**Tính chât Lập trình hướng đối tượng (Object Oriented Programming)**

được xem như kỹ thuật lập trình cho phép Developer tạo ra các đối tượng trong code trừu tượng hóa những đối tượng thực tế trong cuộc sống.

có 4 tính chất

tính đóng gói ( encapsulation ) : không cho phép ng dùng tác động trực tiếp đến dữ liệu bên trong object ( sự vật ) , mà phải thông qua các method( phương thức) nó cung cấp để bảo toàn toàn vẹn của đối tượng

tính kế thừa ( inheratance ) : cho phép xây dựng lớp mới trên những định nghĩa có sẵn, lớp cha sẽ chia sẽ dữ liệu cho các lớp con mà k cần định nghĩa lại, có thể mở rộng các thành phần kế thừa ở lớp con, tránh việc lặp lại code , dư thwuaf khi xử lý cviec tương tự

tính đa hình ( polymorphism ) : là hiện tượng các thuộc các lớp khác nhau có thể hiểu cùng 1 thông điệp theo các cách khác nhau

ví dụ: gà, chó ,mèo được xem là động vật nhưng khi bảo nó kêu thì mỗi con có tiếng khác nhau, 3 con vật là đối tượng khi bảo nó kêu thì đó là thông điệp chung nhưng cách biểu hiện khác nhau

tính trừu tượng ( abstraction ) : chỉ tập chung cốt lỗi của đối tượng, bỏ qua những thứ ko lquan. Chọn ra các phương thức và thuộc tính giúp ích trong quá trình code

khác nhau giữa abstract và interface : điều được sử dụng để điịnh nghĩa các kn trựu tượng và tạo khả năng tái sử dụng mã nguồn.

khác nhau:

**IEnumerable và ICollection khác nhau như nào**

`IEnumerable` và `ICollection` là hai interface trong .NET Framework để làm việc với các tập hợp dữ liệu.

A white text on a white background

Description automatically generated with low confidence

IEnumerable<int> numbers = **new** List<int> { 1, 2, 3, 4, 5 };

**foreach** (**var** number **in** numbers)

{

Console.WriteLine(number);

}

A screenshot of a computer code

Description automatically generated with low confidence

**Có bao nhiêu bộ từ truy cập và nó khác nhau như nào**

có hai khái niệm quan trọng liên quan đến việc xử lý và định tuyến các yêu cầu từ trình duyệt web. Đó là "RouteData" và "HttpContext".

1. RouteData: Là một bộ từ truy cập đại diện cho các đối số và thông tin đường dẫn của URL hiện tại được định tuyến đến một hành động trong ứng dụng MVC. Nó chứa thông tin như tên controller, tên action, và các đối số khác.
2. HttpContext: Là một bộ từ truy cập cung cấp thông tin về yêu cầu hiện tại và các đối tượng cần thiết liên quan đến việc xử lý yêu cầu. Nó chứa thông tin về các đối tượng như HttpRequest và HttpResponse.

## **Làm để để thiết lập Session trong Asp.net Core?**

Middleware cho Session được cung cấp bởi gói **Microsoft.AspNetCore.Session**. Để sử dụng session trong ứng dụng Asp.net Core, chúng ta cần thêm gói này tới file **.csproj** và thêm Session Middleware tới Asp.net Core Request pipeline. A screenshot of a computer program

Description automatically generated

## 2.**Startup class trong ASP.NET Core là gì?**

Startup class là điểm vào của ứng dụng APS.NET Core. Mỗi ứng dụng .NET Core phải có class này. Nó không bắt buộc phải có tên class là “Startup”, tên class có thể là bất kỳ tên gì, chúng ta có thể cấu hình startup class trong Program class.

## 3.**Tag-helper trong Asp.net Core là gì?**

Nó là một tính năng được cung cấp bởi **Razor view-engine** cho phép chúng ta viết mã server-side để tạo và hiển thị các phần tử HTML trong view (Razor). Tag-helper là các lớp C# thường được dùng để tạo view bằng cách thêm các phần tử HTML. Chức năng của tag-helper rất giống với HTML-helper của Asp.net MVC.

