



<https://github.com/shaloml/sivron-DOTNET-Workshop-Day1>



# **.NET CORE DESKTOP REVOLUTION**

---

CREATE BY: SHALOM LEVI



WHO AM I

---

# AGENDA DAY 1

## **Part 1:**

- Overview of .NET Core
- Building Console Applications in .NET Core
- Introduction to .NET Core Command-Line Interface (dotnet CLI)
- Managing Packages with NuGet in .NET Core

## **Part 2:**

- Implementing Dependency Injection in .NET Core
- Advance Dependency Injection in .NET Core

# .NET CORE

---





# DOTNET CORE

```
1 var names = new[]
2 {
3     "Ana",
4     "Felipe",
5     "Emillia"
6 };
7
8 foreach (var name in names)
9 {
10     Console.WriteLine($"Hello {name}");
11 }
```

Compatible

```
let names = [ "Ana"; "Felipe"; "Emillia" ]
```

```
for name in names do
    printfn $"Hello {name}"
```

```
Dim names As New List(Of String)({
    "Ana",
    "Felipe",
    "Emillia"
})
```

```
For Each name In names
    Console.WriteLine($"Hello {name}")
Next
```



Cross Platform



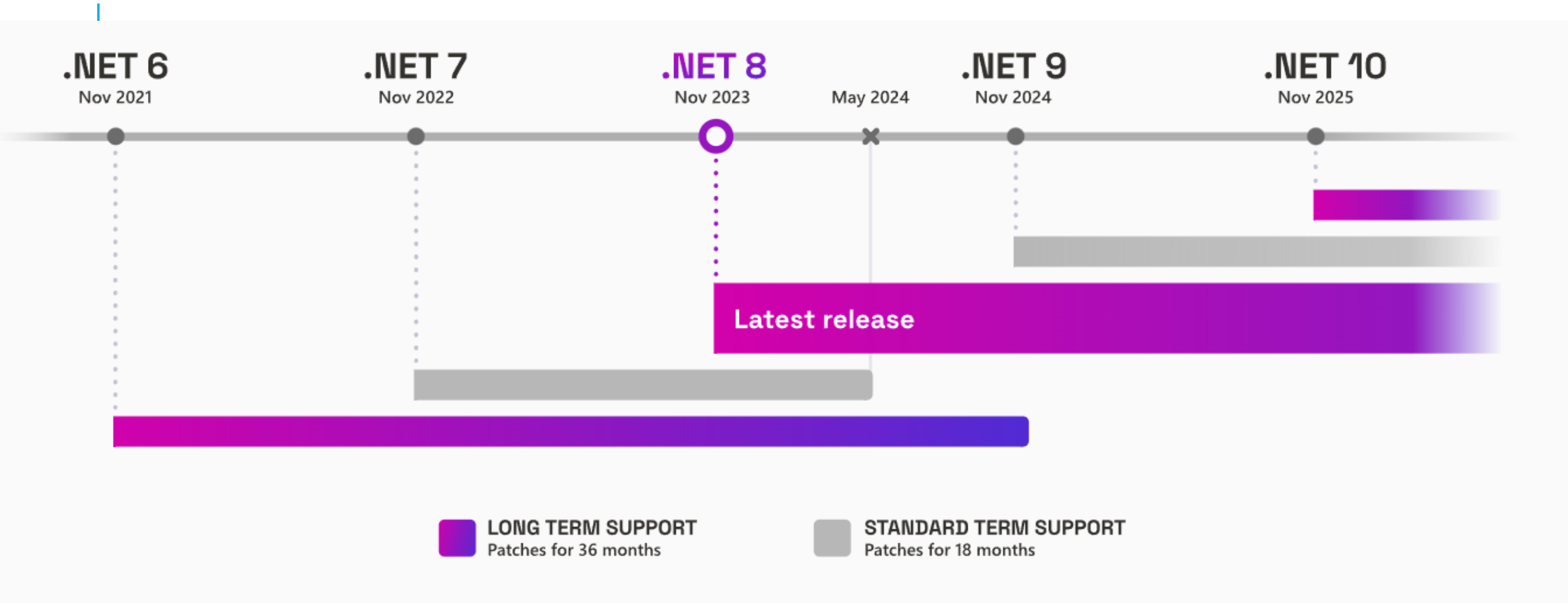
