

# Dependency Injection

Để quản lý khi nào tạo mới hay tái sử dụng lại các service

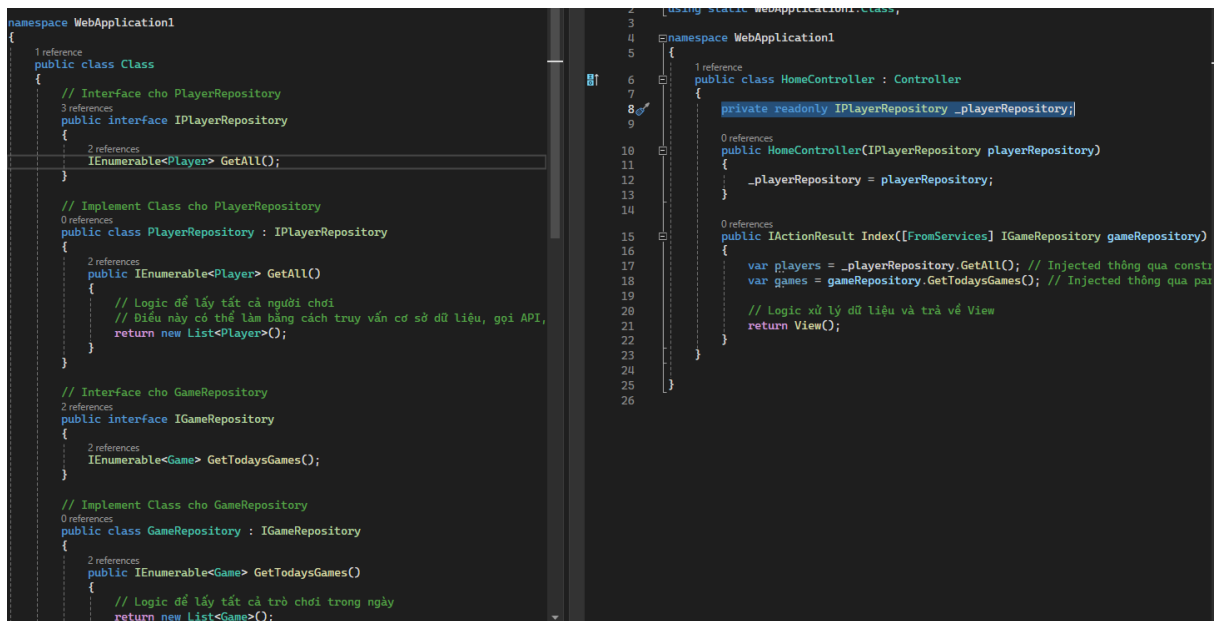
Có 3 cách triển khai

- Constructure Injection
- Setter Injection
- Interface Injection

Trong code trên bn sẽ thắc mắc tại sao lại phải ghi là :

**private readonly IPlayerRepository \_playerRepository;**

bởi vì tôi đang thực hiện **mô hình SOLID** cụ thể **Dependency Inversion**



```
namespace WebApplication1
{
    1 reference
    public class Class
    {
        // Interface cho PlayerRepository
        3 references
        public interface IPlayerRepository
        {
            2 references
            IEnumerable<Player> GetAll();
        }

        // Implement Class cho PlayerRepository
        0 references
        public class PlayerRepository : IPlayerRepository
        {
            2 references
            public IEnumerable<Player> GetAll()
            {
                // Logic để lấy tất cả người chơi
                // Điều này có thể làm bằng cách truy vấn cơ sở dữ liệu, gọi API,
                return new List<Player>();
            }
        }

        // Interface cho GameRepository
        2 references
        public interface IGameRepository
        {
            2 references
            IEnumerable<Game> GetTodaysGames();
        }

        // Implement Class cho GameRepository
        0 references
        public class GameRepository : IGameRepository
        {
            2 references
            public IEnumerable<Game> GetTodaysGames()
            {
                // Logic để lấy tất cả trò chơi trong ngày
                return new List<Game>();
            }
        }
    }
}

2
3
4
5
namespace WebApplication1
{
    1 reference
    public class HomeController : Controller
    {
        8
        private readonly IPlayerRepository _playerRepository;

        0 references
        public HomeController(IPlayerRepository playerRepository)
        {
            _playerRepository = playerRepository;
        }

        0 references
        public IActionResult Index([FromServices] IGameRepository gameRepository)
        {
            17
            var players = _playerRepository.GetAll(); // Injected thông qua const
            18
            var games = gameRepository.GetTodaysGames(); // Injected thông qua par
            20
            // Logic xử lý dữ liệu và trả về View
            21
            return View();
        }
    }
}
```