A screenshot of a computer program

Description automatically generated

## 5.**ASP.NET Core cung cấp các tính năng nào?**

**ASP.NET Core cung cấp các tính năng chính như sau:**

1. Hỗ trợ Dependency Injection.
2. Hỗ trợ cho Logging framework và có thể được mở rộng.
3. Giới thiệu web server mới: Kestrel, nó nhanh và hỗ trợ đa nền tảng. Vì vậy, một ứng dụng web có thể chạy mà không cần IIS, Apache và Nginx.
4. Hỗ trợ nhiều cách Hosting.
5. Hỗ trợ hướng module, vì vậy các developer cần import các module mình cần vào trong ứng dụng. Tuy nhiên, .NET Core framework cũng cung cấp Meta package trong các thư viện.
6. Command-line hỗ trợ tao, build và chạy ứng dụng.
7. Không có file web.config. Chúng ta có thể lưu trữ các thông tin cấu hình vào file appsettings.json.
8. Không có file Global.asax. Chúng ta có thể đăng ký và sử dụng các services vào startup class.
9. Hỗ trợ tốt cho lập trình bất đồng bộ.
10. Hỗ trợ WebSocket và SignalR.
11. Cung cấp bảo vệ chống CSRF ( Cross-Site Request Forgery )

## 6.**Công dụng của hàm “Map” trong IApplicationBuilder trong việc thêm Middleware vào Asp.net Core Pipeline là gì?**

Nó thường dùng để chia nhánh Pipeline. Nó phân nhánh Asp.net Core Pipeline dựa vào việc khớp đường dẫn Request. Nếu đường dẫn Request bắt đầu với đường dẫn đã cho, Middleware trên nhánh đó sẽ được thực thi.

A screenshot of a computer code

Description automatically generated

## 7.**Middleware là gì?**

Middleware là thành phần của phần mềm đóng vai trò tác động vào request pipeline (luồng request) để xử lý chúng và tạo ra response phản hồi lại client. Mỗi một tiến trình middleware thao tác với các request nhận được từ middleware trước nó. Nó cũng có thể quyết định gọi middleware tiếp theo trong pipeline hoặc trả về response cho middleware ngay trước nó. (ngắt pipeline).

## 8.**Làm sao chúng ta có thể thực hiện model-binding tự động trong Razor Pages?**

Razor Pages cung cấp tùy chọn để liên kết thuộc tính một cách tự động khi truyền dữ liệu sử dụng thuộc tính BindProperty. Mặc định nó chỉ liên kết các thuộc tính với các method không phải GET. Chúng ta cần gán thuộc tính SupportsGet thành “true” để liên kết 1 thuộc tính với method GET.

A screenshot of a computer code

Description automatically generated

## 9.**Ứng dụng ASP.NET Core có thể làm việc với framework .NET 4.x full không?**

Có, thông qua thư viện chuẩn .NET

## 10.**Làm sao để disable tag-helper ở cấp phần tử?**

Chúng ta có thể disable tag-helper ở cấp phần tử bằng cách sử dụng ký tự “**!**”. Ký tự này phải được dùng ở tag đóng và mở.

A white rectangular sign with black text

Description automatically generated

## 11.**Hàm ConfigureServices trong startup class dùng để làm gì?**

Đây là một hàm tùy chọn của startup class. Nó được dùng để cấu hình các sevice được sử dụng trong ứng dụng. Hàm này được gọi đầu tiên khi ứng dụng được request lần đầu. Sử dụng hàm này, chúng ta có thể thêm các service vào DI-container, như vậy các service sẽ có sẵn như 1 dependency trong constructor của controller.

## 12.**Các file JSON nào có sẵn trong Asp.net Core?**

* **global.json**
* **launchsettings.json**
* **appsettings.json**
* **bundleconfig.json**
* **bower.json**
* **package.json**

## 13.**Hàm Configure trong startup class dùng để làm gì?**

Nó định nghĩa cách mà ứng dụng sẽ hồi đáp tới mỗi HTTP request. Chúng ta có thể cấu hình Request Pipeline bằng cách cấu hình Middleware. Nó chấp nhận IApplicationBuilder như là 1 parameter và nó cũng có thêm 2 parameter tùy chọn: IHostingEnvironment và ILoggerFactory. Sử dụng hàm này, chúng ta có thể cấu hình Middleware có sẵn như Routing, authentication, session, … cũng như các Third-party Middleware.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## 17.**Asp.net core là gì?**

ASP.NET Core không phải là một phiên bản nâng cấp của ASP.NET. ASP.NET Core hoàn toàn được tạo ra để làm việc với .NET Core framework.