Open Source

<https://github.com/dotnet>

Multi-language

Side by Side

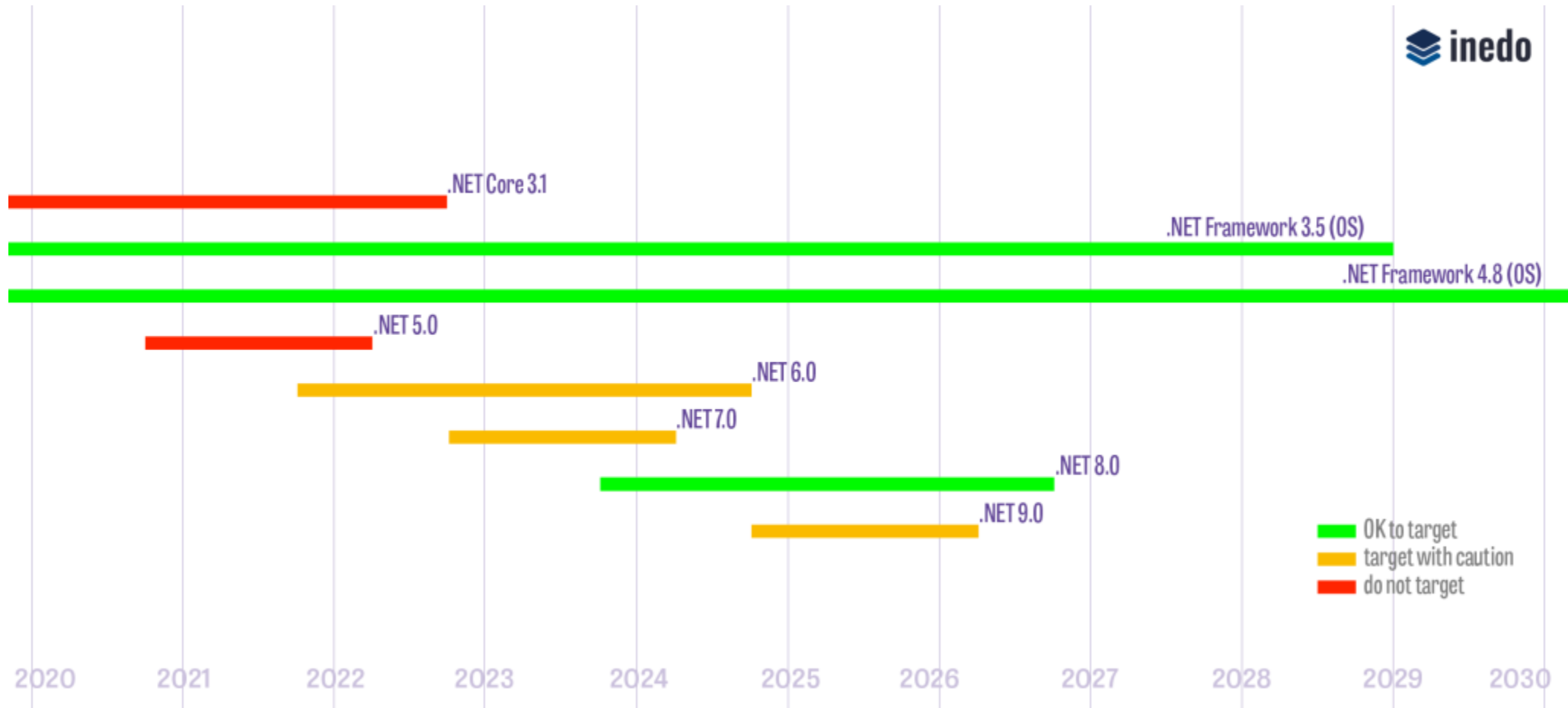
# .NET - END OF LIFE



<https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core>



# .NET FRAMEWORK - END OF LIFE



<https://learn.microsoft.com/en-us/lifecycle/products/microsoft-net-framework>

# TOOLS

Single Platform

Cross Platform

SaaS



Github Codespaces



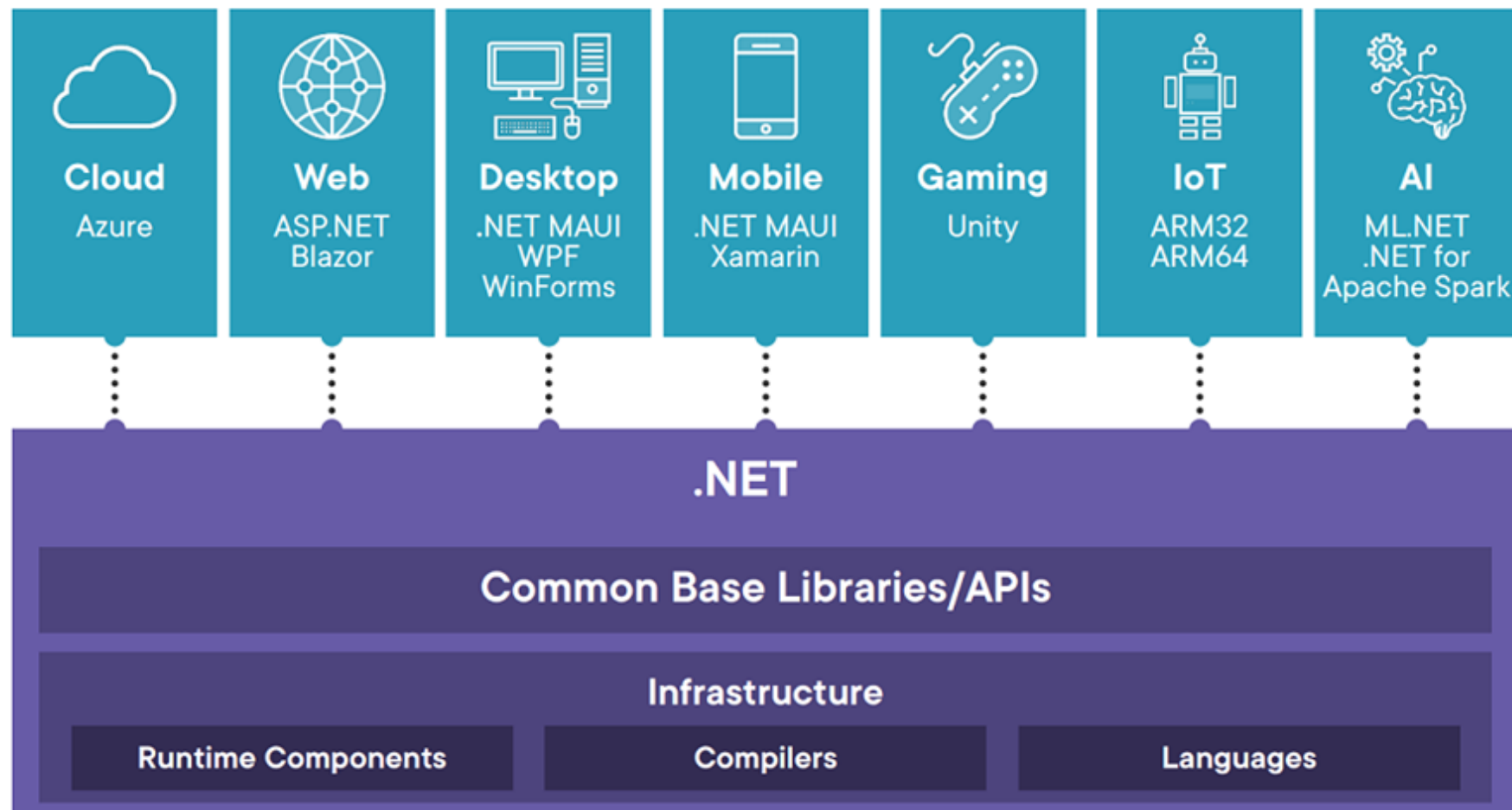
Visual Studio for Mac

August 2024 ,31



```
> dotnet  
Microsoft .NET Core Shared Framework
