

Dependency Injection in ASP.NET Core

Tomáš Herceg

CEO @ RIGANTI

Microsoft Most Valuable Professional

Microsoft Regional Director

[@hercegtomas](https://twitter.com/hercegtomas)

tomas.herceg@riganti.cz

<https://www.tomasherceg.com/blog>



Microsoft Extensions Libraries

- ▶ NuGet packages with .NET infrastructure
 - ▶ Compatible with .NET Standard
- ▶ `Microsoft.Extensions.something`
 - ▶ `DependencyInjection`
 - ▶ `Logging`
 - ▶ `Caching`
 - ▶ `Configuration`
 - ▶ `Localization`
 - ▶ `Identity`
 - ▶ ...

DependencyInjection Package

- ▶ Provides basic DI functionality
- ▶ Ideal for basic scenarios
- ▶ Defines fundamental interfaces for DI
- ▶ Used by ASP.NET Core infrastructure
- ▶ Extensible
 - ▶ Integrates with external DI containers

SOLID

- ▶ Single Responsibility Principle
- ▶ Open Closed Principle
- ▶ Liskov Substitution Principle
- ▶ Interface Segregation Principle
- ▶ Dependency Inversion Principle

DEMO

Solid Principles

Classic .NET Framework

- ▶ No built-in DI interfaces
- ▶ Many existing DI containers
 - ▶ Unity
 - ▶ Castle Windsor
 - ▶ Autofac
 - ▶ SimpleInjector
 - ▶ ...
- ▶ Difficulties for framework and library authors

DependencyInjection Package

- ▶ **IServiceCollection**
 - ▶ Registration rules for services
- ▶ **IServiceProvider**
 - ▶ Abstraction for any DI container
 - ▶ `GetService` method

DependencyInjection Package

- ▶ Creates instances based on the configuration
- ▶ Manages lifetime of instances
 - ▶ Singleton
 - ▶ Transient
 - ▶ Scoped
- ▶ Disposes instances

DEMO

DI in ASP.NET Core

Using External Containers

- ▶ Factories
 - ▶ Injecting Func<T> or Lazy<T>
- ▶ Convention-based registrations
 - ▶ Open generics
 - ▶ Discovery of implementations in specific assemblies
- ▶ Custom lifetime policies
- ▶ Interception

Hooking with Autofac

- ▶ NuGet Packages
 - ▶ Autofac
 - ▶ Autofac.Extensions.DependencyInjection
- ▶ ConfigureServices can return IServiceProvider
 - ▶ If it doesn't, the default one is used

DEMO

Autofac Integration

DEMO

Castle Windsor integration

Best Practices

- ▶ Use modules
- ▶ Use naming conventions
 - ▶ and document them
- ▶ Be careful about lifetime settings
 - ▶ Singletons must be thread-safe
 - ▶ Singletons should not request scoped services

Q&A

Tomáš Herceg

CEO @ RIGANTI

Microsoft Most Valuable Professional

Microsoft Regional Director

[@hercegtomas](#)

[tomas.herceg@riganti.cz](#)

[https://www.tomasherceg.com/blog](#)