<https://tedu.com.vn/lap-trinh-aspnet-core/vong-doi-cua-dependency-injection-transient-singleton-va-scoped-257.html>

## Scoped

Nói dễ hiểu là Chỉ cần tạo mới Scope thì

2 đoàn A, B ,mỗi đoàn có 3 người thì đoàn A mỗi người sẽ sử dụng cốc **màu xanh** và khi uống cốc xanh xong sẽ đem đi vứt và người tiếp theo cũng vậy và đoàn B cũng vậy cốc **màu khác**

\*\*\*\*\* **Màu xanh , màu khác** ở đây là Instance

**AddScoped:** Tương ứng với việc nhân viên phục vụ sử dụng cùng một cái ly cho mỗi khách hàng trong suốt thời gian họ ngồi tại nhà hàng. Tuy nhiên, khi khách hàng đó rời đi và quay lại vào một thời điểm khác, nhân viên sẽ lấy một cái ly mới. Điều này có nghĩa là một instance của đối tượng được tạo ra và sử dụng cho mỗi "phạm vi" (scope), thường là trong một chu kỳ yêu cầu (request) hoặc một kết nối (connection).

BE :

Để Scoped để Service ,

FE Blazor ⇒ Session ,

## Transient

Transient Service luôn tạo mới mỗi lần request tạo một instance mới

```
// Tạo một instance của lớp Person
Person person1 = new Person("Alice");
```

2 đoàn A, B ,mỗi đoàn có 3 người thì đoàn A mỗi người sẽ sử dụng cốc và khi uống xong sẽ đem đi vứt và người tiếp theo cũng vậy

BE để ở Repository

## VD2 :

**AddTransient:** Tương ứng với việc nhân viên phục vụ lấy một cái ly mới mỗi khi có yêu cầu mới từ khách hàng. Điều này có nghĩa là một instance mới của đối tượng được tạo ra mỗi khi có yêu cầu đến dịch vụ đó.

## Singleton

một Service mới sẽ đc tạo ra khi ứng dụng web được tạo ra và sẽ sử dụng trong suốt quá trình web triển khai

**AddSingleton:** Tương ứng với việc nhân viên phục vụ luôn sử dụng cùng một cái ly cho tất cả khách hàng trong suốt thời gian ứng dụng chạy. Điều này có nghĩa là chỉ có một instance duy nhất của đối tượng được tạo ra và sử dụng cho tất cả các yêu cầu (requests).

Các ID tạo ra từ Singleton service luôn giống nhau và sẽ không thay đổi ngay cả khi bạn refresh ứng dụng.

## Singleton Service

localhost:50297

### Index

#### Transient Service

First Instance dd3f328c-3173-4479-bc53-bea6fef9a0c4  
Second Instance 101d57c8-e449-4029-b07a-1a2ba38e99c4

#### Scoped Service

First Instance 027a26e3-c5c3-4f76-bfb7-d1891ef76f9f  
Second Instance 027a26e3-c5c3-4f76-bfb7-d1891ef76f9f

#### Singleton Service

First Instance c347fe45-8674-4c87-b1af-ddb199924369  
Second Instance c347fe45-8674-4c87-b1af-ddb199924369

Always returns the new instance

Instance is created only once per request and shared across the request I.e. why you have same Ids generated  
New ids are generated, when you click on refresh button

Only one instance is created and shared across the application.  
Click on Refresh button, the ids will remain the same

## Dependency Inversion

- Các lớp cấp cao không nên phụ thuộc vào các lớp cấp thấp
- Các lớp cấp cao nên phụ thuộc vào các giao diện hoặc lớp trừu tượng, có thể được triển khai bởi các lớp cấp thấp