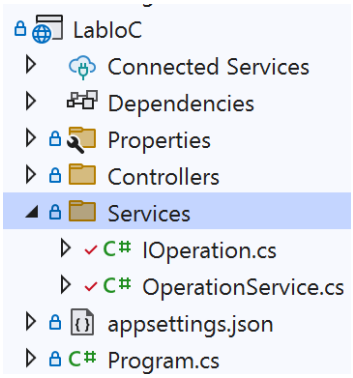
Modul 02: ASP.NET Core - Logging / Dependency Injections

Ziel:	*Verwende Lebenszyklen für Objekten innerhalb ASP.NET Core und Logging mit Serilog und Verwendung von FileSinks*
Tools:	*Visual Studio 2022*
Dauer:	*~30min*

1 Projekt anlegen

Das Beispiel demonstriert eine ASP.NET Core Funktion und kann daher mit allen ASP.NET Core-Templates umgesetzt werden.



1.) Erstelle im ASP.NET Core Projekt das Verzeichnis Services

2.) Erstelle innerhalb des Verzeichnis Services ein Interface mit dem Namen IOpertion.cs und füge folgende Interfaces ein:

```
public interface IOperation
{
    string OperationId { get; }
}

public interface IOperationTransient : IOperation { }

public interface IOperationScoped : IOperation { }

public interface IOperationSingleton : IOperation { }
```

3.) Erstelle innerhalb des Verzeichnis Services eine Klasse mit dem Namen Opertion.cs und füge folgende Klasse ein:

```
public class OperationService :
    IOperationScoped, IOperationSingleton, IOperationTransient
{
    public string OperationId { get; } = Guid.NewGuid().ToString()[^4..];
}
```

2 Dependencies in Program.cs registrieren

Registriere die Interface:

- IOperationSingleton als Singleton Lifetime
- IOperationSclope als Sclope Lifetime
- IOperationTransient als Transient Lifetime

in der Program.cs

```
// Asp.Net sucht nach allen Klassen im Verzeichnis Controllers
// welche mit dem Suffix Controller aufhoeren
builder.Services.AddControllers();

var app = builder.Build();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Test}");

app.Run();
```

3 TestController anlegen in Program.cs registrieren

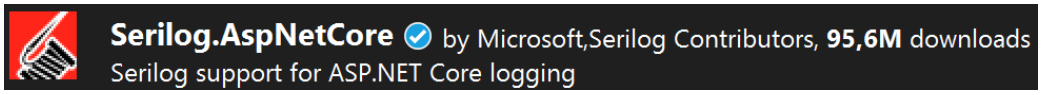
Der TestController soll jeweils via Konstruktor und Methode die Services injizieren. Pro Aufruf soll jedes Interface die Property OperationId in den Default-Logger schreiben.

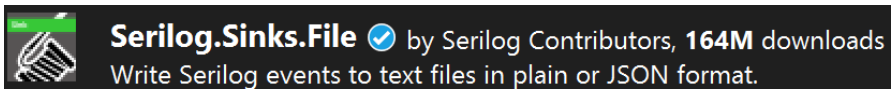
```
[HttpGet]
[Route("/")]
public IActionResult Index(
    [FromServices] IOperationScoped scoped,
    [FromServices] IOperationTransient transient,
    [FromServices] IOperationSingleton singleton
    )
{
    // ... Logger Ausgaben hier ...

    return Ok("Check the logs...");
}
```

4 Serilog installieren

4.1 Mit NuGet Package Manger folgende Packages installieren





4.2 Serilog in Program.cs konfigurieren

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.File("app.log", rollingInterval: RollingInterval.Day)
    .ReadFrom.Configuration(builder.Configuration)
    .CreateLogger();
builder.Host.UseSerilog();

var app = builder.Build();

Log.Information("Starting up");
try
{
    app.MapGet("/", () => "Hello World!");
    app.MapGet("/error", () =>
    {
        throw new InvalidOperationException("Hello Error");
    });
    app.Run();
}
catch (Exception ex)
{
    Log.Error(ex, "Stopped program because of exception");
}
finally
{
    Log.Information("Shutting down");
    Log.CloseAndFlush();
}
```