DOTNEXT

Discovering .NET 5

Raffaele Rialdi - Senior Software Architect

@raffaeler
raffaeler@vevy.com

Who am I?

- Raffaele Rialdi, Senior Software Architect in Vevy Europe – Italy
  - @raffaeler also known as "Raf"
- Consultant in many industries
  - Manufacturing, racing, healthcare, financial, …
- Speaker and Trainer around the globe (development and security)
  - Italy, Romania, Bulgaria, Russia (Moscow, St Petersburg and Novosibirsk), USA, …
- And proud member of the great Microsoft MVP family since 2003

Discount code for my co-authored book

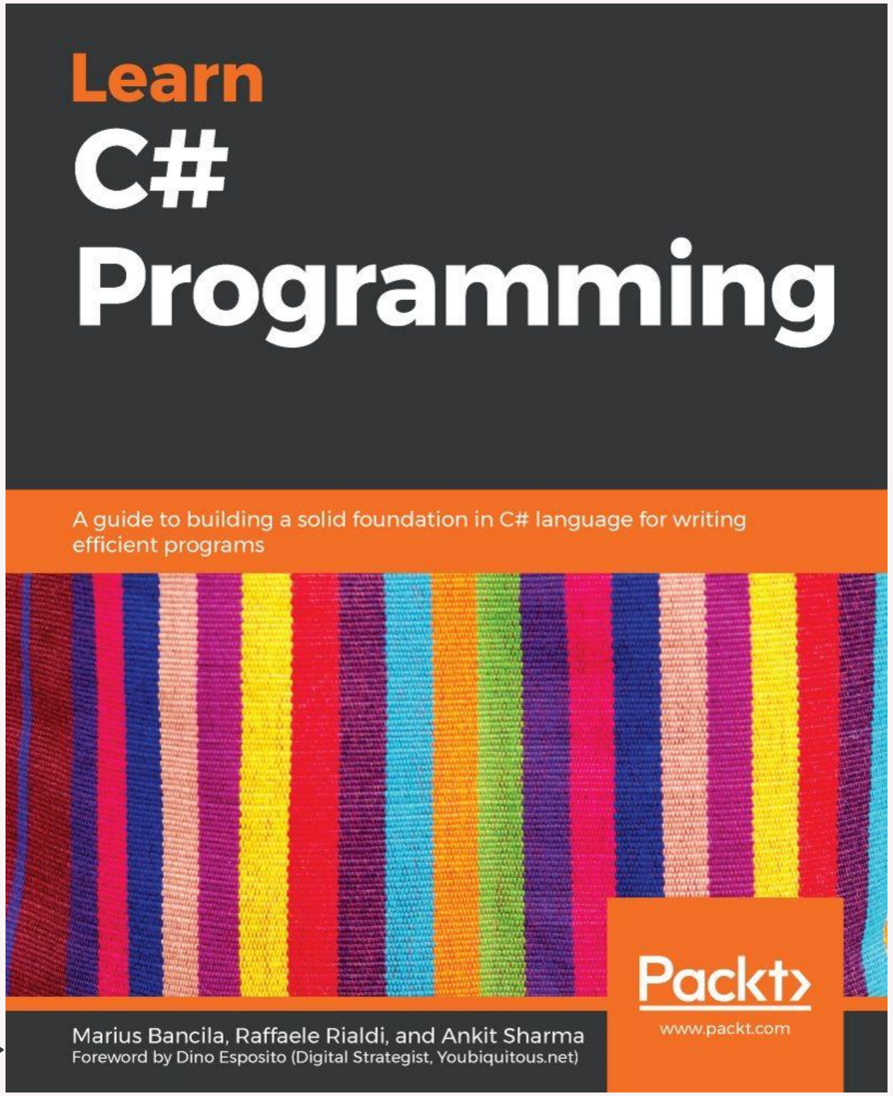
Learn C# Programming

Covers C#8 on .NET Core

- Amazon code: **25CSHARPBK**
  https://www.amazon.com/gp/mpc/AEL3ILD2QGU8K
- Packt code: **25CSHARPBK**
  https://www.packtpub.com/ (ebook)

Codes exipire on December 15

Marius Bancila, Raffaele Rialdi, and Ankit Sharma
Foreword by Dino Esposito (Digital Strategist, Youbiquitous.net)


Learn
C#
Programming
A guide to building a solid foundation in C# language for writing efficient programs
Packt>
www.packt.com
Marius Bancila, Raffaele Rialdi, and Ankit Sharma
Foreword by Dino Esposito (Digital Strategist, Youbiquitous.net)

Agenda

- How is .NET application development changing?
- (Not just the obvious) .NET 5 features
- Performance improvements at various levels
- Publishing improvements in size & bootstrap time

A new .NET era has come

- C# 9: we can count ~18 new features in C# 9
- Web development improvements
  - ASP.NET Core 5 is now the fastest framework!
  - OpenAPI, Swagger, OData, gRPC, …
- Windows development is back
- .NET / Container size has shrinked
- Development+diagnostics is better than ever

C# 9 Don't just stop at the main features!

