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

Microsoft® SQL Server®

.NET Core

ASP.NET Core

Azure Cosmos DB

DOTNETCONSULTING

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

.NET Core Roadmap

Upcoming Ship Dates

Milestone	Release Date
.NET Core 2.2.x, 2.1.x, 1.x (servicing)	Approximately every 1-2 months or as needed (see also releases)
.NET Core 3.0	Preview releases Final version in second half of 2019 - see Schedule in Preview 3 (from 2019/3). Ship date will be announced at the Build 2019 conference. Previous announcements: original 2018/5 and update 2018/10.

Note: Dates are calendar year (as opposed to fiscal year).

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

Update: Build 2019

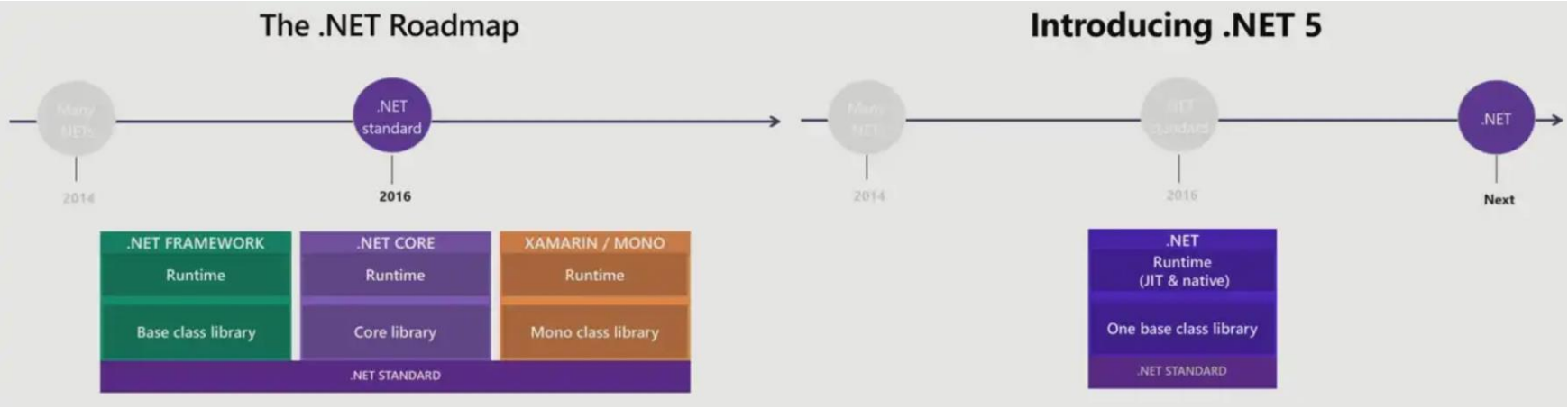
06-08.05.2019

- .NET Core 3.0 (@.NET Conf 23.09.2019)
- .NET Core 3.1 (03.12.2019)
- **.NET 5.0 (10.11.2020)**
- Major releases every year
- LTS (Long Term Support, 3 Jahre) grade Versionen