Nó nhanh hơn, dễ cấu hình chi tiết hơn, theo hướng module, dễ mở rộng và hỗ trợ đa nền tảng. Nó có thể làm với cả .NET Core và .NET framework thông qua .NET standard framework.

ASP.NET Core là thích hợp nhất để phát triển các ứng dụng cloud-based như web, mobile, IoT.

## 18.**Trình bày về routing trong Asp.net Core?**

Routing là chức năng ánh xạ các request tới bộ xử lý định tuyến. Route có thể có nhiều giá trị ( được trích xuất ra từ URL) thường được dùng để xử lý request. Tất cả các Route được đăng ký khi ứng dụng bắt đầu. Asp.net Core hỗ trợ 2 loại Routing:

* Định tuyến thông thường
* Định tuyến thuộc tính (Attribute routing)

Routing sử dụng các Route trong việc ánh xạ các Request với bộ xử lý tuyến và Generate URL được dùng trong việc hồi đáp request. Hầu hết ứng dụng có 1 bộ tập hợp Route và bộ tập hợp này được dùng trong việc xử lý request. Phương thức RouteAsync thường được dùng để ánh xạ request (ứng với URL) có sẵn trong bộ tập hợp Route.

## 19.**Những ưu điểm nào của ASP.NET Core so với ASP.NET ?**

1. Hỗ trợ đa nền tảng, có thể chạy được trên Windows, Linux, Mac.
2. Không có dependency trong cài đặt framework bởi vì tất cả các dependency bắt buộc đều được gửi cùng ứng dụng.
3. ASP.NET Core có thể xử lý nhiều request hơn APS.NET.
4. Nhiều tùy chọn deloy web có sẵn trong ASP.NET Core.

**20. Làm thế nào để thực hiện xác thực người dùng trong ASP.NET Core?**

Windows Authentication, và OAuth để xác thực người dùng. Bạn có thể cấu hình xác thực trong file cấu hình (config file) hoặc trong mã nguồn của ứng dụng.

Ví dụ, để sử dụng Forms Authentication, bạn có thể thực hiện các bước sau:

1. Cấu hình xác thực trong file cấu hình (config file) bằng cách chỉ định provider xác thực và các thông số cần thiết như cookie name, timeout, và đường dẫn đến trang đăng nhập.

2. Tạo trang đăng nhập và trang đăng ký người dùng.

3. Trong mã nguồn của các controller hoặc các action cần xác thực, bạn có thể sử dụng attribute [Authorize] để yêu cầu xác thực trước khi truy cập vào action hoặc controller đó.

**21. Giải thích cách sử dụng Dependency Injection trong ASP.NET Core.**

về Dependency Injection trong ASP.NET Core, nó là một cơ chế quản lý và cung cấp các đối tượng (dependencies) cho các thành phần trong ứng dụng. Dependency Injection giúp giảm sự phụ thuộc giữa các thành phần và tạo điều kiện cho việc kiểm thử và tái sử dụng mã dễ dàng hơn.

