

Online on-demand Project Support



Aus der Praxis WPF und .NET



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 Core
- Was fehlt?
- Windows Compatibility Pack
- WPF & moderne Features
- Clickonce Deployment
- Migration

WPF unter .NET

The background image shows a modern building with a glass facade and a dark metal frame, situated on a waterfront. The building is partially obscured by a blue horizontal band that contains the title text. In the foreground, there is a body of water reflecting the sky. The sky is a clear, light blue with some faint clouds. In the distance, there are green trees and a tall, thin tower. The overall scene is bright and clear, suggesting a sunny day.

Der .NET (Core) Roadmap

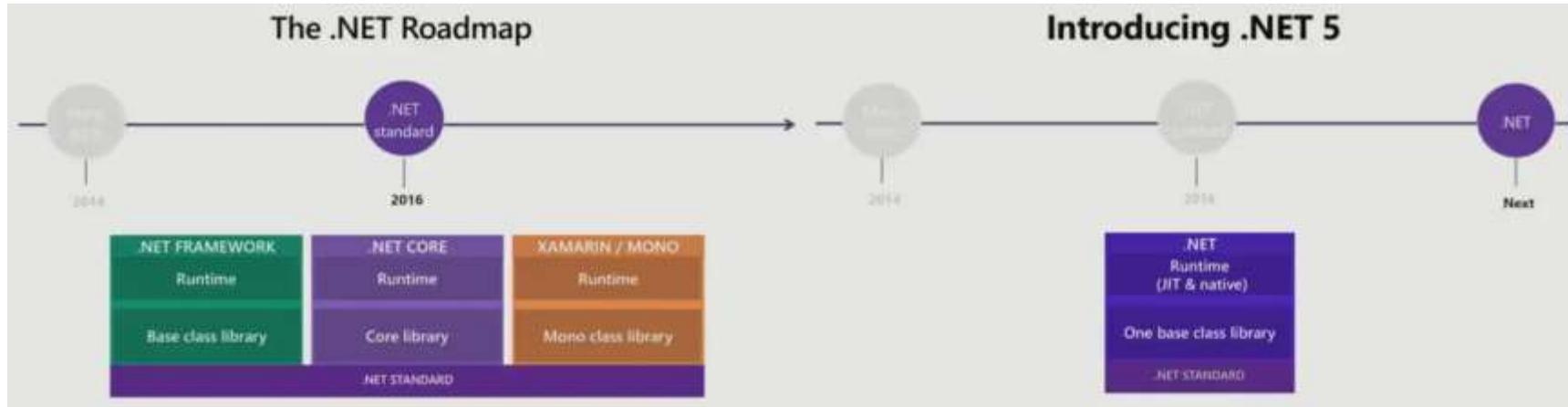


- .NET Core 3.1 (03.12.2019), WPF ab .NET 3.0
- .NET 5.0 (10.11.2020)
- .NET 6 (06.11.2021)
- Major releases every year
- LTS (Long Term Support, 3 Jahre) grade Versionen



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

.NET 6



.NET Framework + .NET Core + Mono/ Xamerin

=

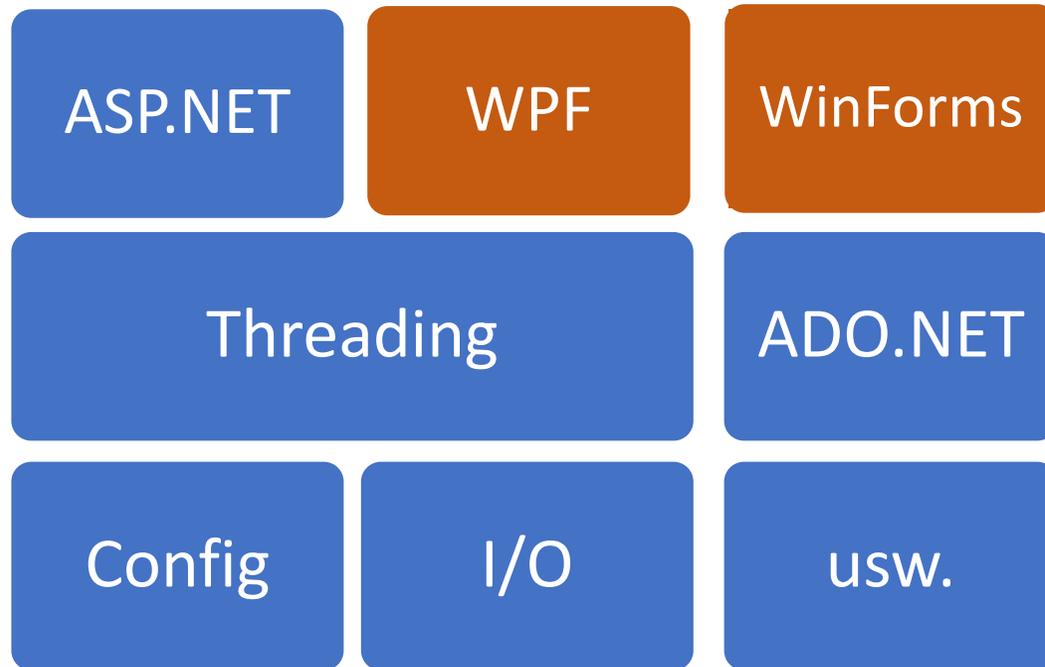
.NET 6 (The one and only)

Zukunft von .NET Framework

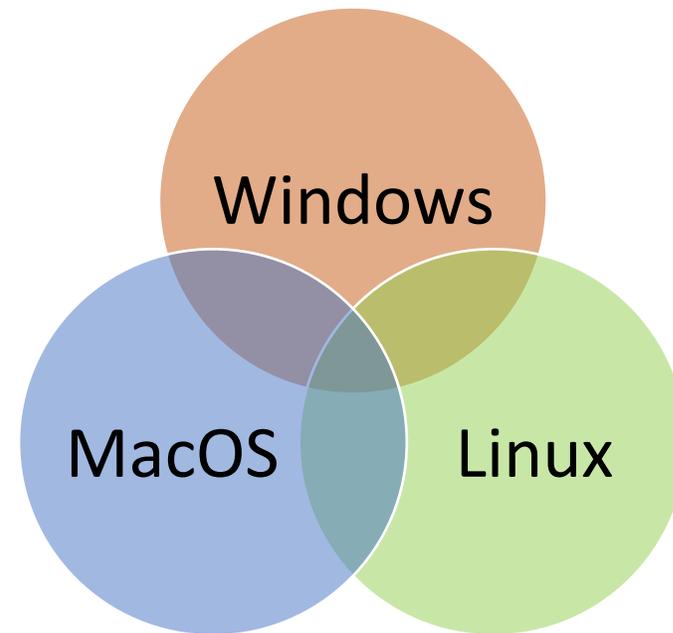
- .NET Framework 4.8 ist die letzte .NET Framework Feature-Version
- Migration auf 4.8 nur bei neuen Features notwendig
- Ende 2028? Wohl doch nicht

.NET Core-Aufbau

Modular

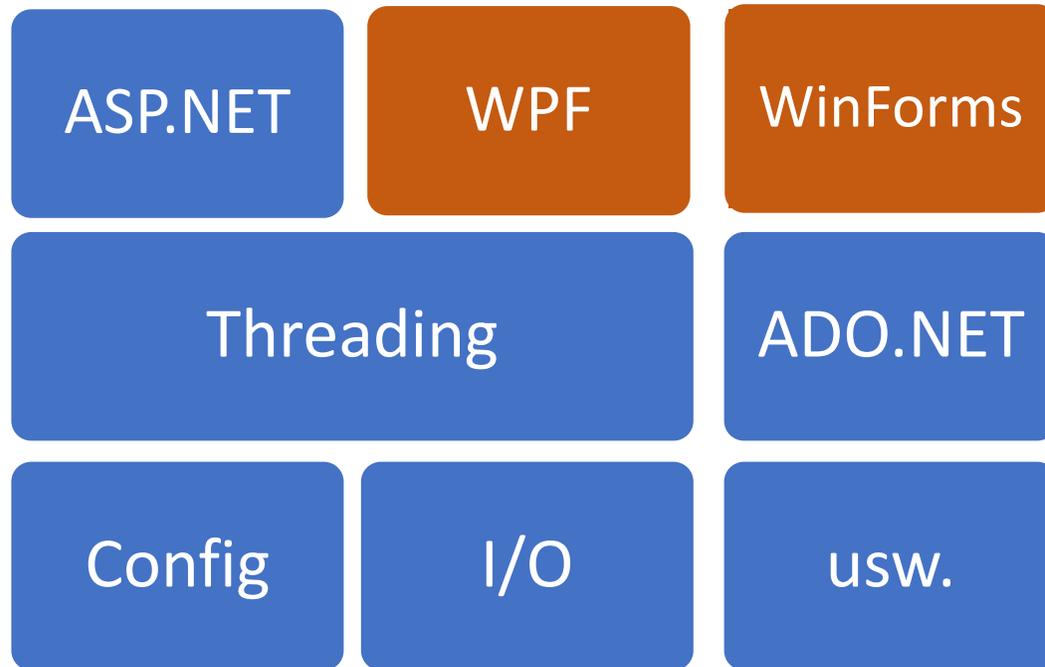


Multi Plattform

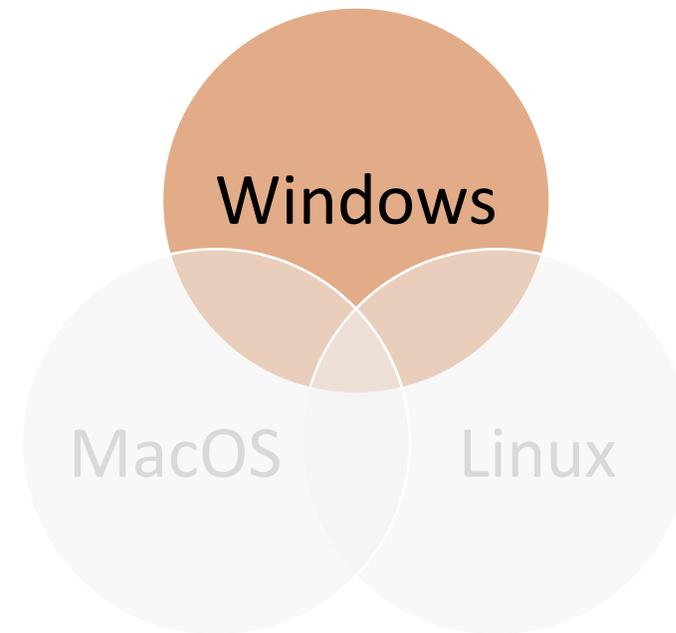


WPF unter .NET

Modular



Multi Plattform



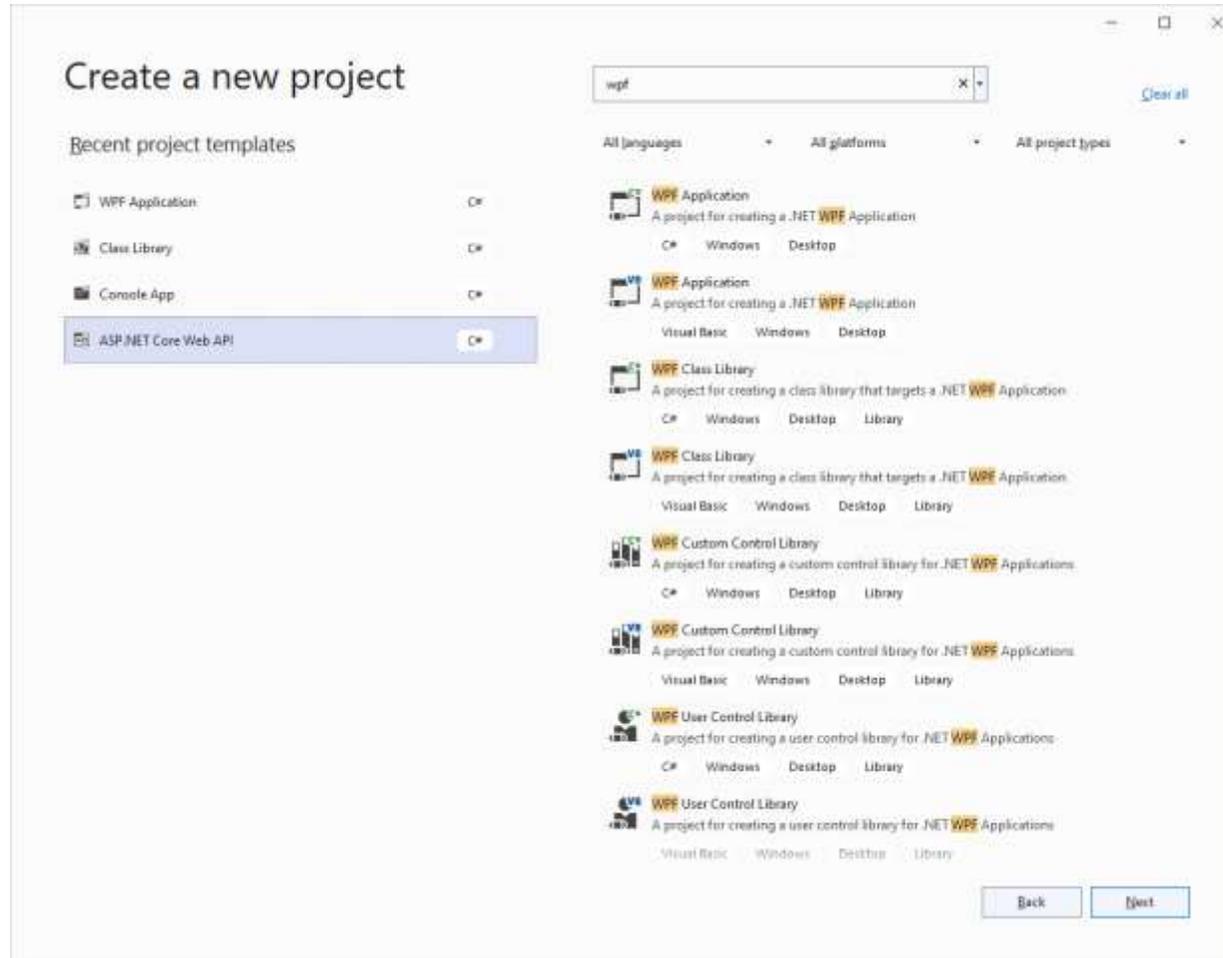
WPF unter .NET

- 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

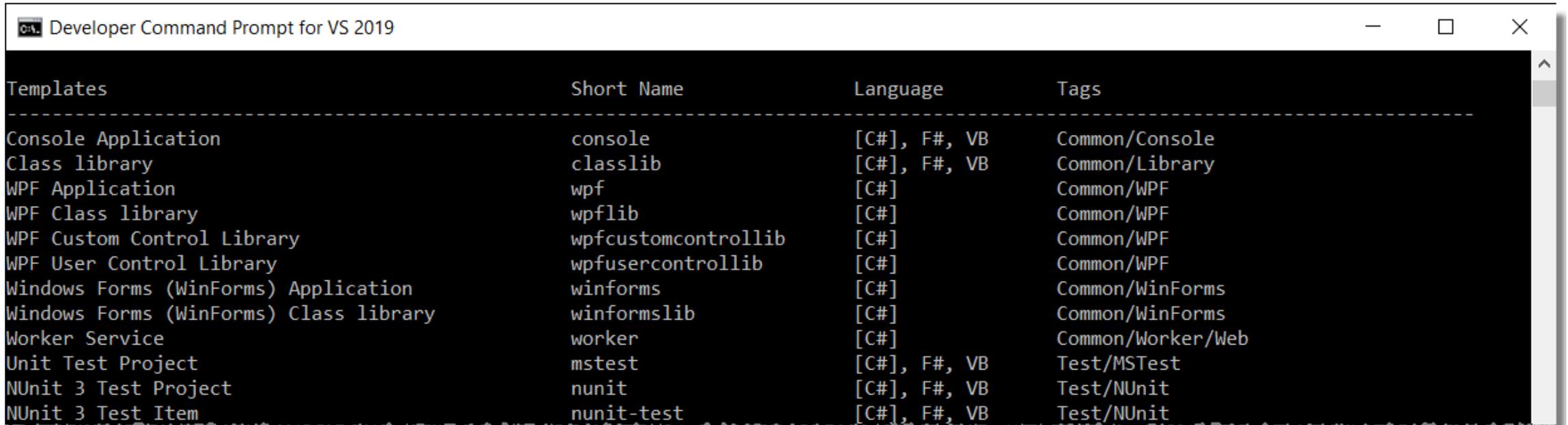
- XAML & Code (behind) sind identisch
- Unterschied .NET Framework vs .NET Core
 - keine App.Config
 - Projektdatei in SDK-Style

Visual Studio 2022+



.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



Was fehlt?



Was fehlt?

- Code Access Security (CAS) => Full trust

Source assembly	Target assembly	Type
<i>WindowsBase.dll</i>	<i>System.Security.Permissions.dll</i>	MediaPermission MediaPermissionAttribute MediaPermissionAudio MediaPermissionImage MediaPermissionVideo WebBrowserPermission WebBrowserPermissionAttribute WebBrowserPermissionLevel
<i>System.Xaml.dll</i>	<i>System.Security.Permissions.dll</i>	XamlLoadPermission
<i>System.Xaml.dll</i>	<i>System.Windows.Extension.dll</i>	XamlAccessLevel



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>



WPF & Moderne Features



Moderne Features

Moderne Features:

- Dependency Injection
- Settings
- Logging

- “Plain” `IServiceProvider` `ServiceProvider`
 - Funktional & einfach, aber eher unelegant
- `GenericHost`
 - Standardisiertes Vorgehen
 - Kontrollierter Shutdown



“Plain” IServiceProvider



“Plain” IServiceProvider

```
public partial class App : Application
{
    7 references | Thorsten, 1 day ago | 1 author, 1 change
    public static IServiceProvider ServiceProvider { get; private set; }

    1 reference | Thorsten, 1 day ago | 1 author, 1 change
    private void Application_Startup(object sender, StartupEventArgs e)
    {
        // IoC konfigurieren
        IServiceCollection serviceCollection = new ServiceCollection();

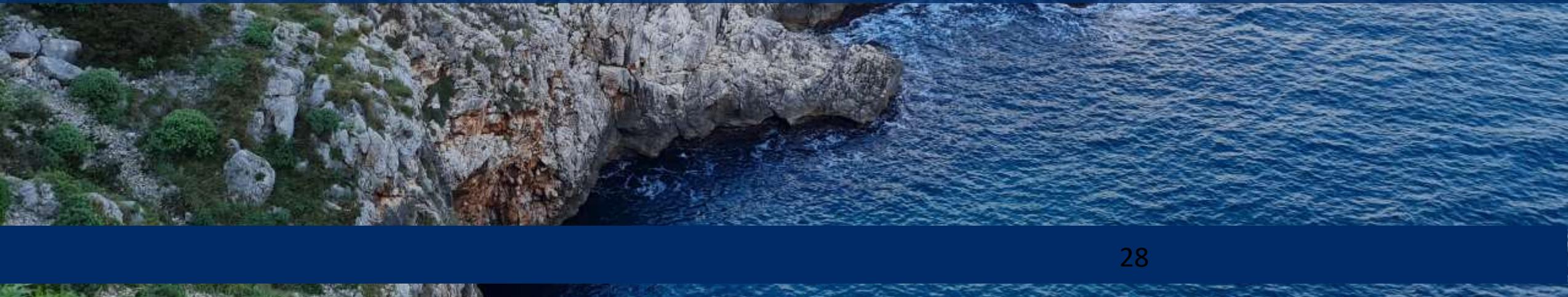
        ConfigureServices(serviceCollection);
        ConfigureViewModels(serviceCollection);
        ConfigureLogging(serviceCollection);

        ServiceProvider = serviceCollection.BuildServiceProvider();
    }
}
```

 Demo 



Generic Host





HostBuilder einrichten

```
_host = new HostBuilder()  
    .ConfigureAppConfiguration((context, configurationBuilder) =>  
    {  
        configurationBuilder.AddJsonFile("CustomSettings.json", optional: true);  
    })  
    .ConfigureServices((context, services) =>  
    {  
        //services.AddScoped();  
        services.AddTransient<MainWindow>();  
        //services.AddSingleton();  
    })  
    .ConfigureLogging(logging =>  
    {  
        logging.AddConsole();  
    })  
    .Build();
```

Settings

```
_host = new HostBuilder()  
    .ConfigureAppConfiguration((context, configurationBuilder) =>  
    {  
        configurationBuilder.AddJsonFile("CustomSettings.json", optional: true);  
    })  
    .ConfigureServices((context, services) =>  
    {  
        //services.AddScoped();  
        services.AddTransient<MainWindow>();  
        //services.AddSingleton();  
    })  
    .ConfigureLogging(logging =>  
    {  
        logging.AddConsole();  
    })  
    .Build();
```

Service Provider

```
_host = new HostBuilder()  
    .ConfigureAppConfiguration((context, configurationBuilder) =>  
    {  
        configurationBuilder.AddJsonFile("CustomSettings.json", optional: true);  
    })  
    .ConfigureServices((context, services) =>  
    {  
        //services.AddScoped();  
        services.AddTransient<MainWindow>();  
        //services.AddSingleton();  
    })  
    .ConfigureLogging(logging =>  
    {  
        logging.AddConsole();  
    })  
    .Build();
```

Logging

```
_host = new HostBuilder()  
    .ConfigureAppConfiguration((context, configurationBuilder) =>  
    {  
        configurationBuilder.AddJsonFile("CustomSettings.json", optional: true);  
    })  
    .ConfigureServices((context, services) =>  
    {  
        //services.AddScoped();  
        services.AddTransient<MainWindow>();  
        //services.AddSingleton();  
    })  
    .ConfigureLogging(logging =>  
    {  
        logging.AddConsole();  
    })  
    .Build();
```

Startup & Exit

```
private async void Application_Startup(object sender, StartupEventArgs e)
{
    await _host.StartAsync();

    var mainWindow = _host.Services.GetRequiredService<MainWindow>();
    mainWindow.Show();
}
```

```
private async void Application_Exit(object sender, ExitEventArgs e)
{
    using (_host)
    {
        await _host.StopAsync(TimeSpan.FromSeconds(3));
    }
}
```

 Demo 



Dependency Injection

Dependency Injection

- Lebenszeit von Instanzen
- DI-Scope

Lebenszeit von Instanzen

Variante	Lebenszeit
<code>AddTransient()</code>	Der IoC-Container liefert jedes Mal eine neu erzeugte Instanz
<code>AddScoped()</code>	Während einer Anfrage (z. B. an einen ASP.NET-Controller) wird die gleiche Instanz geliefert. Bei der nächsten Anfrage wird wiederum eine neue Instanz geliefert
<code>AddSingleton()</code>	Der IoC-Container liefert jedes Mal die gleiche Instanz zurück



DI-Scope

```
// Scope erzeugen (via ASP.NET Core als Request)
using (IServiceScope scope = services.CreateScope())
{
    ISmtpService smtpServiceScope =
        scope.ServiceProvider.GetService<ISmtpService>();
    IOrderService orderServiceScope =
        services.GetService<IOrderService>();

    // ...
}
```



Settings



Konfigurationsprovider

Provider	NuGet-Paket
JSON-Datei	<code>Microsoft.Extensions.Configuration.Json</code>
XML-Datei	<code>Microsoft.Extensions.Configuration.Xml</code>
INI-Datei	<code>Microsoft.Extensions.Configuration.Ini</code>
Umgebungsvariablen	<code>Microsoft.Extensions.Configuration.EnvironmentVariables</code>
Programmargumente	<code>Microsoft.Extensions.Configuration.CommandLine</code>
In-Memory-Collection	–
Azure Key Vault	<code>Microsoft.Extensions.Configuration.AzureKeyVault</code>
User Secrets (nur ASP.NET Core)	<code>Microsoft.Extensions.Configuration.UserSecrets</code>
Custom	–

```
Install-Package Microsoft.Extensions.Configuration.Json (s.o.)
```

Konfiguration verwenden

```
// Konfiguration vorbereiten, Provider nach Bedarf anfügen
 IConfigurationBuilder builder = new ConfigurationBuilder()
     // Umgebungsvariablen hinzufügen
     .AddEnvironmentVariables()
     // Json-Datei hinzufügen
     .AddJsonFile("Config.json");

// Konfiguration abschließen, Werte einlesen
 IConfigurationRoot config = builder.Build();

// Zugriff
 string windowHeight = config["App:Window:Height"];
```



Konfiguration via DI zur Verfügung stellen

IOptions-Pattern

```
private static void ConfigureServices(...)
{
    ...
    IConfigurationSection doSomethingJobConfigurationSection =
        hostBuilderContext.Configuration.GetSection("Jobs:DoSomethingJob");
    services.Configure<DoSomethingJobSettings>(doSomethingJobConfigurationSection);
    ...
}

public DoSomethingJob(IOptions<DoSomethingJobSettings> doSomethingJobSettings) // Konstruktor
{
    _doSomethingJobSettings = doSomethingJobSettings.Value;
}
```



Logging



Logging-Provider

Level	NuGet-Paket
Konsole	<code>Microsoft.Extensions.Logging.Console</code>
Json Konsole (structured)	<code>Microsoft.Extensions.Logging.Console</code>
Einfach Konsole (+Systemd)	<code>Microsoft.Extensions.Logging.Console</code>
Debugger-Monitor (Output window)	<code>Microsoft.Extensions.Logging.Debug</code>
Trace Listener (Output window)	<code>Microsoft.Extensions.Logging.TraceSource</code>
Windows-Ereignisprotokoll	<code>Microsoft.Extensions.Logging.EventLog</code>
EventSource/EventListener	<code>Microsoft.Extensions.Logging.EventSource</code>
Azure-App-Dienste „Diagnoseprotokolle“ und „Log Stream“	<code>Microsoft.Extensions.Logging.AzureAppServices</code>
Datei, Datenbank, SMTP etc.	<code>Serilog.* // NLog.* // usw.</code>
Custom	-



Logging in Action I

```
logger.Log(LogLevel.Trace, "Trace-Log");  
logger.Log(LogLevel.Debug, "Debug-Log");  
logger.Log(LogLevel.Information, "Information-Log");  
  
logger.LogWarning("Warning-Log");  
logger.LogError("Error-Log");  
logger.LogCritical("Critical-Log");
```



Direkt-Ausgabe in die Console



Direkt-Ausgabe in die Console

1 reference | 0 changes | 0 authors, 0 changes

```
private void button_Click(object sender, RoutedEventArgs e)
{
    Console.WriteLine($"Hallo Welt, {DateTime.Now}");
}
```

```
<PropertyGroup>
  <TargetFramework>net6.0-windows</TargetFramework>
  <Nullable>enable</Nullable>
  <UseWPF>>true</UseWPF>
</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)' == 'DEBUG' ">
  <OutputType>Exe</OutputType>
</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)' == 'RELEASE' ">
  <OutputType>WinExe</OutputType>
</PropertyGroup>
```

 Demo 

A large, circular fountain with a central sculpture of two figures embracing. The fountain is surrounded by a stone wall and a clear blue sky. The text "Clickonce Deployment" is overlaid on the image in a white font.

Clickonce Deployment



Clickonce Deployment

Publish

Where are you publishing today?

Target

-  **Azure**
Publish your application to the Microsoft cloud
-  **ClickOnce**
Publish your application with ClickOnce
-  **Docker Container Registry**
Publish your application to any supported Container Registry that works with Docker images
-  **Folder**
Publish your application to a local folder or file share

Back Next Finish Cancel

 Demo 



Migration



Migration zu .NET Core

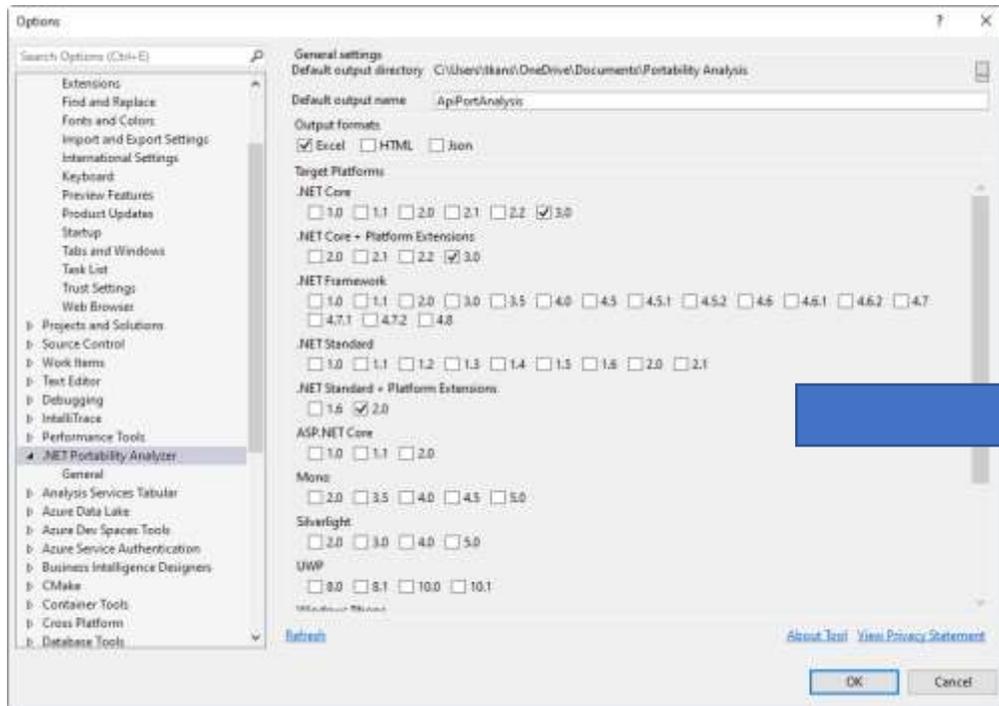
1. Aufwand abschätzen mit Analyzers
 - API Portability Analyzer Tool
 - NET Core 3.0 Desktop Api Analyzer
2. Dritthersteller (UI-)Komponenten prüfen
3. Projekte auf .NET Framework 4.6.2+ migrieren
4. Testportierung
5. Testdeployment
6. ...



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

.NET Portability Analyzer

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



AutoSave: On

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

File Home Insert Draw Page Layout Formulas Data Review View Developer Add-ins Help Team Tell me what you want

Clipboard Font Alignment Number Styles

H25

	A	B	C	D	E	F
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	Header for assembly name entries	Target Framework	.NET Core + Platform Extensions	.NET Core	.NET Framework	.NET Standard
5	dotnetconsulting.DotNetCoreLibrary	.NETFramework,Version=v4.7.2	100	78.07	78.07	100
6						
7	API Catalog last updated on	Tuesday, March 5, 2019				
8	See 'http://go.microsoft.com/fwlink/?LinkId=397652' to learn how to read this table					
9						
10						
11						

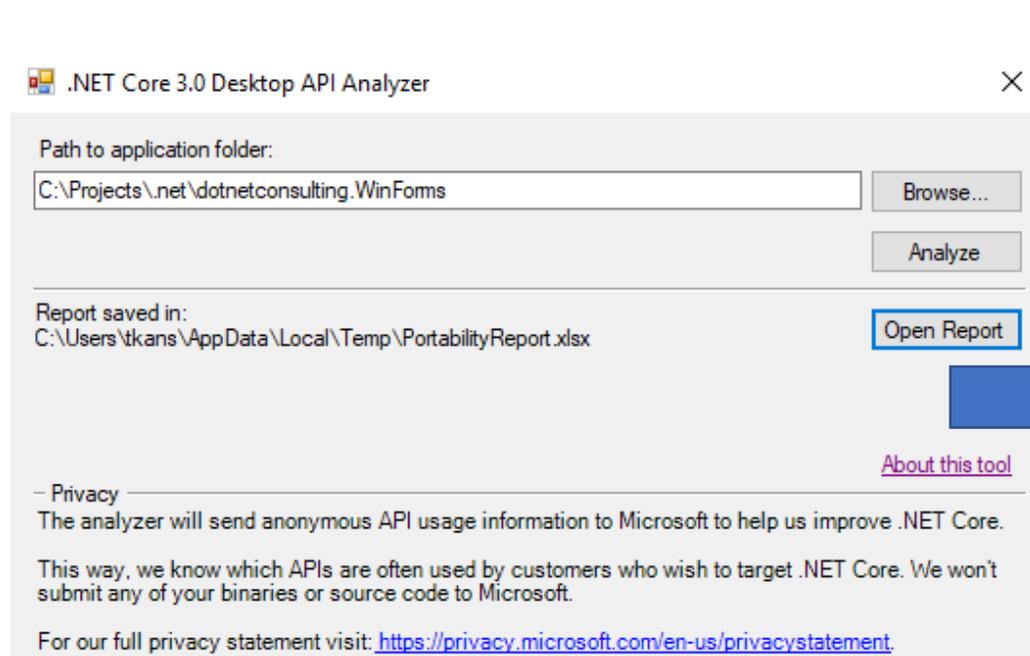


<https://docs.microsoft.com/en-us/dotnet/standard/analyzers/portability-analyzer>

 Demo 



.NET Desktop Api Analyzer



The screenshot shows an Excel spreadsheet with the following data:

Submission Id	Description	Targets
c3952345-4d95-41e9-8f3c-e8344ce240d8	.NET Core	
Header for assembly name entries		Target Framework
		.NET Core
dotnetconsulting.DotNetCoreLibrary		100
ef, Version=2.1.4.0, Culture=neutral, PublicKeyTo .NETCoreApp,Version=v2.0		100
ef, Version=2.1.4.0, Culture=neutral, PublicKeyTo .NETFramework,Version=v4.6.1		98.65
Microsoft.CSharp, Version=4.0.0.0, Culture=neut:		100
Microsoft.CSharp, Version=4.0.0.0, Culture=neut:		100
Microsoft.CSharp, Version=4.0.4.0, Culture=neut:		100
Microsoft.EntityFrameworkCore	.NETStandard,Version=v2.0	100
Microsoft.EntityFrameworkCore.Abstractions	.NETStandard,Version=v2.0	100
Microsoft.EntityFrameworkCore.Analyzers	.NETStandard,Version=v1.3	51.01
Microsoft.EntityFrameworkCore.Design, Version: .NETStandard,Version=v2.0		100
Microsoft.EntityFrameworkCore.Design, Version: .NETFramework,Version=v4.6.1		98.66
Microsoft.EntityFrameworkCore.Relational	.NETStandard,Version=v2.0	100
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 



NET Upgrade Assistant

```
dotnet tool install -g upgrade-assistant
```

```
upgrade-assistant analyze <Path to csproj or sln to upgrade>
```

Erzeugt einen SARIF-Report
(Static Analysis Results Interchange Format)

```
upgrade-assistant upgrade <Path to csproj or sln to upgrade>
```



https://marketplace.visualstudio.com/items?itemName=WDGIS.MicrosoftSarifViewer&WT.mc_id=dotnet-35129-website

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>

 Demo 

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>
- XAML: Microsoft.Xaml.Behaviors einfügen
 - Microsoft.Xaml.Interactivity
 - Microsoft.Xaml.Interactions



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

 Demo 

Fragen? Jetzt oder später!



www.dotnetconsulting.eu

Links



@Tkansy



tkansy@dotnetconsulting.eu



www.dotnetconsulting.eu

Links



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