- There are new types
  - New nint and nuint platform-dependent integers
  - System.Half, 16 bit floating-point
- Lambda discards
- Avoiding local captures
- Target typed new
- Attributes on local functions
- SkipLocalsInitAttribute
- …

```
Func<int, int, int> func = (_, _) => 0;
```

```
var s = x.Select(static n => n.ToString());
```

```
List<string> strings = new();
```

```
[SkipLocalsInit]
public unsafe void DoNotInitializeMemory()
{
    Span<int> storage = stackalloc int[10000];
```

C# 9 Covariant returns

```
interface IArchive { }
class FileArchive : IArchive { }
class MemoryArchive : IArchive { }
```

```
abstract class Storage
{
    public abstract string Name { get; }
    public abstract IArchive CreateArchive();
    public abstract IEnumerable<IArchive> ReadArchives();
}
```

The overriding method can declare a most derived return type

```
class FileStorage : Storage
{
    public override string Name => "File storage";
    public override FileArchive CreateArchive() => new FileArchive();
    public override IEnumerable<FileArchive> ReadArchives() => new List<FileArchive>() { /* ... */ };
}
```

.NET future is "Core" and starts now with .NET 5

- The Framework.NET will not evolve, but supported for long
  - Newest C# features are not available in the .NET Framework
- .NET is the new name for .NET Core

- NetStandard is still useful
  - Are you authoring public/nuget libraries?
    - Use the lowest netstandard version you can
  - Are you publishing apps?
    - Just use a simple .NET5 (not LTS) / .NET Core 3.1 (LTS) library

- In .NET 6 the mono/Xamarin/.NET runtimes will be merged

TFMs and Windows SDK

```
Supported SDKs

 10.0.17763.0
 10.0.18362.0
 10.0.19041.0
```

- .NET can now call any <u>headless</u> API in Windows
  - From .NET Core 3.1

Minimum OS Version

Windows SDK version

```
<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <SupportedOSPlatform>windows7</SupportedOSPlatform>
</PropertyGroup>
<ItemGroup>
  <PackageReference Include="Microsoft.Windows.SDK.Contracts" Version="10.0.19041.1" />
</ItemGroup>
```

  - From .NET 5

.NET + Windows SDK versions

```
<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>net5.0-windows10.0.19041.0</TargetFramework>
  <SupportedOSPlatform>windows7</SupportedOSPlatform>
</PropertyGroup>
```

