Dependency Injection Services Lifetime

Singleton

vs

Scoped

vs

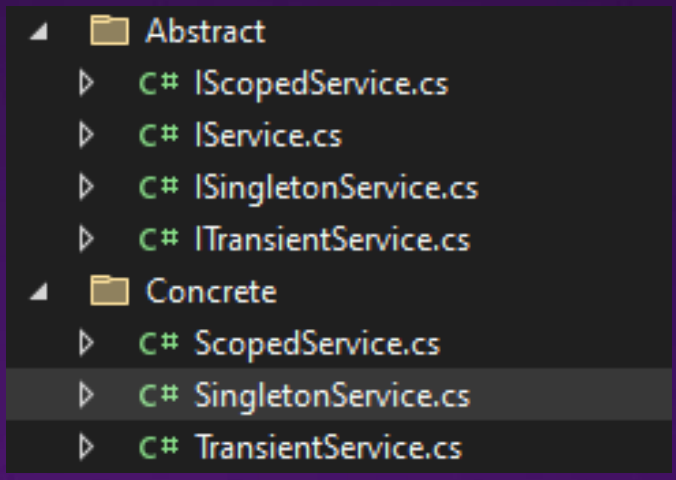
Transient

@djokic-stefan

Setup

We have created 3 services that will serve us for lifecycle testing. We called them the same names as the lifecycles respectively.

All 3 services do the same thing, that is, when creating a service, a random number is generated and placed in a variable.



Singleton Service

```
public class SingletonService : ISingletonService
{
    private int _randomNumber;
    public SingletonService()
    {
        Random rnd = new Random();
        _randomNumber = rnd.Next(1, 1000);
    }

    public int GetRandomNumber()
    {
        return _randomNumber;
    }
}
```

@djokic-stefan

Registering a service with a specific lifecycle.

```
services.AddTransient<ITransientService, TransientService>();
services.AddScoped<IScopedService, ScopedService>();
services.AddSingleton<ISingletonService, SingletonService>();
```

Home controller - DI services

```
public class HomeController : Controller
{
    private ISingletonService _singleton;
    private IScopedService _scoped;
    private ITransientService _transient;

    public HomeController(ISingletonService singleton, IScopedService scoped,
ITransientService transient)
{
     _singleton = singleton;  _scoped = scoped;  _transient = transient;
}
```

We have created a View for each of the lifecycles where we will test each separately.

```
public IActionResult Singleton()
{
    return View(_singleton.GetRandomNumber());
}

public IActionResult Scoped()
{
    return View(_scoped.GetRandomNumber());
}

public IActionResult Transient()
{
    return View(_transient.GetRandomNumber());
}
```

@djokic-stefan

```razor
@model int
@{
    ViewData["Title"] = "Singleton";
}
<div class="text-center">
    <h1 class="display-4">Singleton Random number: @Model</h1>

    <partial name="_Singleton" />
</div>
```

This is a view of each of the 3 services. We print the generated number within the lifecycle that was created and render a partial view for each of them. We do this in order to simulate accessing the same service multiple times.

Partial view _Singleton

```razor
@using ServicesLifetime.Abstract
@inject ISingletonService Singleton

<h3>Singleton from the Partial: @Singleton.GetRandomNumber()</h3>
```

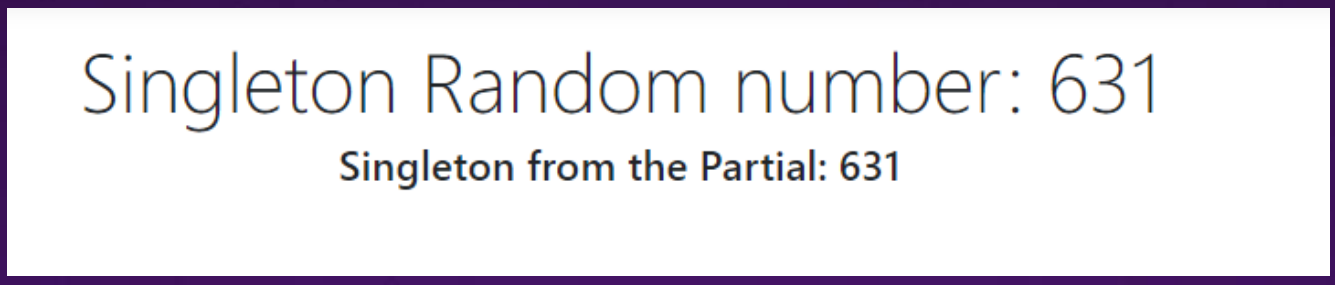
We inject the service in order to use it once again directly in the partial view.

RESULTS

@djokic-stefan

Singleton

/Home/Singleton HTML Page



As we can see, both numbers are exactly the same even though we call the function with 2 different places. The reason for this is that the Singleton is created only once and lasts until the application stops.

Lifetime

Created the first time they are requested (or when ConfigureServices is run if you specify an instance there) and then every subsequent request will use the same instance.

Cases

- Caching Services
- Global Configuration

@djokic-stefan

Scoped

/Home/Scoped HTML Page

Both numbers are also the same, but the difference is that with each refresh of the page, we get a different number. The reason for this is that Scoped is initialized for each Request (rendering the page and the partial is the same request)

Lifetime

Created once per request.

Cases

- Persisting state per request

@djokic-stefan

Transient

/Home/Transient HTML Page

The numbers obtained are different. Whenever we extract a service from DependencyInjection that instance is fresh and new.

Lifetime

Created each time they are requested. This lifetime works best for lightweight, stateless services.

Cases

- **File Access**
- **Database Access**

@djokic-stefan