[ASP].NET Core 2.1-2.2

Demystified

Miroslav Holec

Konzultant a lektor [ASP].NET Core

mirek@miroslavholec.cz
www.miroslavholec.cz

Miroslav Holec

Nezávislý konzultant a lektor [ASP].NET Core

- **školím ve firmách** ASP.NET Core a RESTová API https://www.**miroslavholec.cz**
- **připravuji konferenci** Dotnet Days 2019 https://www.**dotnetdays.cz**
- **píšu novinky** Dotnet News https://www.**dotnetnews.cz**

Miroslav Holec

Nezávislý konzultant a lektor [ASP].NET Core

- **školím ve firmách** ASP.NET Core a RESTová API https://www.**miroslavholec.cz**
- **připravuji konferenci** Dotnet Days 2019 https://www.**dotnetdays.cz**
- **píšu novinky** Dotnet News https://www.**dotnetnews.cz**

Praktické školení

Stavíme RESTová API v ASP.NET Core

https://odkaz.me/skapi

5. září 2019

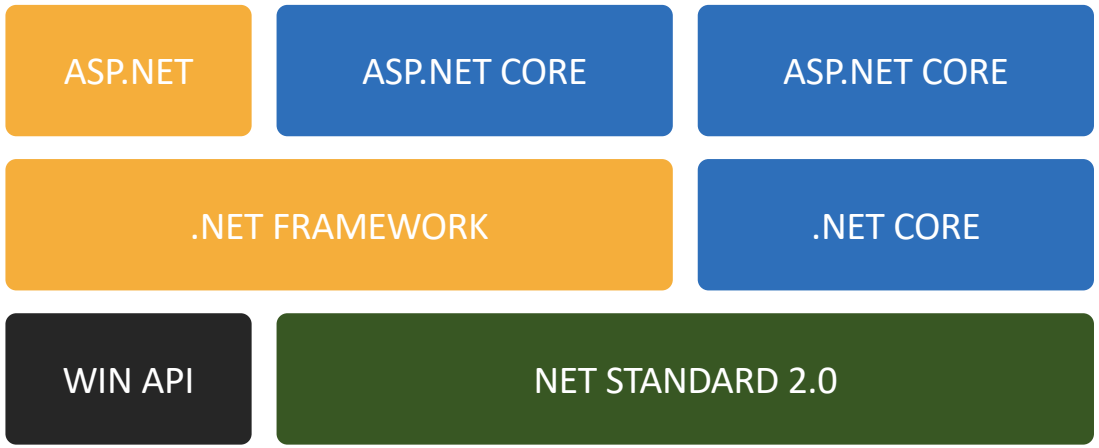
Agenda

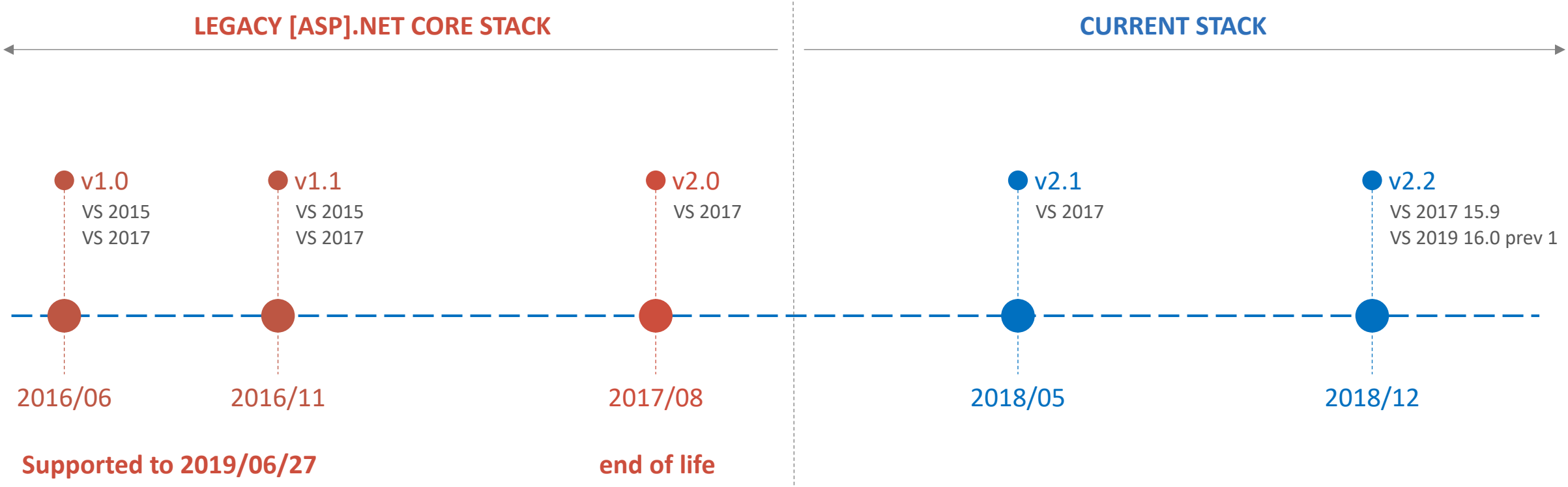
Témata dnešního večera

- Tooling a nástroje pro vývoj .NET Core aplikací
- NET Standard a tvorba NuGet balíčků
- Global Tools
- DI, Configuration & Logging
- ASP.NET Core Lifecycle
- Middlewares
- Publikace a hostování aplikací
- Diskuse

Aktuální vývojářský stack

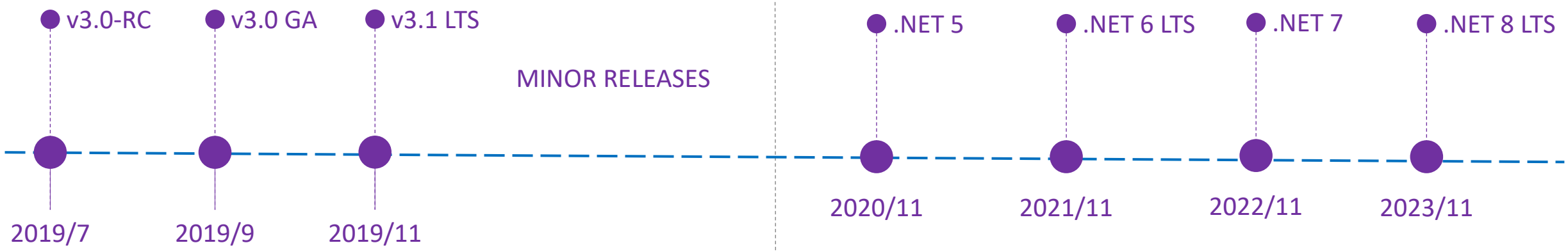
Pohled na současný vývojářský stack

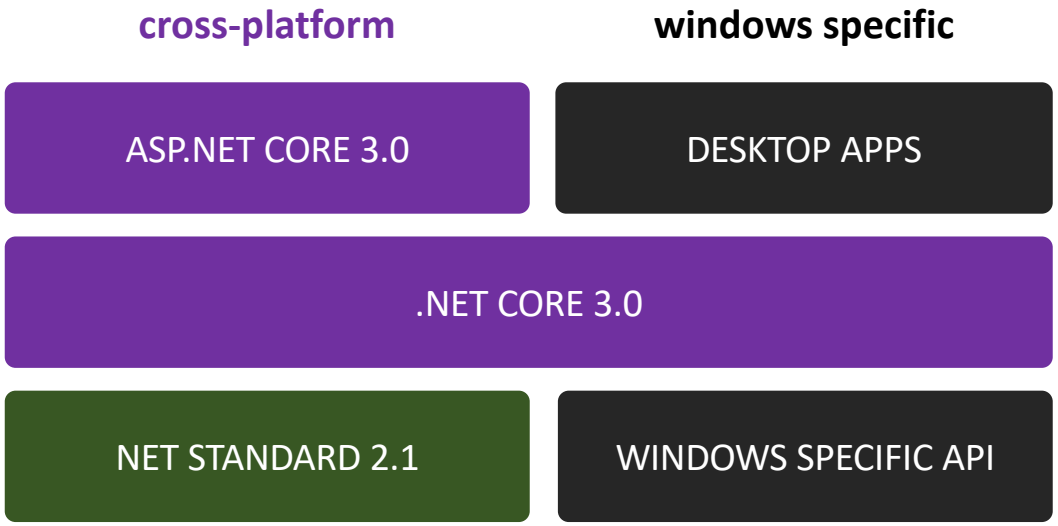




Pohled do budoucnosti

Vývojářský stack pro .NET Core 3.0





.NET – A unified platform

SDK & RUNTIMES

Prostředí pro vývoj aplikací v .NET Core

- **runtime** (typový systém, práce s assembly, GC, interop)
- **framework libraries** (primitivní datové typy a základní utility)
- **CLI tools a kompilátory** (roslyn a další utility pro vývojáře)
- **dotnet tool** (práce s CLI nad vybraným runtime, spouštění aplikací apod.)

NET Core Runtime = runtime + framework libraries

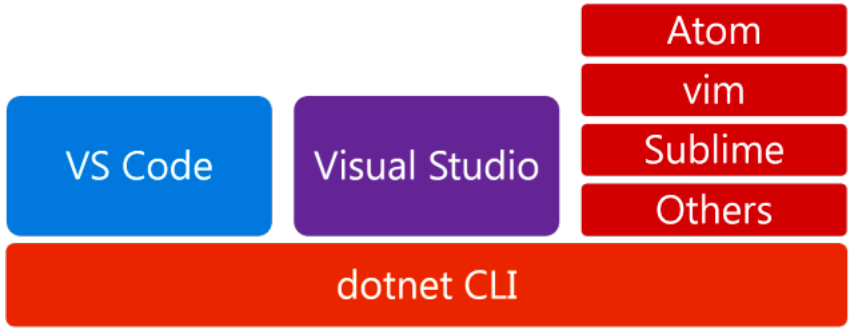
NET Core SDK = všechno v jednom pro vývojáře

NET Core Hosting Bundle = **NET Core Runtime** + **ASP.NET Core Module**

DOTNET & CLI TOOLS

Správa .NET Core aplikací pomocí toolingu

- CLI je **multiplatformní sada nástrojů** pro vývoj .NET aplikací
- **instaluje se společně s SDK** (na disku je více verzí)
- komunikace pomocí IDE nebo nástroje **dotnet**
- **dotnet** slouží mj. k řízení běhu FDD aplikací



Příkazy:

```
dotnet
dotnet new --help
dotnet new console
dotnet new console --help
```

CSPROJ

Moderní projektové soubory s podporou NuGet

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.2</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Newtonsoft.Json" Version="2.2.0" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\NetStd.Lib\NetStd.Lib.csproj" />
```

CSPROJ & NUGET PACKAGES

Moderní projektové soubory s podporou NuGet

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>

    <GeneratePackageOnBuild>true</GeneratePackageOnBuild>
    <Version>1.0.1</Version>
    <Copyright>Miroslav Holec</Copyright>
    <Description>New great library</Description>

  </PropertyGroup>

</Project>
```

Dependency Injection

Nativní podpora pro DI v .NET Core

.NET `dotnet add package Microsoft.Extensions.DependencyInjection`

- Kontejnerem je třída **ServiceProvider** (implementace **IServiceProvider)**
- Vzniká zavoláním metody **BuildServiceProvider()** nad **ServiceCollection**

```
ServiceProvider serviceprovider = new ServiceCollection()
     .AddTransient<MyApp>()
     .AddTransient<SmtpService>()
     .BuildServiceProvider();

var application = serviceprovider.GetService<MyApp>();
```

Konfigurace aplikace

UCHOVÁNÍ KONFIGURACE V .NET CORE

- veškerá konfigurace je založena na párech **KLÍČ**:**HODNOTA**

```
{
  "SmtpServer": {
    "Host": "valuexx",
    "Port": "valueyy"
  }
}
```

```
[SmtpServer]
Host = valuexx
Port = valueyy
```

KLÍČ	HODNOTA
SmtpServer:Host	valuexx
SmtpServer:Port	valueyy

Konfigurace aplikace

Různé zdroje konfigurace s podporou DI

.NET `dotnet add package Microsoft.Extensions.Configuration`

- Veškerou konfiguraci zpřístupňuje třída **IConfigurationRoot**
- Vzniká zavoláním metody **Build()** nad třídou **ConfigurationBuilder**

```
IConfigurationRoot config = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddIniFile("application.ini", true, true)
    .Build();
```

Konfigurace aplikace

Návrhový vzor IOptions<T>

.NET

```
dotnet add package Microsoft.Extensions.Options
+ Microsoft.Extensions.Options.ConfigurationExtensions
```

```
ServiceProvider services = new ServiceCollection()
    .Configure<Server>(configuration.GetSection("SmtpServer"))
    .AddSingleton(configuration)
    .BuildServiceProvider();
```

```
public class MyApp
{
    public MyApp(IOptions<Server> config)
    {
        this.config = config.Value;
```

Logování

Plně rozšiřitelný mechanismus pro logování

.NET `dotnet add package Microsoft.Extensions.Logging`

- Konfiguraci sestavuje **LoggingBuilder**, který je přístupný skrze extension metodu
- Součástí registrace je podpora tzv. **ILogger<T>** patternu pro logování událostí
- Logování probíhá do výstupů dle registrace (obvykle samostatné NuGet pckgs.)

```
var serviceCollection = new ServiceCollection()
    .AddLogging(x =>
    {
        x.AddDebug();
    });
```

Logování

Plně rozšiřitelný mechanismus pro logování

```
public class MyApp
{
    private ILogger<MyApp> logger;

    public MyApp(ILogger<MyApp> logger)
    {
        this.logger = logger;
    }

    public void Run()
    {
        logger.LogInfo(“message“);
    }
}
```

Shared Frameworks

Falešné NuGet balíčky a metabalíčky

- webový framework ASP.NET Core se skládá **z více než stovky balíčků NuGet**
- pro zjednodušení existují **metabalíčky**, které sestavují **stabilní konstelace balíčků**
- každý metabalíček reprezentuje zároveň **shared framework** (sada assemblies)
- shared framework se instaluje společně s runtime a **dll jsou již předkompilované**
- během publikace a hostování probíhá **host roll forward**

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App" />
```

Inicializace webové aplikace (IWebHostBuilder)

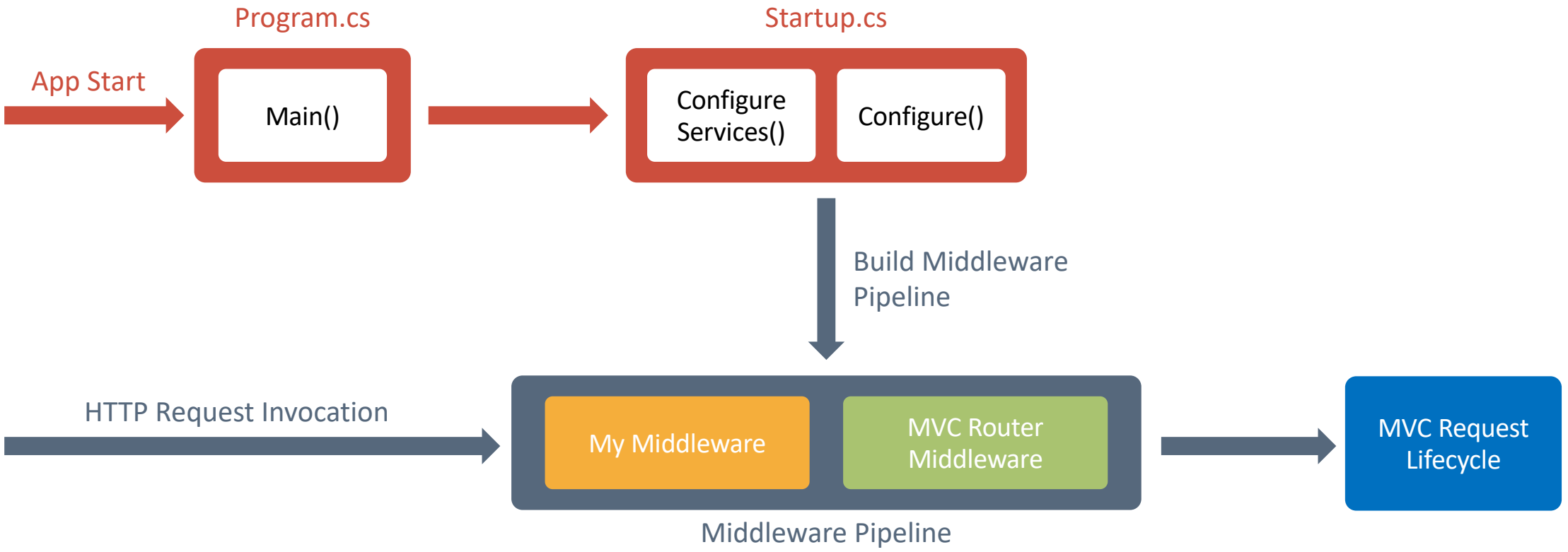
Webová aplikace v .NET Core

- hostování webové aplikace zajišťuje **IWebHostBuilder** (startup + lifecycle)
- základním úkolem je **konfigurace serveru** a **request pipeline**
- komplexní nastavení je skryto v metodě **WebHost.CreateDefaultBuilder()**
- v základu se používá webový server **Kestrel** nebo **IISHttpServer** (pouze IIS)
- lze použít webový server **HTTP.sys** v prostředí Windows