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

Quelle: MicrosoftBuild

Update: Build 2019



.NET Framework + .NET Core + Mono/ Xamerin

=

.NET 5.0

DOTNETCONSULTING

Quelle: MicrosoftBuild

Demo

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

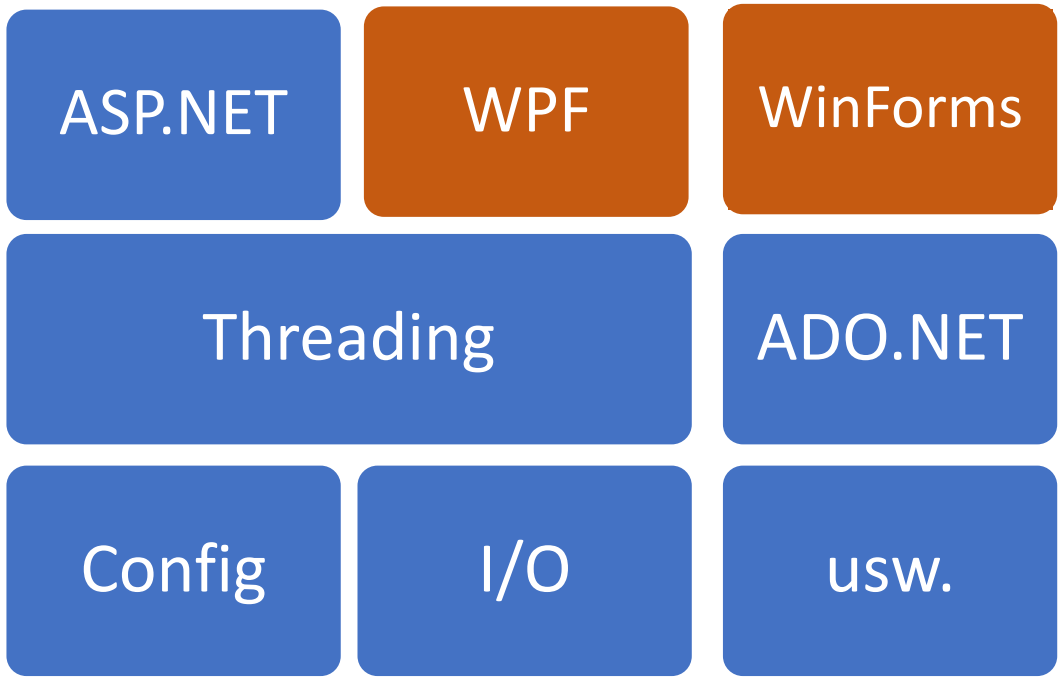
Zukunft von .NET Framework

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

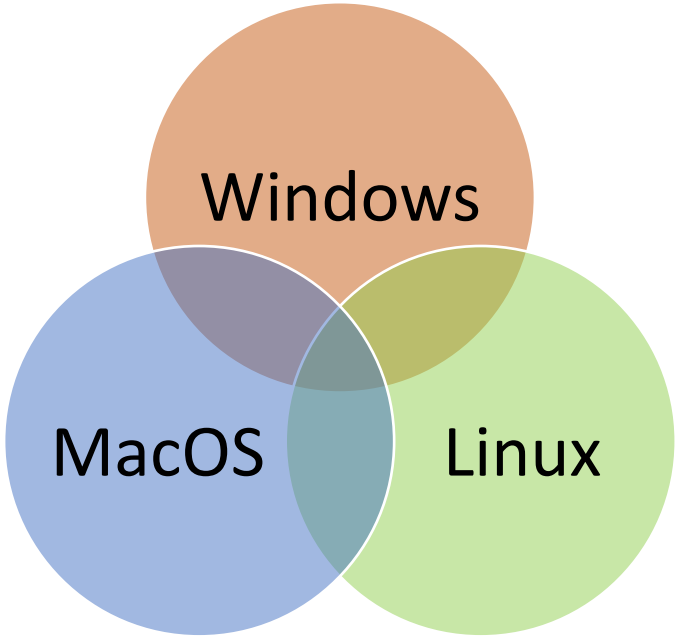
Einführung

.NET Core-Aufbau

Modular

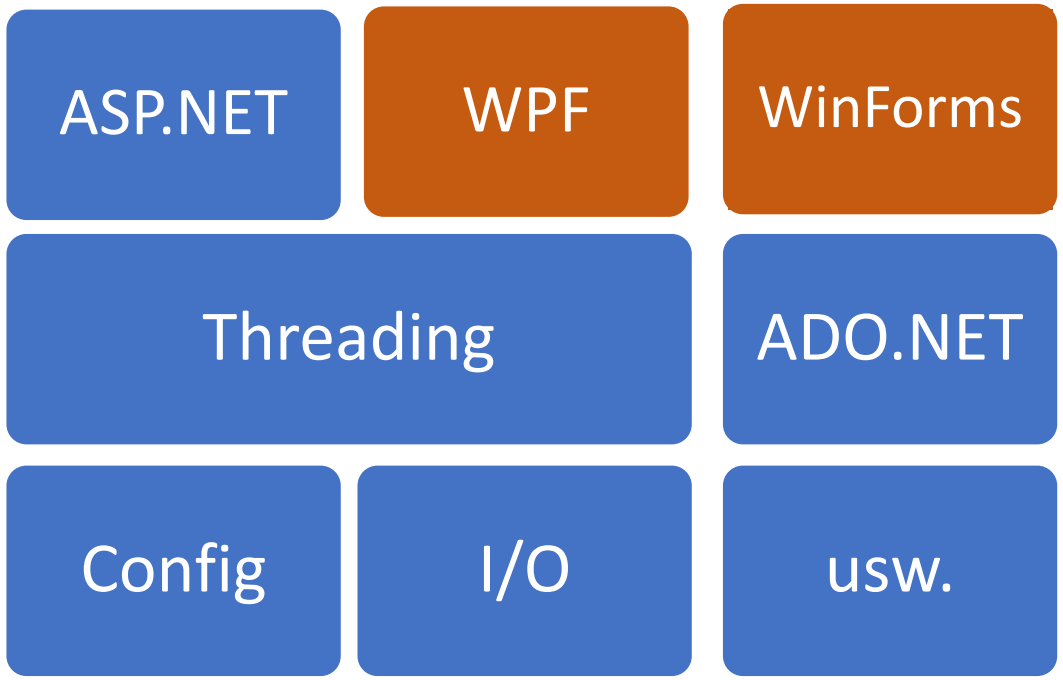


Multi Plattform

