

LẬP TRÌNH WINDOWS

GV: Nguyễn Thanh Tùng
Email: tung.nguyenthanh@stu.edu.vn



NỘI DUNG CHÍNH

2



Giới thiệu ASP.NET Core MVC



Entity Framework Core



HTML và BootStrap



Tag Helper và Validation



JavaScript và JavaScript Ajax



I. Giới thiệu ASP.NET Core MVC

Giới thiệu ASP.NET Core

Giới thiệu mô hình MVC

Ví dụ đầu tiên ASP.NET Core MVC

Sử dụng Controller và Action

Routing

Razor và Sử dụng View

Đối tượng ViewBag, Request và Model

1. Giới thiệu ASP.NET Core

- ❑ ASP.NET Core là Framework mã nguồn mở (open-source), hiệu suất cao (high-performance), đa nền tảng (cross-platform) để xây dựng các ứng dụng, hỗ trợ đám mây (cloud-enabled), kết nối internet (internet-connected).

1. Giới thiệu ASP.NET Core

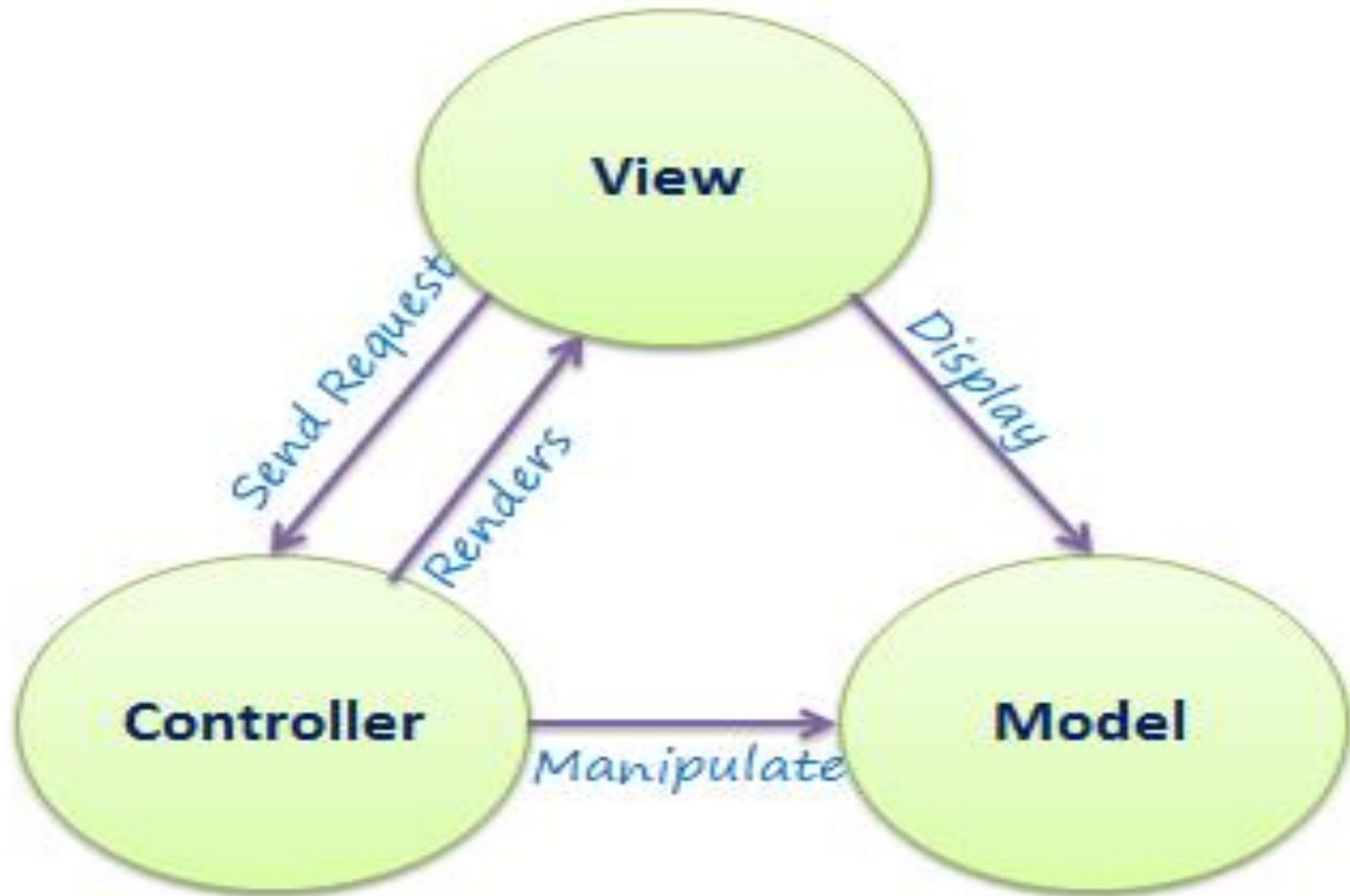
□ ASP.NET Core có thể được sử dụng để:

- Xây dựng các ứng dụng và dịch vụ web, ứng dụng IoT (Internet of Things) và các phần mềm phụ trợ dành cho thiết bị di động (mobile backends).
- Sử dụng các công cụ phát triển trên Windows, macOS và Linux.
- Triển khai lên đám mây (cloud) hoặc tại chỗ (on-premises).

1. Giới thiệu ASP.NET Core

- ❑ Trong phạm vi của môn học này, chúng ta sử dụng ASP.NET Core Web MVC để xây dựng ứng dụng Web.

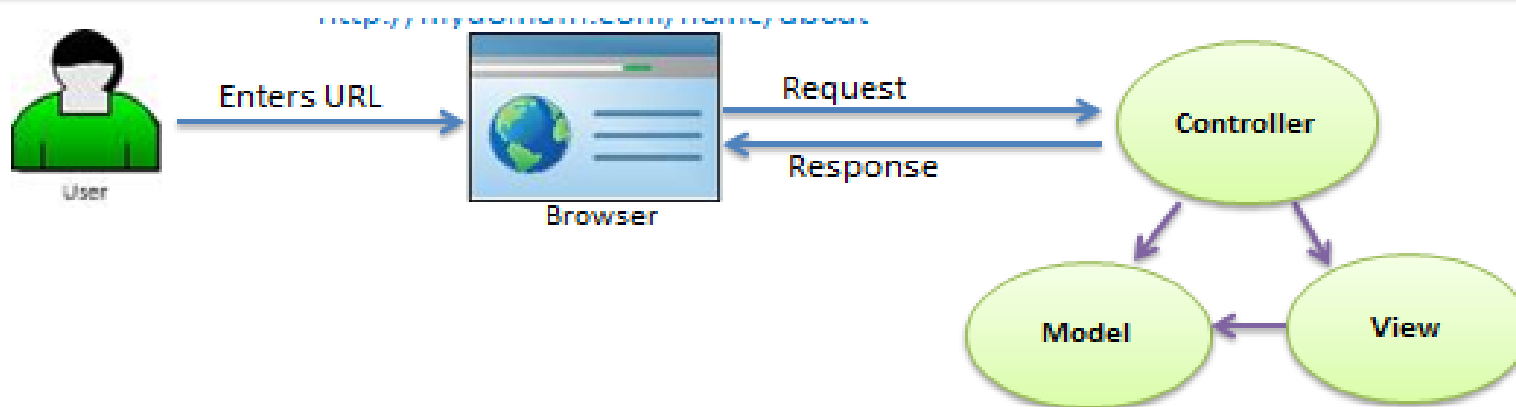
2. Giới thiệu mô hình MVC



2. Giới thiệu mô hình MVC

- ❑ *Models* : Các đối tượng Models là một phần của ứng dụng, các đối tượng này thiết lập các thành phần dữ liệu luận lý của ứng dụng. Thông qua tầng này, chúng ta có thể truy xuất dữ liệu trong CSDL.
- ❑ *Views* : là các thành phần dùng để hiển thị giao diện người dùng (UI). Views còn là nơi người dùng nhập dữ liệu cho hệ thống.
- ❑ *Controllers* : là các thành phần dùng để quản lý tương tác người dùng, làm việc với model và chọn view để hiển thị giao diện người dùng.

2. Giới thiệu mô hình MVC



Request/Response in MVC Architecture

- ❑ Khi người dùng nhập URL trong trình duyệt, nó sẽ gửi yêu cầu đến máy chủ và gọi Controller thích hợp. Sau đó, Controller sử dụng View và Model phù hợp và tạo phản hồi (Response) và gửi lại cho người dùng.

3. Ví dụ đầu tiên ASP.NET Core MVC

- ❑ Action
- ❑ Interface IActionResult (Content, File)

4. Sử dụng Controller và Action

- ❑ Controller trong kiến trúc MVC xử lý mọi yêu cầu gửi đến từ URL, lấy dữ liệu cần thiết từ Model và trả về các phản hồi thích hợp.
- ❑ Controller là một lớp, được dẫn xuất từ lớp cơ sở `Microsoft.AspNetCore.Mvc.Controller`.
- ❑ Lớp Controller chứa các phương thức public gọi là các phương thức Action.

4. Sử dụng Controller và Action

- ❑ Trong ASP.NET Core MVC, mọi tên lớp của Controller phải kết thúc bằng một từ "Controller".
- ❑ Ví dụ: Controller cho trang chủ phải là HomeController và Controller cho Student phải là StudentController.
- ❑ Phương thức Index là một phương thức Action mặc định cho bất kỳ Controller nào.

4. Sử dụng Controller và Action

- ❑ Interface IActionResult: là kiểu trả về của phương thức Action, những lớp đối tượng thường được dùng dẫn xuất từ interface này được liệt kê ở bảng sau:

4. Sử dụng Controller và Action

Result Class	Description	Base Controller Method
ViewResult	Represents HTML and markup.	View()
EmptyResult	Represents No response.	
ContentResult	Represents string literal.	Content()
FileContentResult, FilePathResult, FileStreamResult	Represents the content of a file	File()
JavaScriptResult	Represent a JavaScript script.	JavaScript()
JsonResult	Represent JSON that can be used in AJAX	Json()
RedirectResult	Represents a redirection to a new URL	Redirect()
RedirectToRouteResult	Represent another action of same or other controller	RedirectToRoute()
PartialViewResult	Returns HTML	PartialView()
HttpUnauthorizedResult	Returns HTTP 403 status	

4. Sử dụng Controller và Action

- ❑ **Đối tượng Request** được dùng để nhận những thông tin từ trình duyệt của người dùng gửi về cho máy chủ. Những thông tin này gồm các thông số của Form khi được Submit dùng phương thức POST hoặc GET hay các tham số được ghi cùng với trang ASP.NET trong lời gọi đến trang đó.
- ❑ Đối tượng Request có thể chia sẻ thông tin qua lại giữa các trang ASP.NET trong cùng một ứng dụng và để lấy giá trị các Cookie lưu trữ trên máy Client.

4. Sử dụng Controller và Action

- ❑ Kiểu dữ liệu của đối tượng Request bao gồm các cặp (name, value) cho dữ liệu biểu mẫu hoặc giá trị chuỗi truy vấn hoặc giá trị cookie.
- ❑ Tham số của phương thức Action: Theo mặc định, các giá trị cho tham số phương thức Action được lấy từ đối tượng dữ liệu Request.

4. Sử dụng Controller và Action

- ❑ Mô hình trong ASP.NET liên kết động ánh xạ chuỗi truy vấn URL hoặc thu thập dữ liệu biểu mẫu thành các tham số phương thức Action nếu cả hai tên khớp nhau.
- ❑ Tham số của phương thức action có thể có kiểu Null.
- ❑ Kết luận:

Controller nhận dữ liệu từ View bằng: Đối tượng Request, Đối tượng Model, Các tham số của Action.

5. Routing

- ❑ URL (*Uniform Resource Locator*) được dùng để tham chiếu tới tài nguyên trên Internet. URL mang lại khả năng siêu liên kết cho các trang mạng. Các tài nguyên khác nhau được tham chiếu tới bằng địa chỉ (chính là URL) còn được gọi là *địa chỉ mạng* hay là *liên kết mạng*.
- ❑ Lớp UrlHelper của Asp.net chứa các phương thức để xây dựng URL cho một ứng dụng.

5. Routing

- ❑ Routing cho phép xác định mẫu URL ánh xạ tới trình xử lý yêu cầu. Trình xử lý yêu cầu này có thể là một file hoặc lớp. Trong MVC, đó là lớp Controller và phương thức Action.
- ❑ Tất cả các cấu hình của các Routing được lưu trữ trong RouteTable và sẽ được sử dụng bởi Routing engine để xác định lớp hoặc file xử lý thích hợp cho yêu cầu gửi đến.

5. Routing

- ❑ Mỗi ứng dụng MVC phải cấu hình (đăng ký) ít nhất một Routing, và nó được cấu hình bởi Framework MVC theo mặc định. Hệ thống đã tạo một Routing mặc định trong lớp StartUp, nằm trong file Startup.cs của ứng dụng.

5. Routing

- ❑ Hình dưới đây minh họa cách định cấu hình cho Routing trong lớp StartUp.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

5. Routing

- ❑ Mẫu URL chỉ được xem xét sau phần tên miền trong URL.
- ❑ Ví dụ: mẫu URL "{controller} / {action} / {id}" sẽ trông giống như tên miền localhost: 1234 / {controller} / {action} / {id}.

Controller *Action method*

Id parameter value

http://localhost:1234/home/index/100

Controller *Action method*

http://localhost:1234/home/index

Routing in MVC

6. Razor và Sử dụng View

❑ Razor: là một kỹ thuật trong ASP.NET MVC cho phép người dùng viết code bằng ngôn ngữ lập trình C# trong lớp View. Razor sử dụng ký tự @ để xác định code C#.

❑ Ví dụ:

```
<h2>@DateTime.Now.ToShortDateString()</h2>
```

Khởi lệnh: @{

```
    var date = DateTime.Now.ToShortDateString();  
    var message = "Hello World";  
}
```

```
<h2>Today's date is: @date </h2>
```

```
<h3>@message</h3>
```

6. Razor và Sử dụng View

- ❑ View là các trang HTML
- ❑ Cấu trúc trang HTML
- ❑ Các thẻ cơ bản: Form, Table, Div, Input(Submit), Input(Text), Input(CheckBox), Input(Radio), Select.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

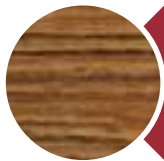
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

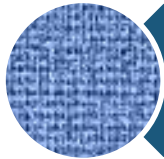
7. Đối tượng ViewBag, Request và Model

- ❑ Truyền dữ liệu từ View sang Controller:
 - Đối tượng Request,
 - Các tham số của Action,
 - Đối tượng Model,
 - Lớp FormCollection.
- ❑ Truyền dữ liệu từ Controller sang View: Đối tượng Model và đối tượng ViewBag.
- ❑ Kiểu trả về của phương thức Action là Interface ActionResult (Content, File, View, PartialView, Json, RedirectToAction, Redirect, JavaScript)

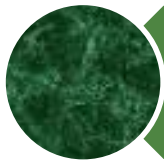
II. Entity Framework Core



Linq to Object



Entity Framework core



Linq to Entities

1. Linq to Object

- ❑ LinQ (Language Integrated Query) là một thư viện cung cấp cho ngôn ngữ lập trình C# và Visual Basic. Thư viện này cung cấp khả năng truy vấn trên tập hợp đối tượng ngay trong ngôn ngữ lập trình.

1. Linq to Object

❑ Khai báo sử dụng thư viện Linq:

`using System.Linq;`

❑ Truy vấn Linq bao gồm ba phần: Chứa dữ liệu nguồn, Tạo câu lệnh truy vấn, và Thực thi câu lệnh truy vấn.

1. Linq to Object

- Chứa dữ liệu nguồn: là dữ liệu được dẫn xuất từ interface **IEnumerable<T>**
- Tạo câu lệnh truy vấn: Cú pháp tạo câu lệnh truy vấn LINQ

from ...where ...select ...

- Thực thi câu lệnh truy vấn: sử dụng từ khóa **var** để khai báo biến chứa kết quả trả về của câu lệnh truy vấn.

1. Linq to Object

□ Ví dụ:

```
//chứa dữ liệu nguồn  
int[] ds = new int[]{26,31,17,22,45,68,79};  
//tạo câu lệnh truy vấn  
var kq = from a in ds  
         where a % 2 == 0  
         select a;  
//thực thi câu lệnh truy vấn  
foreach (int a in kq) {  
    Console.WriteLine("{0} ", a);  
}
```

1. Linq to Object

□ Sử dụng phương thức Linq và biểu thức

Lambda:

- Phương thức Linq: là các phương thức của Interface **IEnumerable<T>**.
- Phép toán Lambda: **=>**
- Biểu thức Lambda:

(input parameters) => expression

1. Linq to Object

□ Ví dụ:

```
//chứa dữ liệu nguồn  
int[] ds = new int[]{26,31,17,22,45,68,79};  
//tạo câu lệnh truy vấn  
IEnumerable<int> kq =  
    ds.Where<int>(x => x % 2 == 0);  
//thực thi câu lệnh truy vấn  
foreach (int a in kq){  
    Console.WriteLine("{0} ", a);  
}
```

1. Linq to Object

□ Sắp xếp dữ liệu:

```
int[] ds = new int[] {26,31,17,22,45,68,79};
```

- Câu lệnh truy vấn Linq:

```
var kq = from a in ds
```

```
where a % 2 == 0
```

```
orderby a ascending
```

```
select a;
```

- Phương thức Linq:

```
IEnumerable<int> kq =
```

```
ds.OrderBy(x => x);
```

2. Entity Framework Core

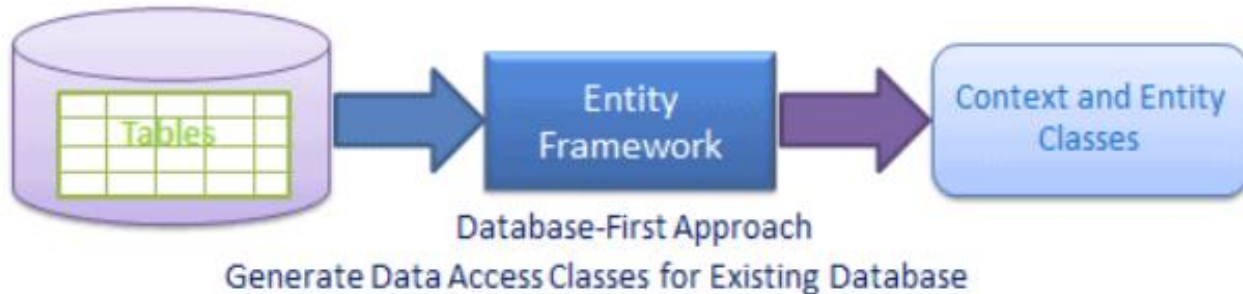
- ❑ Entity Framework Core là phiên bản mới sau Entity Framework 6.x. Nó là mã nguồn mở, nhẹ, có thể mở rộng và là phiên bản đa nền tảng (cross-platform) của công nghệ truy cập dữ liệu Entity Framework.
- ❑ Entity framework core phát sinh mã lệnh C# là các lớp đối tượng và các mối quan hệ giữa các lớp đối tượng tương ứng với các thực thể và các mối quan hệ giữa các thực thể trong CSDL.

2. Entity Framework Core

- ❑ Khi phát sinh mã lệnh, Entity framework core có phát sinh lớp DbContext và lớp này nhận nhiệm vụ mở kết nối CSDL, thực hiện truy vấn hay thay đổi dữ liệu trong CSDL.
- ❑ Khi chúng ta thao tác trên các lớp đối tượng tương ứng với việc thao tác trên CSDL.

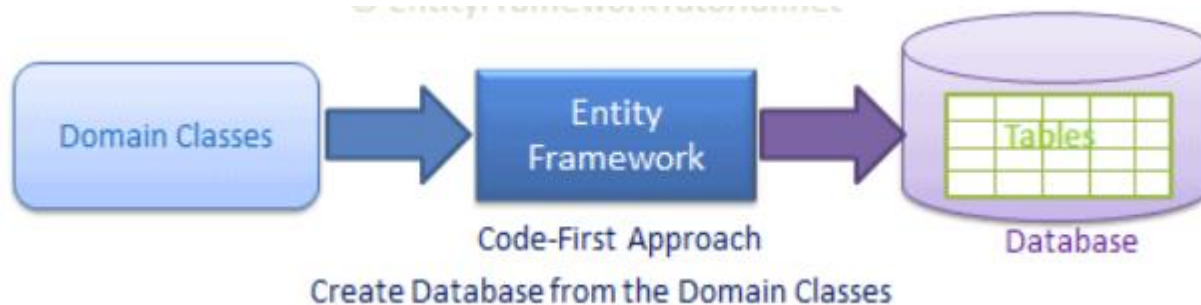
2. Entity Framework Core

- ❑ EF Core cung cấp hai phương pháp sử dụng:
 - Database-First: Sử dụng CSDL có sẵn để phát sinh ra các lớp đối tượng tương ứng.



2. Entity Framework Core

- Code-First: Có sẵn các lớp đối tượng tương ứng và EF Core phát sinh ra CSDL.



2. Entity Framework Core

- ❑ EF Core hỗ trợ cho nhiều CSDL khác nhau. Để sử dụng EF Core phải sử dụng gói nuget tương ứng với CSDL cần dùng.

Database	NuGet Package
SQL Server	Microsoft.EntityFrameworkCore.SqlServer
MySQL	MySql.Data.EntityFrameworkCore
PostgreSQL	Npgsql.EntityFrameworkCore.PostgreSQL
SQLite	Microsoft.EntityFrameworkCore.SQLite
SQL Compact	EntityFrameworkCore.SqlServerCompact40
In-memory	Microsoft.EntityFrameworkCore.InMemory

2. Entity Framework Core

❑ Cài đặt EF Core cho CSDL SqlServer:

- Gói **EF Core DB provider**:

Microsoft.EntityFrameworkCore.SqlServer

- Gói **EF Core tools**:

Microsoft.EntityFrameworkCore.Tools

2. Entity Framework Core

❑ Tạo Model từ CSDL có sẵn (*Database-First*):

Trong cửa sổ **Package Manager Console** sử dụng lệnh

```
Scaffold-DbContext [-Connection] [-Provider] [-OutputDir] [-Context] [-Schemas>] [-Tables>]  
                    [-DataAnnotations] [-Force] [-Project] [-StartupProject] [<CommonParameters>]
```

Ví dụ:

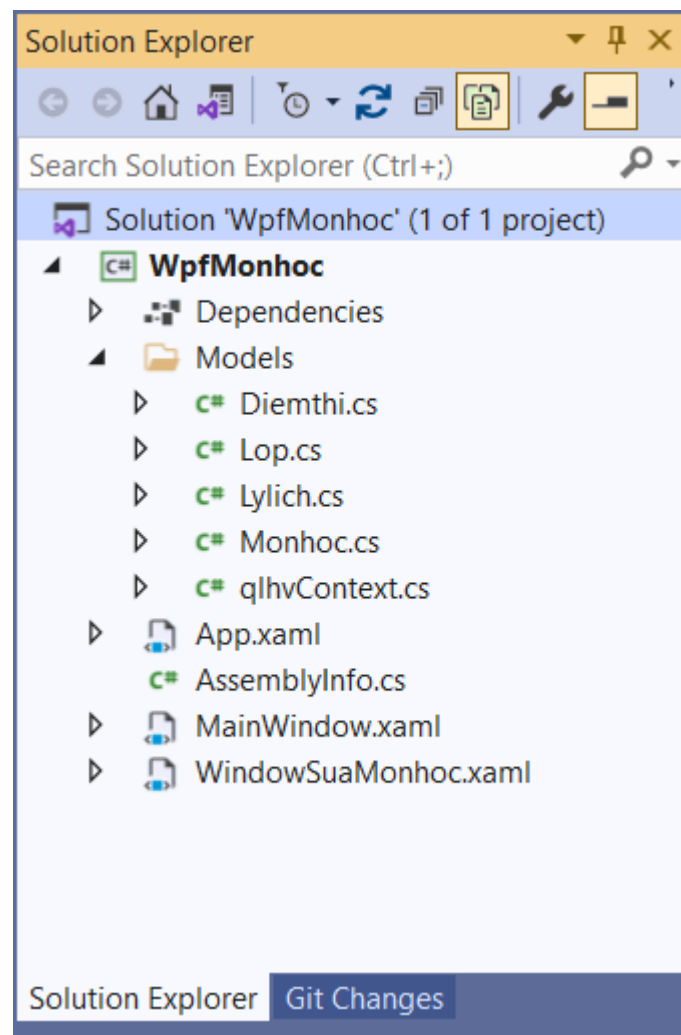
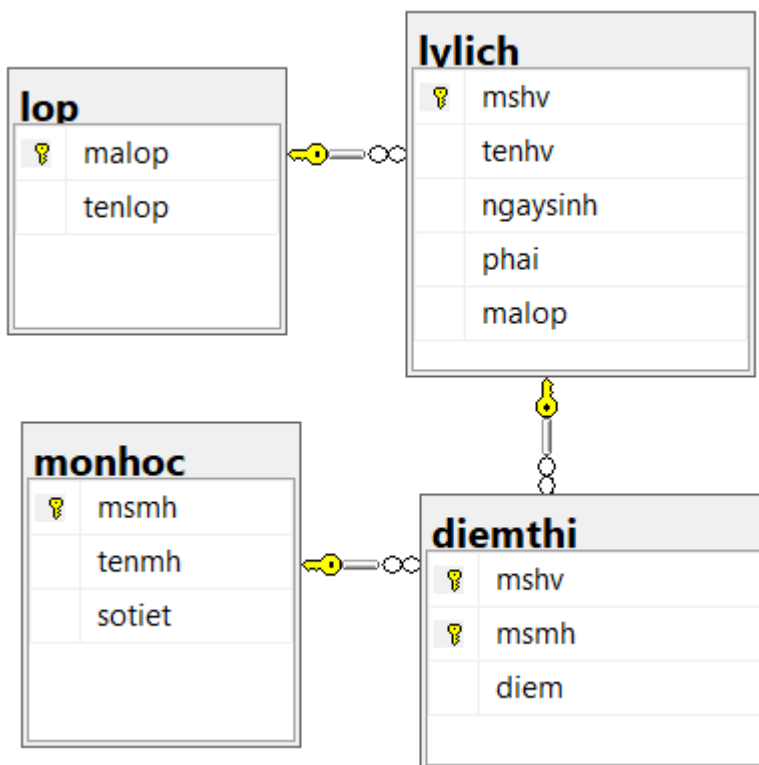
PM> Scaffold-DbContext

"Server=.\SQLEXPRESS;Database=qlhv;Trusted_Connection=True;"

Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

2. Entity Framework Core

- CSDL sử dụng là
SQLServer có tên qlhv.



2. Entity Framework Core

- Lớp **qlhvContext** là ánh xạ giữa CSDL với tập đối tượng:

```
public partial class qlhvContext : DbContext
{
    6 references
    public qlhvContext()...
    0 references
    public qlhvContext(DbContextOptions<qlhvContext> options)...
    0 references
    public virtual DbSet<Diemthi> Diemthis { get; set; }
    0 references
    public virtual DbSet<Lop> Lops { get; set; }
    0 references
    public virtual DbSet<Lylich> Lyliches { get; set; }
    9 references
    public virtual DbSet<Monhoc> Monhocs { get; set; }
    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)...
    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)...
    1 reference
    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

2. Entity Framework Core

- Lớp **Monhoc** chứa thông tin môn học của bảng **monhoc** trong

CSDL:

```
public partial class Monhoc
{
    1 reference
    public Monhoc()
    {
        Diemthis = new HashSet<Diemthi>();
    }
    5 references
    public string Msmh { get; set; }
    4 references
    public string Tenmh { get; set; }
    4 references
    public int? Sotiet { get; set; }
    2 references
    public virtual ICollection<Diemthi> Diemthis { get; set; }
}
```


2. Entity Framework Core

- Lớp **Lop** chứa thông tin lớp học của bảng **lop** trong CSDL:

```
public partial class Lop
{
    0 references
    public Lop()
    {
        Lyliches = new HashSet<Lylich>();
    }
    2 references
    public string Malop { get; set; }
    1 reference
    public string Tenlop { get; set; }
    2 references
    public virtual ICollection<Lylich> Lyliches { get; set; }
}
```


2. Entity Framework Core

- Lớp **Lylich** chứa thông tin học viên của bảng lylich trong CSDL:

```
public partial class Lylich
{
    0 references
    public Lylich()
    {
        Diemthis = new HashSet<Diemthi>();
    }
    2 references
    public string Mshv { get; set; }
    1 reference
    public string Tenhv { get; set; }
    1 reference
    public DateTime? Ngaysinh { get; set; }
    1 reference
    public bool? Phai { get; set; }
    2 references
    public string Malop { get; set; }
    1 reference
    public virtual Lop MalopNavigation { get; set; }
    2 references
    public virtual ICollection<Diemthi> Diemthis { get; set; }
}
```

2. Entity Framework Core

- Lớp **Diemthi** chứa thông tin điểm môn học của học viên của bảng **diemthi** trong CSDL:

```
public partial class Diemthi
{
    3 references
    public string Mshv { get; set; }
    7 references
    public string Msmh { get; set; }
    1 reference
    public string Diem { get; set; }
    1 reference
    public virtual Lylich MshvNavigation { get; set; }
    1 reference
    public virtual Monhoc MsmhNavigation { get; set; }
}
```

2. Entity Framework Core

❑ Tạo CSDL từ Model có sẵn (*Code-First*):

- + B1: Tạo Model.
- + B2: Từ cửa sổ Package Manager Console thêm Migration bằng lệnh

add-migration <tên Migration>

- + B3: Từ cửa sổ Package Manager Console tạo CSDL bằng lệnh

update-database

- ❖ Migration là một cách để giữ cho lược đồ cơ sở dữ liệu đồng bộ với mô hình EF Core bằng cách bảo toàn dữ liệu.

3. Linq to Entities

- ❑ Lớp **DbContext**: ánh xạ CSDL với tập đối tượng trong chương trình.
 - Thư viện: **System.Data.Entity**
 - Phương thức **SaveChange()**: Cập nhật dữ liệu trong tập đối tượng xuống CSDL.

3. Linq to Entities

- ❑ Lớp **DBSet**: ánh xạ bảng dữ liệu trong CSDL với tập đối tượng trong chương trình.
 - Thư viện: **System.Data.Entity**
 - Các phương thức:
 - object Add (object entity)**: thêm đối tượng vào tập đối tượng.
 - object Find (params object[] keyValues)**: Tìm đối tượng trong tập đối tượng dựa vào khóa.
 - object Remove (object entity)**: Xóa đối tượng trong tập đối tượng.

3. Linq to Entities

❑ Ví dụ: `qlhvContext dc = new qlhvContext();`

- Ghi thông tin của đối tượng môn học mh vào CSDL.

```
dc.Monhocs.Add(mh);  
dc.SaveChanges();
```

- Xóa môn học trong CSDL có mã số môn học là msmh

```
Monhoc mh = dc.Monhocs.Find(msmh);  
if (mh != null)  
{  
    dc.Monhocs.Remove(mh);  
    dc.SaveChanges();  
}
```

3. Linq to Entities

- Truy vấn đối tượng: Trả về danh sách môn học có tên môn học chứa giá trị tenmh.
 - Câu lệnh truy vấn Linq:
`var kq = from x in dc.Monhocs
where x.Tenmh.Contains(tenmh)
select x;`
 - Phương thức Linq:
`var kq = dc.Monhocs
.Where(x => x.Tenmh.Contains(tenmh))
.ToList();`

3. Linq to Entities

- Mệnh đề group...by: Trả về số học sinh đã đăng ký cho từng môn học.

- Câu lệnh truy vấn Linq:

```
var kq = from x in dc.Diemthis.ToList()..  
group x by x.Msmh into dx..  
select new { Msmh=dx.First().Msmh  
            ,slhs=dx.ToList().Count};
```

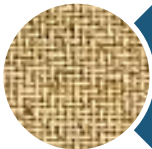
- Phương thức Linq:

```
var kq = dc.Diemthis.ToList().GroupBy(x => x.Msmh)  
        .Select(y => new { Msmh = y.First().Msmh,  
                           slhs = y.ToList().Count  
        });
```

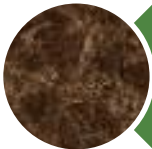

III. HTML và Bootstrap



Các thẻ HTML cơ bản



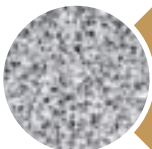
Thẻ Form



Table



Tổ chức bố cục trang Web



Bootstrap

1. Các thẻ HTML cơ bản

- ❖ **HTML (Hyper Text Markup Language):** là một ngôn ngữ được thiết kế để tạo nên các trang web với các mẫu thông tin được trình bày trên World Wide Web. Cùng với CSS và JavaScript, HTML tạo ra bộ ba nền tảng kỹ thuật cho World Wide Web.
- ❖ **CSS (Cascading Style Sheets):** được dùng để miêu tả cách trình bày các tài liệu viết bằng ngôn ngữ HTML và XHTML.

1. Các thẻ HTML cơ bản

1) Cấu trúc trang HTML:

❑ Thẻ `<head>`: chứa những khai báo thông tin cho trang web.

❑ Thẻ `<body>`: chứa phần hiển thị định dạng nội dung của trang web.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

1. Các thẻ HTML cơ bản

2) Tiêu đề HTML: Là các thẻ từ <h1> đến <h6>

Thẻ <h1> là tiêu đề quan trọng nhất và Thẻ <h6> là tiêu đề ít quan trọng nhất.

3) Đoạn HTML: là thẻ <p>

4) Liên kết HTML: là thẻ <a>

Ví dụ:

```
<a href="https://www.w3schools.com">This is a link</a>
```

1. Các thẻ HTML cơ bản

5) Hình ảnh HTML: là thẻ

Ví dụ:

```

```

- Thuộc tính src: chứa tên file hình ảnh.
- Thuộc tính alt: chứa văn bản thay thế.

6) Nút HTML: là thẻ <button>

7) Thẻ : được dùng để nhóm các phần tử nội tuyến với nhau, tiện cho việc định dạng CSS.

1. Các thẻ HTML cơ bản

8) Danh sách HTML: là thẻ `` (unordered/bullet list) hoặc thẻ `` (ordered/numbered list) với các phần tử trong danh sách được xác định bởi thẻ `` (list items).

Ví dụ:

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

- Coffee
- Tea
- Milk

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

1. Coffee
2. Tea
3. Milk

2. Thẻ Form

- ❖ Thẻ `<form>` HTML xác định một biểu mẫu được sử dụng để nhập dữ liệu của người dùng.
- ❖ Trong thẻ `<form>` chứa nhiều thẻ nhập dữ liệu có kiểu khác nhau.

1) Thẻ `<input>`: có thể được hiển thị theo nhiều cách, tùy thuộc vào thuộc tính type của nó.

2. Thẻ Form

Ví dụ:

```
<h2>Radio Buttons</h2>
```

```
<form>
```

```
  <input type="radio" name="gender" value="male" checked> Male<br>
```

```
  <input type="radio" name="gender" value="female"> Female<br>
```

```
  <input type="radio" name="gender" value="other"> Other
```

```
</form>
```

Kết quả:

Radio Buttons

- ☒ Male
- ☐ Female
- ☐ Other

2. Thẻ Form

Ví dụ:

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey">
    <br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse">
    <br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

Kết quả:

Personal information:

First name:

Mickey

Last name:

Mouse

Submit

2. Thẻ Form

- 2) Thuộc tính action: Xác định hành động sẽ được thực hiện khi biểu mẫu được Submit.
- 3) Thuộc tính target: Chỉ định nếu kết quả được gửi sẽ mở trong tab trình duyệt mới, khung hoặc trong cửa sổ hiện tại.
- 4) Thẻ <textarea>: Xác định trường dữ liệu nhập có thể nhập nhiều dòng dữ liệu.

2. Thẻ Form

- 5) Thuộc tính method: Chỉ định phương thức HTTP (GET hoặc POST) sẽ được sử dụng khi Submit dữ liệu form.
- GET: dữ liệu form đã gửi sẽ hiển thị trong trường địa chỉ trang.
 - POST: dữ liệu form chứa thông tin nhạy cảm hoặc thông tin cá nhân. Phương thức POST không hiển thị dữ liệu form đã gửi trong trường địa chỉ trang.

Ví dụ:

```
<form action="/action_page.php" method="get">
```

2. Thẻ Form

- 6) Thuộc tính name: Mỗi trường đầu vào phải có một thuộc tính name được gửi. Nếu thuộc tính name bị bỏ qua, dữ liệu của trường đầu vào đó sẽ không được gửi đi.
- 7) Thuộc tính value: Chỉ định giá trị ban đầu cho trường đầu vào.
- 8) Thẻ <fieldset>: Tạo nhóm dữ liệu form.
Thẻ <legend>: Xác định chú thích cho phần tử <fieldset>.
- 9) Thẻ <select>: Xác định Dropdown List.
Thẻ <option>: Xác định một mục chọn có thể được lựa chọn trong thẻ <select>.

2. Thẻ Form

Ví dụ:

```
<h2>The select Element</h2>
```

```
<p>The select element defines a drop-down list:</p>
```

```
<form action="/action_page.php">  
  <select name="cars">  
    <option value="volvo">Volvo</option>  
    <option value="saab">Saab</option>  
    <option value="fiat">Fiat</option>  
    <option value="audi">Audi</option>  
  </select>  
  <br><br>  
  <input type="submit">  
</form>
```

Kết quả:

The select Element

The select element defines a drop-down list:

Volvo ▼

Submit

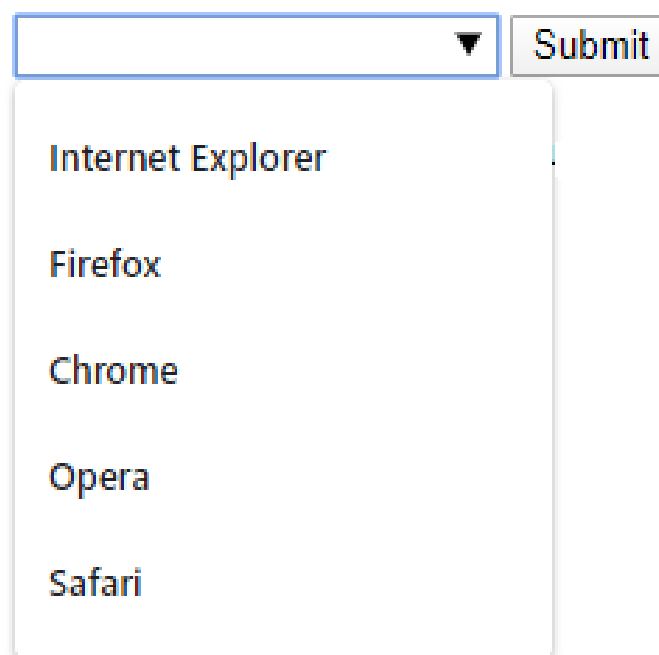
2. Thẻ Form

10) Thẻ <datalist> (trong HTML5): Xác định danh sách mục chọn cho thẻ <input>.

Ví dụ:

```
<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit">
</form>
```

Kết quả:



The screenshot shows a web form with a dropdown menu and a Submit button. The dropdown menu is open, displaying a list of browsers: Internet Explorer, Firefox, Chrome, Opera, and Safari. The Submit button is located to the right of the dropdown menu.

3. Table

- ❑ Thẻ `<table>`: Xác định bảng gồm nhiều dòng và nhiều cột.
- ❑ Thẻ `<tr>` (table row): Xác định một dòng trong bảng.
- ❑ Thẻ `<th>` (table header): Xác định tiêu đề của một cột.
- ❑ Thẻ `<td>` (table data/cell): Xác định một ô trong bảng.

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
  </tr>
</table>
```

3. Table

Kết quả:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

- ☐ Bảng có kẻ
khung dùng
CSS:

```
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
</style>
```


3. Table

- ❑ Xác định không gian của ô giữa nội dung ô và viền của nó:

```
th, td {  
    padding: 15px;  
}
```

- ❑ Căn lề tiêu đề của bảng:

```
th {  
    text-align: left;  
}
```

- ❑ Xác định khoảng cách giữa các ô trong bảng.

```
table {  
    border-spacing: 5px;  
}
```

3. Table

- ❑ Thuộc tính
colspan: Xác
định một ô có
nhiều cột.

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>
```

Kết quả:

Name	Telephone	
Bill Gates	55577854	55577855

3. Table

- ❑ Thuộc tính
rowspan: Xác
định một ô có
nhiều dòng.

```
<table style="width:100%">  
  <tr>  
    <th>Name:</th>  
    <td>Bill Gates</td>  
  </tr>  
  <tr>  
    <th rowspan="2">Telephone:</th>  
    <td>55577854</td>  
  </tr>  
  <tr>  
    <td>55577855</td>  
  </tr>  
</table>
```

Kết quả:

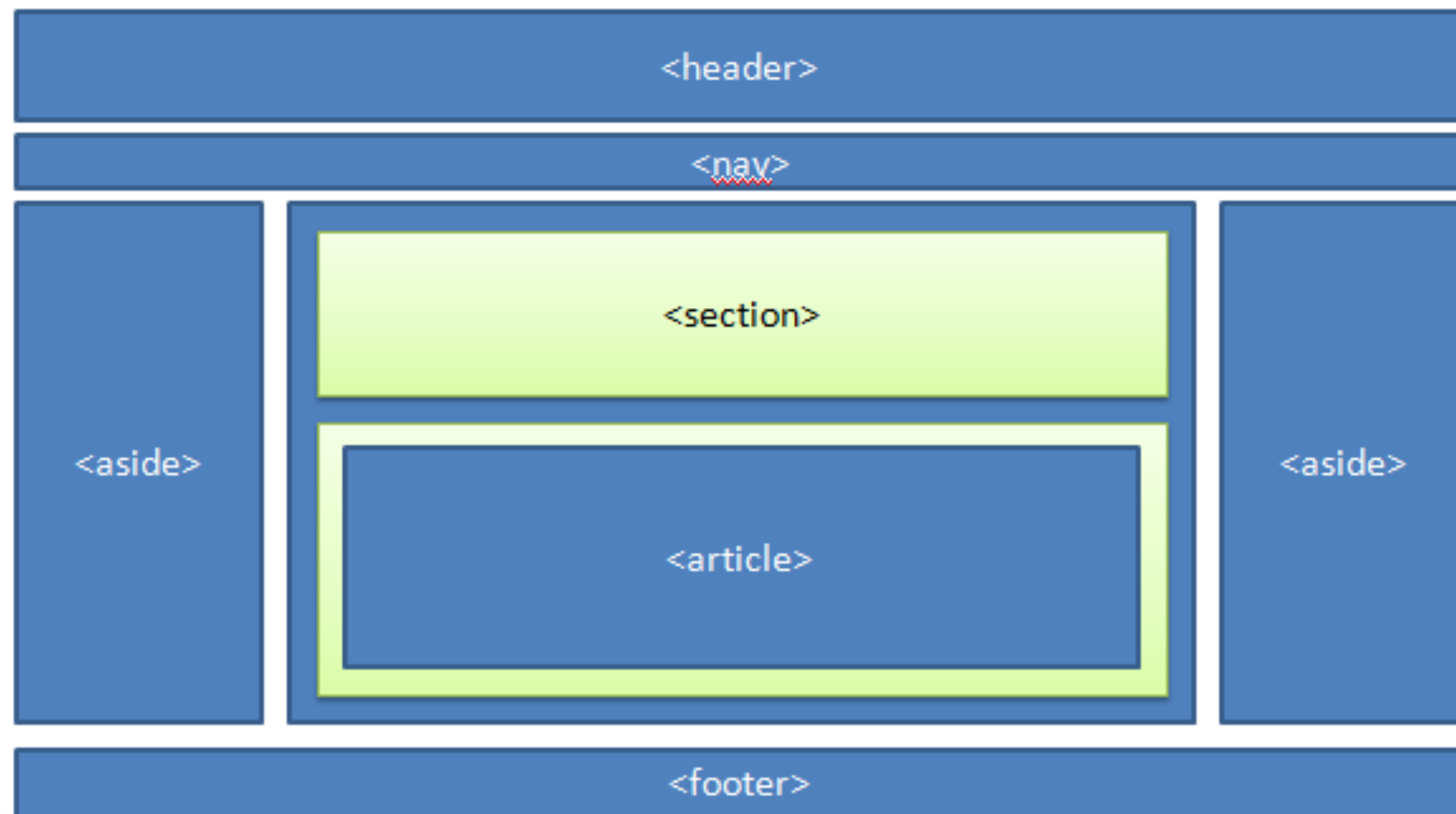
Name:	Bill Gates
Telephone:	55577854
	55577855

4. Tổ chức bố cục trang Web

- ❑ HTML cung cấp một số yếu tố ngữ nghĩa xác định các phần khác nhau của trang web như sau:
 - `<header>` : Xác định tiêu đề ở đầu cho tài liệu hoặc cho phần.
 - `<footer>` : Xác định tiêu đề ở cuối cho tài liệu hoặc cho phần.
 - `<nav>` : Xác định vùng chứa cho các liên kết điều hướng.
 - `<section>` : Xác định một phần trong tài liệu.
 - `<article>` : Xác định một bài viết độc lập.
 - `<aside>` : Xác định nội dung ngoài nội dung.

4. Tổ chức bố cục trang Web

❑ Ví dụ bố cục trang Web:



4. Tổ chức bố cục trang Web

- ❑ Có nhiều cách khác nhau để tạo bố cục của trang Web. Mỗi cách đều có ưu và nhược điểm.

1) Thuộc tính float CSS:

Việc thực hiện toàn bộ bố cục web bằng thuộc tính float CSS là điều phổ biến. Float rất dễ học - bạn chỉ cần nhớ các thức hoạt động của các thuộc tính float.

2) CSS flexbox:

Việc sử dụng flexbox đảm bảo rằng các thành phần hoạt động có thể dự đoán được khi bố cục trang được xây dựng phù hợp với các kích thước màn hình khác nhau và các thiết bị hiển thị khác nhau.

4. Tổ chức bố cục trang Web

3) CSS framework:

Sử dụng CSS framework để tạo bố cục trang Web nhanh, như W3.CSS hoặc Bootstrap.

4) CSS grid:

CSS grid cung cấp một hệ thống bố cục trang Web dựa trên lưới, với các hàng và cột, giúp thiết kế trang web dễ dàng hơn mà không phải sử dụng floats và position.

5. Bootstrap

- ❖ Bootstrap là một Framework Front-End miễn phí để phát triển web nhanh hơn và dễ dàng hơn.
- ❖ Bootstrap bao gồm các mẫu thiết kế dựa trên HTML và CSS cho kiểu chữ, biểu mẫu, nút, bảng, điều hướng, phương thức, băng chuyền hình ảnh và nhiều mẫu khác, cũng như các plugin JavaScript tùy chọn.
- ❖ Bootstrap cũng cung cấp cho bạn công cụ dễ dàng tạo ra các thiết kế trang Web đáp ứng nhu cầu của bạn.
- ❖ Thiết kế web đáp ứng việc tạo các trang web tự động điều chỉnh để trông đẹp hơn trên tất cả các thiết bị, từ điện thoại nhỏ đến máy tính để bàn lớn.

5. Bootstrap

1) Bắt đầu Bootstrap:

- ❑ Bạn có thể download Bootstrap từ địa chỉ

<https://getbootstrap.com/>

- ❑ Thêm bố cục HTML5.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="utf-8">
```

```
</head>
```

```
</html>
```

5. Bootstrap

❑ Bootstrap được thiết kế để đáp ứng cho các thiết bị di động.

- Phần `width=device-width` thiết lập chiều rộng của trang theo chiều rộng màn hình của thiết bị (sẽ thay đổi tùy theo thiết bị).
- Phần `initial-scale=1` đặt mức khởi tạo zoom khi trang web được trình duyệt tải lần đầu tiên.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

5. Bootstrap

- ❑ Bootstrap cũng yêu cầu một thành phần chứa (containing element) để bao bọc nội dung trang web.
 - Lớp `.container` cung cấp một thành phần chứa có chiều rộng cố định.
 - Lớp `.container-fluid` cung cấp một thành phần chứa có chiều rộng đầy đủ, trải rộng toàn bộ chiều rộng của khung nhìn.

```
<div class="container">  
  <h1>My First Bootstrap Page</h1>  
  <p>This is some text.</p>  
</div>
```

5. Bootstrap

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This part is inside a .container class.</p>
  <p>The .container class provides a responsive fixed width container.</p>
</div>

</body>
</html>
```

5. Bootstrap

2) Hệ thống lưới Bootstrap:

❑ Hệ thống lưới của Bootstrap được xây dựng cho phép tối đa 12 cột trên trang.

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

5. Bootstrap

- ❑ Hệ thống lưới linh hoạt và các cột sẽ tự động sắp xếp lại tùy thuộc vào kích thước màn hình.
- ❑ Hệ thống lưới Bootstrap có năm lớp. Các lớp ở trên có thể được kết hợp để tạo ra bố cục trang web năng động và linh hoạt hơn.

`.col-` (extra small devices - screen width less than 576px)

`.col-sm-` (small devices - screen width equal to or greater than 576px)

`.col-md-` (medium devices - screen width equal to or greater than 768px)

`.col-lg-` (large devices - screen width equal to or greater than 992px)

`.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)

5. Bootstrap

- ❑ Cấu trúc cơ bản của lưới Bootstrap: `<div class="row">` tạo dòng, còn `<div class="col-*-*">` tạo cột.

```
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
```

5. Bootstrap

3) Bảng Bootstrap:

- ❑ Lớp `.table` thêm kiểu dáng cơ bản vào bảng.
- ❑ Lớp `.table-striped` thêm sọc ngang vào bảng.

```
<table class="table">  
  <thead>  
    <tr>  
      <th>Firstname</th>  
      <th>Lastname</th>  
      <th>Email</th>  
    </tr>  
  </thead>  
  <tbody>  
    <tr>  
      <td>John</td>  
      <td>Doe</td>  
      <td>john@example.com</td>  
    </tr>  
    <tr>  
      <td>Mary</td>  
      <td>Moe</td>  
      <td>mary@example.com</td>  
    </tr>  
    <tr>  
      <td>July</td>  
      <td>Dooley</td>  
      <td>july@example.com</td>  
    </tr>  
  </tbody>  
</table>
```


5. Bootstrap

- ☐ Lớp `.table-bordered` thêm đường viền trên tất cả các mặt của bảng và các ô.
- ☐ Lớp `.table-hover` thêm hiệu ứng di chuột (màu nền xám) trên các hàng của bảng.
- ☐ Lớp `.table-dark` thêm nền đen vào bảng.

5. Bootstrap

- ❑ Các lớp ngữ cảnh có thể được sử dụng để tô màu cho toàn bộ bảng (<table>), các hàng của bảng (<tr>) hoặc các ô của bảng (<td>): `.table-primary`, `.table-success`, `.table-danger`, `.table-info`, `.table-warning`, `.table-active`, `.table-secondary`, `.table-light`, `.table-dark`.

5. Bootstrap

4) List Groups:

- ❑ Sử dụng thẻ `` với lớp `.list-group` để tạo List Groups cơ bản, và các phần tử của List Groups dùng thẻ `` với lớp `.list-group-item`.

```
<div class="container">
  <h2>Basic List Group</h2>
  <ul class="list-group">
    <li class="list-group-item">First item</li>
    <li class="list-group-item">Second item</li>
    <li class="list-group-item">Third item</li>
  </ul>
</div>
```

5. Bootstrap

❑ Sử dụng thẻ

`` với lớp `.list-group` để tạo List Groups cơ bản, và các phần tử của List Groups dùng thẻ `` với lớp `.list-group-item`.

```
<div class="container">
  <h2>Basic List Group</h2>
  <ul class="list-group">
    <li class="list-group-item">First item</li>
    <li class="list-group-item">Second item</li>
    <li class="list-group-item">Third item</li>
  </ul>
</div>
```

Basic List Group

First item

Second item

Third item

5. Bootstrap

❑ Nếu bạn muốn List Groups hiển thị theo chiều ngang thay vì theo chiều dọc (cạnh nhau thay vì chồng lên nhau), hãy thêm lớp `.list-group-horizontal` vào lớp `.list-group`.

```
<ul class="list-group list-group-horizontal">  
  <li class="list-group-item">First item</li>  
  <li class="list-group-item">Second item</li>  
  <li class="list-group-item">Third item</li>  
  <li class="list-group-item">Fourth item</li>  
</ul>
```

First item	Second item	Third item	Fourth item
------------	-------------	------------	-------------

5. Bootstrap

5) Thẻ Card:

❑ Một thẻ Card trong Bootstrap 4 là một hộp có viền với một số phần đệm xung quanh nội dung của nó. Nó bao gồm các tùy chọn cho tiêu đề đầu trang (Header), tiêu đề cuối trang (Footer), nội dung, màu sắc, v.v.

```
<div class="container">  
  <h2>Card Header and Footer</h2>  
  <div class="card">  
    <div class="card-header">Header</div>  
    <div class="card-body">Content</div>  
    <div class="card-footer">Footer</div>  
  </div>  
</div>
```

5. Bootstrap

Card Header and Footer

Header

Content

Footer

- ❑ Sử dụng lớp `.card-title` để thêm tiêu đề thẻ vào bất kỳ yếu tố tiêu đề nào.

5. Bootstrap

❑ Lớp `.card-link` thêm màu xanh lam vào bất kỳ liên kết nào và hiệu ứng di chuột.

```
<div class="card">
  <div class="card-body">
    <h4 class="card-title">Card title</h4>
    <p class="card-text">Some example text. Some example text.</p>
    <a href="#" class="card-link">Card link</a>
    <a href="#" class="card-link">Another link</a>
  </div>
</div>
```

Card title

Some example text. Some example text.

[Card link](#) [Another link](#)

5. Bootstrap

6) Lớp Nav:

❑ Tạo thanh menu theo chiều ngang: sử dụng thẻ `` với lớp `.nav` , với mỗi phần tử trong thanh menu sử dụng thẻ `` với lớp `.nav-link`.

5. Bootstrap

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Link

Link

Link

Disabled

5. Bootstrap

7) Thẻ Form

❑ Thẻ `<form>` Bootstrap xác định một biểu mẫu được sử dụng để nhập dữ liệu của người dùng. Lớp `.form-group` để nhóm các phần tử trên Form thành một nhóm hiển thị theo chiều dọc.

5. Bootstrap

```
<div class="container">
  <h2>Stacked form</h2>
  <form action="/action_page.php">
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email" name="email">
    </div>
    <div class="form-group">
      <label for="pwd">Password:</label>
      <input type="password" class="form-control" id="pwd" placeholder="Enter password" name="pswd">
    </div>
    <div class="form-group form-check">
      <label class="form-check-label">
        <input class="form-check-input" type="checkbox" name="remember"> Remember me
      </label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
</div>
```

5. Bootstrap

Stacked form

Email:

Password:

☐ Remember me

5. Bootstrap

❑ Sử dụng lớp `.row` tạo các dòng và lớp `.col` để tạo các cột trên Form.

```
<form action="/action_page.php">
  <div class="row">
    <div class="col">
      <input type="text" class="form-control" id="email" placeholder="Enter email" name="email">
    </div>
    <div class="col">
      <input type="password" class="form-control" placeholder="Enter password" name="pswd">
    </div>
  </div>
  <button type="submit" class="btn btn-primary mt-3">Submit</button>
</form>
```

5. Bootstrap

❑ Bootstrap hỗ trợ cho các Controls: input, textarea, checkbox, radio, select.

```
<div class="form-group">
  <label for="usr">Name:</label>
  <input type="text" class="form-control" id="usr">
</div>

<div class="form-check">
  <label class="form-check-label">
    <input type="radio" class="form-check-input" name="optradio">Option 1
  </label>
</div>
<div class="form-check">
  <label class="form-check-label">
    <input type="radio" class="form-check-input" name="optradio">Option 2
  </label>
</div>
```

5. Bootstrap

8) Dropdown Menu:

- ❑ Lớp `.dropdown` dùng để tạo Dropdown Menu.
- ❑ Lớp `.dropdown-toggle` và `data-toggle="dropdown"` tạo nút thả xuống.
- ❑ Lớp `.dropdown-menu` và `.dropdown-item` tạo các phần tử trong danh sách lựa chọn.

5. Bootstrap

```
<div class="dropdown">  
  <button type="button" class="btn btn-primary dropdown-toggle" data-toggle="dropdown">  
    Dropdown button  
  </button>  
  <div class="dropdown-menu">  
    <a class="dropdown-item" href="#">Link 1</a>  
    <a class="dropdown-item" href="#">Link 2</a>  
    <a class="dropdown-item" href="#">Link 3</a>  
  </div>  
</div>
```

Dropdown button ▼

Link 1

Link 2

Link 3

5. Bootstrap

9) Collapsibles: dùng để ẩn và hiển thị số lượng lớn nội dung. Có hai phần:

- ❑ Phần 1: Nút dùng để ẩn/hiện nội dung. Trong thẻ `<button>` sử dụng hai thuộc tính `data-toggle="collapse"` và `data-target="#id"`.
- ❑ Phần 2: Đoạn chứa nội dung hiển thị. Sử dụng lớp `.collapse`.

5. Bootstrap

```
<button data-toggle="collapse" data-target="#demo">Collapsible</button>
```

```
<div id="demo" class="collapse">
```

```
  Lorem ipsum dolor text....
```

```
</div>
```

5. Bootstrap

10) Hộp thông báo: Tạo hộp thông báo có hai phần:

- ❑ Phần 1: Nút mở hộp thông báo. Sử dụng thẻ `< button >` với hai thuộc tính `data-toggle="modal"` và `data-target="#id"`.
- ❑ Phần 2: Hộp thông báo. Sử dụng lớp `.modal`, lớp `.modal-dialog` và lớp `.modal-content`.

5. Bootstrap

```
<!-- Button to Open the Modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
  Open modal
</button>

<!-- The Modal -->
<div class="modal" id="myModal">
  <div class="modal-dialog">
    <div class="modal-content">

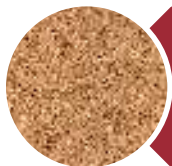
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Modal Heading</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>

      <!-- Modal body -->
      <div class="modal-body">
        Modal body..
      </div>

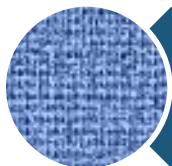
      <!-- Modal footer -->
      <div class="modal-footer">
        <button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>
      </div>

    </div>
  </div>
</div>
```

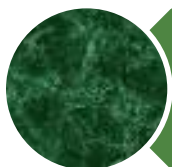
IV. Tag Helpers và Validation



View Model



Tag Helpers



Validation



Session, Application và Cookie

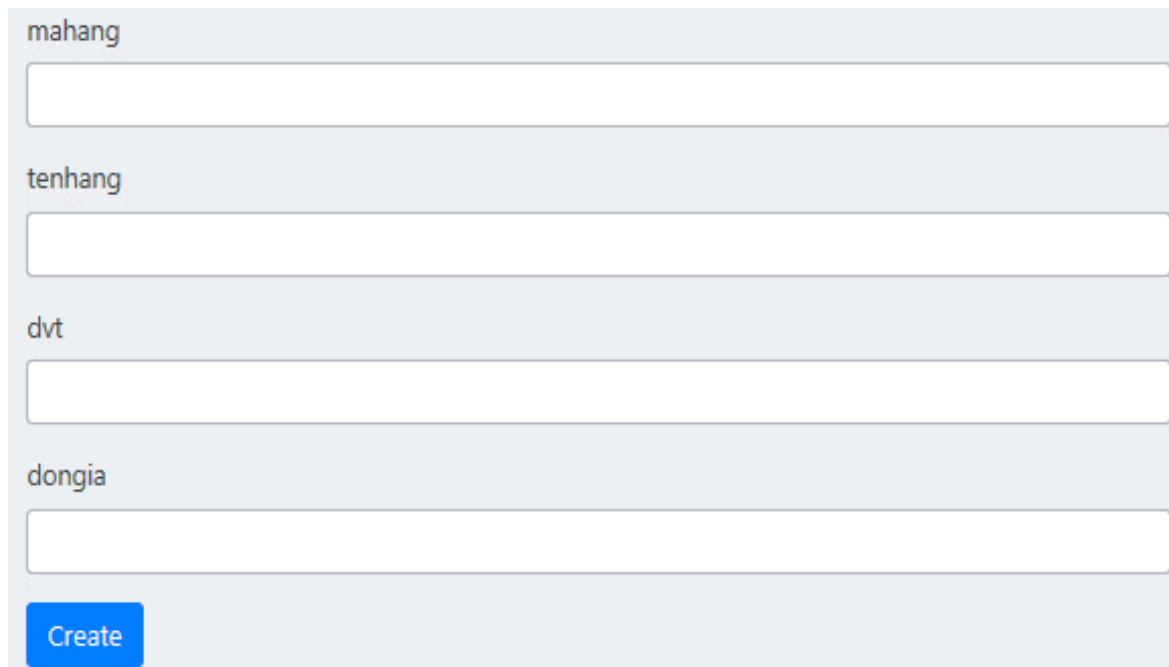
1. View Model

- ❑ View Model là Model được thiết kế đặc biệt để sử dụng trong View. Nó cung cấp giao diện đơn giản dựa trên Model giúp duy trì việc ra quyết định trên View ở mức tối thiểu.
- ❑ Ví dụ:
 - Tạo lớp CHanghoa là ViewModel chứa thông tin Hàng Hóa.

```
public class CHanghoa
{
    public string mahang { get; set; }
    public string tenhang { get; set; }
    public string dvt { get; set; }
    public string dongia { get; set; }
}
```

1. View Model

- Tạo Form nhập Thông tin hàng hóa với các thẻ HTML có tên trùng với tên của các thuộc tính trong đối tượng của lớp CHanghoa.



The image shows a web form for creating a product. It consists of four input fields and a 'Create' button. The labels for the fields are 'mahang', 'tenhang', 'dvt', and 'dongia'. The 'Create' button is blue and located at the bottom left of the form.

mahang
tenhang
dvt
dongia
Create

1. View Model

- Trong Controller Home tạo Action CreateHanghoa(), action này có tham số a là đối tượng nhận dữ liệu nhập của Form sau khi Form được Submit.

[HttpPost]

```
public ActionResult CreateHanghoa(models.CHanghoa a)
{
    models.hanghoa x = new models.hanghoa();
    x.mahang = a.mahang;
    x.tenhang = a.tenhang;
    x.dvt = a.dvt;
    x.dongia = double.Parse(a.dongia);

    dc.hanghoas.Add(x);
    dc.SaveChanges();

    return RedirectToAction("Index");
}
```

2. Tag Helpers

- ❑ Tag Helpers là các thành phần tạo ra các thẻ HTML.
- ❑ Tất cả các Tag Helpers sinh ra thẻ HTML như forms, controls, input validation và trả về kết quả là chuỗi.

2. Tag Helpers

- ❑ Tag Helpers liên kết đối tượng Model với các thẻ html để hiển thị giá trị của các thuộc tính Model lên các thẻ html và cũng gán giá trị của các thẻ html cho các thuộc tính Model sau khi Submit biểu mẫu.
- ❑ Để làm việc với Tag Helpers thì cần có dòng khai báo `@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers` trong file `Views/_ViewImports.cshtml` của ứng dụng.

2. Tag Helpers

1) Anchor Tag Helper:

- ☐ Mở rộng thẻ `<a ...> ` trong HTML.
- ☐ Các thuộc tính:
 - **asp-controller**: xác định Controller
 - **asp-action**: xác định phương thức action

`<a asp-controller="Speaker"`

`asp-action="Index">All Speakers`

2. Tag Helpers

- **asp-route-{value}**: xác định tham số cho định tuyến

```
<a asp-controller="Speaker" asp-action="Detail"  
asp-route-id="@Model.SpeakerId">SpeakerId:  
@Model.SpeakerId</a>
```

tương đương với

```
<a href="/Speaker/Detail/12">SpeakerId: 12</a>
```

2. Tag Helpers

```
<a asp-controller="Speaker" asp-action="Detail"  
asp-route-speakerid="@Model.SpeakerId">SpeakerId:  
@Model.SpeakerId</a>
```

tương đương với

```
<a href="/Speaker/Detail?speakerid=12"> SpeakerId:12  
</a>
```

2. Tag Helpers

- **asp-all-route-data**: xác định tham số là một tập hợp gồm các bộ (key,value) cho định tuyến

```
@{ var parms = new Dictionary<string, string> {  
    { "speakerId", "11" }, { "currentYear", "true" } }; }
```

```
<a asp-controller="Speaker" asp-action="Detail" asp-all-  
route-data="parms">SpeakerId</a>
```

tương đương với

```
<a href="/Speaker/Detail?speakerId=11&currentYear=true">  
SpeakerId</a>
```

2. Tag Helpers

2) Form Tag Helper:

Tạo thẻ `<form ...>`.

```
<form asp-controller="Demo" asp-action="Register" method="post">
```

```
<!-- Input and Submit elements -->
```

```
</form>
```


2. Tag Helpers

3) Input Tag Helper:

Tạo thẻ `<input ... >`

- Thuộc tính **asp-for** xác định thuộc tính **id** và **name** trong thẻ `input` của HTML.

```
<input asp-for="Msmh" class="form-control" />
```

tương đương với

```
<input id="Msmh" name="Msmh" class="form-control" />
```

2. Tag Helpers

- Kiểu dữ liệu của thuộc tính trong đối tượng Model xác định thuộc tính **type** trong thẻ input của HTML và thuộc tính chú thích dữ liệu (data annotation attributes) xác định kiểu thuộc tính của đối tượng Model.

2. Tag Helpers

```
public class CMonhoc
{
    0 references
    public string Msmh { get; set; }
    0 references
    public string Tenmh { get; set; }
    0 references
    public int? Sotiet { get; set; }
}
```

Thuộc tính Sotiet có kiểu int nên thẻ input có type="number".

```
<input asp-for="Sotiet" class="form-control" />
```

Thuộc tính Tenmh có kiểu string nên thẻ input có type="text".

```
<input asp-for="Tenmh" class="form-control" />
```

2. Tag Helpers

```
public class CHocvien
{
    4 references
    public string Mshv { get; set; }
    4 references
    public string Tenhv { get; set; }
    [DataType(DataType.Date)]
    4 references
    public DateTime? Ngaysinh { get; set; }
    4 references
    public bool Phai { get; set; }
    4 references
    public string Malop { get; set; }
}
```

Thuộc tính Phai có kiểu bool nên thẻ input có type="checkbox".

```
<input asp-for="Phai" class="form-control" />
```

Thuộc tính Ngaysinh có kiểu Date nên thẻ input có type="date".

```
<input asp-for="Ngaysinh" class="form-control" />
```

2. Tag Helpers

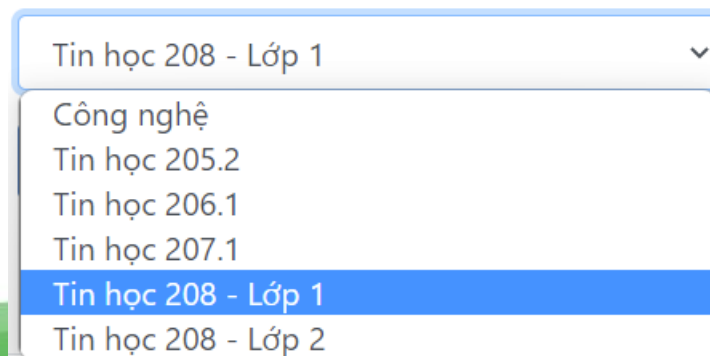
4) Select Tag Helper:

Tạo thẻ `<select ...>` và mối quan hệ với các phần tử trong thẻ `<option ...>`

```
<select asp-for="Malop" asp-items="@ViewBag.DSLop"
class="form-control"></select>
```

```
ViewBag.DSLop = new SelectList(db.Lops.ToList(),"Malop", "Tenlop");
```

Malop



Tin học 208 - Lớp 1	▼
Công nghệ	
Tin học 205.2	
Tin học 206.1	
Tin học 207.1	
Tin học 208 - Lớp 1	
Tin học 208 - Lớp 2	

2. Tag Helpers

5) Label Tag Helper:

Tạo thẻ <label ...>

<label **asp-for**="Msmh" class="control-label"></label>

<input **asp-for**="Msmh" class="form-control" />

```
public class CMonhoc
{
    [Display(Name = "Mã số môn học")]
    0 references
    public string Msmh { get; set; }
    0 references
    public string Tenmh { get; set; }
    0 references
    public int? Sotiet { get; set; }
}
```

3. Validation

- ❖ Kiểm tra dữ liệu nhập từ phía Client và phía Server.
- ❖ ASP.NET MVC framework sẽ tự động thực thi các quy tắc kiểm tra dữ liệu nhập và hiển thị các thông báo xác thực trong View.

1) Code từ Model:

Sử dụng thư viện

`System.ComponentModel.DataAnnotations` để xác định các thuộc tính dữ liệu:

3. Validation

Thuộc tính	Mô tả
[Required]	Sử dụng xác định thuộc tính dữ liệu là bắt buộc.
[StringLength]	Sử dụng xác định số ký tự tối đa và tối thiểu của thuộc tính dữ liệu.
[Range]	Sử dụng xác định miền giá trị số của thuộc tính dữ liệu.
[RegularExpression]	Sử dụng xác định thuộc tính dữ liệu phải khớp với biểu thức quy định.
[MinLength]	Sử dụng xác định độ dài tối thiểu của mảng dữ liệu hay chuỗi trong thuộc tính dữ liệu.
[MaxLength]	Sử dụng xác định độ dài tối đa của mảng dữ liệu hay chuỗi trong thuộc tính dữ liệu.
[Compare]	Sử dụng kiểm tra hai thuộc tính dữ liệu của một Model.
[CustomValidation]	Phương thức xác thực tùy chọn được chỉ định để xác thực giá trị trường dữ liệu.

3. Validation

2) Code từ View:

❑ **Validation Message Tag Helper:** Hiển thị thông báo xác thực nếu có lỗi trong mục nhập được chỉ định trong đối tượng ModelStateDictionary

```
<span asp-validation-for="Email"></span>
```

tương ứng với

```
<span class="field-validation-valid"
```

```
data-valmsg-for="Email"
```

```
data-valmsg-replace="true"></span>
```

3. Validation

- ❑ **Validation Summary Tag Helper:** Tạo một danh sách không có thứ tự các thông báo xác thực trong đối tượng ModelStateDictionary và được sử dụng hiển thị tất cả các thông báo lỗi cho các trường.

`<div asp-validation-summary="ModelOnly" class="text-danger"></div>`

asp-validation-summary	Validation messages displayed
All	Property and model level
ModelOnly	Model
None	None

3. Validation

3) Ví dụ: Tạo ứng dụng hiển thị View

Danh sách môn học

Thêm môn học

Mã số môn học	Tên môn học	Số tiết	
a	Anh văn 3	35	Xóa
csdl	Nhập môn cơ sở dữ liệu	45	Xóa
ctks	Công tác kỹ sư	60	Xóa
hdh	Hệ điều hành	60	Xóa
ktmt	Kiến trúc máy tính	45	Xóa
ltw	Lập trình Windows	30	Xóa
nmth	Nhập môn tin học	45	Xóa

3. Validation

□ Code HTML

```
@model IEnumerable<waMonhoc_Ajax.Models.Monhoc>
@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h3>Danh sách môn học</h3>
<p>
    <a asp-action="formThemMonhoc" asp-controller="QLMonhoc"
        class="btn btn-primary">Thêm môn học</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>Mã số môn học </th>
            <th>Tên môn học </th>
            <th>Số tiết </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>@item.Msmh </td>
                <td>@item.Tenmh </td>
                <td>@item.Sotiet </td>
                <td>
                    <a asp-action="formXoaMonhoc" asp-controller="QLMonhoc"
                        asp-route-id="@item.Msmh" class="btn btn-primary">Xóa</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

3. Validation

□ Code C#

```
public class QLMonhocController : Controller
{
    private qlhvContext db = new qlhvContext();
    0 references
    public IActionResult Index()
    {
        return View(db.Monhocs.ToList());
    }
    0 references
    public IActionResult formThemMonhoc()...
    [HttpPost]
    0 references
    public IActionResult themMonhoc(Monhoc mh)...
    0 references
    public IActionResult formXoaMonhoc(string id)...
    0 references
    public IActionResult xoaMonhoc(string mamh)...
}
```

```
public partial class Monhoc
{
    1 reference
    public Monhoc()...
    [Display(Name = "Mã môn học")]
    [Required(ErrorMessage = "Bạn chưa nhập Mã số môn học")]
    23 references
    public string Msmh { get; set; }
    [Display(Name = "Tên môn học")]
    [Required(ErrorMessage = "Bạn chưa nhập Tên môn học")]
    16 references
    public string Tenmh { get; set; }
    [Display(Name = "Số tiết")]
    [Required(ErrorMessage = "Bạn chưa nhập Số tiết")]
    16 references
    public int? Sotiet { get; set; }
    2 references
    public virtual ICollection<Diemthi> Diemthis { get; set; }
```

3. Validation

❑ View cho nút “Thêm môn học”

Form thêm môn học

Trở về

Mã môn học

Tên môn học

Số tiết

Thêm

3. Validation

❑ Code

HTML

cho nút

“Thêm

môn

học”

```
@model waMonhoc_Ajax.Models.Monhoc
@{
    ViewData["Title"] = "formThemMonhoc";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h3>Form thêm môn học</h3>
<div>
    <a asp-action="Index" asp-controller="QLMonhoc" class="btn btn-primary">Trở về</a>
</div>
<hr />
<div class="row">
    <div class="col-md-12">
        <form asp-action="themMonhoc" asp-controller="QLMonhoc" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Msmh" class="control-label"></label>
                <input asp-for="Msmh" class="form-control" />
                <span asp-validation-for="Msmh" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Tenmh" class="control-label"></label>
                <input asp-for="Tenmh" class="form-control" />
                <span asp-validation-for="Tenmh" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Sotiet" class="control-label"></label>
                <input asp-for="Sotiet" class="form-control" />
                <span asp-validation-for="Sotiet" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Thêm" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
```

3. Validation

❑ Code C#

```
public class QLMonhocController : Controller
{
    private qlhvContext db = new qlhvContext();
    0 references
    public IActionResult Index()...
    0 references
    public IActionResult formThemMonhoc()
    {
        return View();
    }
    [HttpPost]
    0 references
    public IActionResult themMonhoc(Monhoc mh)
    {
        if (ModelState.IsValid)
        {
            db.Monhocs.Add(mh);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        return View("formThemMonhoc");
    }
    0 references
    public IActionResult formXoaMonhoc(string id)...
    0 references
    public IActionResult xoaMonhoc(string mamh)...
```


3. Validation

❑ View cho nút “Xóa”

Form xóa môn học

[Trở về](#)

Bạn có thật sự muốn xóa môn học này không?

Thông tin môn học

Mã môn học	a
Tên môn học	Anh văn 3
Số tiết	35

[Xóa](#)

Form xóa môn học

[Trở về](#)

Bạn không thể xóa môn học này!

3. Validation

❑ Code

HTML

cho nút

“Xóa”

```
@model waMonhoc_Ajax.Models.Monhoc
@{
    ViewData["Title"] = "formXoaMonhoc";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h3>Form xóa môn học</h3>
<a asp-action="Index" class="btn btn-primary">Trở về</a>
@if((bool)ViewBag.flagDelete){
    <h4>Bạn có thật sự muốn xóa môn học này không?</h4>
    <hr />
    <div>
        <h6>Thông tin môn học</h6>
        <hr />
        <dl class="row">
            <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Msmh)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Msmh)
            </dd>
            <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Tenmh)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Tenmh)
            </dd>
            <dt class="col-sm-2">
                @Html.DisplayNameFor(model => model.Sotiet)
            </dt>
            <dd class="col-sm-10">
                @Html.DisplayFor(model => model.Sotiet)
            </dd>
        </dl>
        <form asp-action="xoaMonhoc" asp-controller="QLMonhoc">
            <input type="hidden" value="@Model.Msmh" name="mamh" />
            <input type="submit" value="Xóa" class="btn btn-danger" />
        </form>
    </div>
}
}
else { <h4>Bạn không thể xóa môn học này!</h4> }
```

3. Validation

□ Code C#

```
public class QLMonhocController : Controller
{
    private qlhvContext db = new qlhvContext();
    0 references
    public IActionResult Index()...
    0 references
    public IActionResult formThemMonhoc()...
    [HttpPost]
    0 references
    public IActionResult themMonhoc(Monhoc mh)...
    0 references
    public IActionResult formXoaMonhoc(string id)
    {
        ViewBag.flagDelete = true;
        Monhoc x = db.Monhocs.Find(id);
        if (x == null) ViewBag.flagDelete = false;
        else
        {
            if(db.Diemthis.Where(a=>a.Msmh==id).ToList().Count>0)
                ViewBag.flagDelete = false;
        }
        return View(x);
    }
    0 references
    public IActionResult xoaMonhoc(string mamh)
    {
        Monhoc x = db.Monhocs.Find(mamh);
        db.Monhocs.Remove(x);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
}
```

4. Session, Application và Cookie

1) Session:

- ❑ Session là một vùng nhớ tồn tại trong một phiên làm việc giữa client và server, bắt đầu khi có yêu cầu từ client và kết thúc khi người dùng giải phóng vùng nhớ Session hoặc khi session hết thời gian timeout.

4. Session, Application và Cookie

- ❑ Session được dùng để lưu trữ thông tin liên quan đến người dùng, nhưng thay vì được lưu trên máy client, các biến session được lưu trên máy server.
- ❑ Mỗi session có một định danh duy nhất (SessionID).

4. Session, Application và Cookie

- ❑ Cấu hình session trong ASP.NET Core: khai báo các thành phần trong lớp Startup

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddDistributedMemoryCache();
    services.AddSession(options =>
    {
        options.IdleTimeout = TimeSpan.FromSeconds(10);
        options.Cookie.HttpOnly = true;
        options.Cookie.IsEssential = true;
    });
}
```

4. Session, Application và Cookie

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())...
    else...
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseSession();

    app.UseEndpoints(endpoints =>...);
}
```

- ❑ Sau khi cấu hình Session thì thành phần **HttpContext.Session** có hiệu lực.

4. Session, Application và Cookie

❑ Set và Get giá trị Session

- Các phương thức thuộc Namespace

`Microsoft.AspNetCore.Http`

- Phương thức GetString()

`public static string? GetString (this`

`Microsoft.AspNetCore.Http.ISession session, string key);`

- Phương thức SetString()

`public static void SetString (this Microsoft.AspNetCore.Http.ISession
session, string key, string value);`

4. Session, Application và Cookie

2) Application:

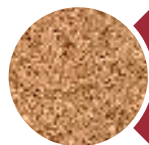
- ❑ Application là một vùng nhớ được sử dụng cho mọi Client trên toàn bộ ứng dụng. Bất cứ Client nào cũng có thể thực hiện các thao tác truy xuất, thay đổi, xóa dữ liệu trên Application.

4. Session, Application và Cookie

3) Cookie:

- ❑ Cookie là những mẫu tin nhỏ được lưu trữ trên client bằng các file văn bản. Cookie được giao tiếp với Server thông qua đối tượng Request và Response.
- ❑ Hạn chế của Cookie là chỉ lưu trữ dữ liệu text và kích thước tối đa là 4KB. Việc lưu trữ Cookie còn tùy thuộc vào Client có cho phép lưu trữ hay không.

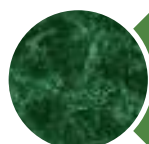
V. JavaScript và JavaScript Ajax



JavaScript HTML DOM



The Browser Object Model



Events



JSON



JavaScript Ajax

1. JavaScript HTML DOM

- ❑ JavaScript HTML DOM (Document Object Model) có thể truy cập và thay đổi tất cả các phần tử trong tài liệu HTML.
- ❑ HTML DOM là một mô hình đối tượng tiêu chuẩn và giao diện lập trình cho HTML. Các phần tử HTML được xem là các đối tượng nên có các thuộc tính (property), các phương thức (Method) và các sự kiện (event).

1. JavaScript HTML DOM

- ❑ The HTML DOM Document Object: đối tượng tài liệu biểu diễn một trang web. Do đó để truy cập các phần tử trong trang HTML phải bắt đầu từ đối tượng tài liệu.
- ❑ Phương thức tìm kiếm phần tử HTML

Method	Description
<code>document.getElementById(<i>id</i>)</code>	Find an element by element id
<code>document.getElementsByTagName(<i>name</i>)</code>	Find elements by tag name
<code>document.getElementsByClassName(<i>name</i>)</code>	Find elements by class name

1. JavaScript HTML DOM

❑ Thay đổi các phần tử HTML

Property	Description
<i>element.innerHTML = new html content</i>	Change the inner HTML of an element
<i>element.attribute = new value</i>	Change the attribute value of an HTML element
<i>element.style.property = new style</i>	Change the style of an HTML element
Method	Description
<i>element.setAttribute(attribute, value)</i>	Change the attribute value of an HTML element

1. JavaScript HTML DOM

❑ Thêm trình xử lý sự kiện

Method	Description
<code>document.getElementById(<i>id</i>).onclick = function(){<i>code</i>}</code>	Adding event handler code to an onclick event

❑ Tìm những phần tử HTML bằng CSS Selectors:

Phương thức: **querySelectorAll()**

```
const x = document.querySelectorAll("p.intro");
```

Trả về tất cả các thẻ <p...> có thuộc tính class="intro"

1. JavaScript HTML DOM

- ❑ Thuộc tính `innerHTML` dùng để thay đổi nội dung HTML.