```
public static void Main(string[] args)
{
    WebHost.CreateDefaultBuilder(args)
           .UseStartup<Startup>()
           .Build().Run();
}
```

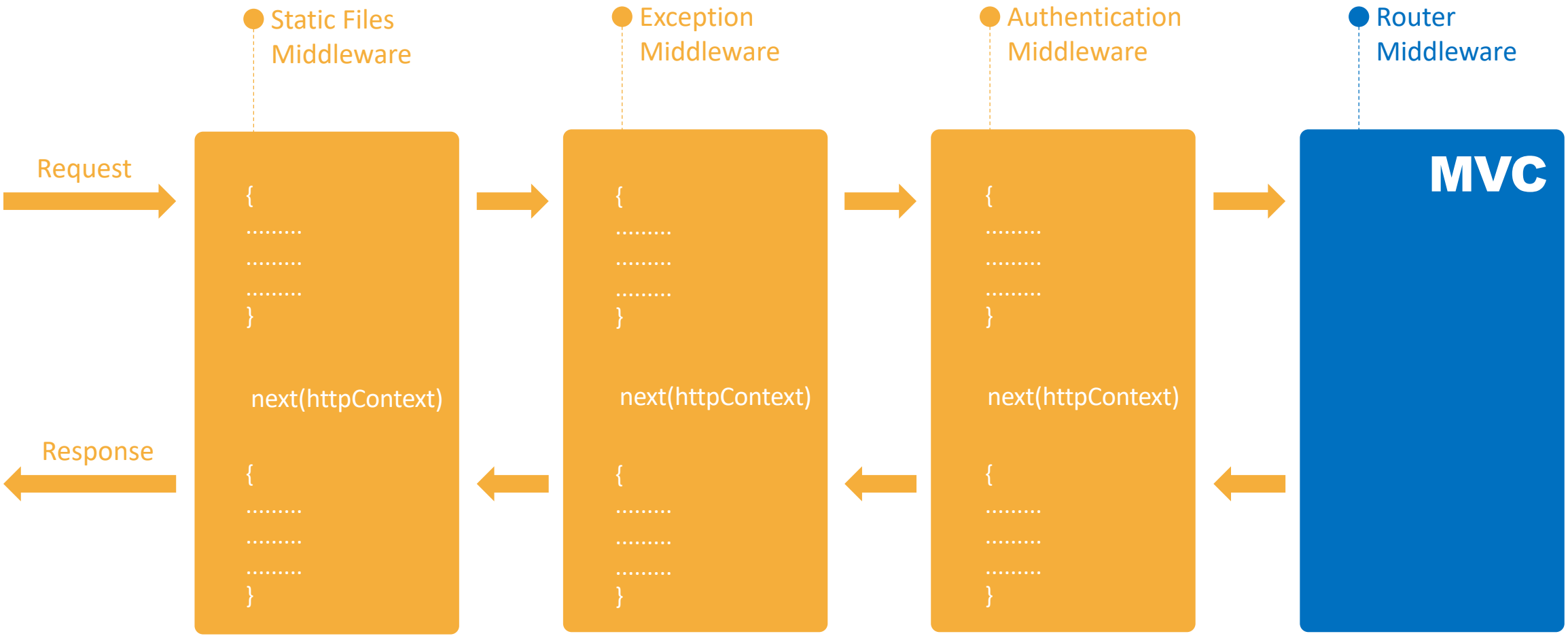
Životní cyklus ASP.NET Core aplikace

Vznik webové aplikace a souvislosti s .NET Core



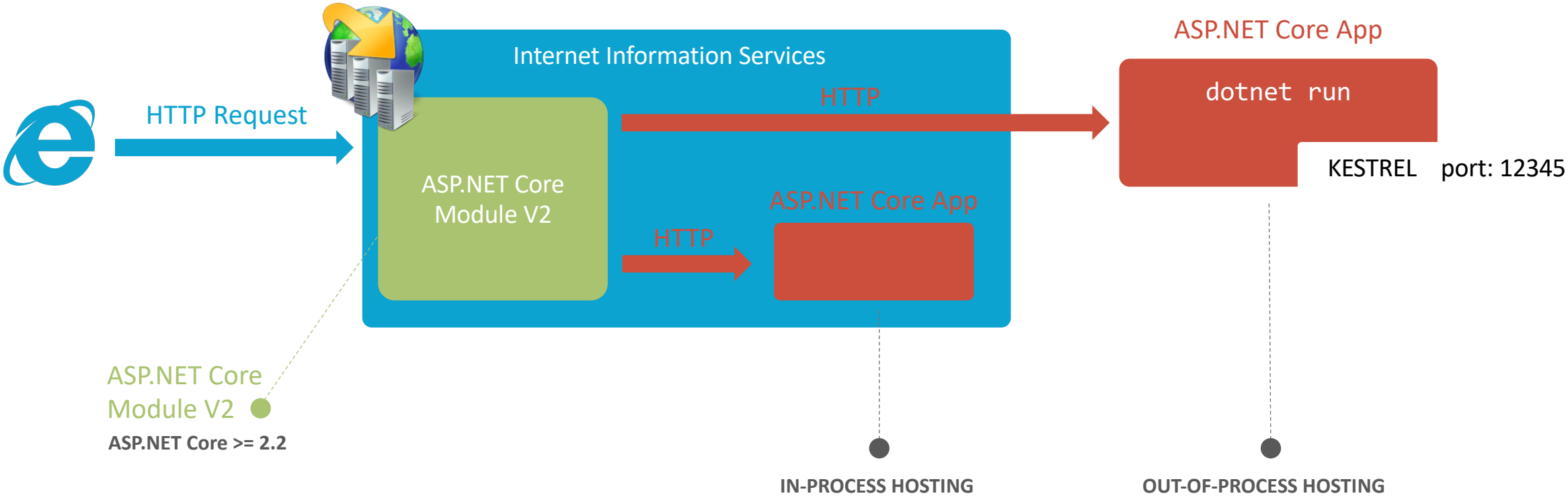
Middleware Pipeline

Jednotný způsob odbavování HTTP požadavků



Publikace a hostování ASP.NET Core aplikací

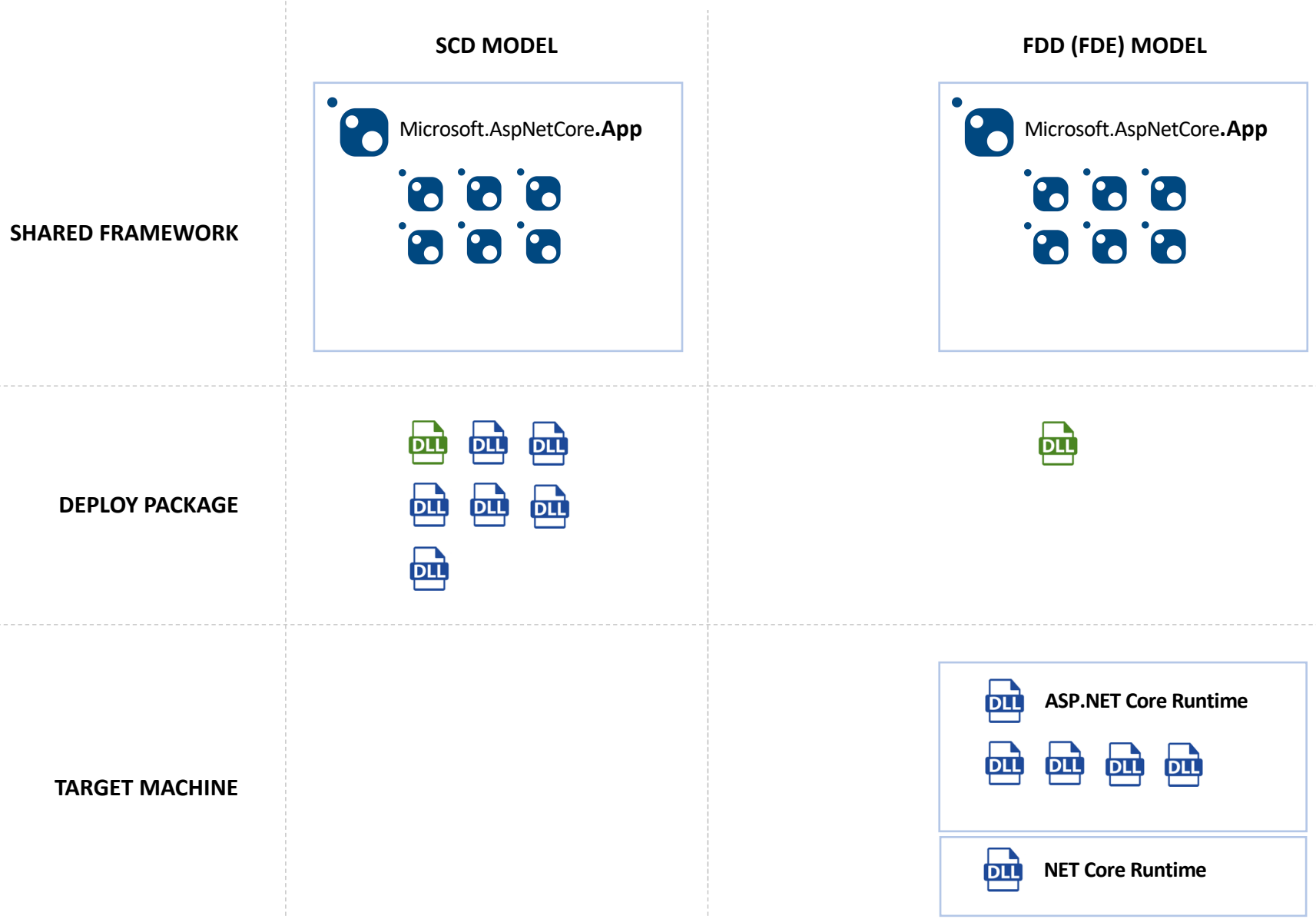
ASP.NET CORE HOSTING MODULE V2




```
<PropertyGroup>
   <AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>
</PropertyGroup>
```

Publikace a hostování ASP.NET Core aplikací

FDD, SCD



Shrnutí

Co byste měli o [ASP].NET Core vědět

- SDK & Runtimes + správa pomocí CLI
- Nový csproj a tvorba NuGet balíčků a Global Tools
- Obecné moduly DI, Configuration, Logging
- Životní cyklus ASP.NET Core aplikace a výchozí šablona
- Publikování aplikací v režimech SCD a FDD