```

# .NET Architecture

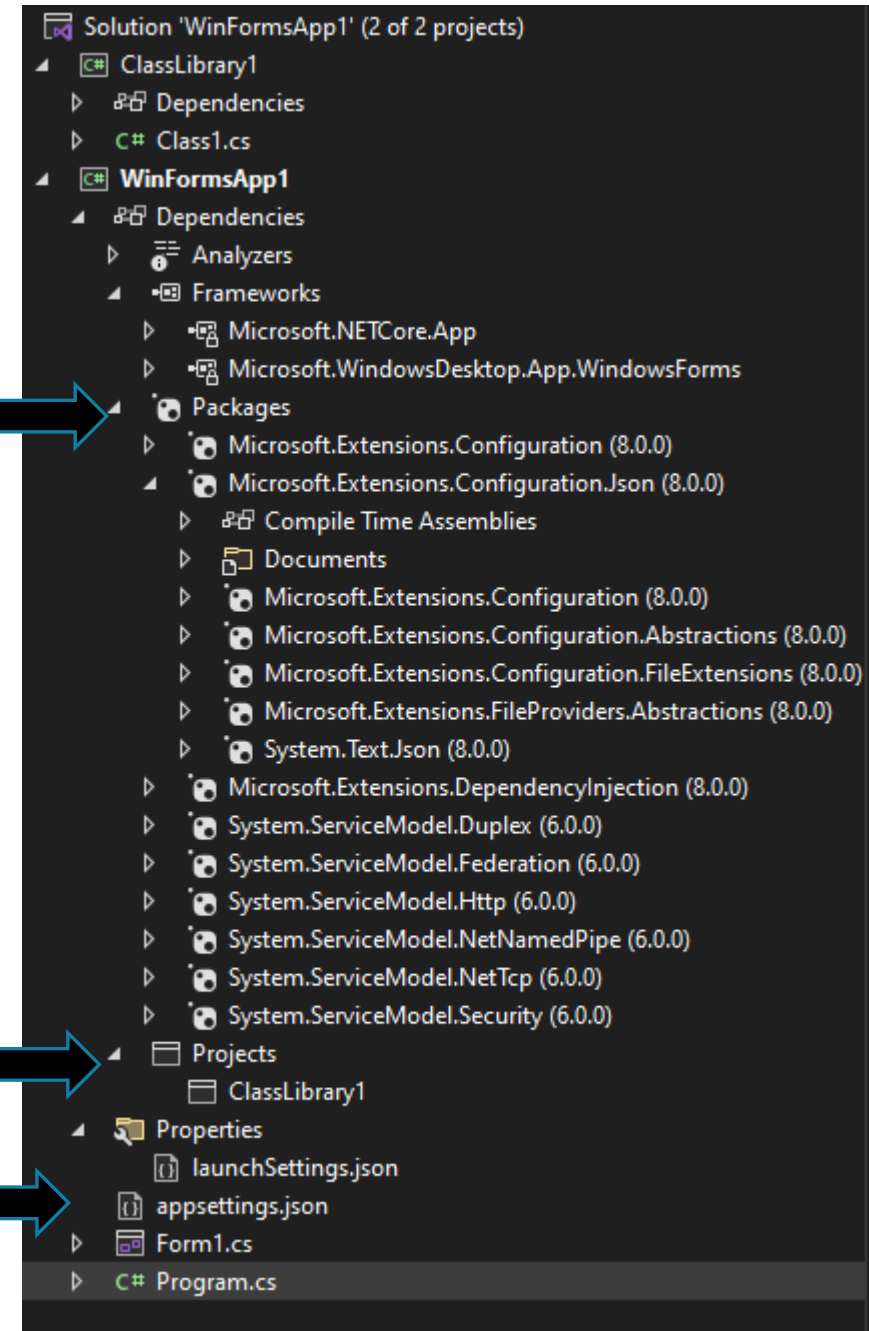


# EXPLORING THE PROJECT STRUCTURE

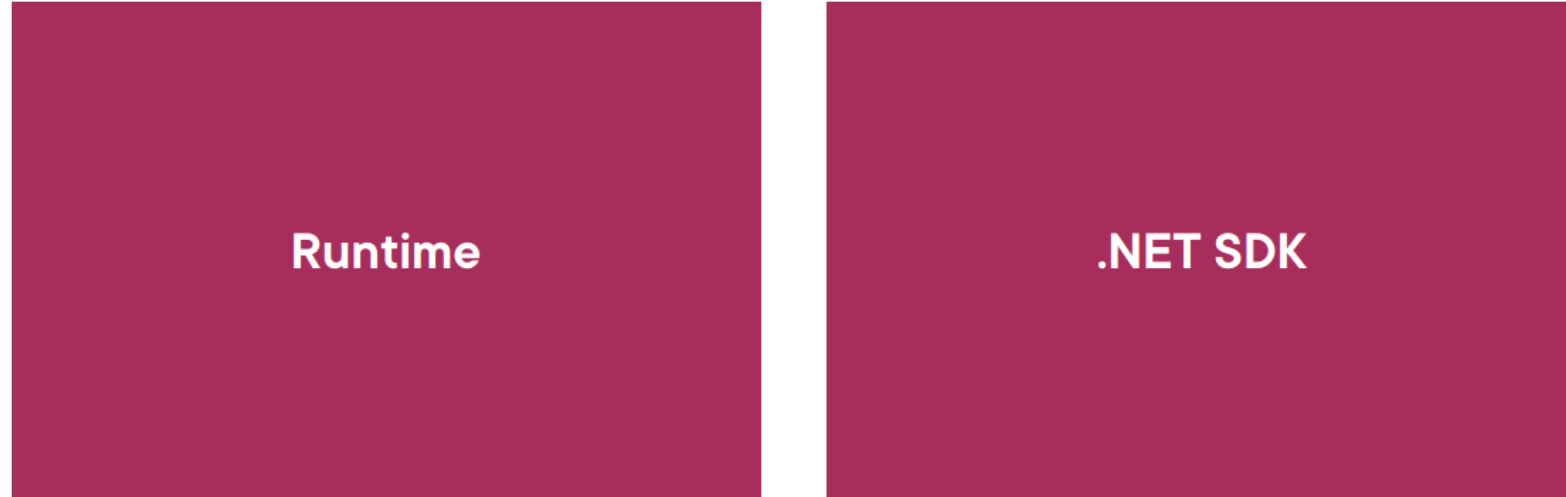
**NuGet/ project ref / npm /dll's / other**

**Debug profile running**

**settings**



# GETTING STARTED WITH .NET APPLICATIONS



## In short

The SDK is what you use to build and run your application.

The Runtime is to run the application

# SDK Components



**.NET CLI**



**.NET driver**



**Roslyn and F# compilers**



**MSBuild build engine**



**.NET runtime**



**Runtime libraries**



**ASP.NET Core runtime**



**Desktop runtime**

# CLI

Dotnet

dotnet --version

dotnet --list-sdks

Dotnet --help

Dotnet new

Dotnet new console -o console\_demo

Dotnet new wpf -o wpf\_demo

Dotnet new wepapp -o web\_app

Dotnet build

Dotnet publish

# INSTALL ON DOCKER

```
docker run --rm mcr.microsoft.com/dotnet/core/samples
```

```
docker run -it --rm -p 8000:80 --name aspnetcore_sample  
mcr.microsoft.com/dotnet/core/samples:aspnetapp
```

