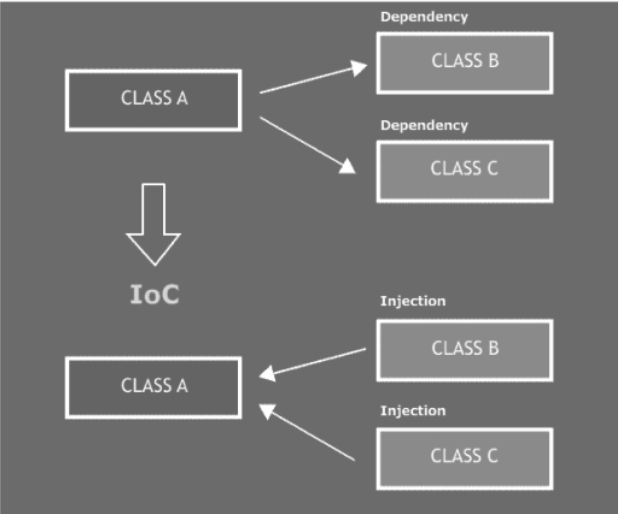
# Dependency Injection in ASP.NET Core:

## Giới thiệu

* **Dependency Injection (DI)** là một mẫu thiết kế phần mềm được hỗ trợ sẵn trong ASP.NET Core. Đây là một kỹ thuật giúp đạt được Inversion of Control (IoC) giữa các lớp và các phần phụ thuộc của chúng.[1]
* **Inversion of Control** là một nguyên lý thiết kế trong công nghệ phần mềm trong đó các thành phần mà nó dựa vào để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hướng thủ thục truyền thống.[2]

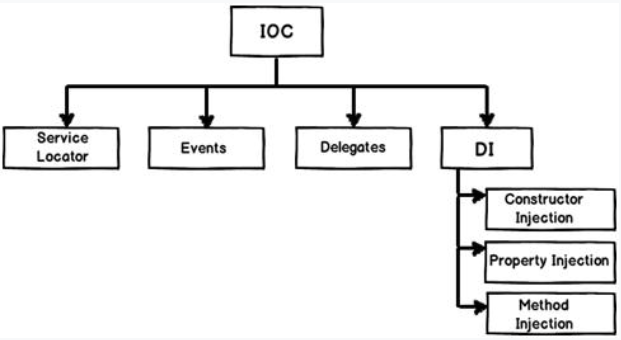


Mô phỏng không có IoC và mô hình IoC [2]

- Như ở hình trên, Class A ở hai trường hợp, áp dụng IoC và không. Lớp này làm việc phụ thuộc vào hai Class B và Class C (B và C gọi là các dependency). [2]

+ Ở mô hình không IoC, Class A khi cần chủ động tạo ra đối tượng từ lớp Class B và Class C (nó nắm quyền khởi tạo và điều khiển). [2]

+ Với mô hình IoC thì Class A không tự khởi tạo và cũng không chịu trách nhiệm quản lý Class B, Class C. Nó nhận được hai dependency này từ bên ngoài thông qua một cơ chế nào đó (như bằng setter, bằng tham số từ phương thức khởi tạo của Class A, bằng gán thuộc tính ...). [2]



Mô hình triển khai nguyên lý IoC[2]

* Việc triển khai nguyên lý IoC thường được thực hiện bởi các Framework theo từng loại ngôn ngữ lập trình, các mô hình lập trình (pattern) triển khai từ IoC như:

+ Service Locator

+ Events

+ DI (Dependency Injection) [2]

* Ở đề tài này, ta sẽ chủ yếu sử dụng Dependency Injection, trong kỹ thuật này một đối tượng cung cấp những phụ thuộc (dependency - là đối tượng, dịch vụ, chức năng) của đối tượng khác. Injection - Bơm vào (tiêm vào) ám chỉ một phụ thuộc (đối tượng, dịch vụ) đưa vào đối tượng để đối tượng sử dụng nó. [2]
* Các ba kiểu để đưa một dependency vào đối tượng cần sử dụng nó:

+ **Inject thông qua phương thức khởi tạo**: cung cấp các Dependency cho đối tượng thông qua phương thức khởi tạo (constructor). Đây là cách được thư viện .NET hỗ trợ sẵn, mang tính linh hoạt, mềm dẻo, dễ xây dựng và được sử dụng chính trong ứng dụng ASP.NET Core của đồ án.

+ **Inject thông qua setter**.

+ **Inject thông qua các Interface**[2]

* **Lớp ServiceCollection**: ServiceCollection là lớp triển khai giao diện IServiceCollection nó có chức năng quản lý các dịch vụ (đăng ký dịch vụ - tạo dịch vụ - tự động inject - và các dependency của dịch vụ ...). ServiceCollection là trung tâm của kỹ thuật DI, nó là thành phần rất quan trọng trong ứng dụng ASP.NET Core.
* **ServiceLifetime**: Mỗi dịch vụ (lớp) khi đăng ký vào ServiceCollection thì có một đối tượng ServiceDescriptor chứa thông tin về dịch vụ đó, căn cứ vào ServiceDescriptor để ServiceCollection khởi tạo dịch vụ khi cần. Trong ServiceDescriptor có thuộc tính Lifetime để xác định dịch vụ tạo ra tồn tại trog bao lâu. Lifetime có kiểu ServiceLifetime (kiểu enum) có các giá trị cụ thể:

+ **Scoped**: Một bản thực thi (instance) của dịch vụ (Class) được tạo ra cho mỗi phạm vi, tức tồn tại cùng với sự tồn tại của một đối tượng kiểu ServiceScope (đối tượng này tạo bằng cách gọi ServiceProvider.CreateScope, đối tượng này hủy thì dịch vụ cũng bị hủy).

AddScoped<ServiceType, ImplementationType>()

+ **Singleton:** Duy nhất một phiên bản thực thi (instance of class) (dịch vụ) được tạo ra cho hết vòng đời của ServiceProvider. Phương thức

AddSingleton<ServiceType, ImplementationType>().

+ **Transient:** Một phiên bản của dịch vụ được tạo mỗi khi được yêu cầu.

AddTransient<ServiceType, ImplementationType>()

[1] <https://docs.microsoft.com/vi-vn/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0>

[2] <https://xuanthulab.net/nguyen-ly-lap-trinh-ioc-inversion-of-control-ioc.html#IoC>