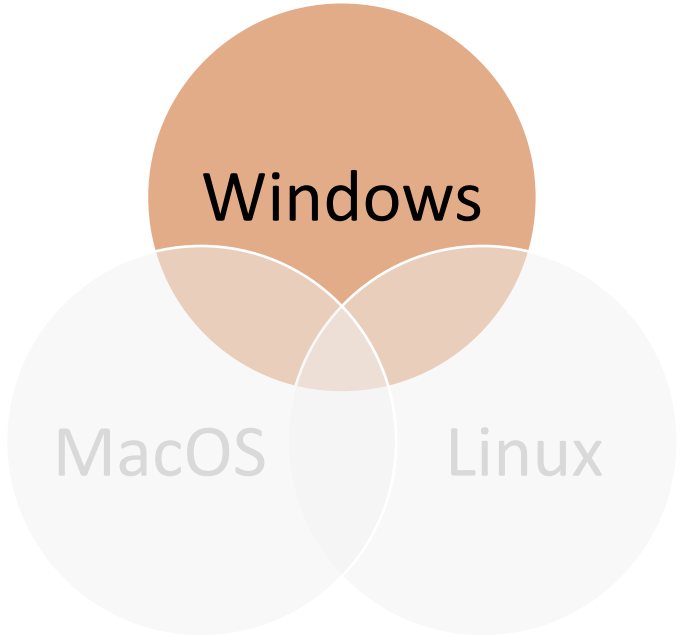


WPF unter .NET Core-Aufbau

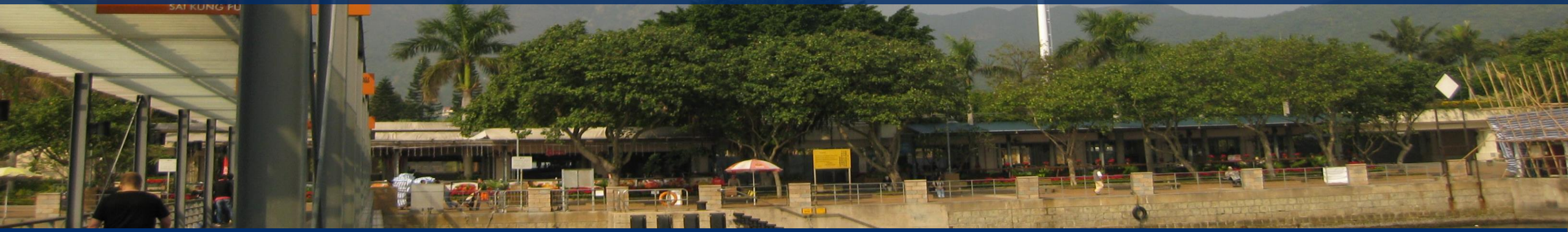
Modular



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

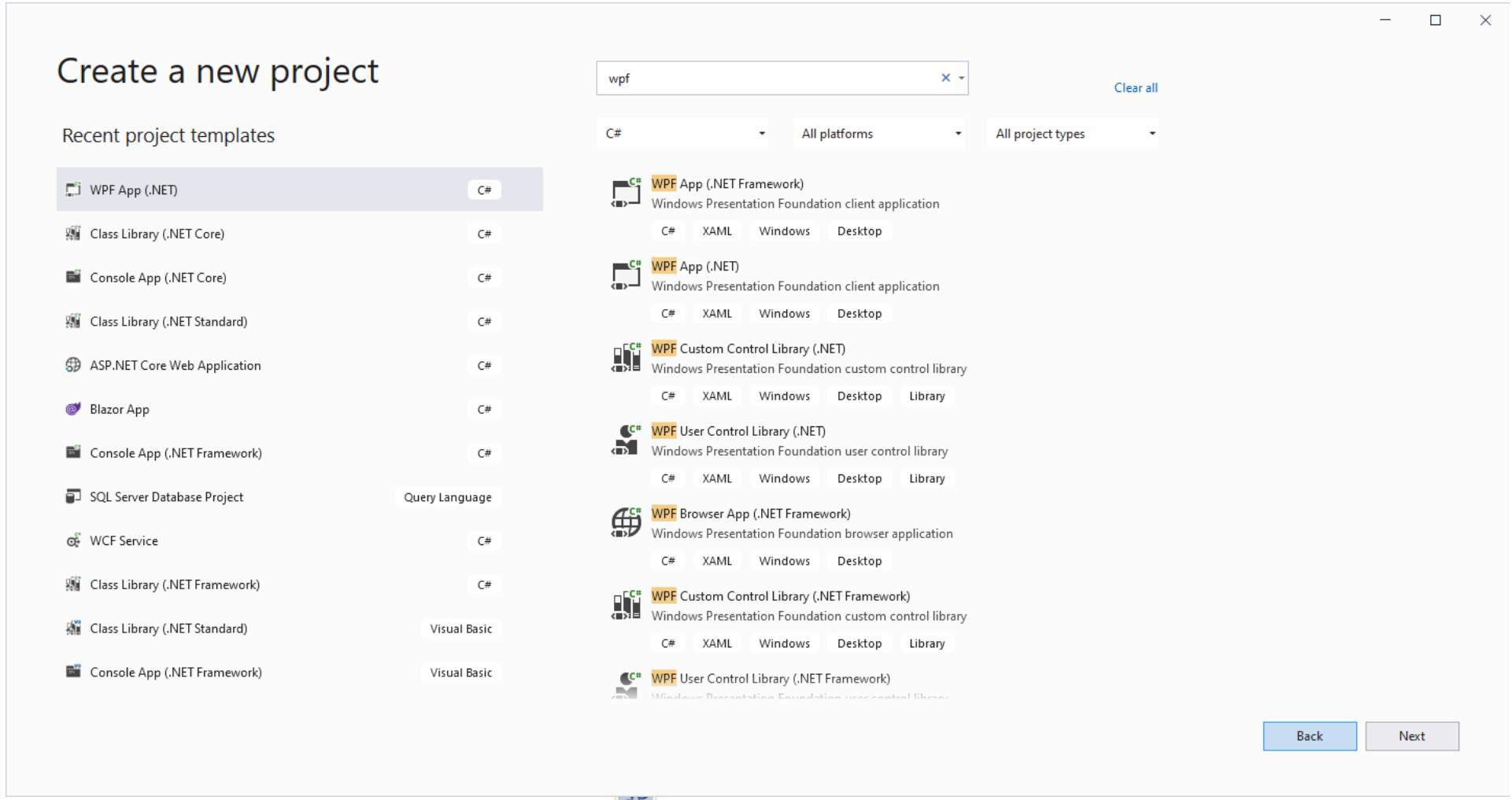
- Gleiches gilt für WinForms
  - Basiert allerdings auf GDI+

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
```

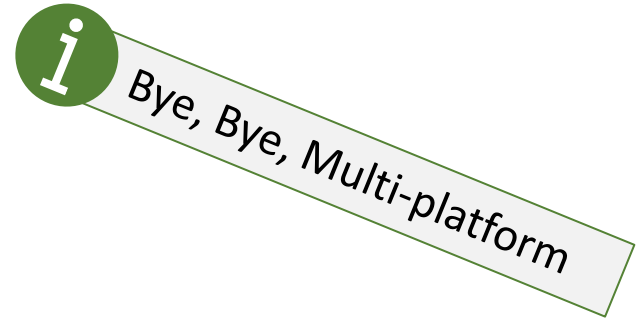
Developer Command Prompt for VS 2019

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

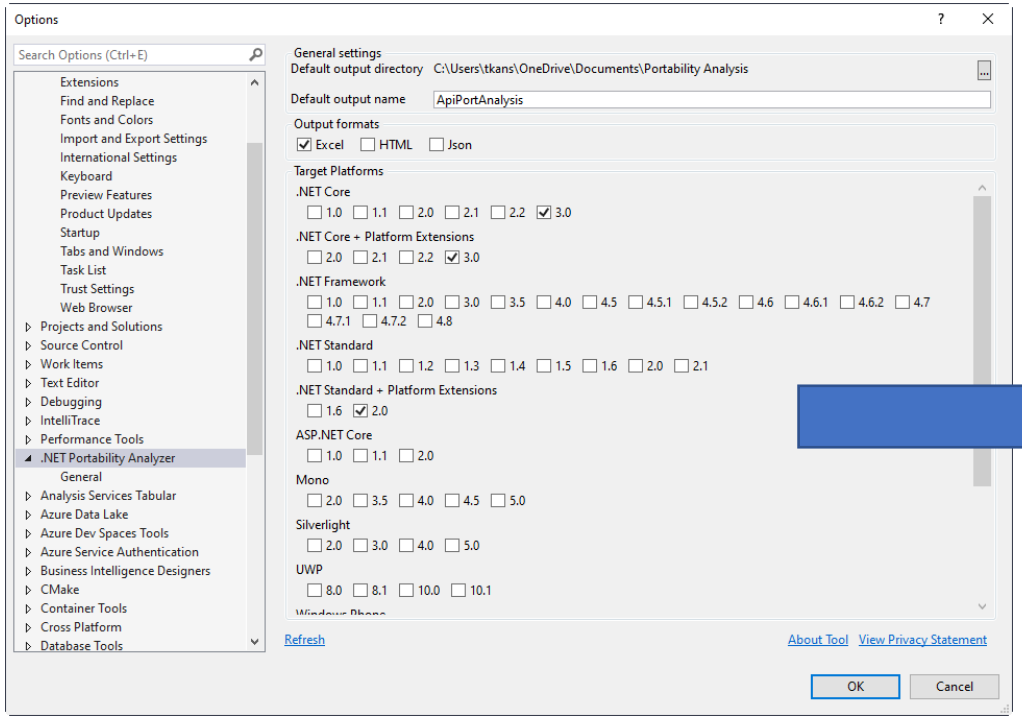
Migration zu .NET Core

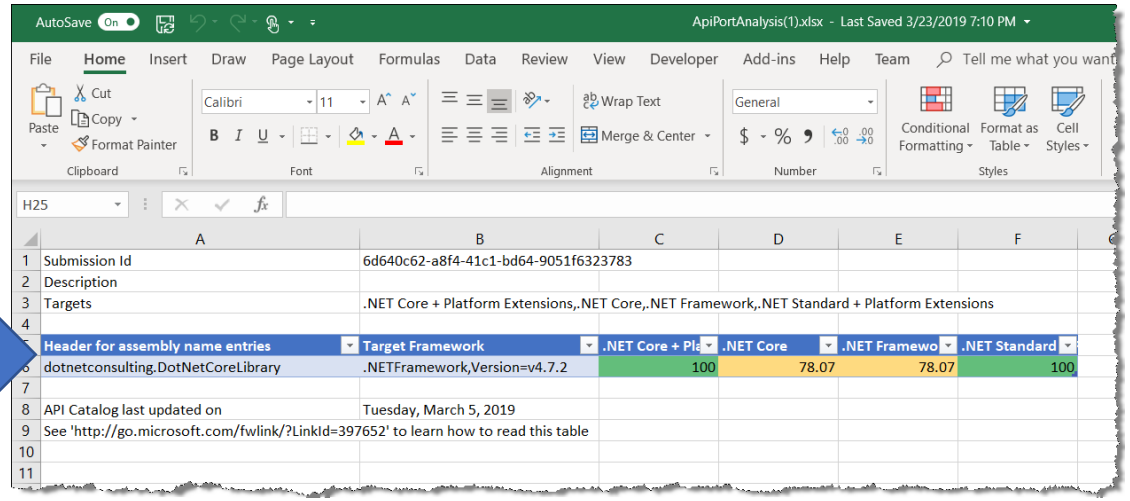
1. Aufwand abschätzen mit Analyzers
   - API Portability Analyzer Tool
   - NET Core 3.0 Desktop Api Analyzer
2. Dritthersteller (UI-)Komponenten prüfen
3. Projekte auf .NET Framework 4.6.2+ migrieren
4. Testportierung
5. Testdepolyment
6. …

```
https://github.com/Microsoft/dotnet-apiport/
```

.NET Portability Analyzer

Visual Studio Plug-in für 2015/ 2017/ 2019



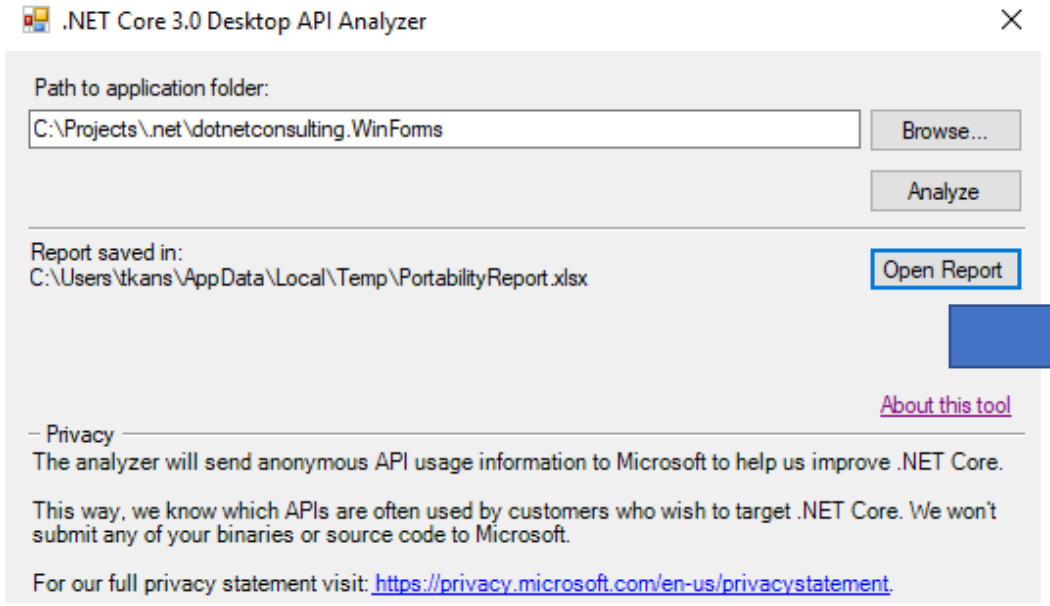


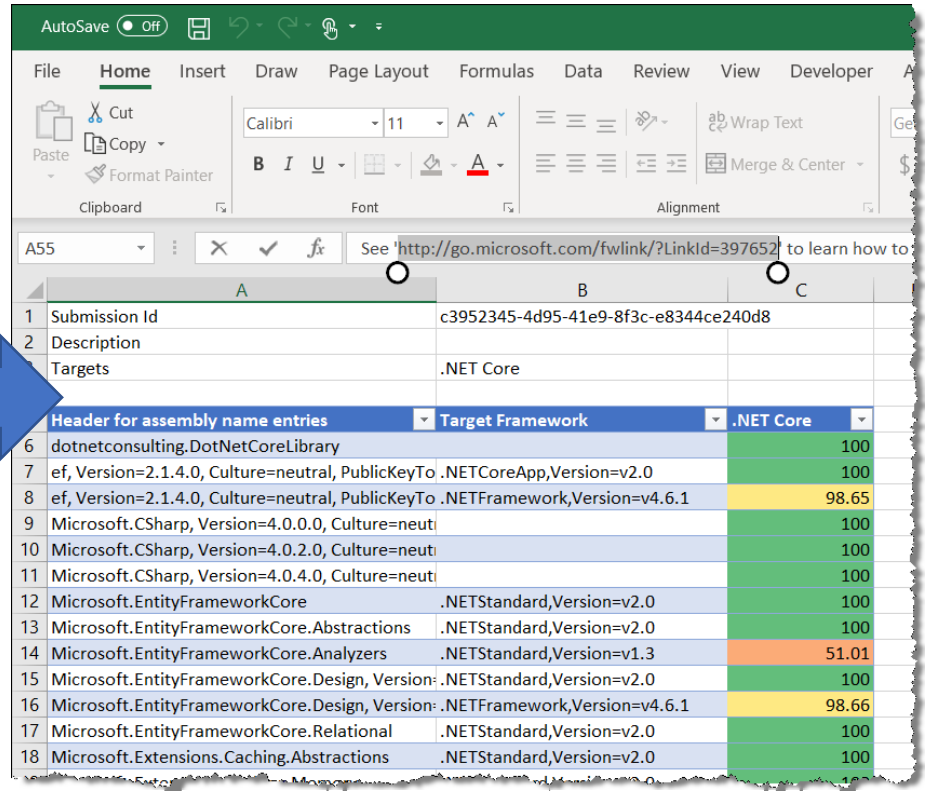
https://docs.microsoft.com/en-us/dotnet/standard/analyzers/portability-analyzer

DOTNETCONSULTING

Demo

.NET Desktop Api Analyzer





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

Demo

GUI Migration

Kein Assistent & viel Arbeit

- 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? Nicht bei Real Live-Projekten…

```
https://github.com/dotnet/try-convert
```

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

glob-Pattern

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

Demo

Test Migration

- Update NuGet dependencies
  - Standard oder .NET Core
- Update Project SDK-style Format
- Fix build issues
- Runtime testing

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

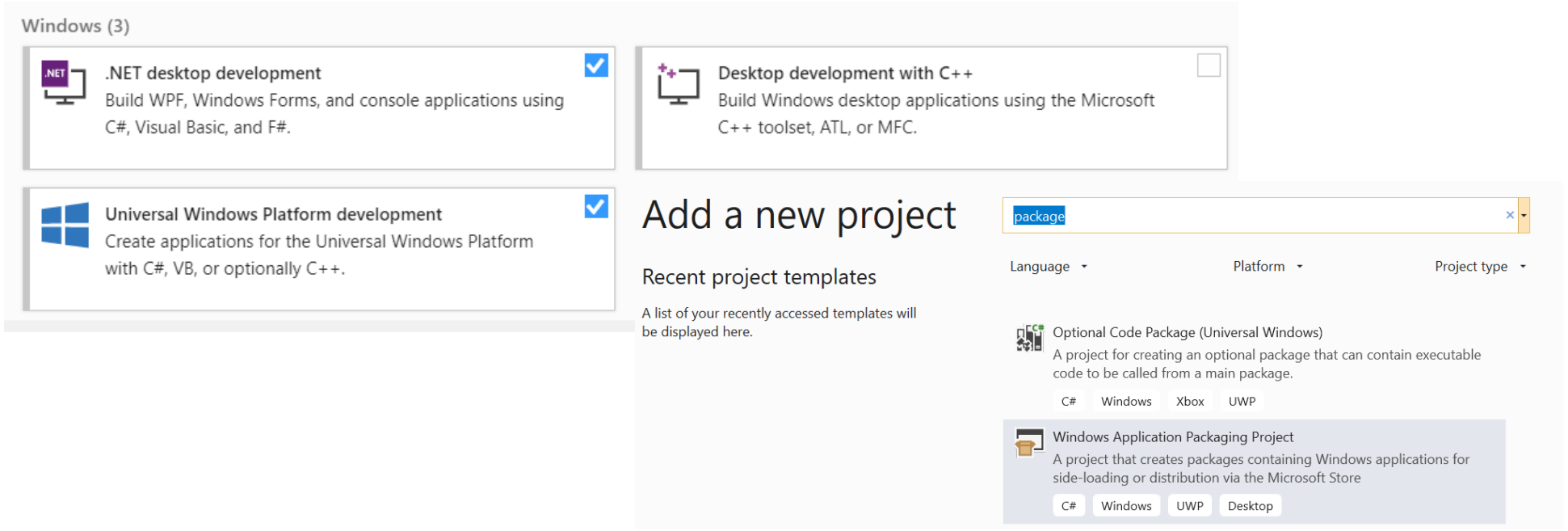
MSIX

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/
```

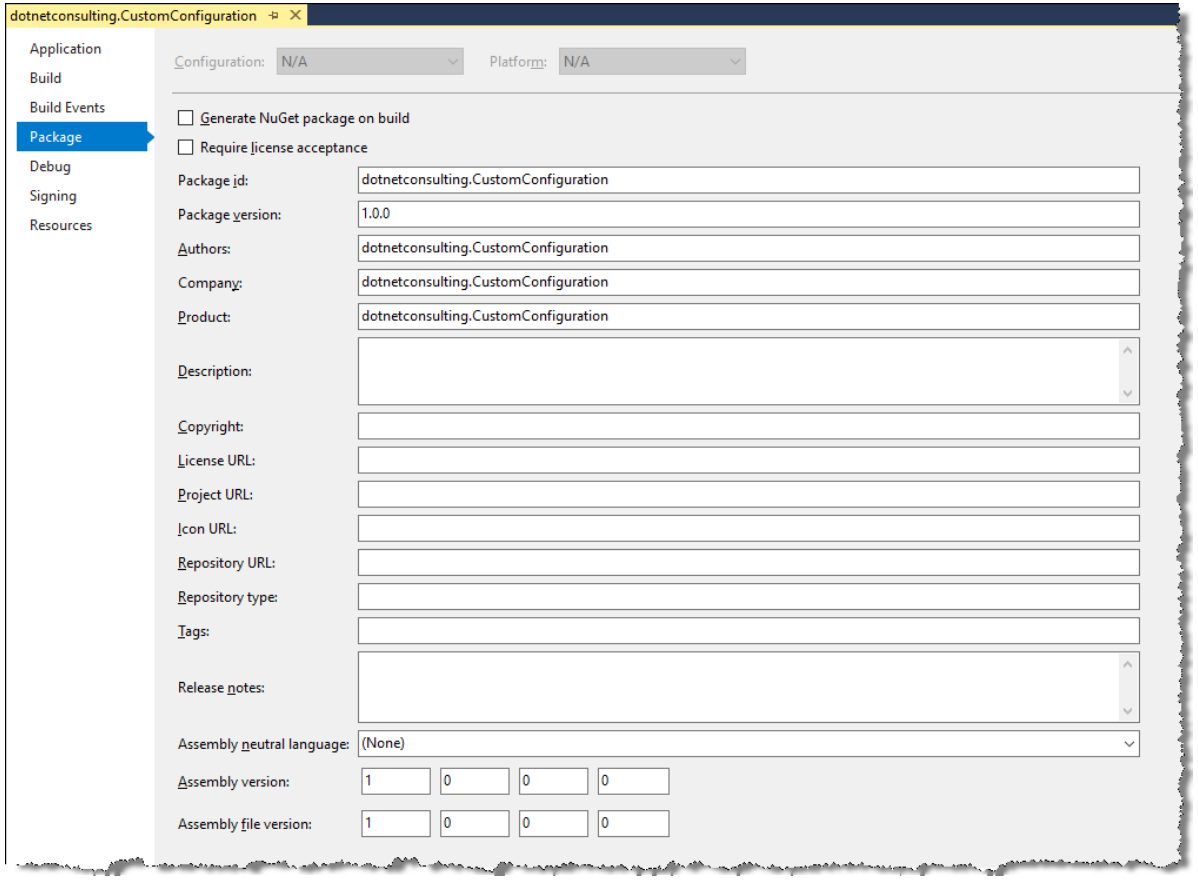
MSIX



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?

Links

@Tkansy

tkansy@dotnetconsulting.eu

www.dotnetconsulting.eu

DOTNETCONSULTING

Links

```
https://www.visualstudio.com/de/downloads
```

```
https://github.com/ElectronNET/Electron.NET
```

```
https://github.com/AvaloniaUI/Avalonia
```

```
https://github.com/mellinoe/ImGui.NET
```