Data-driven systems

Dependency Injection in ASP.NET Core environment

Dependency Injection in ASP.NET Core environment

- The material is accessible here (there are no Powerpoint slides):

  https://bmeviauac01.github.io/datadriven/en/lecture-notes/di

- The next slides are just a bit of additional illustrations

Register (type mappings) at startup

INotificationService → NotificationService (as transient)
Ilogger → Logger (as singleton)
IEMailSender → IMailSender (as transient)
IContactRepository → ContactRepository (as scoped)

Register (the above) TYPE mappings at application startup in
the Dependency Injection (DI) container: AddSingleton, AddTransient, AddScoped

DI/IoC container

IServiceCollection

Note: type mappings are typically interface-to-concrete-type mappings

Resolve step

To resolve INotificationService: we call GetService<INotificationService>() on the container, which takes care of the creation/AUTOWIRING of a new NotificationService object

ISeviceProvider

DI/IoC container
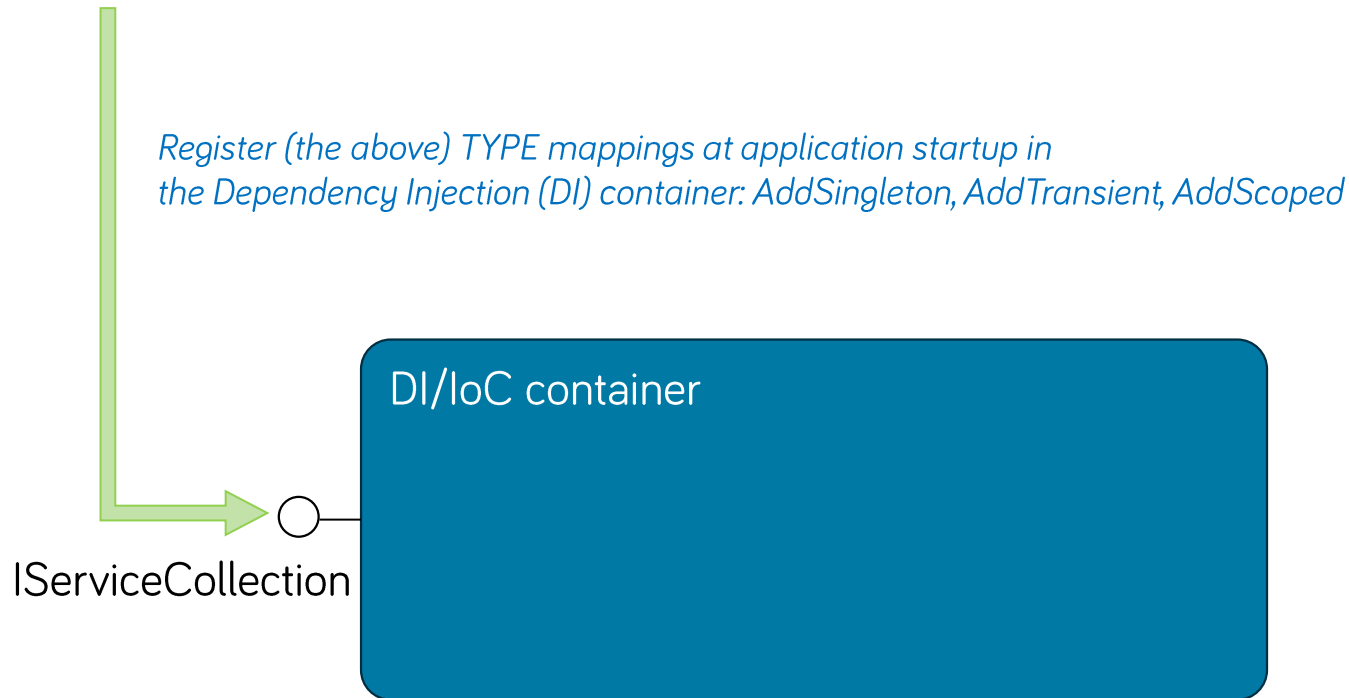
INotificationService → NotificationService (as transient)
Ilogger → Logger (as singleton)
IEMailSender → IMailSender (as transient)
IContactRepository → ContactRepository (as scoped)

Creates a NotificationService object with all of its registered dependencies (calls NotificationService constructor)

The parameters of the NotificationService constructor are also automatically resolved by the DI container (and the ctor parameters of these parameters recursively), an entire object graph is created: this process is calles AUTOWIRING

```
NotificationService(
    ILogger logger,
    IEMailSender emailSender,
    IContactRepository contactRepository)

EMailSender(
        ILogger logger,
        ...)
```