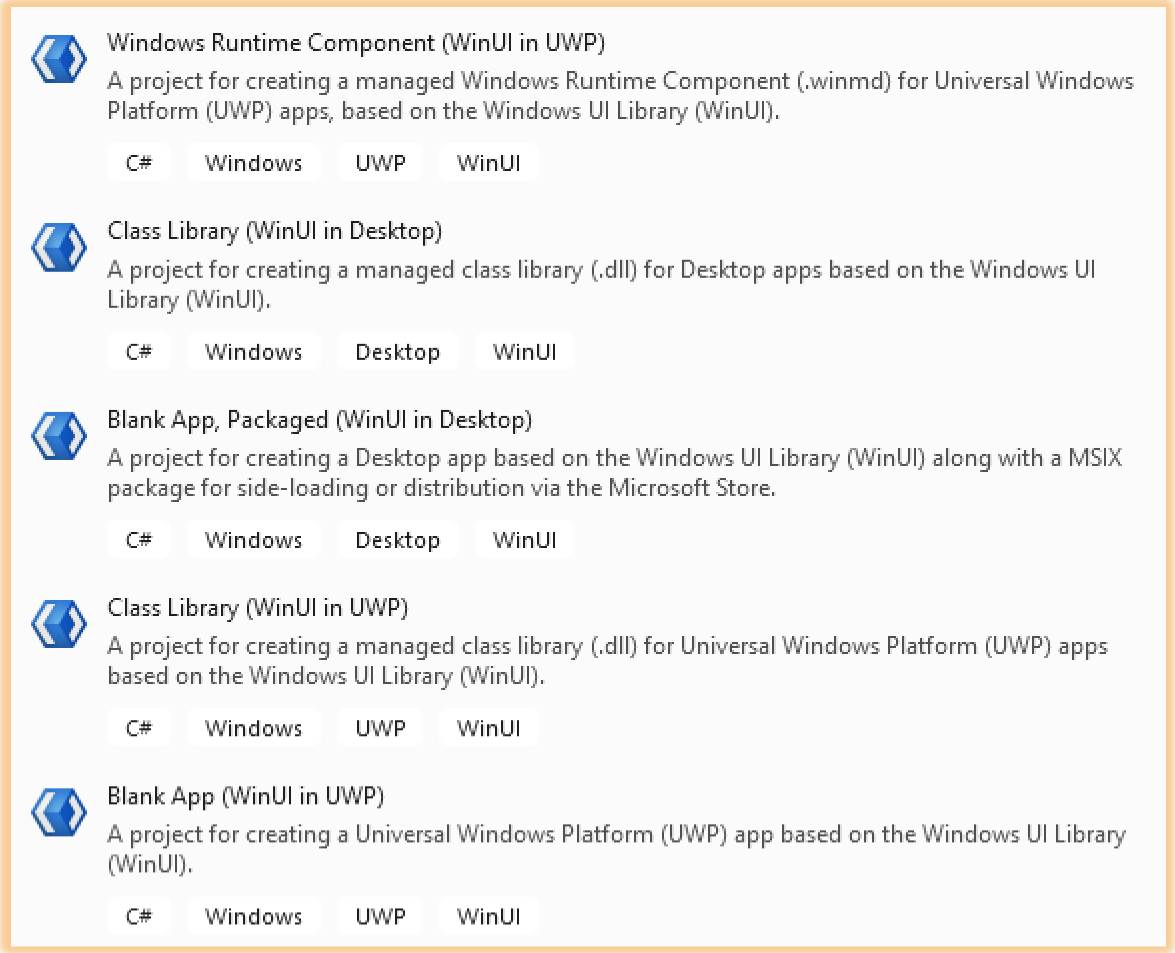
Minimum OS Version

Using the Windows SDK from .NET 5

Using the Windows UI controls from .NET 5

- WinUI3 allows .NET 5 apps to use UI controls
  - Available in WPF, Winform, UWP
- Notable features:
  - SwapChainPanel
  - Multi windows
  - ARM64
  - RenderTargetBitmap
- WebView2 is a separate control
  - available on Winform/WPF as well!

WinUI3 Preview 3 templates in Visual Studio



Windows Runtime Component (WinUI in UWP)
A project for creating a managed Windows Runtime Component (.winmd) for Universal Windows Platform (UWP) apps, based on the Windows UI Library (WinUI).
C# Windows UWP WinUI

Class Library (WinUI in Desktop)
A project for creating a managed class library (.dll) for Desktop apps based on the Windows UI Library (WinUI).
C# Windows Desktop WinUI

Blank App, Packaged (WinUI in Desktop)
A project for creating a Desktop app based on the Windows UI Library (WinUI) along with a MSIX package for side-loading or distribution via the Microsoft Store.
C# Windows Desktop WinUI

Class Library (WinUI in UWP)
A project for creating a managed class library (.dll) for Universal Windows Platform (UWP) apps based on the Windows UI Library (WinUI).
C# Windows UWP WinUI

Blank App (WinUI in UWP)
A project for creating a Universal Windows Platform (UWP) app based on the Windows UI Library (WinUI).
C# Windows UWP WinUI

FAQ: Will WinUI3 work cross-platform?

- It depends from your votes!
  - Kevin Gallo clearly said that it only depends on the feedback they get

- If you care, please vote here:
  - https://github.com/microsoft/microsoft-ui-xaml/issues/2024

WinUI3 Preview 3
WebView2

The magics behind accessing Windows SDKs

The CsWinRT nuget package

- A code generator to Consume and Produce WinRT components
  - Other code generators that Microsoft is working on: C++, Rust, Python

- Generates the C# code (projection) that makes the native component appear as it was .NET code
  - Widely used in Microsoft (WinUI is just an example)

- *Note: replaces the old projection generator that was part of .NET*

C# 9 Function Pointers (used by CsWinRT)

Calling convention

The C# method below

```
delegate* unmanaged[Cdecl]<int, byte*, int, int> _export = &OnNativeMessage;
```

in

out ← Func<> like signature

```
[UnmanagedCallersOnly(CallConvs = new[] { typeof(CallConvCdecl) })]
public static int OnNativeMessage(int id, byte* message, int length)
{
    Console.WriteLine(Encoding.UTF8.GetString(message, length));
    return 0;
}
```

- Allow exporting "functions" to the native world
- Remove the need of creating a delegate instance
- Reduce the cost of runtime calling
- Transition the GC to cooperative mode automatically
- The pointers point straight to the JITted native code.

System.Text.Json

System.Text.Json Tips

Immutable *record* and structs

```
public record Person(string FirstName, string LastName);
```

Select the appropriate ctor

```
public class Person {
    [JsonConstructor]
    public Person(string firstName, string lastName) { ... }
    public Person(string fullName) { ...}
```

Don't serialize default values

```
[JsonIgnore(Condition = JsonIgnoreCondition.WhenWritingDefault)]
public string LastName { get; init; }
```

Allow using private setter

```
[JsonInclude]
public int FullNameLen { get; private set; }
```

Extra data goes here

```
[JsonExtensionData]
[JsonInclude]
public IDictionary<string, object> Extra { get; private set; }
```

Allow circular references

```
new System.Text.Json.JsonSerializerOptions() {
    ReferenceHandler = ReferenceHandler.Preserve,
};
```

Serialization Benchmark

- Depth: 3 levels (about 250 objects)
- Circular references (Parent property)
- camelCase

```
.NET 5 - Serialization
|   Method |        Mean |     Error |    StdDev |     Gen 0 |    Gen 1 | Gen 2 |  Allocated |
|--------- |-----------:|---------:|----------:|----------:|---------:|------:|-----------:|
|  JsonNet |    484.3 us |   9.62 us |  20.71 us |   46.8750 |   8.7891 |     - |  195.73 KB |
| TextJson |    221.7 us |   4.42 us |   6.48 us |   20.7520 |   0.2441 |     - |   85.51 KB |

.NET Framework 4.8 - Serialization
|   Method |        Mean |     Error |    StdDev |     Gen 0 |    Gen 1 | Gen 2 |  Allocated |
|--------- |-----------:|---------:|----------:|----------:|---------:|------:|-----------:|
|  JsonNet |    648.8 us |  12.95 us |  31.78 us |   41.0156 |        - |     - |   171.8 KB |
| TextJson |    568.2 us |  11.01 us |  17.14 us |   17.5781 |        - |     - |   75.09 KB |
```

Deserialization Benchmark

- Depth: 3 levels (about 250 objects)
- Circular references (Parent property)
- camelCase

BenchmarkDotNet
Powerful .NET library for benchmarking

```
.NET 5 - Deserialization
|   Method |       Mean |     Error |    StdDev |     Gen 0 |    Gen 1 | Gen 2 | Allocated |
|--------- |-----------:|----------:|----------:|----------:|---------:|------:|----------:|
|  JsonNet | 1,288.1 us | 25.38 us | 37.21 us | 121.0938 | 52.7344 |     - | 682.76 KB |
| TextJson |   417.1 us |  8.31 us | 13.17 us |  17.5781 |  2.4414 |     - |   72.7 KB |

.NET Framework 4.8 - Deserialization
|   Method |       Mean |     Error |    StdDev |     Gen 0 |    Gen 1 | Gen 2 | Allocated |
|--------- |-----------:|----------:|----------:|----------:|---------:|------:|----------:|
|  JsonNet | 1,469.3 us | 27.12 us | 43.79 us |  80.0781 | 29.2969 |     - | 390.32 KB |
| TextJson |   821.1 us | 16.24 us | 13.56 us |  11.7188 |       - |     - |  49.42 KB |
```

Prepare for publishing

Publishing options

- Self-contained: remove the need to deploy the runtime

```
<RuntimeIdentifiers>linux-x64</RuntimeIdentifiers>
<SelfContained>true</SelfContained>
```

- SingleFile: creates a single file containing the App and its dependencies

```
<PublishSingleFile>true</PublishSingleFile>
```

- Trimming: remove the need for IL for types not used from the App

```
<PublishTrimmed>true</PublishTrimmed>
```

- AOT/R2R: pre-generate the native assembler to speed up App bootstrap

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

- CLI command:

```
dotnet publish -r linux-x64 --self-contained true
/p:PublishTrimmed=true /p:PublishReadToRun=true
/p:PublishSingleFile=true /p:IncludeNativeLibrariesForSelfExtract=true
```

Trimming more code!

