ASP.NET Core
Dependency Injection & Unit Testing

Robert Haken
software architect, HAVIT, s.r.o.
haken@havit.cz, @RobertHaken
Microsoft MVP, MCT, MCSD

ShowIT
2017

ASP.NET Core - MVC6 Controller
IEmailSender Dependency Injection demo

OCHUTNÁVKA

Dependency Injection

"Dependency Injection is a set of software design principles and patterns that enable us to develop loosely coupled code."

Mark Seeman

Dependency Injection Benefits

loosely coupled code =>

- Late-binding
- Extensibility
- Parallel Development
- Maintainability
- Testability

Price Resolver - Dependency Injection, Unit-Testing

DEMO

"Good" Unit Test

- automated + repeatable
- fully isolated
- consistent in its results
- runs quickly
- full control of the unit under test (all dependencies)
- relevant tomorrow
- easy to implement
- able to run it at the push of a button
- if fails => easy to detect what was expected

Dependency Injection Containers

- Object Composition
- Object Lifetime
- Interception

- Code-based / XML-based configuration
  - Auto-wiring, convention-based configuration, ...

ASP.NET Core DI "Container" - Register

- ## Startup.cs
  - Configure[Env]Services(IServiceCollection services)

- ## API - Lifestyles
  - services.AddTransient<TAbstraction, TImplementation>()
  - services.AddScoped<TAbstraction, TImplementation>()
  - services.AddSingleton<TAbstraction, TImplementation>()
  - services.AddSingleton<TAbstraction>(instance)
  - plus factory-method verze

ASP.NET Core DI "Container" - Injection

Controllers
- constructor injection
- Action - method injection `[FromServices]`

Views
- property injection `@inject TService PropertyName`

Models
- property injection `[FromServices]`

Startup: Configure()
- method injection

ASP.NET Core DI "Container" - Injection

Middleware
- constructor injection
- method injection `Invoke(...)`

Filters
- "attribute" injection `[ServiceFilter(typeof(...))]`
- "attribute" injection `[TypeFilter(typeof(...))]`

Other
- constructor injection

Q & A