vào dự án)

Lượng truy xuất thường xuyên

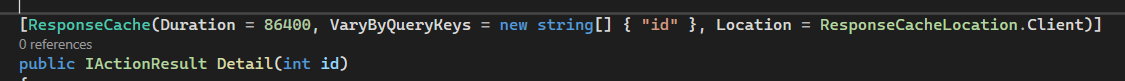
Các thuộc tính của OutputCache

Duration:thời gian tồn tại của cache,đơn vị giây

Location:vị trí của cache,mặc định là any

VaryByQueryKeys:các tham số của 1 method muốn cache

Vd:

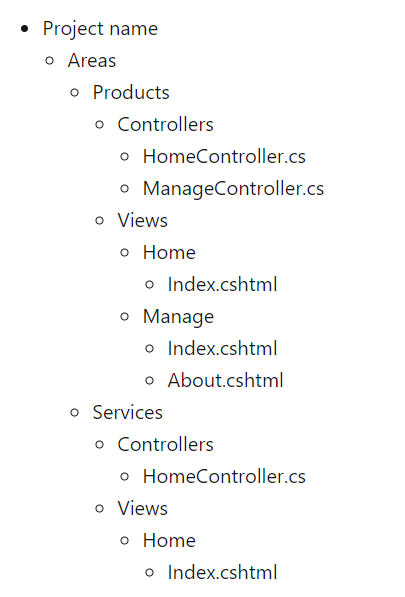


Do ResponseCache giống với OutputCache nên em sử dụng cho phần vd(tại em xài .Net core)

**Area**

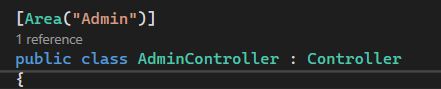
Là tính năng dùng để phân vùng thành nhóm chức năng nhỏ hơn,mỗi nhóm đều có Razor Pages,Controller,Model,View riêng

Cấu trúc của folder Area



Các Controller được chỉ định với thuộc tính [Area]

Vd:



**SCRIPTBUNDLE, STYLEBUNDLE**

SCRIPTBUNDLE:thực hiện việc thu nhỏ và đóng gói Javascript

STYLEBUNDLE:thực hiện việc rút gọn và đóng gói CSS

**Kendo**

Textbox

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Numeric Textbox

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with medium confidence

RadioButton

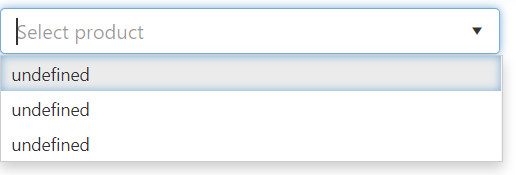
Text

Description automatically generated with low confidence

Text

Description automatically generated

Combobox



Dropdownlist

Background pattern

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generatedText

Description automatically generated

Multiselect

Graphical user interface, application

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

Datepicker

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Timepicker

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Grid

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

splitter

Graphical user interface, text, application, email

Description automatically generatedA picture containing text

Description automatically generated

Tooltip

Text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Popup

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated