Workshop

.NET Core 3

Introduction

Rainer Stropek

software architects gmbh

Web	http://www.timecockpit.com
Mail	rainer@timecockpit.com
Twitter	@rstropek

time cockpit

Saves the day.

Intro

Tooling

Versioning

Runtime

- Semantic versioning

SDK

- Not semantic versioning
- First two parts of version: Runtime SDK was released with (e.g. 2.2.100 → runtime 2.2)
- Third part: Minor version * 100 + patch number

Version selection

- Always use latest SDK installed (e.g. for *dotnet new*), even if project for earlier runtime
- Use *global.json* in path hierarchy to specify use of an older SDK (docs)

https://docs.microsoft.com/en-us/dotnet/core/versions/index

Versioning

Rolling forward for framework-dependent deployments

- Latest patch for specified target framework is used
- E.g. netcoreapp2.0 → latest 2.0.x runtime used; if not found, latest 2.x runtime is used
- Docs

Self-contained deployments

- Uses highest patch runtime on the publishing machine

Important commands

- *dotnet --list-sdks*
- *dotnet --list-runtimes*
- Remove old sdks in Windows' *Add/Remove Programs*
- Remove old runtimes by deleting corresponding folders

https://docs.microsoft.com/en-us/dotnet/core/versions/index

Versioning

New in 3: *In-place* upgrade of SDK by MSI installer

- E.g. MSI for 3.0.103 will replace 3.0.101

Publishing

```
dotnet new console
dotnet publish -o out
dir out
  → Note framework-dependent exe

dotnet publish -o out-fdd -r linux-x64 --no-self-contained
  → Create framework-dependent exe for Linux on Windows

dotnet publish -o out-scd -r win-x64
  → Create self-contained exe
  → Before running on WSL:
    export DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=1

dotnet publish -o out-sf -r win10-x64 /p:PublishSingleFile=true
  → Alternative: Property PublishSingleFile in .csproj
  Run out-sf/demoapp.exe

  set DOTNET_BUNDLE_EXTRACT_BASE_DIR=…\out-sf
  Run out-sf/demoapp.exe and look into extracted files

→ Repeat previous demo with --no-self-contained
→ Repeat previous demo with PolygonDesigner
   Add to csproj: <UseAppHost>true</UseAppHost>
```

Architecture identifiers

- Docs

Single-file exe

- Extraction logic docs

```
Check size of single-file exe

Add to .csproj:
  <PropertyGroup>
    <PublishTrimmed>true</PublishTrimmed>
  </PropertyGroup>

Create self-contained single-file exe again

Compare size of single-file exe
```

Assembly linking

Removes unused libraries
- Especially useful with self-contained exe

Careful with reflection
- *Test entire app* before shipping!

Global Tools

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3</TargetFramework>

    <PackAsTool>true</PackAsTool>
    <ToolCommandName>...</ToolCommandName>
    <PackageOutputPath>./nupkg</PackageOutputPath>
  </PropertyGroup>
</Project>
```

```
dotnet pack

dotnet tool install --global --add-source ./nupkg eolfixer
dotnet tool list --global

eolfixer --help
eolfixer --pattern *.txt –v

dotnet tool uninstall --global eolfixer
```

NuGet packages that are console apps

Install
- *dotnet tool install -g ...*
- Install location (Windows):
  - *%USERPROFILE%\.dotnet\tools*

Sample:
https://github.com/rstropek/Samples/tree/master/CSharp8/LocalTools

Local Tools

Tools associated with a location on disk

Manifest-based

Dotnet-tools.json

```
dotnet new tool-manifest
  → .config/dotnet-tools.json

dotnet tool install  --add-source ./nupkg eolfixer
  → .config/dotnet-tools.json

# Restore with dotnet tool restore

dotnet eolfixer –help
  → Try running in different folder
dotnet eolfixer --pattern *.txt –v

dotnet tool uninstall eolfixer
```

Performance

Performance

.NET Libraries use platform-dependent intrinsics

- Docs

Tiered compilation

- Fast JIT compilation during startup → faster startup
- Optimized JIT compilation is called multiple times → faster steady-state
- Docs

Ready-to-run images

- Ahead-of-time compilation
- Similar native code *and* IL in assemblies → JITer has less work and will be faster
- *PublishReadyToRun* setting in *.csproj* (docs)
- Careful: Assemblies will become larger
- *No* cross-compiling (Linux on Windows, Windows on Linux)

Ready-to-Run

Demo with *PolygonDesigner* sample

```
Add <PublishReadyToRun>true</PublishReadyToRun>

Measure JIT with PerfView

Remove PublishReadyRun

Measure again JIT with PerfView
```

Performance

Built-in JSON Parser

- Based on *Span<T>*
- High performance, low allocation
- Up to 3x faster than *Json.NET*
- Sample: https://github.com/rstropek/Samples/tree/master/CSharp8/Json

Protocols

HTTP/2 Support in *HttpClient*

HttpClient now supports HTTP/2

- *HttpRequestMessage.Version*
- *HttpClient.DefaultRequestVersion*
- Sample: https://github.com/rstropek/Samples/tree/master/CSharp8/Http2Client

Support for TLS 1.3 & OpenSSL 1.1.1 on Linux

- Details: https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-core-3-0#tls-13--openssl-111-on-linux

Generic Host

Generic Host

Microsoft.Extensions.Hosting

ASP.NET Core Generic Host for non-HTTP-driven apps

Adds cross-cutting capabilities

- Configuration, DI, Logging

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/generic-host

C# 8

Time cockpit eLearning Library

Q&A

Thank your for coming!

Rainer Stropek

software architects gmbh

Mail	rainer@timecockpit.com
Web	http://www.timecockpit.com
Twitter	@rstropek

time cockpit
Saves the day.