

**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**  
**KHOA CÔNG NGHỆ THÔNG TIN**

**CÔNG NGHỆ PHÁT TRIỂN WEBSITE**  
**(LẬP TRÌNH WEBSITE VỚI ASP.NET MVC 5)**

**CHƯƠNG 3:**

**NGÔN NGỮ & KIẾN TRÚC TƯƠNG TÁC CSDL**  
**(LINQ TO SQL VÀ ENTITY FRAMEWORK)**



**Giảng Viên: ThS. Dương Thành Phết**  
**Email: phetcm@gmail.com**  
**Website: <http://www.thayphet.net>**  
**Tel: 091815867**

1. Entity Framework
2. LinQ To SQL

# 1. ENTITY FRAMEWORK

---

**1.1. Kiến trúc tổ chức**

**1.2. Các mô hình lập trình**

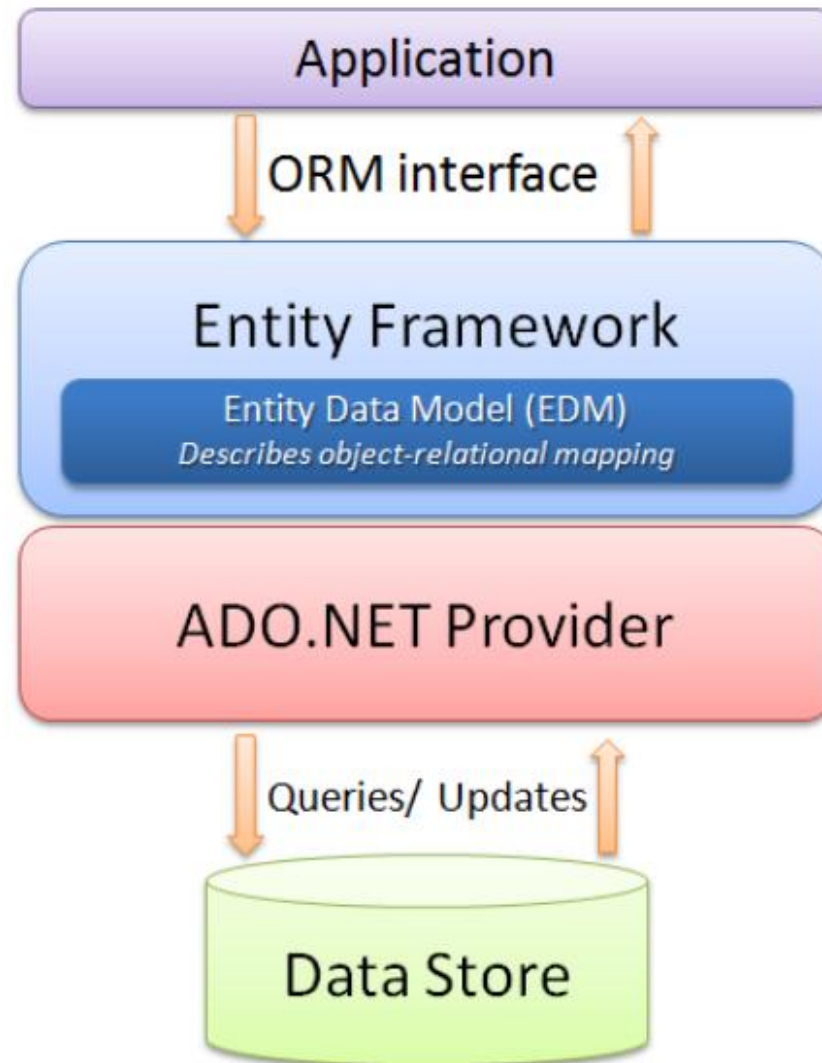
**1.3. Mô hình Database First**

**1.4. EF API**

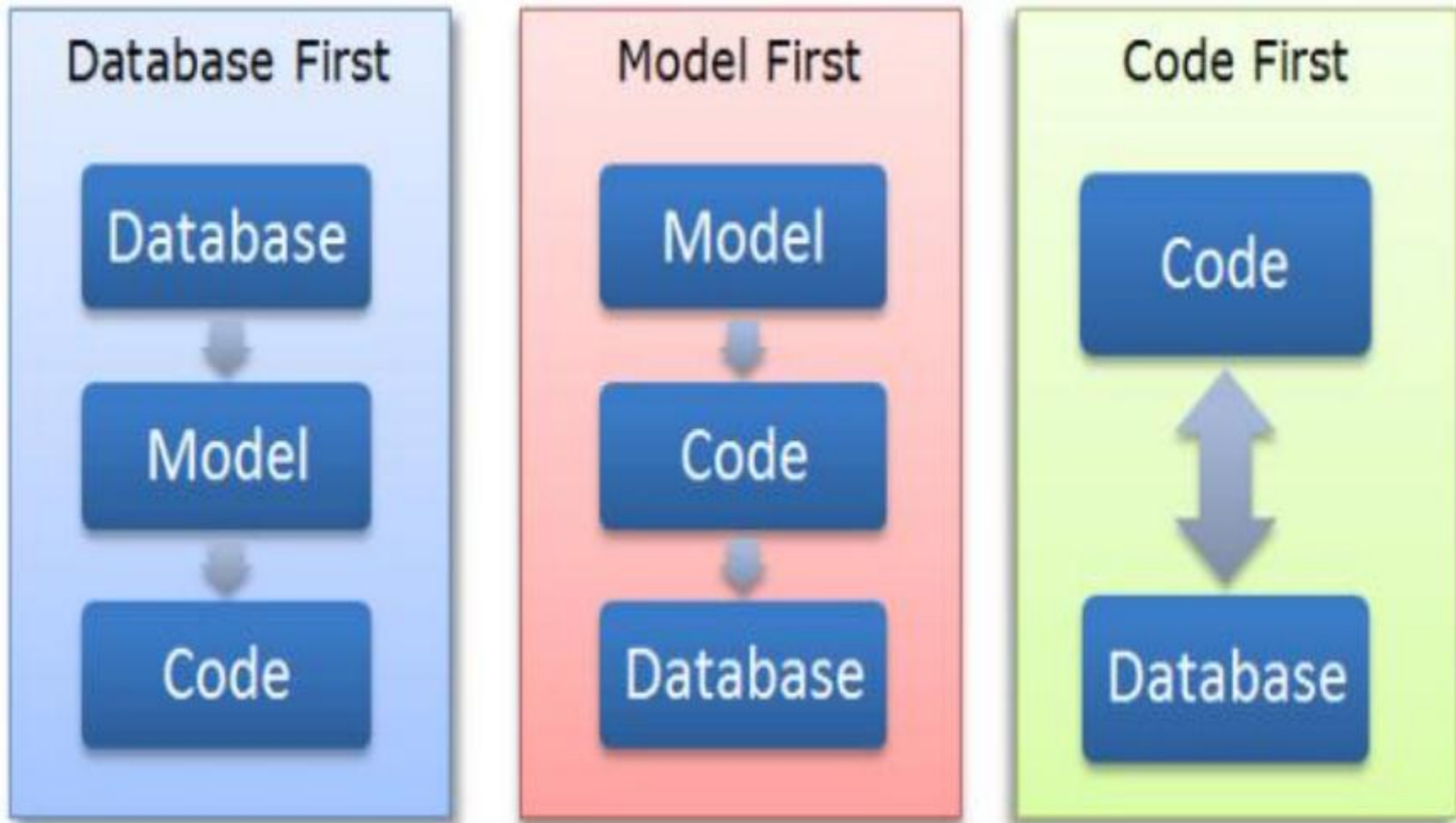
**1.5. Mô hình Code First**

- ✓ **Xây dựng Entity**
- ✓ **Xây dựng Database Context**
- ✓ **Khởi tạo dữ liệu**
- ✓ **Thay đổi thuộc tính**

# 1.1. KIẾN TRÚC TỔ CHỨC

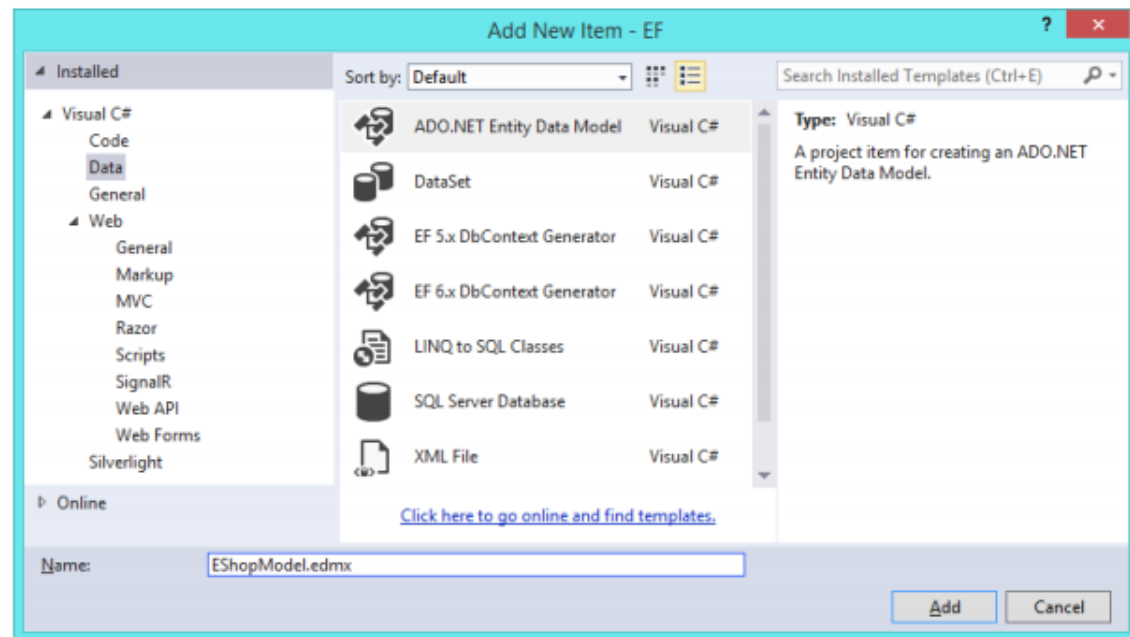
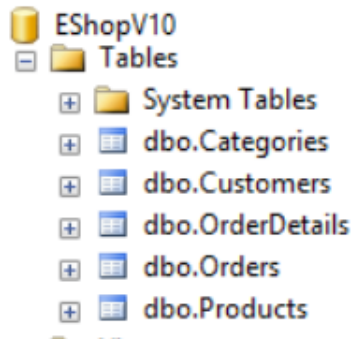
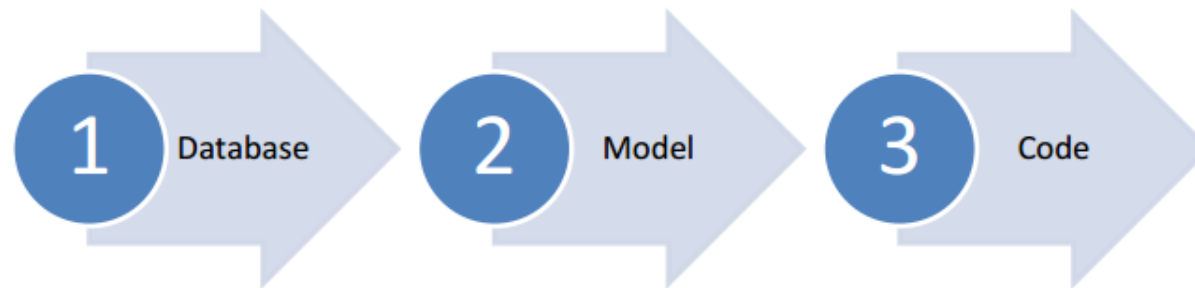


## 1.2. CÁC MÔ HÌNH LẬP TRÌNH



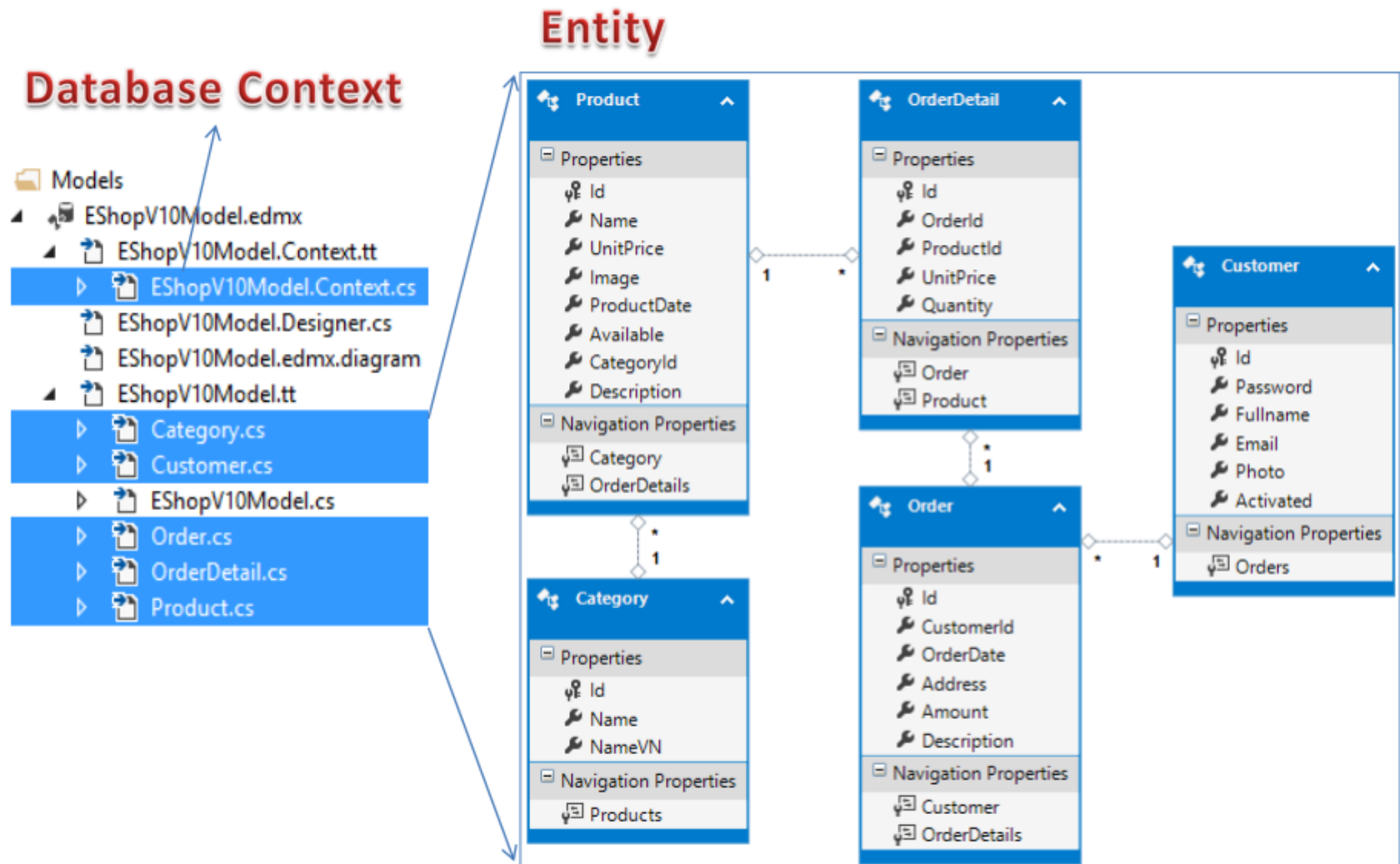
# 1.3. DATABASE FIRST

## Data First



# 1.3. DATABASE FIRST

## Database Model



## 1.3. DATABASE FIRST

### Database Context

```
public partial class EShopV10 : DbContext
{
    public EShopV10()
        : base("name=EShopV10")
    {
    }

    public DbSet<Category> Categories { get; set; }
    public DbSet<Customer> Customers { get; set; }
    public DbSet<OrderDetail> OrderDetails { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<Product> Products { get; set; }
}
```

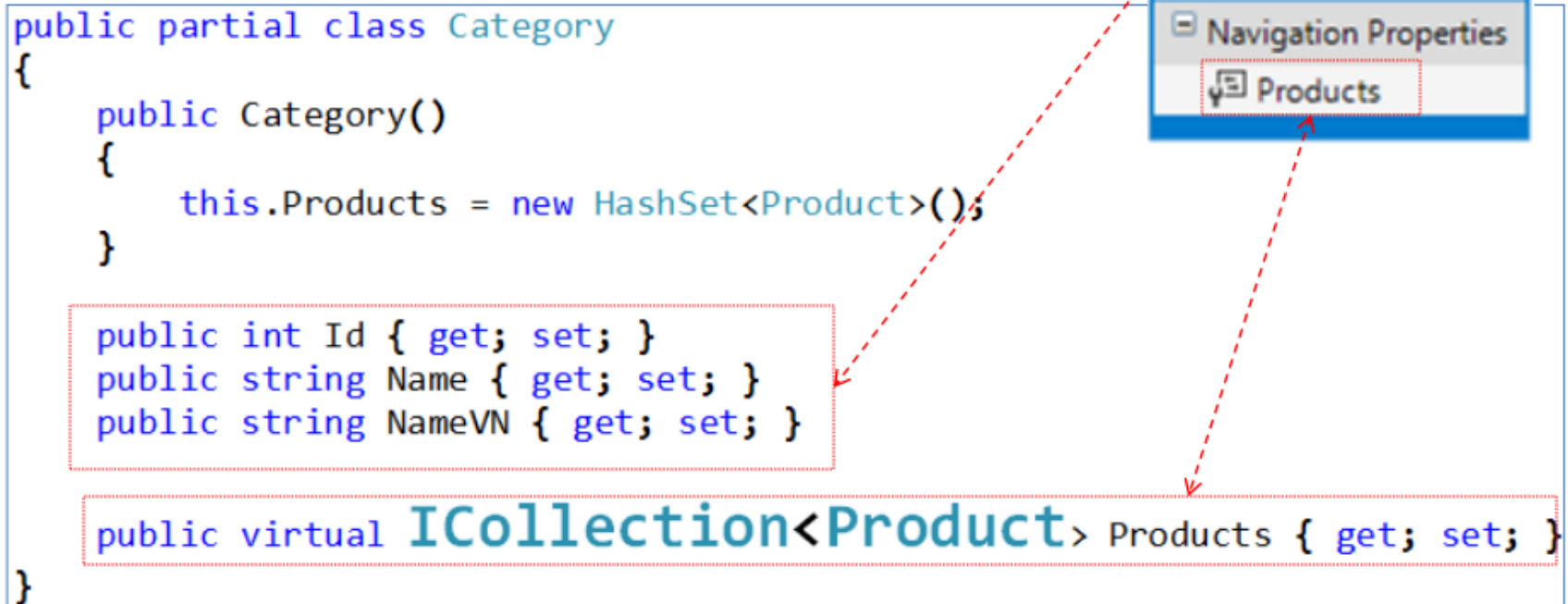
```
<connectionStrings>
    <add name="EShopV10" connectionString="..."
        providerName="System.Data.EntityClient" />
</connectionStrings>
```



## 1.3. DATABASE FIRST

### Chi tiết thực thể

- ❑ Tên thực thể <=> Tên bảng
- ❑ Tên thuộc tính <=> Tên cột
- ❑ Thực thể kết hợp 1-Nhiều



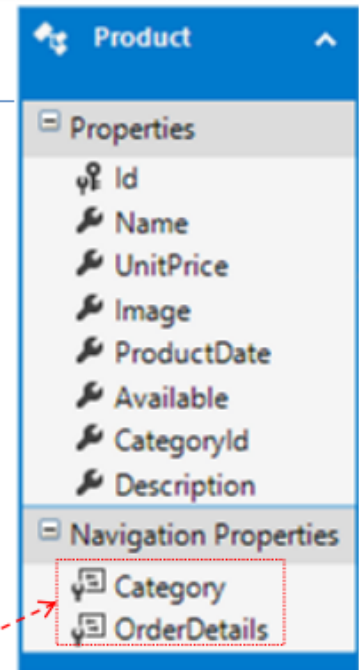
# 1.3. DATABASE FIRST

## Thực thể kết hợp

```
public partial class Product
{
    public Product()
    {
        this.OrderDetails = new HashSet<OrderDetail>();
    }

    public int Id { get; set; }
    public string Name { get; set; }
    public double UnitPrice { get; set; }
    public string Image { get; set; }
    public System.DateTime ProductDate { get; set; }
    public bool Available { get; set; }
    public int CategoryId { get; set; }
    public string Description { get; set; }

    public virtual Category Category { get; set; }
    public virtual ICollection<OrderDetail> OrderDetails { get; set; }
}
```



## 1.4. ENTITY FRAMEWORK API

### Thực thể kết hợp

- ❑ Tạo đối tượng db context

var db = **new EShopV10();**

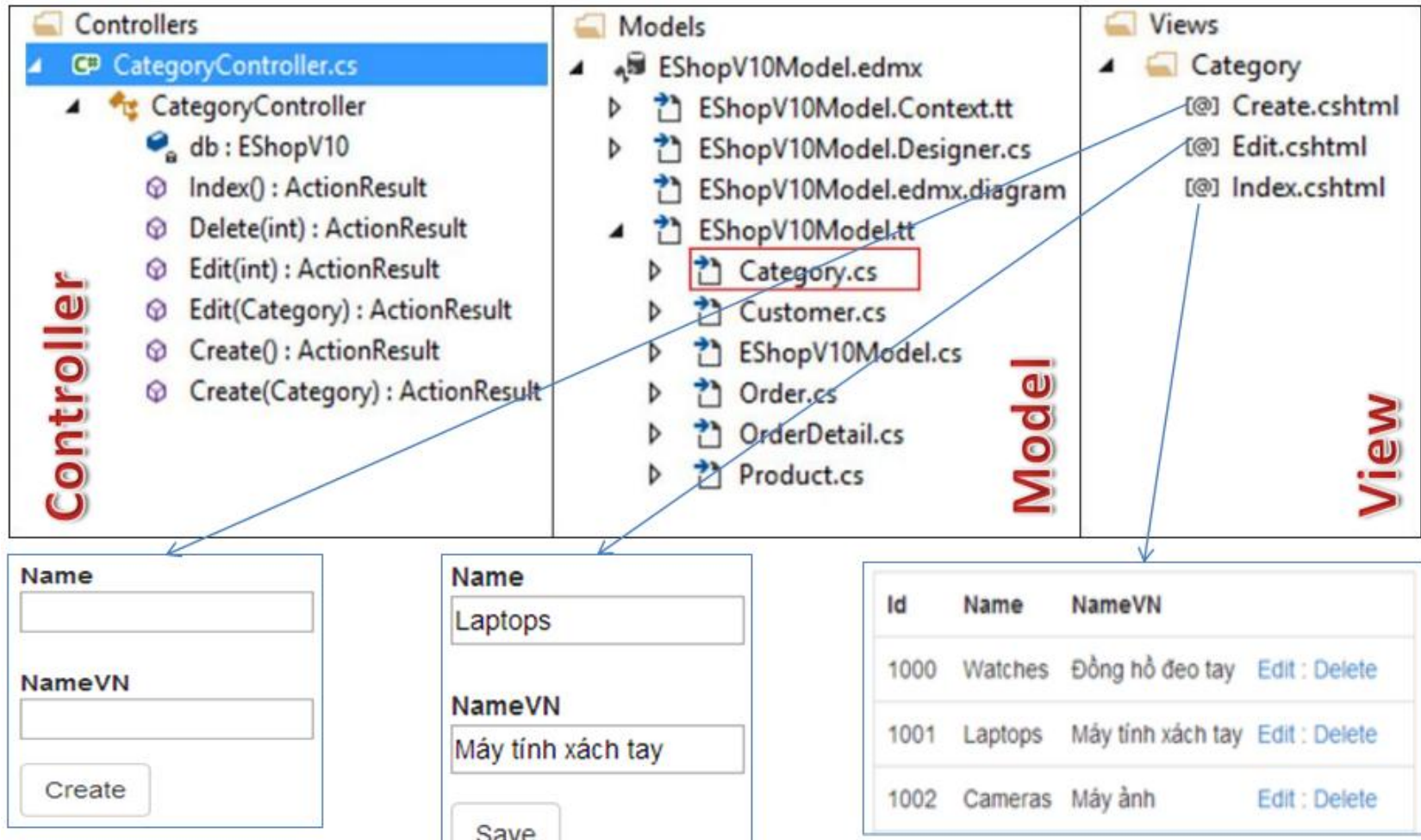
- ❑ Thao tác & truy vấn thực thể

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

- ❑ Lưu sự thay đổi  
**db.SaveChanges();**

# 1.4. ENTITY FRAMEWORK API

## Demo ứng dụng



## 1.4. ENTITY FRAMEWORK API

### Demo ứng dụng - Create

```
// GET:/Category/Create
public ActionResult Create()
{
    return View();
}

// POST:/Category/Create
[HttpPost]
public ActionResult Create(Category model)
{
    // Thêm mới
    db.Categories.Add(model);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Name

NameVN

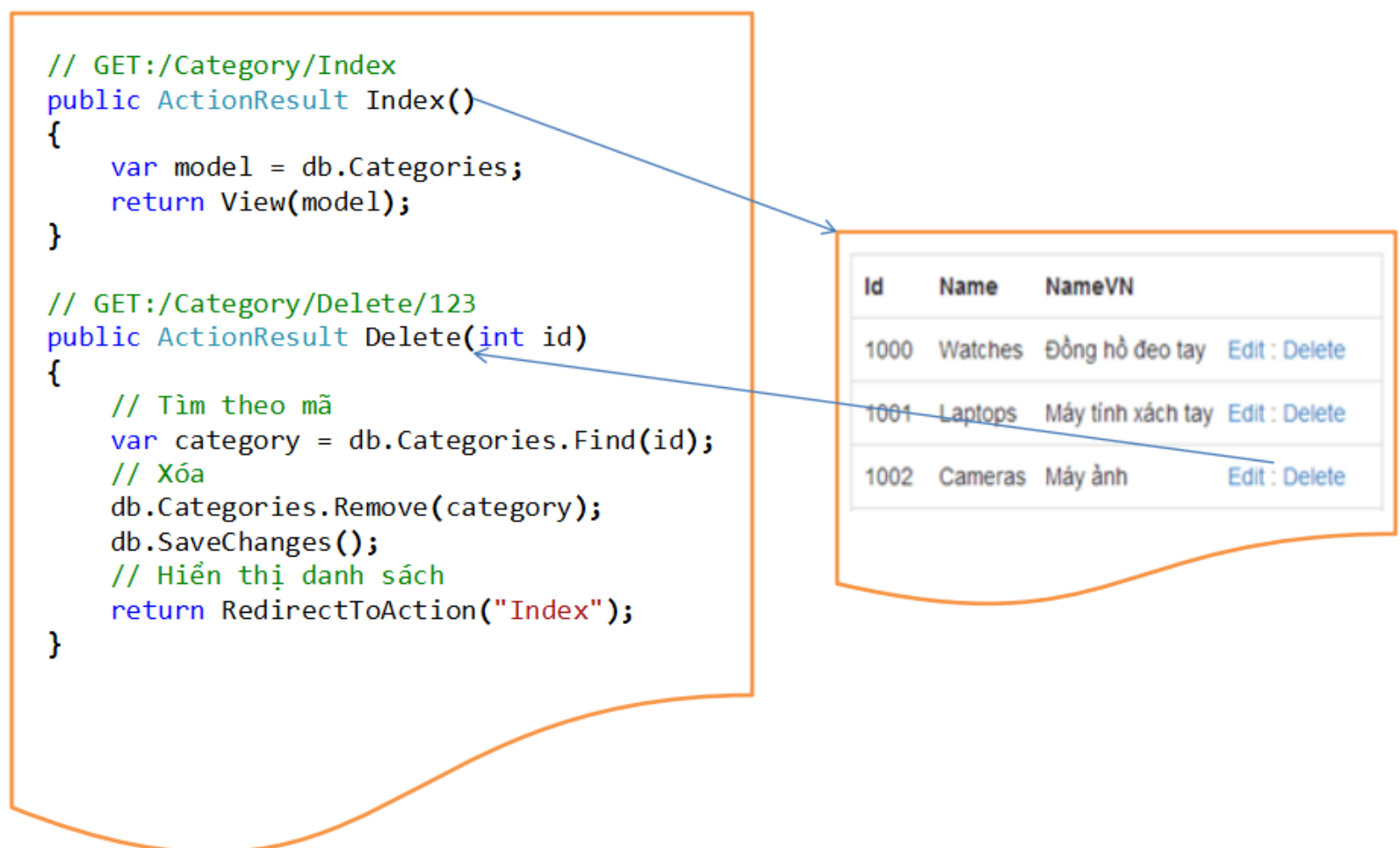
Create

# 1.4. ENTITY FRAMEWORK API

## Demo ứng dụng – List & Delete

```
// GET:/Category/Index
public ActionResult Index()
{
    var model = db.Categories;
    return View(model);
}

// GET:/Category/Delete/123
public ActionResult Delete(int id)
{
    // Tìm theo mã
    var category = db.Categories.Find(id);
    // Xóa
    db.Categories.Remove(category);
    db.SaveChanges();
    // Hiển thị danh sách
    return RedirectToAction("Index");
}
```



Id	Name	NameVN	
1000	Watches	Đồng hồ đeo tay	Edit : Delete
1001	Laptops	Máy tính xách tay	Edit : Delete
1002	Cameras	Máy ảnh	Edit : Delete



# 1.4. ENTITY FRAMEWORK API

## Demo ứng dụng – Edit

```
// GET:/Category/Edit/123
public ActionResult Edit(int id)
{
    var model = db.Categories.Find(id);
    return View(model);
}

// POST:/Category/Edit
[HttpPost]
public ActionResult Edit(Category model)
{
    db.Entry(model).State = EntityState.Modified;
    db.SaveChanges();
    return View(model);
}
```

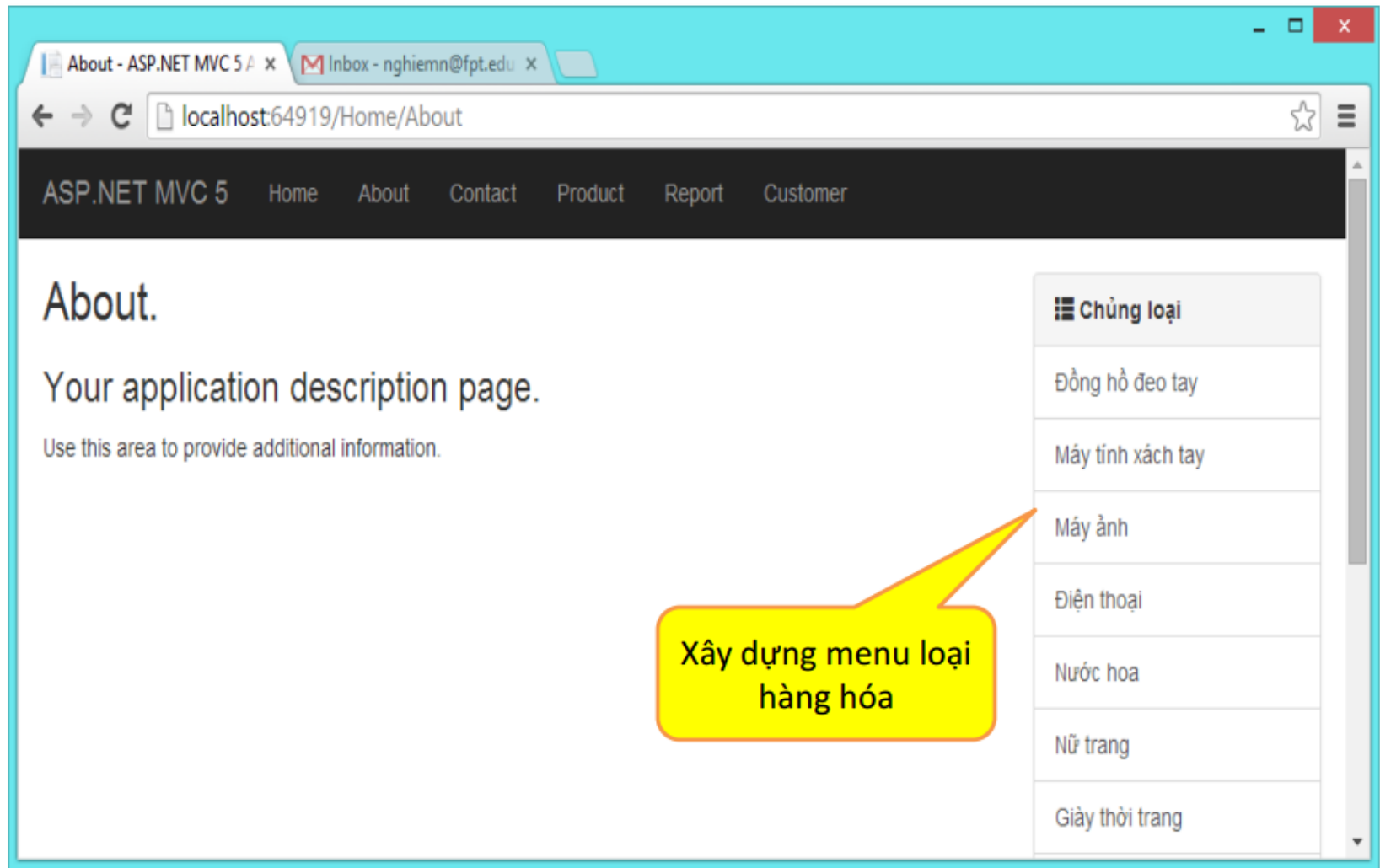
Id	Name	NameVN	
1000	Watches	Đồng hồ đeo tay	<a href="#">Edit</a> : <a href="#">Delete</a>
1001	Laptops	Máy tính xách tay	<a href="#">Edit</a> : <a href="#">Delete</a>
1002	Cameras	Máy ảnh	<a href="#">Edit</a> : <a href="#">Delete</a>

Name

NameVN

# 1.4. ENTITY FRAMEWORK API

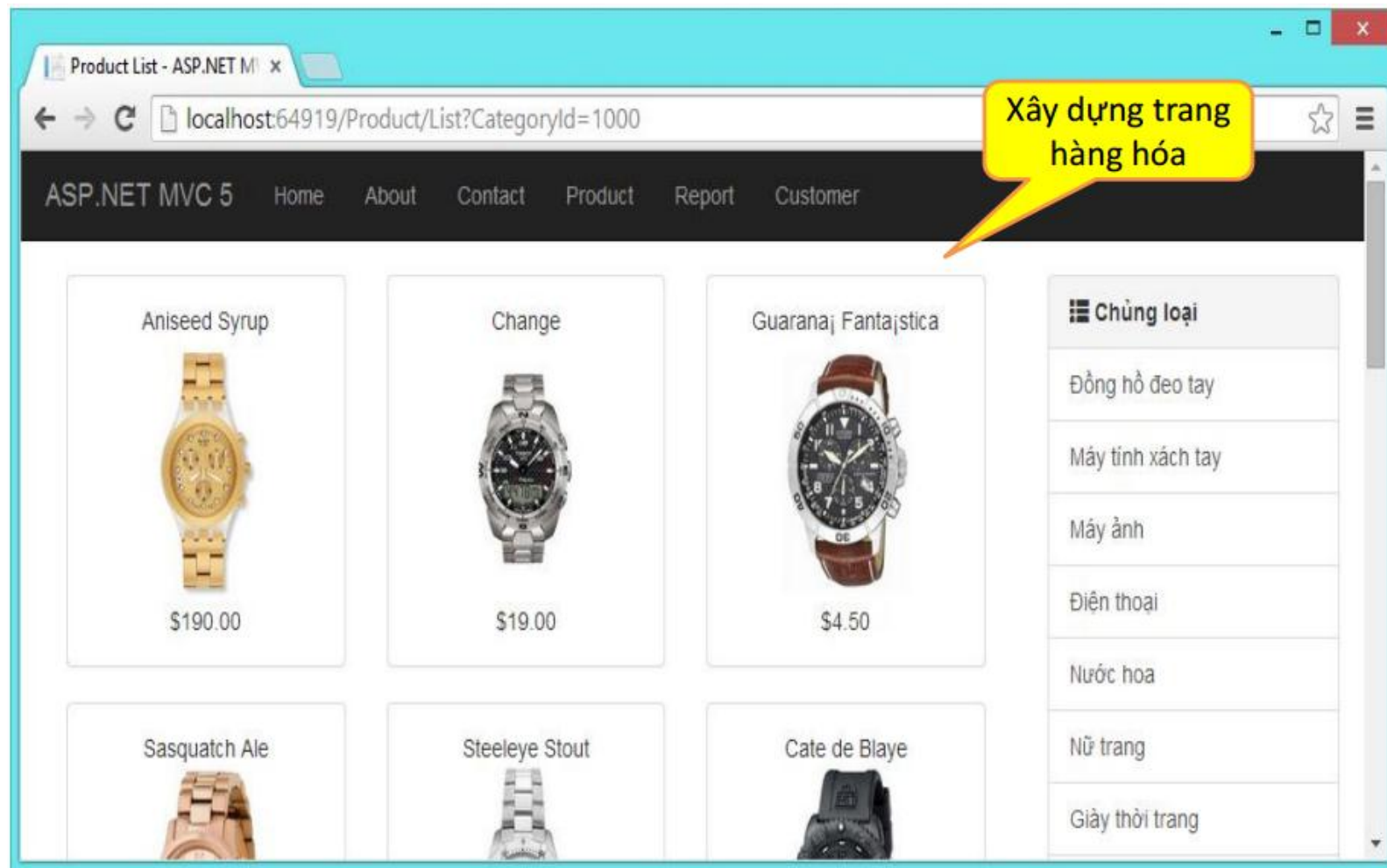
## Thực hành 1:





# 1.4. ENTITY FRAMEWORK API

## Thực hành 2:



# 1.4. ENTITY FRAMEWORK API

## Thực hành 3:

Detail - ASP.NET MVC 5 A x

localhost:64919/Product/Detail/1055

ASP.NET MVC 5 Home About Contact Product Report Customer

### Chi tiết hàng hóa

- Mã: 1055
- Tên: Pacta chinois
- Đơn giá: 24
- Ngày SX: 7/8/2007 12:00:00 AM
- Chủng loại: Nữ trang
- Trạng thái: Còn hàng

EmEditor uses JavaScript or VBScript for its macro language, so those who are familiar with HTML or Windows scripting will be able to create macros with little difficulty. For those unfamiliar with scripting languages, EmEditor can record keystrokes, which can then be saved in a macro file that can easily be loaded and executed in various situations. With the use of JavaScript or VBScript, you can also troubleshoot your code easily. For example, with JavaScript, you can use the following statement to troubleshoot errors

### Hàng cùng loại

- Pacta chinois
- Mishi Kobe Niku
- Alice Mutton
- Tharinger Rostbratwurst
- Perth Pasties
- Tourtiare
- Mishi Kobe Niku

**Xây dựng trang chi tiết hàng hóa**

**Hàng cùng loại với hàng hóa đang xem**

**Chủng loại**

- Đồng hồ đeo tay
- Máy tính xách tay
- Máy ảnh
- Nữ trang
- Giày thời trang
- Túi xách du lịch
- Loại mới

# 1.4. ENTITY FRAMEWORK API

## Thực hành 4:

**Customer Management - x**

localhost:64919/Customer/Edit/ALFKI

ASP.NET MVC 5 Home About Contact Product Report Customer

### Edit Customer

Mã đăng nhập:

Mật khẩu:

Họ và tên:

Địa chỉ email:

Hình ảnh:

Trạng thái kích hoạt: ☒

### Customer List

Id	Password	Fullname	Email	Photo	Activated?
ALFKI	iloveyou	Maria Anders	ALFKI@yahoo.com	ALFKI.jpg	Yes <a href="#">Edit</a>
ANATR	iloveyou	Ana Trujillo	ANATR@gmail.com	ANATR.jpg	No <a href="#">Edit</a>

**Chủng loại**

- Đồng hồ đeo tay
- Máy tính xách tay
- Máy ảnh
- Điện thoại
- Nước hoa
- Nữ trang
- Giày thời trang
- Túi xách du lịch
- Loại mới
- Thời trang
- Thời trang

Xây dựng trang quản lý khách hàng

Cho phép upload hình lên server

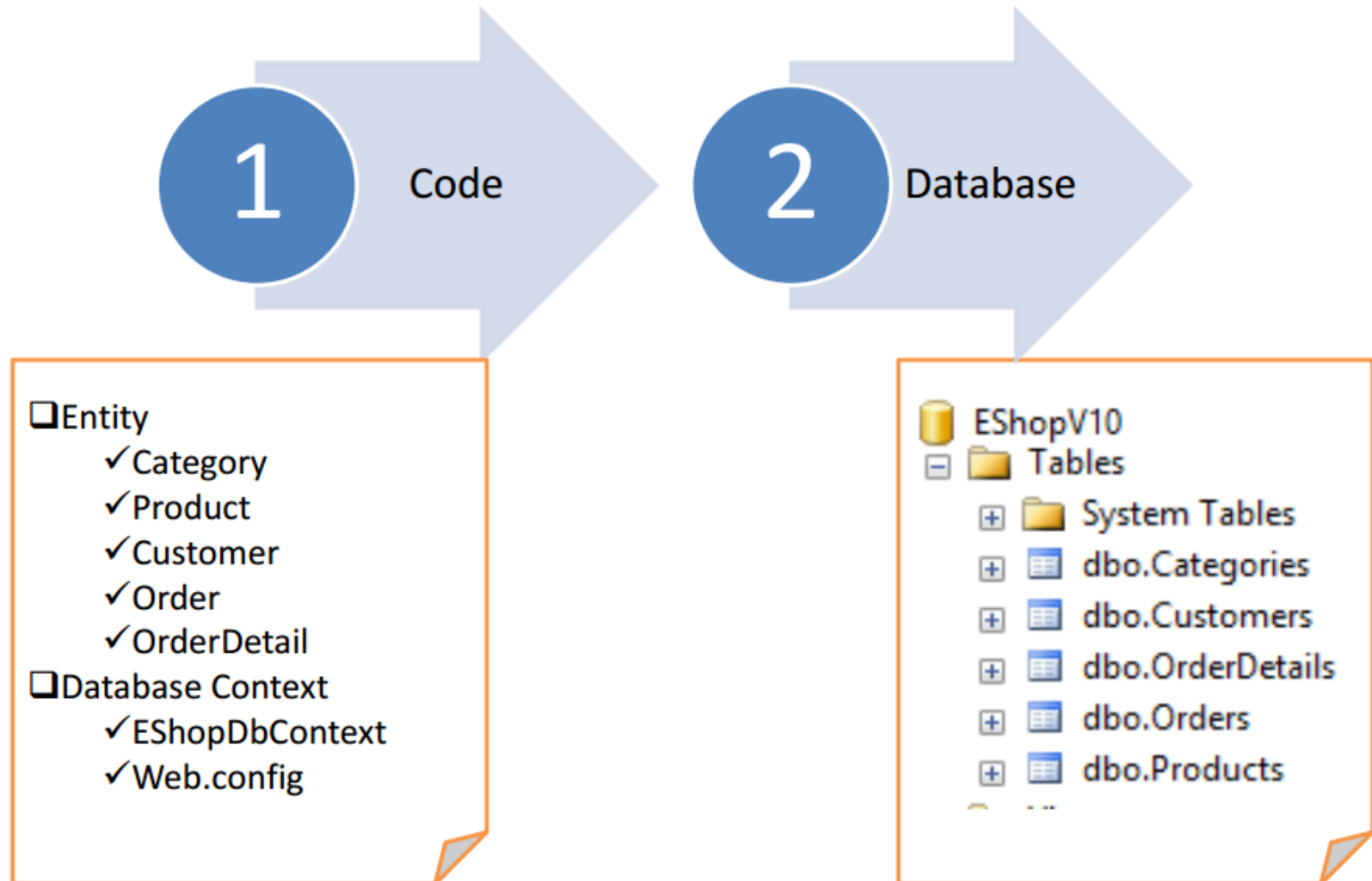
## 1.5. CODE FIRST

---



# 1.5. CODE FIRST

## Code First



# 1.5. CODE FIRST

## Entity Class

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public double UnitPrice { get; set; }
    public string Image { get; set; }
    public DateTime ProductDate { get; set; }
    public bool Available { get; set; }
    public int CategoryId { get; set; }
    public string Description { get; set; }

    public virtual Category Category { get; set; }
    public virtual List<Order> Orders { get; set; }
}
```

```
public class Category
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string NameVN { get; set; }

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

```
public class Customer
{
    public string Id { get; set; }
    public string Password { get; set; }
    public string Fullname { get; set; }
    public string Email { get; set; }
    public string Photo { get; set; }
    public bool Activated { get; set; }

    public virtual List<Order> Orders { get; set; }
}
```

```
public class OrderDetail
{
    public int Id { get; set; }
    public int OrderId { get; set; }
    public int ProductId { get; set; }
    public double UnitPrice { get; set; }
    public int Quantity { get; set; }
}
```

```
public class Order
{
    public int Id { get; set; }
    public string CustomerId { get; set; }
    public DateTime OrderDate { get; set; }
    public string Address { get; set; }
    public double Amount { get; set; }
    public string Description { get; set; }

    public virtual Customer Customer { get; set; }
    public virtual List<OrderDetail> OrderDetails { get; set; }
}
```

Thực thể kết hợp: Nhiều-1 và 1-Nhiều

# 1.5. CODE FIRST

## Database Context

```
public class EShopV10 : DbContext
{
    public EShopV10() : base("name=EShopV10") { }

    public DbSet<Category> Categories { get; set; }
    public DbSet<Customer> Customers { get; set; }
    public DbSet<OrderDetail> OrderDetails { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<Product> Products { get; set; }
}
```

```
<connectionStrings>
    <add name="EShopV10"
        connectionString="Server=.;Database=EShopV10;Integrated Security=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

## 1.5. CODE FIRST

### Qui ước ánh xạ thực thể

```
[Table("KhoaHoc")]
public class Course
{
    [Key]
    [Column("MaKH")]
    public int Id { get; set; }
    [Column("TenKH")]
    public String Name { get; set; }
    [Column("HocPhi")]
    public double UnitPrice { get; set; }
}
```

- ❑ 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 khóa phải là **Id** hoặc **EntityId**.

✎ Tùy biến với `[Key]`

- ❑ **Khóa int** được hiểu là **tự tăng**.

✎ Tùy biến với `[DatabaseGenerated(DatabaseGeneratedOption.Identity)]`



## 1.5. CODE FIRST

### Khởi tạo dữ liệu

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        Database.SetInitializer(new MusicStoreDbInitializer());
    }
}
```

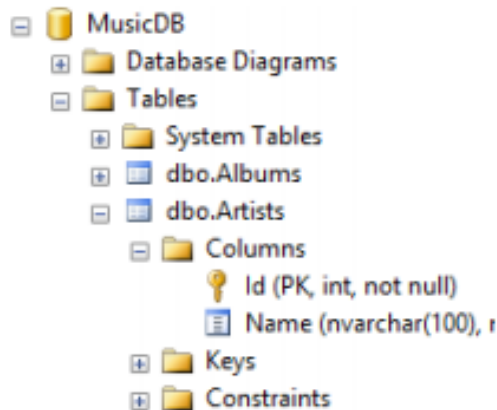


```
public class MusicStoreDbInitializer
    : DropCreateDatabaseIfModelChanges<MusicStoreDbContext>
{
    protected override void Seed(MusicStoreDbContext context)
    {
        base.Seed(context);
        var genres = new List<Genre>
        {
            new Genre { Name = "Rock" },
        };
        var artists = new List<Artist>
        {
            new Artist { Name = "Aaron Copland & London Symphony Orchestra" },
        };
        var albums = new List<Album>
        {
            new Album { Title = "A Copland Celebration, Vol. I", Genre :
            };
        };
        albums.ForEach(a => context.Albums.Add(a));
    }
}
```

# 1.5. CODE FIRST

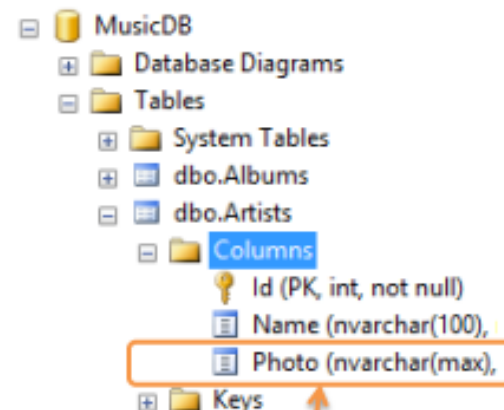
## Thay đổi Entity Class

- Chỉ cần 1 sự thay đổi các thuộc tính của EntityClass thì CSDL sẽ bị xóa đi và tạo lại



```
public class Artist
{
    public int Id { get; set; }
    [Required]
    [StringLength(100)]
    public String Name { get; set; }

    public List<Album> Albums { get; set; }
}
```



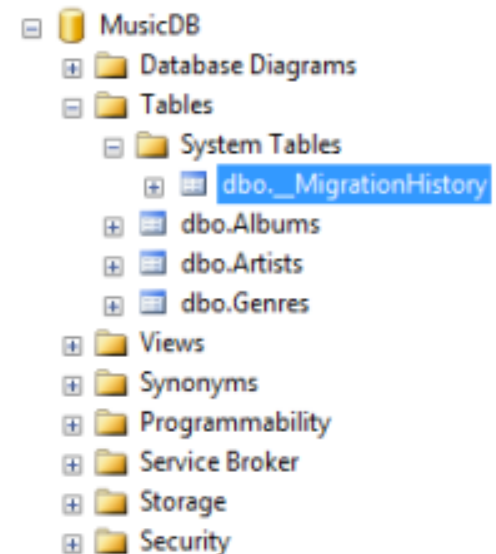
```
public class Artist
{
    public int Id { get; set; }
    [Required]
    [StringLength(100)]
    public String Name { get; set; }
    public String Photo { get; set; }

    public List<Album> Albums { get; set; }
}
```

## 1.5. CODE FIRST

### Bảng - MigrationHistory

- ❑ Trong CSDL được sinh ra bởi mô hình lập trình Code-First của EF có chứa 1 bảng có tên là **MigrationHistory** được sử dụng để theo dõi phiên bản ánh xạ.



- ❑ Xóa bảng này sẽ làm mất liên lạc về sự thay đổi của EntityClass khi đó sẽ trở lại làm việc như CSDL đã tồn tại.

## 1.5. CODE FIRST

### Làm việc với CSDL đã tồn tại

- ❑ Nếu CSDL của bạn đã có, thì bạn vẫn thực hiện các bước như cũ

- ~~✗~~ Định nghĩa Entity Class
- ~~✗~~ Định nghĩa DbContext Class
- ~~✗~~ Khai báo Connection String



- ❑ Ngoại trừ khai báo khởi đầu giá trị cho các bảng trong CSDL tại Global.asax

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        Database.SetInitializer(new MusicStoreDbInitializer());
    }
}
```

Bỏ dòng mã lệnh này

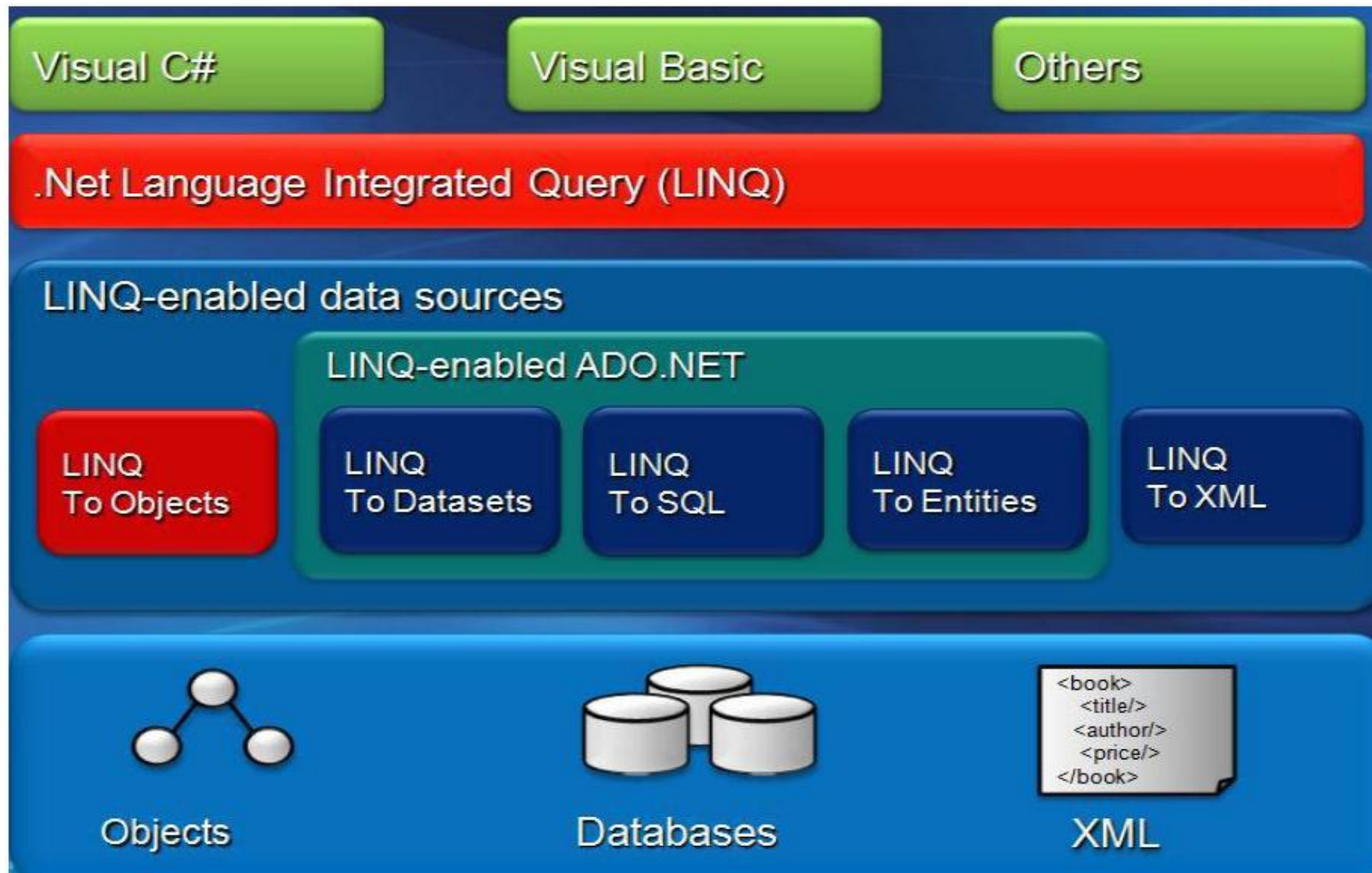
## 2. LinQ To SQL

---

2.1. Kiến trúc tổ chức của LINQ.

2.2. Truy vấn dữ liệu

## 2.1. KIẾN TRÚC TỔ CHỨC CỦA LINQ





## 2.2. TRUY VẤN DỮ LIỆU

```
var result = from s in students  
            where s.Marks > 9  
            orderby s.Marks descending  
            select new { s.Name, s.Marks };
```

Biểu thức  
truy vấn

Kiểu nội bộ  
tự suy

Kiểu nhắc danh

Khởi tạo đối tượng

```
var result2 = students  
    .Where(s => s.Marks > 9)  
    .OrderByDescending(s => s.Marks)  
    .Select(s => new { s.Name, s.Marks });
```

Biểu thức  
lambda

Phương thức  
mở rộng

## 2.2. TRUY VẤN DỮ LIỆU

### VD1: Truy Vấn các số chẵn:

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };
```

```
var evens = from n in numbers  
            where n % 2 == 0  
            select n;
```

```
foreach(int n in numbers){  
    if(n % 2 == 0){  
        tích lũy số chẵn  
    }  
}
```

- ❑ **from**: chỉ ra phần tử được lấy từ tập hợp cần truy vấn
- ❑ **where**: chỉ ra điều kiện lọc
- ❑ **select**: chỉ ra đối tượng nhận được



## 2.2. TRUY VẤN DỮ LIỆU

### VD2: Truy Vấn các số chẵn:

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };

var evens = from n in numbers
            where n % 2 == 0
            let rate = n / numbers.Sum()
            orderby n descending
            select new { number = n, rate = rate };
```

Đối tượng

```
foreach (var e in evens)
{
    int n = e.N
}
```

N	
Equals	
GetHashCode	
GetType	
number	int 'a.number
rate	
ToString	

Anonymous Types:  
'a is new { int number, int rate }

## 2.2. TRUY VẤN DỮ LIỆU

### Tổng hợp - Thống kê

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };  
  
var evens = from n in numbers  
            group n by n % 3 into g  
            select new  
            {  
                Nhom = g.Key,  
                Tong = g.Sum(),  
                SoLuong = g.Count(),  
                SoNN = g.Min(),  
                SoLN = g.Max(),  
                SoTB = g.Average()  
            };
```

- ❑ Nhóm chia 3 dư 0: gồm 6, 45, 87
- ❑ Nhóm chia 3 dư 1: gồm 19, 13
- ❑ Nhóm chia 3 dư 2: gồm 23, 56, 5, 8

## 2.2. TRUY VẤN DỮ LIỆU

### Sử dụng phương thức mở rộng

```
var evens = numbers
    .Where(n => n % 2 == 0)
    .Select(n => n);
```

```
var evens = numbers
    .Where(n => n % 2 == 0)
    .OrderByDescending(n => n)
    .Select(n => new
    {
        number = n,
        rate = n / numbers.Sum()
    });
```

```
var evens = numbers.GroupBy(n => n % 3)
    .Select(g => new
    {
        Nhom = g.Key,
        Tong=g.Sum(),
        SoLuong=g.Count(),
        SoNN=g.Min(),
        SoLN=g.Max(),
        SoTB=g.Average()
    });
```

## 2.2. TRUY VẤN DỮ LIỆU

### Sử dụng phương thức mở rộng

```
var evens = from n in numbers
            where n % 2 == 0
            select n;
```



```
var evens = numbers
            .Where(n => n % 2 == 0)
            .Select(n => n);
```

```
var evens = from n in numbers
            where n % 2 == 0
            orderby n descending
            select new
            {
                number = n,
                rate = n / numbers.Sum()
            };
```



```
var evens = numbers
            .Where(n => n % 2 == 0)
            .OrderByDescending(n => n)
            .Select(n => new
            {
                number = n,
                rate = n / numbers.Sum()
            }));
```

```
var evens = from n in numbers
            group n by n % 3 into g
            select new
            {
                Nhom = g.Key,
                Tong = g.Sum(),
                SoLuong = g.Count(),
                SoNN = g.Min(),
                SoLN = g.Max(),
                SoTB = g.Average()
            };
```



```
var evens = numbers.GroupBy(n => n % 3)
            .Select(g => new
            {
                Nhom = g.Key,
                Tong=g.Sum(),
                SoLuong=g.Count(),
                SoNN=g.Min(),
                SoLN=g.Max(),
                SoTB=g.Average()
            }));
```

## 2.2. TRUY VẤN DỮ LIỆU

### Truy vấn cơ bản

Phương thức	Mô tả	Ví dụ
.Where(e=>điều kiện)	Lọc	Students.Where(s=>s.Marks > 9)
.GroupBy(e=>biểu thức)	Nhóm	Students.GroupBy(s=>s.Class)
.OrderBy(e=>biểu thức) .OrderByDescending(e=>biểu thức)	Sắp xếp	Students.OrderBy(s=>s.Name)
.Select(e=>đối tượng)	Chọn	Students.Select(s=>new{s.Name, s.Marks})
.Distinct()	Giữ 1 của các đối tượng giống nhau	Numbers.Distinct()

```
var studs = Students
    .Where(s=>s.Marks > 9)
    .OrderBy(s=>s.Marks)
    .Select(s=>s);
```

## 2.2. TRUY VẤN DỮ LIỆU

### Truy vấn phân trang

Phương thức	Mô tả	Ví dụ
.Take(số lượng)	Lấy các phần tử đầu	Students.Take(5)
.Skip(số lượng)	Bỏ qua các phần tử đầu	Students.Skip(3).Take(6)
.TakeWhile(e=>đ.kiện)	Lấy các phần tử đầu thỏa điều kiện	Students.TakeWhile(s=>s.Marks < 4)
.SkipWhile(e=>đ.kiện)	Bỏ qua các phần tử đầu thỏa điều kiện	Students.SkipWhile(s=>s.Marks < 0)

```
var result = db.Products
    .Skip(10).Take(20)
```



## 2.2. TRUY VẤN DỮ LIỆU

### Truy vấn một thực thể

```
var result = db.Customers
```

```
.Single(c=>c.Id=="A" && c.Password=="B")
```

Phương thức	Mô tả	Ví dụ
.Single(e=>đ.kiện)	Lấy 1 phần tử thỏa điều kiện. Ngoại lệ nếu không tìm thấy hoặc nhiều hơn một.	Students.Single(s=>s.Id=="Hoa")
.First()	Lấy phần tử đầu	Students.First()
.Last()	Lấy phần tử cuối	Students.Last()

## 2.2. TRUY VẤN DỮ LIỆU

### Tổng hợp số liệu

Phương thức	Mô tả	Ví dụ
.Sum(e=>biểu thức số học)	Tính tổng	Students.Sum(s=>s.Marks)
.Count(e=>biểu thức số học)	Đếm số lượng	Students.Count(s=>s.Id)
.Min(e=>biểu thức số học)	Giá trị nhỏ nhất	Students.Min(s=>s.Marks)
.Max(e=>biểu thức số học)	Giá trị lớn nhất	Students.Max(s=>s.Marks)
.Average(e=>biểu thức số học)	Giá trị trung bình	Students.Average(s=>s.Marks)

❑ Var result = db.Products

✗.GroupBy(p=>p.Category)

✗.Select(g=>new{g.Key.Name, g.Count})



## 2.2. TRUY VẤN DỮ LIỆU

### VD: Thống kê doanh số

```
var items7 = db.Products.GroupBy(p => p.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại
        Sum = g.Sum(p=>p.UnitPrice), //--tổng đơn giá hàng hóa của loại
        Count = g.Count(), //--số hàng hóa của loại
        Min = g.Min(p => p.UnitPrice), //--giá hàng hóa thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá hàng hóa cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items8 = db.OrderDetails.GroupBy(d=>d.Product)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên hàng hóa
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị đã bán
        Count = g.Sum(p => p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

## 2.2. TRUY VẤN DỮ LIỆU

### VD: Thống kê doanh số

```
var items9 = db.OrderDetails.GroupBy(d => d.Product.Category)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Name, //--tên loại hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

```
var items10 = db.OrderDetails.GroupBy(d => d.Order.Customer)
    .Select(g => new ReportInfo
    {
        Group = g.Key.Fullname, //--họ và tên khách hàng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã mua
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã mua
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

## 2.2. TRUY VẤN DỮ LIỆU

### VD: Thống kê doanh số

```
var items11 = db.OrderDetails.GroupBy(d => d.Order.OrderDate.Month)
    .Select(g => new ReportInfo
    {
        Group = g.Key, //--tháng
        Sum = g.Sum(p => p.UnitPrice * p.Quantity), //--tổng giá trị hàng hóa đã bán
        Count = g.Sum(p=>p.Quantity), //--tổng số lượng đã bán
        Min = g.Min(p => p.UnitPrice), //--giá thấp nhất
        Max = g.Max(p => p.UnitPrice), //--giá cao nhất
        Avg = g.Average(p => p.UnitPrice) //--giá trung bình
    });
```

## 2.2. TRUY VẤN DỮ LIỆU

### Truy vấn có kiểm tra

Phương thức	Mô tả	Ví dụ
.Contains(phần tử)	Tập có chứa phần tử?	Students.Contains(hoa)
.Any(e=>đ.kiện)	Ít nhất một phần tử trong tập thỏa điều kiện	Students.Any(s=>s.Marks < 3)
.All(e=>đ.kiện)	Tất cả các phần tử trong tập thỏa điều kiện	Students.All(s=>s.Marks > 5)

```
int[] numbers = { 19, 23, 6, 56, 45, 87, 5, 8, 13 };

if (numbers.All(n => n % 2 == 0))
{
    //tất cả các số trong numbers đều là số chẵn
}
if (numbers.Any(n => n % 3 == 0))
{
    //ít nhất một số trong numbers chia hết cho 3
}
if (numbers.Contains(8))
{
    //tập numbers có chứa số 8
}
```

## 2.2. TRUY VẤN DỮ LIỆU

### VD Lọc dữ liệu

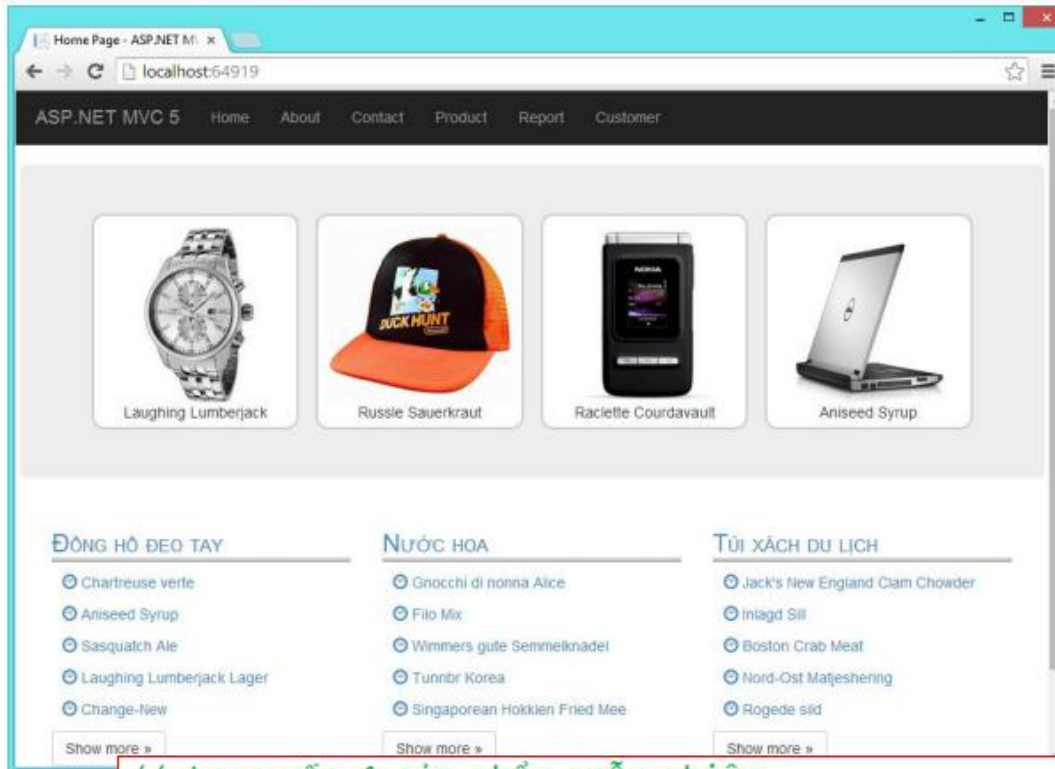
Id	Name	Price	Date	Category	Available
1023	Tunnbr Korea	\$9.00	31-08-2011	Nước hoa	Yes
1054	Tourtiare	\$7.45	07-10-2009	Nữ trang	Yes
1075	RhanbrAu Klosterbier	\$7.75	31-10-1982	Đồng hồ đeo tay	Yes

```
public ActionResult Search(String Name = "",
                           double Min = double.MinValue, double Max = double.MaxValue)
{
    var list = db.Products
        .Where(p => p.Name.Contains(Name) && p.UnitPrice >= Min
                && p.UnitPrice <= Max).ToList();

    return View(list);
}
```

## 2.2. TRUY VẤN DỮ LIỆU

### VD Thực thể kết hợp

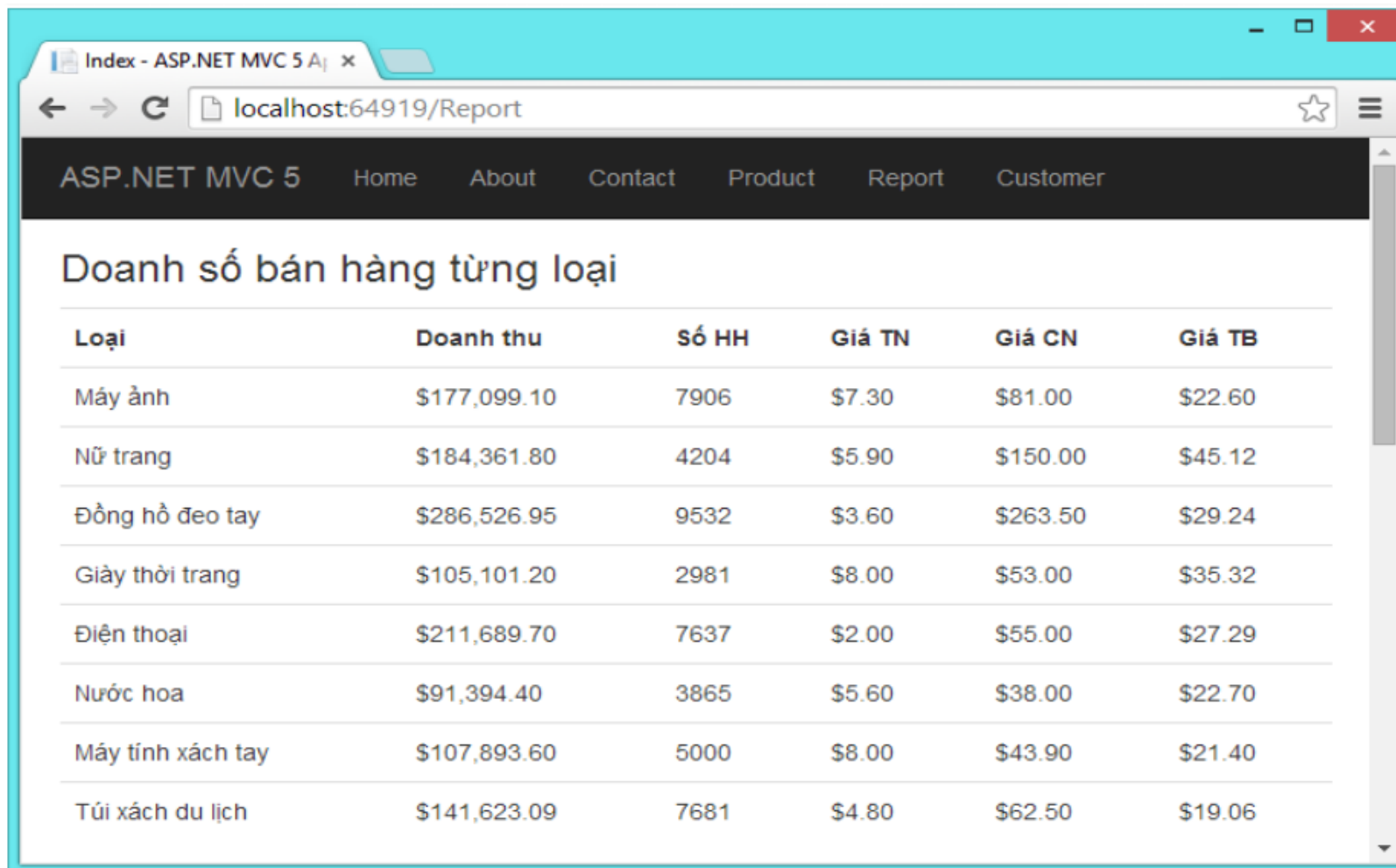


```
// truy vấn 4 sản phẩm ngẫu nhiên
ViewBag.Products = db.Products.OrderBy(p => Guid.NewGuid())
    .Take(4).ToList();

// truy vấn 3 loại có ít nhất 5 sản phẩm ngẫu nhiên
var model = db.Categories
    .Where(c=>c.Products.Count > 5)
    .OrderBy(c => Guid.NewGuid()).ToList().Take(3).ToList();
```

## 2.2. TRUY VẤN DỮ LIỆU

### VD Doanh số bán hàng



Loại	Doanh thu	Số HH	Giá TN	Giá CN	Giá TB
Máy ảnh	\$177,099.10	7906	\$7.30	\$81.00	\$22.60
Nữ trang	\$184,361.80	4204	\$5.90	\$150.00	\$45.12
Đồng hồ đeo tay	\$286,526.95	9532	\$3.60	\$263.50	\$29.24
Giày thời trang	\$105,101.20	2981	\$8.00	\$53.00	\$35.32
Điện thoại	\$211,689.70	7637	\$2.00	\$55.00	\$27.29
Nước hoa	\$91,394.40	3865	\$5.60	\$38.00	\$22.70
Máy tính xách tay	\$107,893.60	5000	\$8.00	\$43.90	\$21.40
Túi xách du lịch	\$141,623.09	7681	\$4.80	\$62.50	\$19.06



**HẾT**