Contents

[Generic host (HostBuilder) //TODO 1](#_Toc158809944)

[Configuration //TODO 1](#_Toc158809945)

[Dependency injection //TODO 1](#_Toc158809946)

[Options pattern 2](#_Toc158809947)

[Logging //TODO 2](#_Toc158809948)

[Hosted service // TODO 3](#_Toc158809949)

[ASP .NET Core 3](#_Toc158809950)

[ASP .NET Core vs ASP .NET Core MVC 3](#_Toc158809951)

[Kestrel vs IIS 3](#_Toc158809952)

Generic host (HostBuilder) //TODO

Просто все отсюда: <https://docs.microsoft.com/en-us/dotnet/core/extensions/generic-host>

Configuration //TODO

<https://docs.microsoft.com/en-us/dotnet/core/extensions/configuration>

Appsettings - <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-6.0>

Dependency injection //TODO

**Construction Injection**

Зависимости нужны на этапе создания, чтобы проинициализировать объекты. Т.е. конструктор использует параметры для внедрения зависимостей.

Плюсы/минусы

зависимость можно сделать immutable (неизменяемой), чтобы предотвратить повторное внедрение зависимости

все зависимости должны быть переданы в конструкторе

**Setter Injection**

Для внедрения сложных зависимостей в указанный момент

Плюсы/минусы

не нужно все прописывать в конструкторе

всегда надо отслеживать, какая зависимость нужна, чтоб подгрузить ее

**Interface Based Injection**

…

DI - <https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection>

Типы регистрации сервисов:

<https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection#service-registration-methods>

| **Method** | **Automatic object disposal** | **Multiple implementations** | **Pass args** |
| --- | --- | --- | --- |
| Add{LIFETIME}<{SERVICE}, {IMPLEMENTATION}>()  Example:  services.AddSingleton<IMyDep, MyDep>(); | Yes | Yes | No |
| Add{LIFETIME}<{SERVICE}>(sp => new {IMPLEMENTATION})  Examples:  services.AddSingleton<IMyDep>(sp => new MyDep()); services.AddSingleton<IMyDep>(sp => new MyDep(99)); | Yes | Yes | Yes |
| Add{LIFETIME}<{IMPLEMENTATION}>()  Example:  services.AddSingleton<MyDep>(); | Yes | No | No |
| AddSingleton<{SERVICE}>(new {IMPLEMENTATION})  Examples:  services.AddSingleton<IMyDep>(new MyDep()); services.AddSingleton<IMyDep>(new MyDep(99)); | No | Yes | Yes |
| AddSingleton(new {IMPLEMENTATION})  Examples:  services.AddSingleton(new MyDep()); services.AddSingleton(new MyDep(99)); |  |  |  |

IOC - <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles#dependency-inversion>

* [TryAdd](https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.dependencyinjection.extensions.servicecollectiondescriptorextensions.tryadd)
* [TryAddTransient](https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.dependencyinjection.extensions.servicecollectiondescriptorextensions.tryaddtransient)
* [TryAddScoped](https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.dependencyinjection.extensions.servicecollectiondescriptorextensions.tryaddscoped)
* [TryAddSingleton](https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.dependencyinjection.extensions.servicecollectiondescriptorextensions.tryaddsingleton)

один интерфейс можно регистрировать для разных сервисов

Options pattern

<https://docs.microsoft.com/en-us/dotnet/core/extensions/options>

Logging //TODO

<https://docs.microsoft.com/en-us/dotnet/core/extensions/logging?tabs=command-line>

Hosted service // TODO

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-6.0&tabs=visual-studio>

ASP .NET Core

ASP .NET Core vs ASP .NET Core MVC

<https://docs.microsoft.com/en-us/dotnet/architecture/porting-existing-aspnet-apps/middleware-modules-handlers>

Kestrel vs IIS

<https://www.tutorialspoint.com/what-is-kestrel-and-how-does-it-differ-from-iis-asp-net>