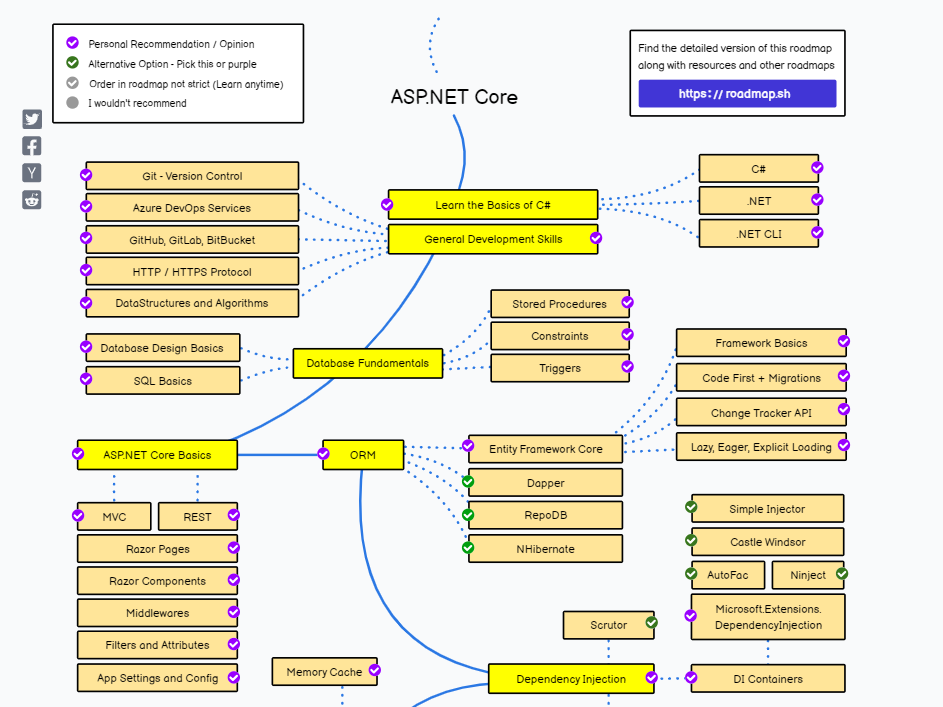
ASP.NET Core cung cấp một hệ thống Dependency Injection tích hợp sẵn.

Bạn có thể đăng ký các dịch vụ (services) trong container DI của ASP.NET Core bằng cách sử dụng phương thức ConfigureServices trong file Startup.cs. Sau đó, bạn có thể tiêm (inject) các dịch vụ này vào các controller, middleware, hoặc các thành phần khác bằng cách sử dụng constructor injection hoặc property injection.

Dependency Injection giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng bằng cách cho phép thay thế dễ dàng các implementation của các dịch vụ và tái sử dụng các thành phần đã được đăng ký.

**Trong ASP.NET Core, View Component** là một thành phần cung cấp khả năng tái sử dụng và phân chia logic giao diện người dùng. Nó cho phép bạn phân chia giao diện thành các phần nhỏ hơn và đóng gói chúng thành các thành phần độc lập có thể được sử dụng lại trong nhiều trang hoặc layout khác nhau.

View Component tương tự như Partial View, nhưng có tính linh hoạt cao hơn và có thể điều khiển logic phía sau. Nó thường được sử dụng để hiển thị các phần giao diện người dùng như menu điều hướng, danh sách bài viết mới nhất, widget thông tin, v.v.



A screenshot of a computer

Description automatically generated

**Trong ASP.NET Core, async/await** là một cách để xử lý các tác vụ không đồng bộ một cách dễ dàng và hiệu quả.

Khi một phương thức được đánh dấu là async, nghĩa là nó có khả năng thực hiện các tác vụ không đồng bộ. Bằng cách sử dụng từ khóa await, chúng ta có thể đợi cho đến khi một tác vụ không đồng bộ hoàn thành mà không phải chặn luồng chính của ứng dụng. Trong khi đợi, luồng chính có thể tiếp tục thực hiện các công việc khác mà không bị tắc nghẽn.Sử dụng async/await trong ASP.NET Core giúp tăng cường hiệu suất và khả năng mở rộng của ứng dụng bằng cách cho phép các tác vụ không đồng bộ thực hiện mà không làm đứt quãng luồng chính và giữ cho ứng dụng phản hồi nhanh chóng và mượt mà.

**Để truy vấn cơ sở dữ liệu trong ASP.NET Core 5**, bạn có thể sử dụng Entity Framework Core (EF Core). EF Core là một ORM (Object-Relational Mapping) framework cho phép bạn làm việc với cơ sở dữ liệu bằng cách sử dụng các đối tượng và truy vấn LINQ.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated