

Online on-demand Project Support



WPF und .NET (Core) – Back again



Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor



Azure Cosmos DB

Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
 - Problemanalyse und Lösungen
 - Technologieentscheidungen



Agenda

- WPF unter .NET Core
- Was fehlt?
- Windows Compatibility Pack
- Migration
- Publishing/ Deployment



.NET Core Roadmap

Update: Build 2021



- .NET Core 3.1 (03.12.2019)
- **.NET 5.0 (10.11.2020)**
- .NET 6.0 LTS (Nov of 2021)
- Major releases every year
- LTS (Long Term Support, 3 Jahre) grade Versionen

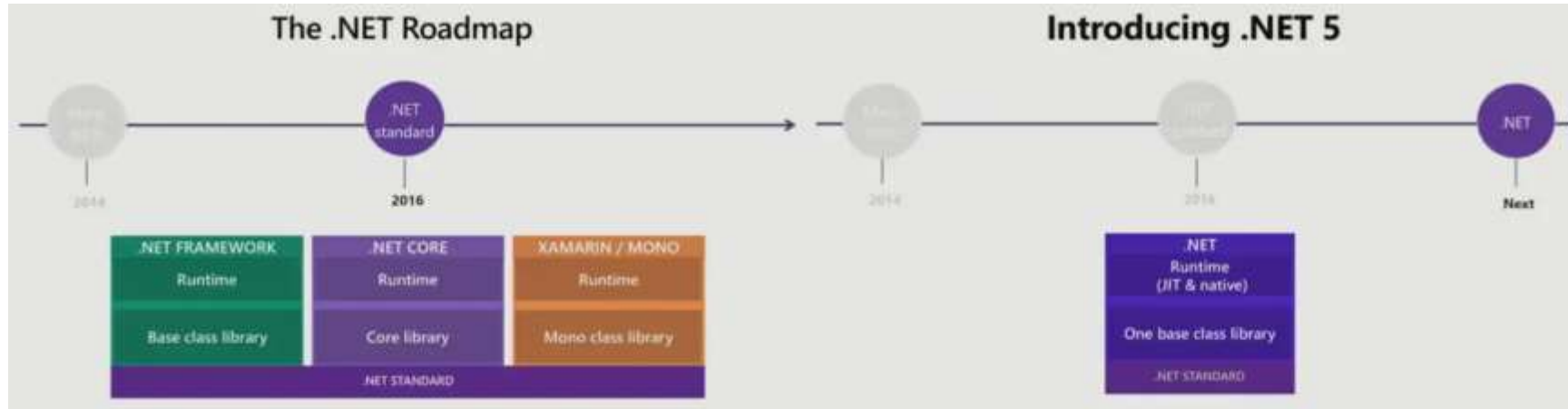


<https://www.microsoft.com/en-us/build>

Roadmap

Milestone	Release Date
.NET Core 2.1.x (servicing)	LTS (Long Term Support) release. Approximately every 1-2 months or as needed.
.NET Core 3.1.x (servicing)	LTS (Long Term Support) release. Approximately every 1-2 months or as needed.
.NET 5.0 (servicing)	Current release. Approximately every 1-2 months or as needed.
.NET 6.0	LTS (Long Term Support) release, scheduled for November 2021
.NET 7.0	Release scheduled for November 2022
.NET 8.0	LTS (Long Term Support) release, scheduled for November 2023

Update: Build 2021



.NET Framework + .NET Core + Mono/ Xamerin
=
.NET 5.0



<https://github.com/dotnet/core/blob/master/roadmap.md>

Zukunft von .NET Framework

- .NET Framework 4.8 ist die letzte .NET Framework Feature-Version
- Migration auf 4.8 nur bei neuen Features notwendig
- Ende 2028?!

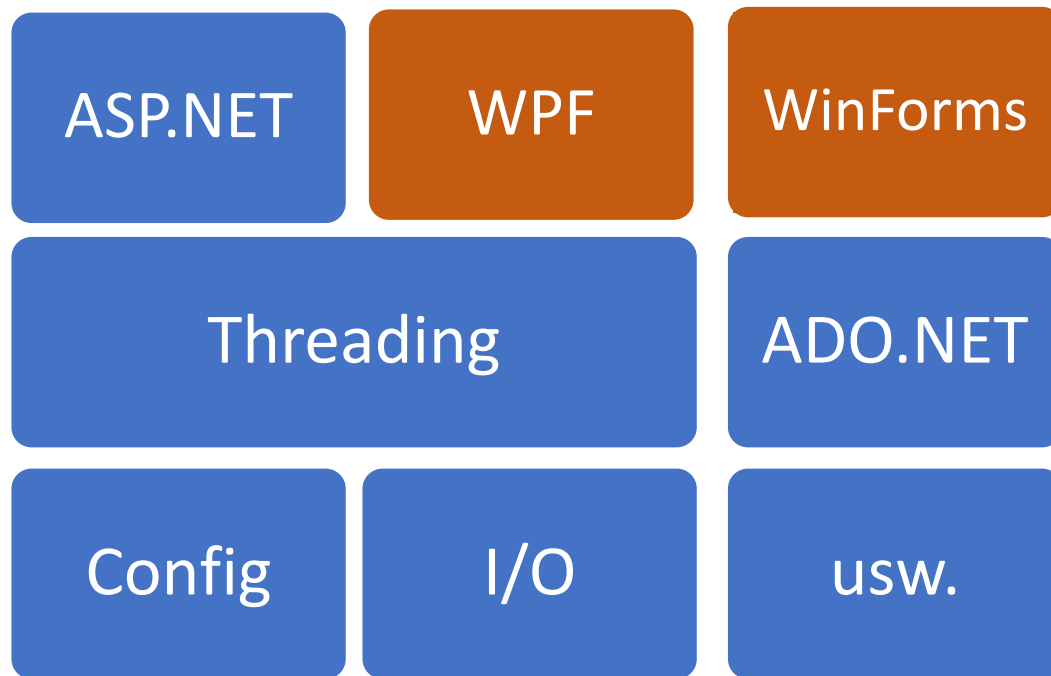


Einführung

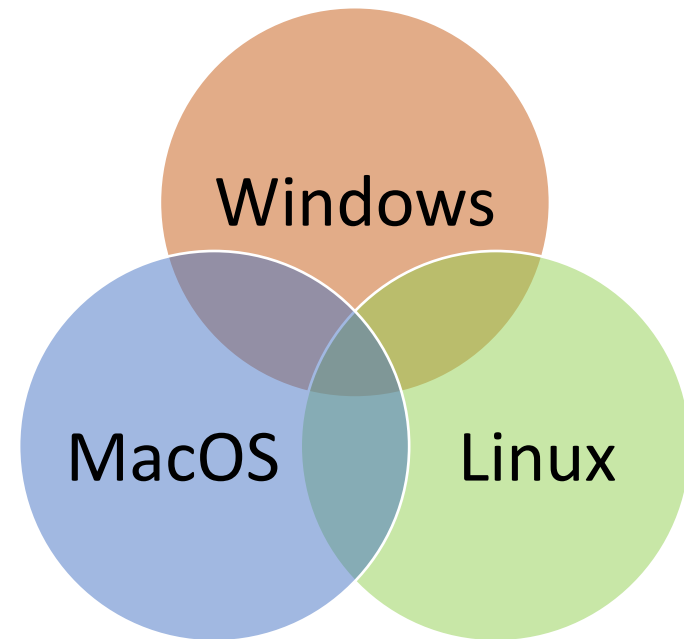


.NET Core-Aufbau

Modular

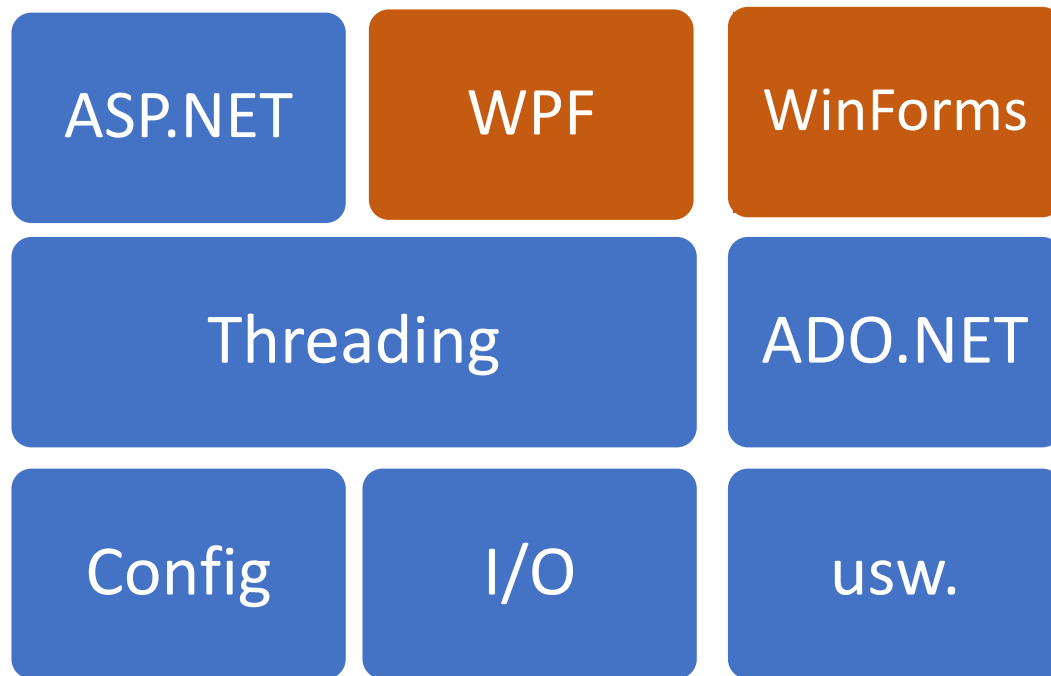


Multi Plattform

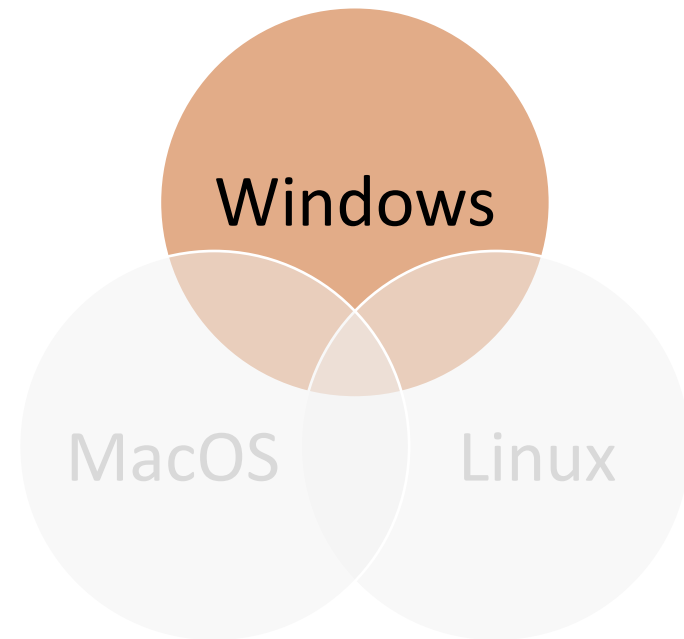


WPF unter .NET Core-Aufbau

Modular



Multi Plattform





WPF unter .NET Core

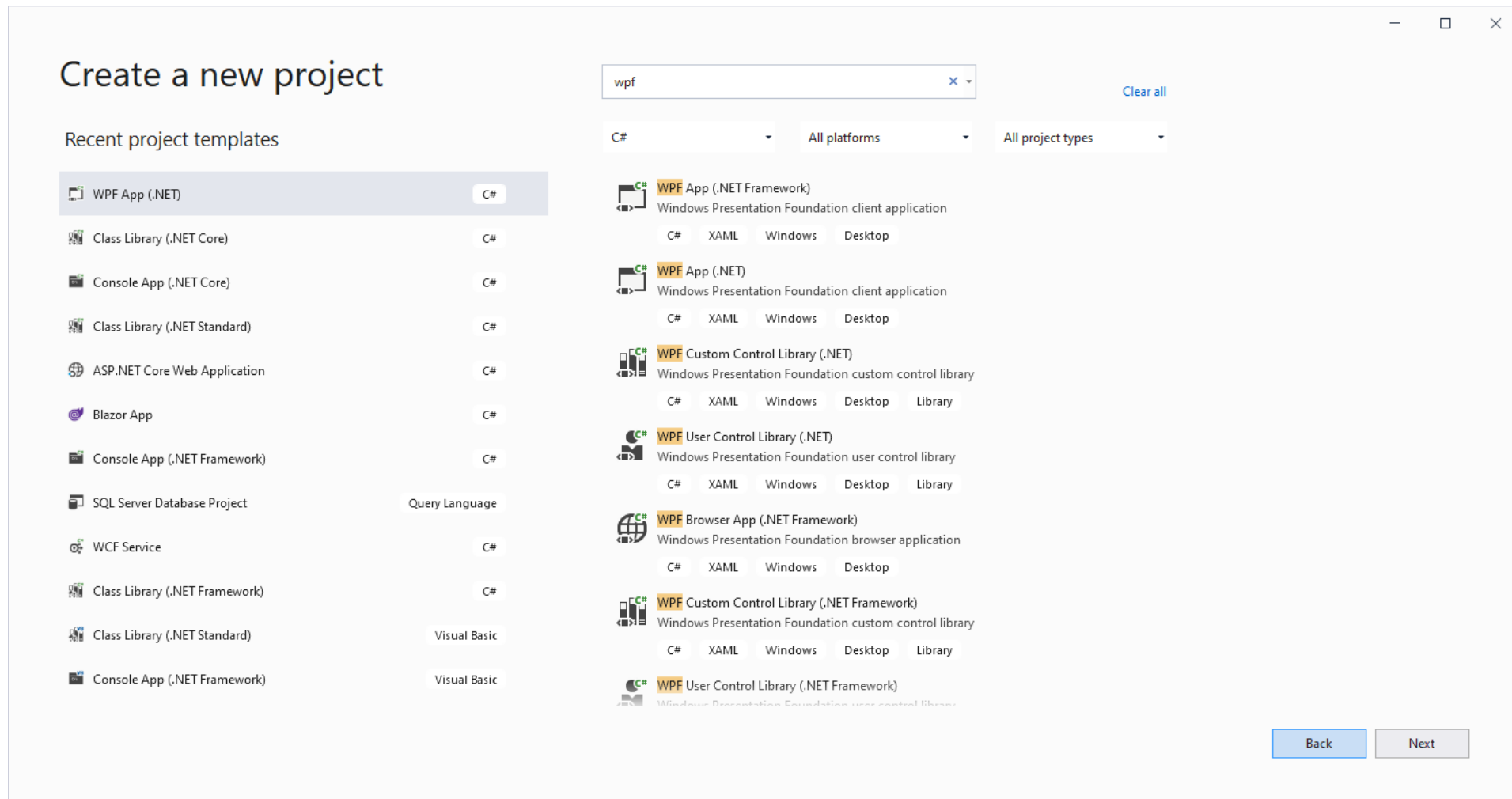
WPF unter .NET Core

- Ab .NET Core 3.0
- Visual Studio 2019+ (auch) mit Designer
- Nur unter Windows lauffähig
 - WPF basiert auf DirectX!
- Gleiches gilt für WinForms
 - Basiert allerdings auf GDI+

Vieles gleich, manches unterschiedlich

- XAML & Code (behind) sind identisch
- Unterschied .NET Framework vs .NET Core
 - keine App.Config
 - Projektdatei in SDK-Style

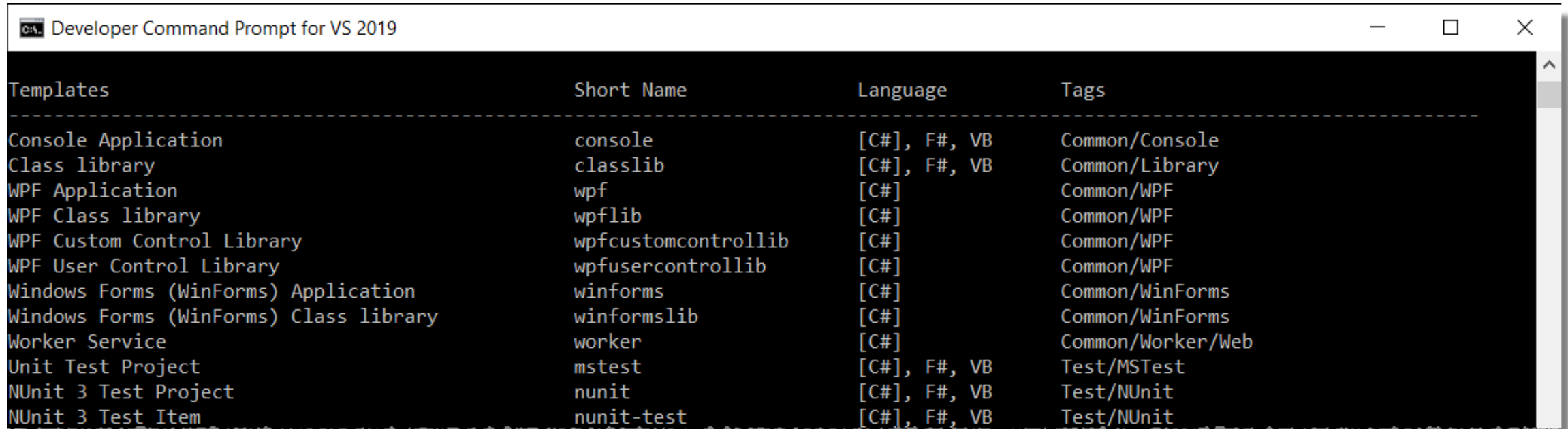
Visual Studio 2019+



 Demo 

.NET Core CLI

```
C:\>dotnet new
```



Templates	Short Name	Language	Tags
Console Application	console	[C#], F#, VB	Common/Console
Class library	classlib	[C#], F#, VB	Common/Library
WPF Application	wpf	[C#]	Common/WPF
WPF Class library	wpflib	[C#]	Common/WPF
WPF Custom Control Library	wpfcustomcontrollib	[C#]	Common/WPF
WPF User Control Library	wpfusercontrollib	[C#]	Common/WPF
Windows Forms (WinForms) Application	winforms	[C#]	Common/WinForms
Windows Forms (WinForms) Class library	winformslib	[C#]	Common/WinForms
Worker Service	worker	[C#]	Common/Worker/Web
Unit Test Project	mstest	[C#], F#, VB	Test/MSTest
NUnit 3 Test Project	nunit	[C#], F#, VB	Test/NUnit
NUnit 3 Test Item	nunit-test	[C#], F#, VB	Test/NUnit

Demo



Windows Compatibility Pack

Windows Compatibility Pack

- Code Pages
- CodeDom
- Configuration
- Directory Services
- Drawing
- ODBC
- Permissions
- Ports
- Windows Access Control Lists (ACL)
- Windows Communication Foundation (WCF)
- Windows Cryptography
- Windows EventLog
- Windows Management Instrumentation (WMI)
- Windows Performance Counters
- Windows Registry
- Windows Runtime Caching
- Windows Services



<https://docs.microsoft.com/en-us/dotnet/core/porting/windows-compat-pack>



Migration



Migration zu .NET Core

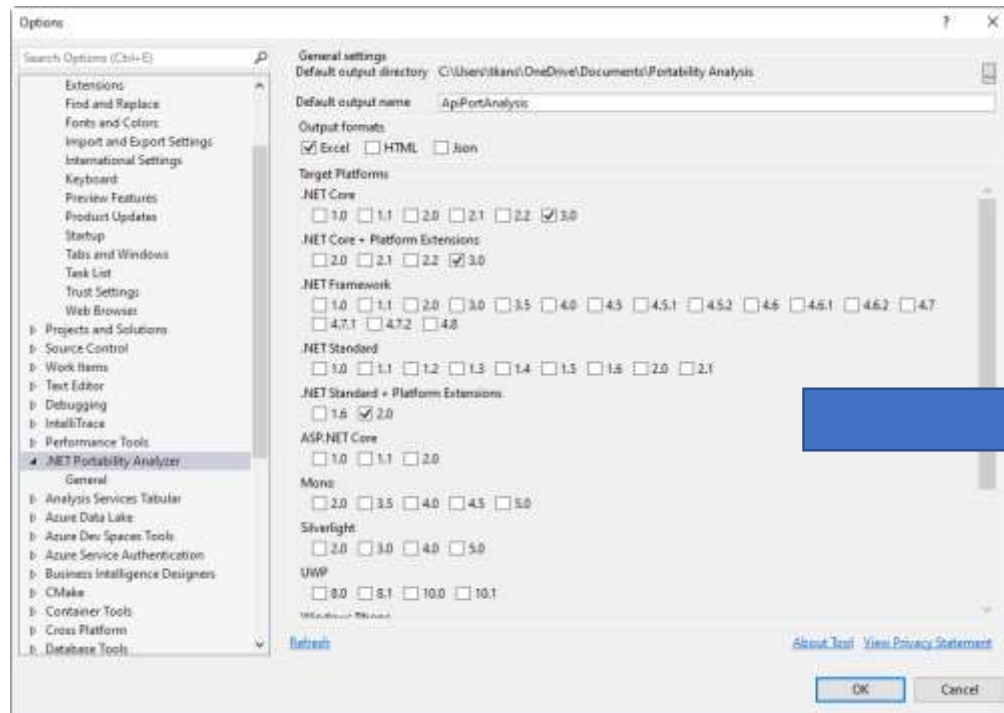
1. Aufwand abschätzen mit Analyzers
 - API Portability Analyzer Tool
 - NET Core 3.0 Desktop Api Analyzer
2. Dritthersteller (UI-)Komponenten prüfen
3. Projekte auf .NET Framework 4.6.2+ migrieren
4. Testportierung
5. Testdeployment
6. ...



<https://github.com/Microsoft/dotnet-apiport/>

.NET Portability Analyzer

Visual Studio Plug-in für 2015/ 2017/ 2019/ 2021



AutoSave

ApPortAnalysis(T).xlsx · Last Saved 3/23/2019 7:10 PM

FileHomeInsertDrawPage LayoutFormulasDataReviewViewDeveloperAdd-insHelpTeamTell me what you want

Cut

Copy

Format Painter

Clipboard

Calibri

11

A⁺A⁻

B

I

U

Text Color

Background Color

Font

Wrap Text

Align Left

Align Center

Align Right

Justify

Alignment

General

Number

Percentage

Decimals

Number

Conditional Formatting

Format as Table

Cell Styles

Styles

H25

✕

✓

f

	A	B	C	D	E	F	G
1	Submission Id	6d640c62-a8f4-41c1-bd64-9051f6323783					
2	Description						
3	Targets	.NET Core + Platform Extensions, .NET Core, .NET Framework, .NET Standard + Platform Extensions					
4							
5	Header for assembly name entries	Target Framework	.NET Core + Platform Extensions	.NET Core	.NET Framework	.NET Standard	
6	dotnetconsulting.DotNetCoreLibrary	.NETFramework,Version=v4.7.2	100	78.07	78.07	100	
7							
8	API Catalog last updated on	Tuesday, March 5, 2019					
9	See 'http://go.microsoft.com/fwlink/?LinkId=397652' to learn how to read this table						
10							
11							



<https://docs.microsoft.com/en-us/dotnet/standard/analyzers/portability-analyzer>

 Demo 

.NET Desktop Api Analyzer

 .NET Core 3.0 Desktop API Analyzer

Path to application folder:

Report saved in:
C:\Users\vkans\AppData\Local\Temp\PortabilityReport.xlsx

[About this tool](#)

– Privacy –
The analyzer will send anonymous API usage information to Microsoft to help us improve .NET Core.
This way, we know which APIs are often used by customers who wish to target .NET Core. We won't submit any of your binaries or source code to Microsoft.
For our full privacy statement visit: <https://privacy.microsoft.com/en-us/privacystatement>.



Header for assembly name entries			Target Framework	.NET Core
1	Submission Id	c3952345-4d95-41e9-8f3c-e8344ce240d8		
2	Description			
	Targets	.NET Core		
6	dotnetconsulting.DotNetCoreLibrary			100
7	ef, Version=2.1.4.0, Culture=neutral, PublicKeyTo .NETCoreApp,Version=v2.0			100
8	ef, Version=2.1.4.0, Culture=neutral, PublicKeyTo .NETFramework,Version=v4.6.1			98.65
9	Microsoft.CSharp, Version=4.0.0.0, Culture=neut			100
10	Microsoft.CSharp, Version=4.0.0.0, Culture=neut			100
11	Microsoft.CSharp, Version=4.0.0.0, Culture=neut			100
12	Microsoft.EntityFrameworkCore		.NETStandard,Version=v2.0	100
13	Microsoft.EntityFrameworkCore.Abstractions		.NETStandard,Version=v2.0	100
14	Microsoft.EntityFrameworkCore.Analyzers		.NETStandard,Version=v1.3	51.01
15	Microsoft.EntityFrameworkCore.Design, Version=		.NETStandard,Version=v2.0	100
16	Microsoft.EntityFrameworkCore.Design, Version=		.NETFramework,Version=v4.6.1	98.66
17	Microsoft.EntityFrameworkCore.Relational		.NETStandard,Version=v2.0	100
18	Microsoft.Extensions.Caching.Abstractions		.NETStandard,Version=v2.0	100



<https://devblogs.microsoft.com/dotnet/are-your-windows-forms-and-wpf-applications-ready-for-net-core-3-0>

 Demo 

Mit Assistent & viel Arbeit

GUI Migration

- Erstellung neuer Projekte in der Solution mit SDK-Projektformaten
 - WPF/ WinForm => .NET Core
 - Libraries => .NET Core oder .NET Standard
- Verlinken der ursprünglichen Quelldateien
 - Somit eine Quelldatei in mehreren Projekten
- (Nuget-)Package-Referenzen anpassen
- Namespace- und Assemblyname anpassen
- Erstellung der Assembly-Info unterdrücken
- Compiler-Fehler beheben
 - Windows Compatibility Pack?
- Try-Convert? Upgrade-Assistant? Nicht bei realen Live-Projekten...



<https://github.com/dotnet/try-convert>

<https://docs.microsoft.com/en-us/dotnet/core/porting/upgrade-assistant-wpf-framework>

SDK-Projektdatei

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">
  <PropertyGroup>
    ...
    <GenerateAssemblyInfo>false</GenerateAssemblyInfo>
    <!-- Namespace- und Assemblyname anpassen -->
    <RootNamespace>WpfGui</RootNamespace>
    <AssemblyName>WpfGui</AssemblyName>
  </PropertyGroup>

  <ItemGroup>
    <!-- Ggf. Code Dateien (jedoch nicht die der der XAML-Dateien) -->
    <Compile Include="..\WpfGui/**/*.*.cs"
             Exclude="..\WpfGui/**/*.*.xaml.cs" />
    <!-- Ggf. Ressourcen -->
    <EmbeddedResource Include="..\WpfGui/**/*.*.resx" />
  </ItemGroup>
```

glob-Pattern



<https://docs.microsoft.com/en-us/dotnet/core/tools/csproj>

Und Blender: Behaviors, etc.

- Nuget: Expression.Blend.Sdk => Microsoft.Xaml.Behaviors.Wpf
- C# <http://schemas.microsoft.com/xaml/behaviors> einfügen
 - <http://schemas.microsoft.com/expression/2010/interactivity>
 - <http://schemas.microsoft.com/expression/2010/interactions>
- XMAL: Microsoft.Xaml.Behaviors einfügen
 - Microsoft.Xaml.Interactivity
 - Microsoft.Xaml.Interactions



Demo

Test Migration

- Update NuGet dependencies
 - Standard oder .NET Core
- Update Project SDK-style Format
 - Nuget-Packages
(*packages.config* => *<PackageReference>-Elemente*)
- Fix build issues
- Runtime testing

A lush green landscape featuring a pond in the foreground with several flamingos wading in the water. The background is filled with dense tropical vegetation, including large trees and various plants. A dark blue horizontal band with a faint flamingo pattern is overlaid across the middle of the image, containing the word "Deployment" in white text.

Deployment

Deployment

- Visual Studio 2019: Publish
 - Framework-dependent Deployment (FDD)
 - Self-contained Deployment (SCD)
 - ClickOnce Deployment
- .NET Core CLI

Visual Studio 2019+: Publish

- FDD
 - Project -> Publish
- SCD
 - Project -> Publish

```
== *.csproj-Daten ==  
<PropertyGroup>  
  <RuntimeIdentifiers>win10-x64;osx.10.11-x64;linux-x64</RuntimeIdentifiers>  
</PropertyGroup>
```



<https://docs.microsoft.com/en-us/dotnet/core/deploying/deploy-with-vs>

 Demo 

MSIX

MSIX ist ein Paketformat, das auf Sicherheit, Schutz und Zuverlässigkeit ausgelegt ist und auf einer Kombination aus .msi-, .appx-, App-V- und ClickOnce-Installationstechnologien basiert.

Pro	Contra
Für alle Anwendungstypen (wenn Core 3.0+)	Windows 10+
Uri Launching	Sandbox Apps
Sandbox Apps	



<https://docs.microsoft.com/de-de/windows/msix/>

MSIX

Windows (3)

**.NET desktop development** ☒
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

**Desktop development with C++** ☐
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

**Universal Windows Platform development** ☒
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Add a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Language: Platform: Project type:

**Optional Code Package (Universal Windows)**
A project for creating an optional package that can contain executable code to be called from a main package.
C# Windows Xbox UWP

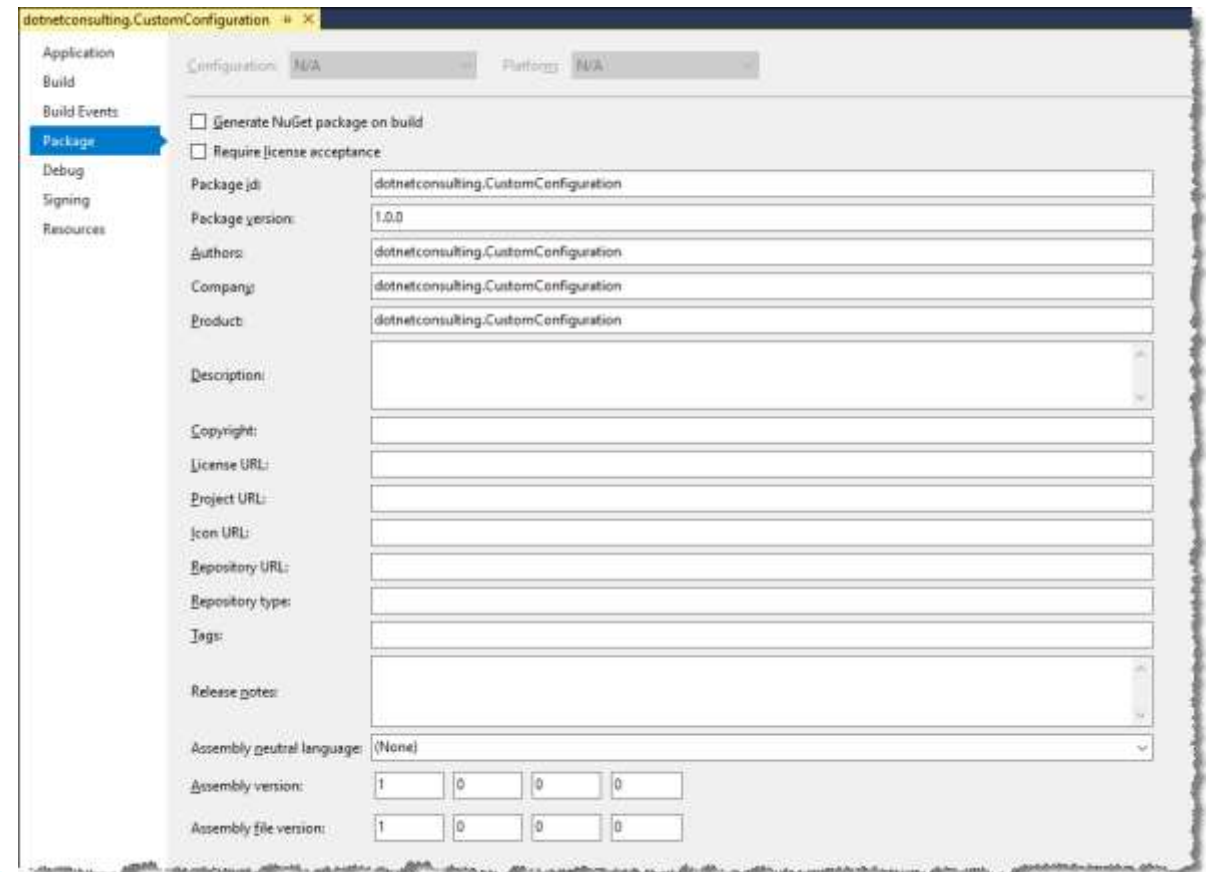
**Windows Application Packaging Project**
A project that creates packages containing Windows applications for side-loading or distribution via the Microsoft Store.
C# Windows UWP Desktop



<https://montemagno.com/distributing-a-net-core-3-wpf-and-winforms-app-with-msix/>

Visual Studio 2019+: Pack

- Erstellt ein NuGet-Paket
 - `nuget.exe` aus VS heraus
- Details:
 - Projekt -> Properties -> Package



Demo

.NET Core CLI

Framework-dependent Deployment

```
dotnet publish -c Release
```

Self-contained Deployment

```
dotnet publish -c Release -r win10-x64  
dotnet publish -c Release -r ubuntu.16.10-x64  
dotnet publish -c Release -r osx.10.11-x64
```

 Demo 

Fragen? Jetzt oder später!



www.dotnetconsulting.eu

Links



@Tkansy



tkansy@dotnetconsulting.eu



www.dotnetconsulting.eu

Links



<https://www.visualstudio.com/de/downloads>



<https://github.com/ElectronNET/Electron.NET>



<https://github.com/AvaloniaUI/Avalonia>



<https://github.com/mellinoe/ImGui.NET>