- .NET Core 3 can only trim entire types
- .NET 5 has an <u>optional</u> feature trimming <u>just</u> unused members

```
<PublishTrimmed>true</PublishTrimmed>
<TrimMode>link</TrimMode>
```

```
/p:PublishTrimmed=true /p:TrimMode=Link
```

- Removes R2R images as well unless PublishReadyToRun is true
- It is an experimental feature, but very promising
  - Not all the system libraries have been annotated yet
  - This is why it does not play well with WPF and Winform
  - Implemented to address WebAssembly/Blazor needs
- App manual testing is **<u>mandatory</u>**
  - Unit tests can fool the results

Trimming: adding or removing "Features"

- This is how Blazor trims away parts of the BCL for good reasons
- You can do the same with your own libraries
- .NET 5 Features: sets of functionalities

.NET5 MSBuild pre-defined properties	Feature	Trimmed when
DebuggerSupport	Code such as debugger visualizers	false
EnableUnsafeUTF7Encoding	Unsafe UTF-7 encoding	false
EnableUnsafeBinaryFormatterSerialization	Binary formatter	false
EventSourceSupport	EventSource and related types	false
InvariantGlobalization	Invariant globalization conversions	**true**
UseSystemResourceKeys	Localized resources for system assemblies	**true**
HttpActivityPropagationSupport	Distributed tracing over System.Net.Http	false

A new «Superhost» is scheduled for .NET 6

- Statically link the CLR, Jitter (if needed) and native libraries
- Cross-platform friendly

- Goals:
  - Total removal of the Intermediate Language IL (obfuscation purposes)
  - Smallest possible binary file
  - Fastest possible startup time

Size comparison in publishing

AnyCPU	Self Contained	Single File	R2R AOT	Trim	Trim Link	Console	WebApp	WPF *	WPF * Browser
●						153.0 K	4.4 M	159.0 K	1.2 M
	●					64.7 M	88.8 M	148.0 M	149.0 M
		●				153.0 K	4.4 M	159.0 K	1.0 M
	●	●				58.7 M	82.8 M	142.0 M	143.0 M
	●	●	●			58.7 M	82.8 M	142.0 M	143.0 M
	●	●	●	●		18.8 M	42.0 M	81.2 M	82.4 M
	●	●	●	●	●	10.6 M	30.5 M	72.0 M	73.3 M
	●			●		24.9 M	48.0 M	87.3 M	88.5 M
	●			●	●	16.7 M	36.5 M	78.1 M	79.3 M

** WPF apps need additional work to avoid trimming vital types*

Better diagnostics

our app

- dotnet-trace can now start a process

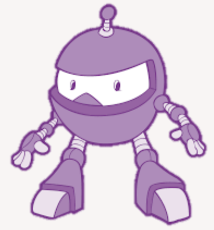
```
dotnet-trace collect --providers Microsoft-Windows-DotNETRuntime:4:4  --  Browsing.exe
```

  - Saves a trace file on disk that can be analyzed with PerfView

- Analyze Linux dumps on Windows
  - Windbg or `dotnet dump analyze`

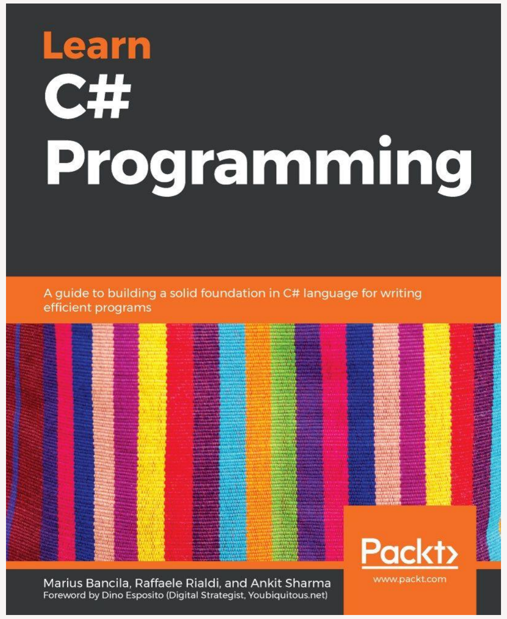
- Capture ELF dumps on macOS

Questions?



Thank you!

@raffaeler - raffaeler@vevy.com



Learn C# Programming

A guide to building a solid foundation in C# language for writing efficient programs

Packt
www.packt.com

Marius Bancila, Raffaele Rialdi, and Ankit Sharma
Foreword by Dino Esposito (Digital Strategist, Youbiquitous.net)

Amazon code: **25CSHARPBK**
https://www.amazon.com/gp/mpc/AEL3ILD2QGU8K

Packt code: **25CSHARPBK**
https://www.packtpub.com/ (ebook)