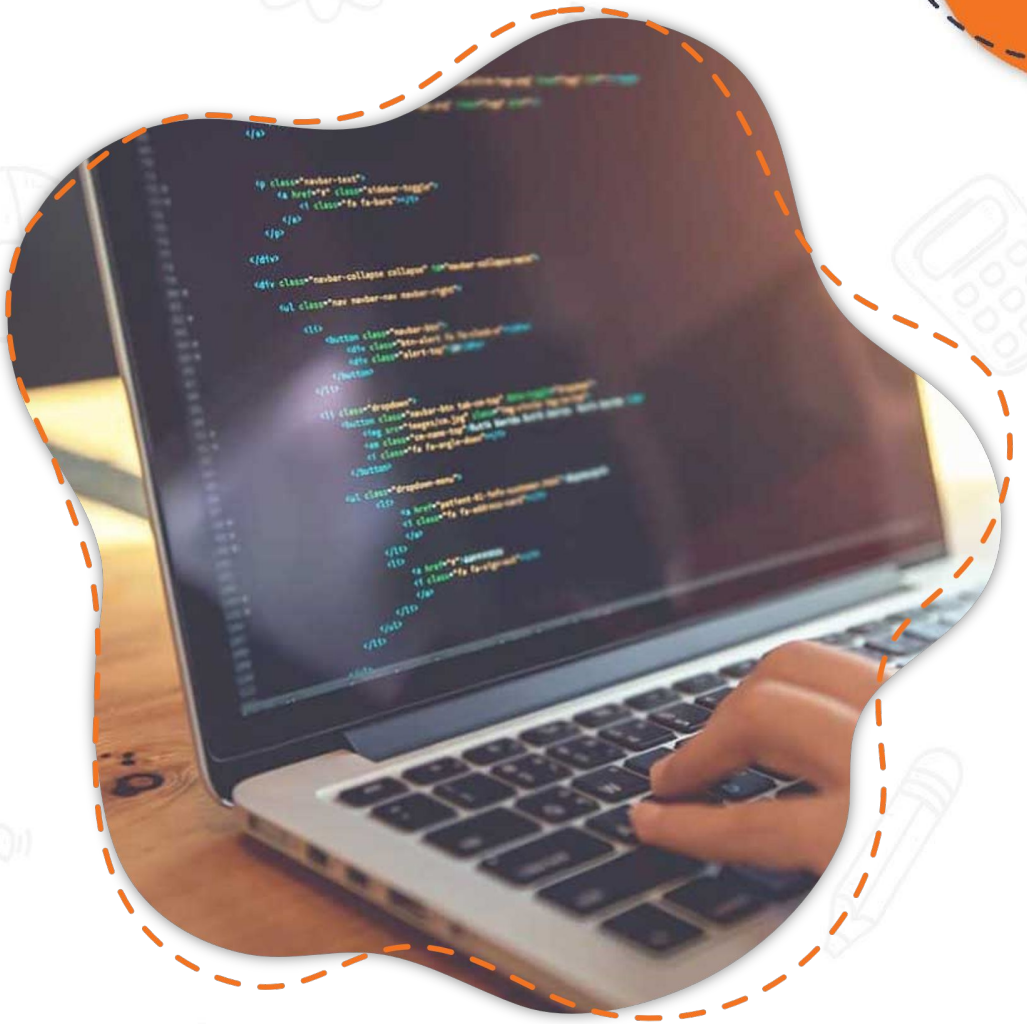
T3H
IT-Institute

VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

C# Backend



www.t3h.edu.vn

NỘI DUNG

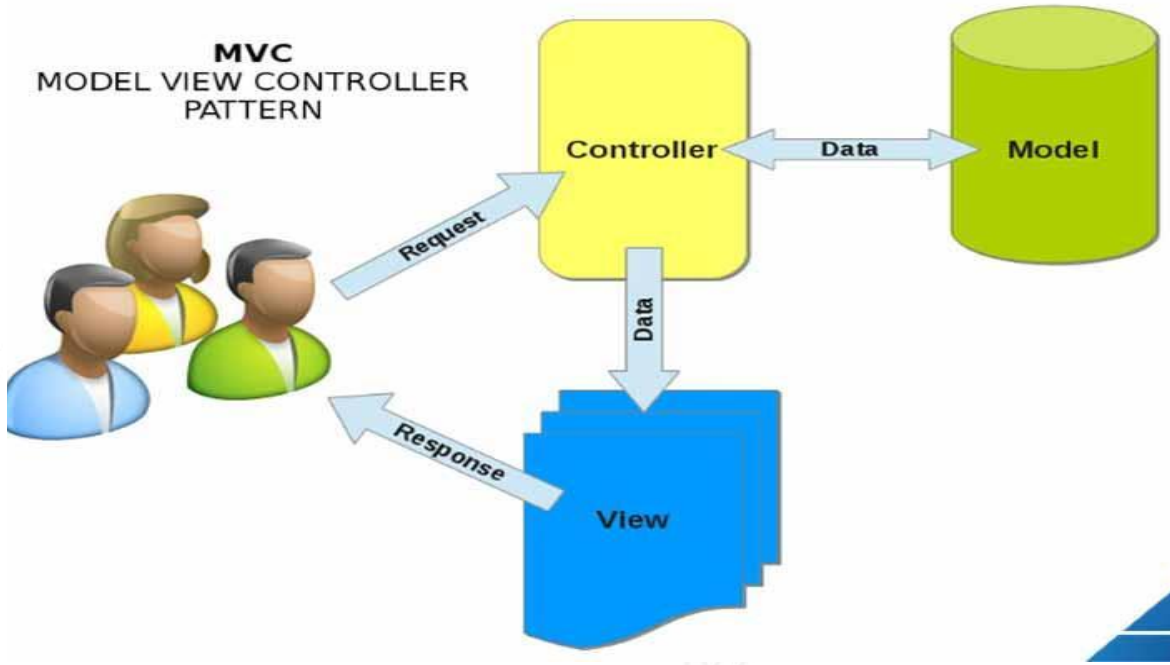
01 Mô hình MVC

Trong ASP.NET Core, mô hình MVC (Model-View-Controller) là một trong những phương pháp phổ biến để xây dựng ứng dụng web. Mô hình MVC chia ứng dụng thành ba phần chính:

Model: Là phần của ứng dụng chứa logic kinh doanh và dữ liệu. Model thường đại diện cho dữ liệu từ cơ sở dữ liệu hoặc các dịch vụ khác.

View: Là phần giao diện người dùng của ứng dụng. View là nơi hiển thị dữ liệu cho người dùng và lắng nghe các sự kiện từ người dùng.

Controller: Là phần chịu trách nhiệm xử lý các yêu cầu từ người dùng và tương tác với Model để lấy dữ liệu cần thiết. Sau đó, Controller sẽ chuyển dữ liệu này cho View để hiển thị cho người dùng.



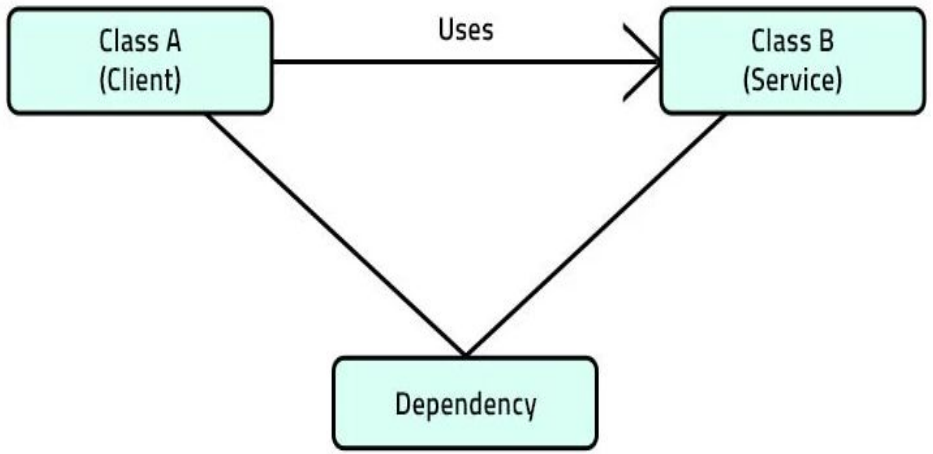
01 Dependency Injection

Nguyên tắc số 5 trong solid: Các modun bậc cao không nên phụ thuộc trực tiếp vào modun bậc thấp mà nên phụ thuộc thông qua interface

Dependency Injection (DI) là một design partern nhằm mục đích đảm bảo nguyên tắc này

Dependency Injection được sử dụng để giảm Mối Liên Kết Chặt Chẽ giữa các thành phần phần mềm. Kết quả là, chúng ta có thể dễ dàng quản lý các thay đổi trong ứng dụng của chúng ta. Trong trường hợp này, nếu chúng ta thay đổi một thành phần, thì nó sẽ không ảnh hưởng đến các thành phần khác.

Trong ASP.NET Core, Dependency Injection (DI) là một phần quan trọng của cấu trúc ứng dụng và là một tính năng được tích hợp sẵn. ASP.NET Core cung cấp một DI container tích hợp sẵn mà bạn có thể sử dụng để đăng ký và giải quyết các phụ thuộc của ứng dụng.



01 Dependency Injection

Đăng ký dịch vụ (Service Registration): Trước tiên, bạn cần đăng ký các dịch vụ (services) mà ứng dụng của bạn cần sử dụng trong DI container. Bạn thường thực hiện việc này trong phương thức ConfigureServices của lớp Startup.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddTransient<IMyService, MyService>();
    services.AddScoped<IOtherService, OtherService>();
    services.AddSingleton<IAnotherService, AnotherService>();
}
```

01 Dependency Injection

Tiêm (Inject) dịch vụ vào các thành phần của ứng dụng: Sau khi các dịch vụ đã được đăng ký, bạn có thể tiêm chúng vào các thành phần khác nhau của ứng dụng như controllers, middleware, hoặc các thành phần khác.

```
public class MyController : Controller
{
    private readonly IMyService _myService;

    public MyController(IMyService myService)
    {
        _myService = myService;
    }

    // Sử dụng _myService ở đây
}
```

03 Các mode làm việc của dependency injection

Transient Lifetime: Các dịch vụ được đăng ký với Transient Lifetime sẽ được tạo ra mỗi khi chúng được yêu cầu. Mỗi yêu cầu sẽ tạo ra một instance mới của dịch vụ.

```
services.AddTransient<IService, MyService>();
```

Scoped Lifetime: Các dịch vụ được đăng ký với Scoped Lifetime sẽ tồn tại trong phạm vi của một request HTTP. Mỗi request sẽ có một instance mới của dịch vụ.

```
services.AddScoped<IService, MyService>();
```

Singleton Lifetime: Các dịch vụ được đăng ký với Singleton Lifetime sẽ tồn tại trong suốt vòng đời của ứng dụng. Một instance duy nhất của dịch vụ được tạo ra và sử dụng lại cho tất cả các yêu cầu.

```
services.AddSingleton<IService, MyService>();
```