`document.getElementById(id).innerHTML = new HTML`

- ❑ Thuộc tính `document.forms` tham chiếu đến tất cả các thẻ form.

- ❑ Ví dụ tham chiếu đến phần tử trong thẻ form

`<form name="frmThemMonhoc">`

`<input asp-for="Msmh" class="form-control" />`

`</form>`

`let ms = document.forms["frmThemMonhoc"]["Msmh"].value;`

2. The Browser Object Model (BOM)

- ❑ The Browser Object Model (BOM): cho phép JavaScript “nói chuyện với” trình duyệt.
- ❑ The Window Object: biểu diễn cửa sổ của trình duyệt.
- ❑ Đối tượng `window.screen` chứa thông tin về màn hình của người dùng.
- ❑ Đối tượng `window.history` chứa lịch sử trình duyệt.
 - Phương thức `history.back()` load URL trước đó trong danh sách history.
 - Phương thức `history.forward()` load URL tiếp theo trong danh sách history.

2. The Browser Object Model (BOM)

- ❑ Đối tượng `window.location` có thể được sử dụng để lấy địa chỉ trang hiện tại (URL) và chuyển hướng trình duyệt đến một trang mới.
 - Thuộc tính `window.location.href` trả về URL của trang hiện tại.
 - Thuộc tính `window.location.hostname` trả về tên của máy chủ lưu trữ internet (của trang hiện tại).
 - Thuộc tính `window.location.pathname` trả về tên đường dẫn của trang hiện tại.
 - Phương thức `window.location.assign()` load một tài liệu mới.

2. The Browser Object Model (BOM)

❑ Hộp thông báo của JS (JavaScript Popup Boxes)

- Hộp alert:

`window.alert("sometext");`

- Hộp confirm: Trên hộp có hai nút “OK” và “Cancel” tương ứng với giá trị true và false được trả về khi người dùng nhấn nút để đóng hộp.

`window.confirm("sometext");`

- Hộp prompt: Trên hộp có TextBox cho phép người dùng nhập dữ liệu vào TextBox và hai nút “OK” và “Cancel”. Khi nhấn “OK” phương thức trả về giá trị trong TextBox, ngược lại trả về null.

`window.prompt("sometext", "defaultText");`

3. Sự kiện (Events)

❑ Sự kiện HTML:

<element event="some JavaScript">

❑ Các sự kiện thường dùng:

Event	Description
onchange	Phần tử HTML bị thay đổi
onclick	Phần tử HTML bị click vào
onmouseover	Phần tử HTML bị di chuyển chuột đi vào
onmouseout	Phần tử HTML bị di chuyển chuột đi ra
onkeydown	Khi người dùng nhấn phím
onload	Khi trang web hoàn thành load trang

3. Sự kiện (Events)

- ❑ Khai báo hàm trong Javascript:

```
Function functionName(parameters) {  
    // code to be executed  
}
```

4. JSON (JavaScript Object Notation)

❑ JSON là định dạng văn bản dùng để lưu trữ dữ liệu và truyền dữ liệu.

❑ Ví dụ cho đối tượng JSON như sau:

```
{"name":"John", "age":30, "car":null}
```

- Đối tượng này có các thuộc tính (properties) là: name, age và car.
- Thuộc tính name có giá trị “John”, age có giá trị 30 và car có giá trị null.

❑ Đối tượng JSON được chứa trong hai dấu ngoặc nhọn.

4. JSON (JavaScript Object Notation)

- ❑ JSON có các kiểu dữ liệu sau: string, number, object, array, Boolean, null.
- ❑ Mảng JSON được chứa trong hai dấu ngoặc vuông.
- ❑ Cả hai JSON và XML đều dùng để lưu trữ dữ liệu và truyền dữ liệu.
- ❑ Ví dụ mảng JSON

```
{"employees":  
  { "firstName":"John", "lastName":"Doe" },  
  { "firstName":"Anna", "lastName":"Smith" },  
  { "firstName":"Peter", "lastName":"Jones" }  
}]
```


4. JSON (JavaScript Object Notation)

❑ Ví dụ mảng XML

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```


4. JSON (JavaScript Object Notation)

- ❑ Hàm `JSON.parse()` dùng để chuyển đổi một chuỗi thành đối tượng JSON.
- ❑ Hàm `JSON.stringify()` dùng để chuyển đổi một đối tượng JSON thành một chuỗi.

5. JavaScript Ajax

- ❑ AJAX (Asynchronous JavaScript and XML) là một bộ công cụ cho phép load dữ liệu từ server mà không yêu cầu tải lại trang.
- ❑ AJAX sử dụng chức năng sẵn có XMLHttpRequest(XHR) của trình duyệt để gửi một yêu cầu đến server và nhận kết quả xử lý dữ liệu trả về từ server.
- ❑ Dữ liệu trả về từ Server có các định dạng dữ liệu là HTML, văn bản thuần túy (Text), XML và JSON (JavaScript Object Notation).

5. JavaScript Ajax

❑ Đối tượng XMLHttpRequest: là nội dung cốt lõi của AJAX. Các bước sử dụng XMLHttpRequest:

B1: Tạo đối tượng XMLHttpRequest.

B2: Định nghĩa hàm Callback.

B3: Gọi hàm Open để mở đối tượng XMLHttpRequest.

B4: Gọi hàm Send để gửi yêu cầu đến Server.

5. JavaScript Ajax

❑ Ví dụ:

```
// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onload = function() {
    // Here you can use the Data
}

// Send a request
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

5. JavaScript Ajax

❑ Các phương thức của đối tượng XMLHttpRequest :

Method	Description
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(<i>method</i> , <i>url</i> , <i>async</i> , <i>user</i> , <i>psw</i>)	Specifies the request <i>method</i> : the request type GET or POST <i>url</i> : the file location <i>async</i> : true (asynchronous) or false (synchronous) <i>user</i> : optional user name <i>psw</i> : optional password
send()	Sends the request to the server Used for GET requests
send(<i>string</i>)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

5. JavaScript Ajax

❑ Các thuộc tính của đối tượng XMLHttpRequest :

Property	Description
onload	Defines a function to be called when the request is recieved (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the
statusText	Returns the status-text (e.g. "OK" or "Not Found")

5. JavaScript Ajax

- ❑ Ví dụ:
Tạo View
hiển thị
Danh
sách môn
học như
sau:

Danh sách môn học

Thêm môn học

Mã số môn học	Tên môn học	Số tiết	
a	Anh văn 3	35	Xóa
csdl	Nhập môn cơ sở dữ liệu	45	Xóa
ctks	Công tác kỹ sư	60	Xóa
hdh	Hệ điều hành	60	Xóa
ktmt	Kiến trúc máy tính	45	Xóa
ltw	Lập trình Windows	30	Xóa
nmth	Nhập môn tin học	45	Xóa

5. JavaScript Ajax

❑ Code

HTML

hiển

thị

View

```
<h3>Danh sách môn học</h3>
<p>
    <a asp-action="formThemMonhoc" asp-controller="ViduMonhoc"
        class="btn btn-primary">Thêm môn học</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>Mã số môn học </th>
            <th>Tên môn học </th>
            <th>Số tiết </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td> @item.Msmh </td>
                <td> @item.Tenmh </td>
                <td> @item.Sotiet </td>
                <td>
                    <input type="button" onclick="formXoaMonhoc_Click('@item.Msmh')"
                        class="btn btn-primary" value="Xóa" />
                </td>
            </tr>
        }
    </tbody>
</table>
```


5. JavaScript Ajax

❑ Code javascript cho nút “Xóa”

```
<script type="text/javascript">
    function formXoaMonhoc_Click(id)
    {
        if (confirm("Bạn có thật sự muốn xóa môn học '" + id + "' này không?")) {
            const xhttp = new XMLHttpRequest();
            xhttp.onload = function () {
                if (JSON.parse(xhttp.responseText))
                    window.location.href = "/ViduMonhoc/Index";
                else alert("Không xóa được môn học này!");
            }
            const url = "/ViduMonhoc/xoaMonhoc/" + id;
            xhttp.open("GET", url);
            xhttp.send();
        }
    }
</script>
```

5. JavaScript Ajax

❑ Code C# cho nút “Xóa”

```
public class ViduMonhocController : Controller
{
    private Models.qlhvhContext db = new Models.qlhvhContext();
    0 references
    public IActionResult Index()
    {
        return View(db.Monhocs.ToList());
    }
    0 references
    public IActionResult xoaMonhoc(string id)
    {
        try
        {
            Models.Monhoc a = db.Monhocs.Find(id);
            if (a == null) return Json(false);
            db.Monhocs.Remove(a);
            db.SaveChanges();
            return Json(true);
        }
        catch (Exception)
        {
            return Json(false);
        }
    }
}
```

5. JavaScript Ajax

- ❑ Ví dụ:
- Nhấn nút
“Thêm môn
học” sẽ hiển
thị Form
thêm môn
học như sau:

Form thêm môn học

Trở về

Mã môn học

Tên môn học

Số tiết

Thêm

5. JavaScript Ajax

❑ Code
HTML
hiển
thị
Form
thêm
môn
học

```
<h3>Form thêm môn học</h3>
<div>
  <a asp-action="Index" class="btn btn-primary">Trở về</a>
</div>
<hr />
<div class="row">
  <div class="col-md-12">
    <form id="formThemMonhoc">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <div class="form-group">
        <label asp-for="msmh" class="control-label"></label>
        <input asp-for="msmh" class="form-control" />
        <span asp-validation-for="msmh" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="tenmh" class="control-label"></label>
        <input asp-for="tenmh" class="form-control" />
        <span asp-validation-for="tenmh" class="text-danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="sotiet" class="control-label"></label>
        <input asp-for="sotiet" class="form-control" />
        <span asp-validation-for="sotiet" class="text-danger"></span>
      </div>
      <div class="form-group">
        <input type="button" value="Thêm" class="btn btn-primary"
          onclick="themMonhoc_Click()" />
      </div>
    </form>
  </div>
</div>
```

5. JavaScript Ajax

❑ Code javascript cho nút “Thêm”

```
<script type="text/javascript">
    function themMonhoc_Click() {
        const mamh = document.forms["formThemMonhoc"]["msmh"].value;
        const tenmon = document.forms["formThemMonhoc"]["tenmh"].value;
        const st = document.forms["formThemMonhoc"]["sotiet"].value;
        let mh = { msmh: mamh, tenmh: tenmon, sotiet: st };
        const xhttp = new XMLHttpRequest();
        xhttp.onload = function () {
            if (JSON.parse(xhttp.responseText))
                window.location.href = "/ViduMonhoc/Index";
            else alert("Không thêm được môn học này!");
        }
        const url = "/ViduMonhoc/themMonhoc";
        xhttp.open("POST", url);
        xhttp.setRequestHeader("Content-Type", "application/json");
        xhttp.send(JSON.stringify(mh));
    }
</script>
```

5. JavaScript Ajax

- ❑ Code C#
cho nút
“Thêm”

```
public IActionResult formThemMonhoc()  
{  
    return View();  
}  
[HttpPost]  
0 references  
public IActionResult themMonhoc([FromBody] CMonhoc mh)  
{  
    try  
    {  
        Monhoc x = new Monhoc  
        {  
            Msmh=mh.msmh,  
            Tenmh=mh.tenmh,  
            Sotiet=mh.sotiet  
        };  
        db.Monhocs.Add(x);  
        db.SaveChanges();  
        return Json(true);  
    }  
    catch (Exception)  
    {  
        return Json(false);  
    }  
}
```

Tài liệu tham khảo

- 1) [ASP.NET MVC 4 In Action](#), Jeffrey Palermo - Jimmy Bogard - Eric Hexter - Matthew Hinze - Jeremy Skinner, 2012, Manning.
- 2) [Pro ASP.NET MVC 5](#), Adam Freeman, 2013, Apress.
- 3) <https://www.tutorialsteacher.com/mvc/>
- 4) <https://www.w3schools.com/html/>