ASP.NET Core 2

Tomáš Jecha

Microsoft Most Valuable Professional (MVP), MCSD
.NET team lead, software architect, lector

www.jecha.net
tomas@jecha.net
What is .NET Core?

**Cross-platform**
- Windows
- Linux
- macOS

**Modular**
- Only packages you need
- Lightweight deployment

**Modern**
- C#
- LINQ
- Async support
- Much more!

**Open Source**
- Runtime, libraries, compiler, languages and tools – on GitHub
- Contributions are reviewed and accepted
.NET Core Web App Stack

- .NET Core Libraries
  - Collections, IO, LINQ, XML, ADO.NET...
  - Implements .NET Standard
- Extensions
  - Logging, DI, Caching, Configuration...
- ASP.NET Core
  - Web server abstraction, Authentication, WebSockets...
- ASP.NET MVC Core
  - MVC Framework
- Your application & 3rd party libraries

App
Current .NET Core Versions

- .NET Core 1.0
- .NET Core 1.1
- .NET Core 2.0

-.NET Core 2.0 SDK and/or Visual Studio 2017 15.3

https://github.com/dotnet/core/blob/master/roadmap.md
.NET Core App
ASP.NET Core

- .NET Core App project type
- Set of packages **Microsoft.AspNetCore.All**
- Not based on IIS – starting with clean HTTP pipeline
- HTTP pipeline is configured via Startup class
- ASP.NET MVC Core is optional part of ASP.NET Core
ASP.NET Core HTTP pipeline: Middlewares Chain
ASP.NET Core App
Microsoft Extensions

- Namespace Microsoft.Extensions.*
- Used by ASP.NET Core
- Important parts:

- Logging
- DI
- Config
- Caching
- File Providers
- Identity
- Localization
Configuration

• No static configuration accessors
• Configuration is optional
• Build-in sources:
  – JSON
  – XML
  – INI
  – Command line
  – Environment vars
  – Custom
Configuration & Options
Logging
Caching, file providers, default documents
Front-end features

• Visual Studio Bower integration (bower.json)
• Visual Studio NPM integration (packages.json)
• Task runner integrations (gulp, WebPack)
• Templates with CSPROJ integrations
• Bundling and minification
• NodeServices
• WebPackMiddleware
• SPA integrations for Angular and React
Front-end features
Cross-platform
.NET Standard
Deployment
.NET Core Runtime vs .NET Core SDK

.NET Core SDK
• Allows you to compile
• Installation package or binaries
• Supports side-by-side SDK versions and runtimes

.NET Core Runtime
• Allows to run compiled code
• Delivered with SDK or with application
• Identified by:
  • Version (1.0, 1.1, 2.0)
  • And Runtime Identifier (win10-x64, centos.7-x64...)
• Some runtimes have prerequisites (libunwind & libicu on CentOS 7 etc.)
SDK installation and versions
Deployment Strategies

- Compiled with runtime
- Compiled as portable
- Source Code
Publishing
.NET Core Runtime IDentifier (RID)

- RIDs are used to identify target operating systems
- [os].[version]-[architecture]
- Examples:
  - win10-x64
  - win10-arm
  - win7-x86
  - centos.7-x64
  - ubuntu.16.10-x64
  - osx.10.12-x64

- Full catalog: https://docs.microsoft.com/en-us/dotnet/core/rid-catalog
.NET platforms

Application Types
- WinForms, WPF
- ASP.NET
- Console App
- Windows Service

Base Libraries
- Base Class Library

.NET Framework
- ASP.NET

.NET Core
- ASP.NET Core
- Universal Apps
- Windows Service

.NET Core Libraries

Xamarin
- Android
- iOS
- Windows App

Mono Class Libraries
NuGet packages using API available on all platforms
Sharing code with multiple runtimes?

- Compile per-platform
- Portable Class Library
- .NET Standard
What is .NET Standard?

- Set of APIs that all .NET platforms have to implement
- Unifies the .NET platforms and prevents future fragmentation
- .NET Standard 2.0 is implemented by .NET Framework, .NET Core, and Xamarin
- Use .NET Standard to write and share universal libraries.

  - [https://github.com/dotnet/standard](https://github.com/dotnet/standard)
  - Easy to use!
.NET Standard Libraries
.NET Standard Versioning

<table>
<thead>
<tr>
<th>.NET Standard</th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
<th>1.5</th>
<th>1.6</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET Core</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>.NET Framework</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5.1</td>
<td>4.6</td>
<td>4.6.1</td>
<td>4.6.1</td>
<td>4.6.1</td>
<td>4.6.1</td>
</tr>
<tr>
<td>Mono</td>
<td>4.6</td>
<td>4.6</td>
<td>4.6</td>
<td>4.6</td>
<td>4.6</td>
<td>4.6</td>
<td>4.6</td>
<td>vNext</td>
</tr>
<tr>
<td>Xamarin.iOS</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>vNext</td>
</tr>
<tr>
<td>Xamarin.Mac</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>3.0</td>
<td>vNext</td>
</tr>
<tr>
<td>Xamarin.Android</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>vNext</td>
</tr>
<tr>
<td>Universal Windows</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
<td>vNext</td>
<td>vNext</td>
<td>vNext</td>
</tr>
<tr>
<td>Platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Full compatibility matrix: [https://github.com/dotnet/standard/blob/master/docs/versions.md](https://github.com/dotnet/standard/blob/master/docs/versions.md)
Compatibility
What’s next?

• Learn .NET Core 2 & ASP.NET Core 2 when released
• Consider using .NET Standard as target for your libraries
• Install Visual Studio 2017 Preview (side-by-side)
Tomáš Jecha
www.jecha.net
tomas@jecha.net