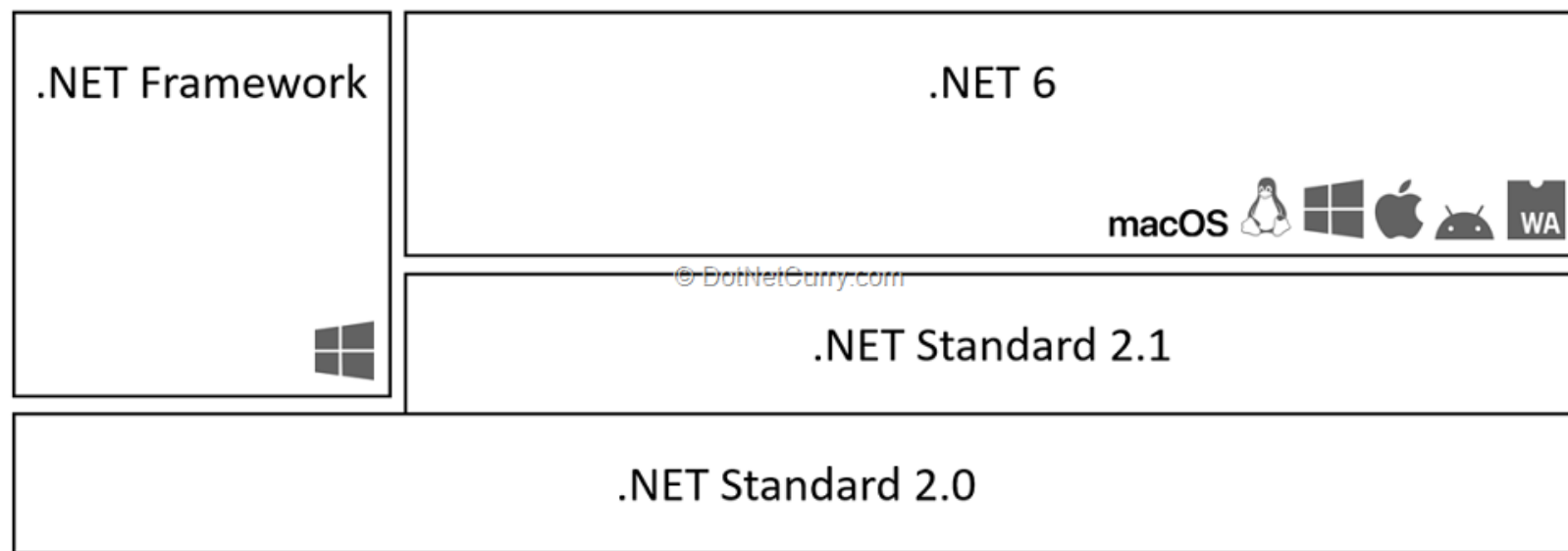


# .NET STANDARD CONTEXT

---



# .NET STANDARD CONTEXT



# .NET STANDARD 2.0

[1.0](#)[1.1](#)[1.2](#)[1.3](#)[1.4](#)[1.5](#)[1.6](#)[2.0](#)[2.1](#)

.NET Standard 2.0 has 32,638 of the 37,118 available APIs.

.NET implementation	Version support
.NET and .NET Core	2.0, 2.1, 2.2, 3.0, 3.1, 5.0, 6.0
.NET Framework <sup>1</sup>	4.6.1 <sup>2</sup> , 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
Mono	5.4, 6.4
Xamarin.iOS	10.14, 12.16
Xamarin.Mac	3.8, 5.16
Xamarin.Android	8.0, 10.0
Universal Windows Platform	10.0.16299, TBD
Unity	2018.1

# .NET STANDARD 2.1

[1.0](#)[1.1](#)[1.2](#)[1.3](#)[1.4](#)[1.5](#)[1.6](#)[2.0](#)[2.1](#)

.NET Standard 2.1 has 37,118 of the 37,118 available APIs.

.NET implementation	Version support
.NET and .NET Core	3.0, 3.1, 5.0, 6.0
.NET Framework <sup>1</sup>	N/A <sup>2</sup>
Mono	6.4
Xamarin.iOS	12.16
Xamarin.Mac	5.16
Xamarin.Android	10.0
Universal Windows Platform	TBD
Unity	2021.2

# .NET STANDARD

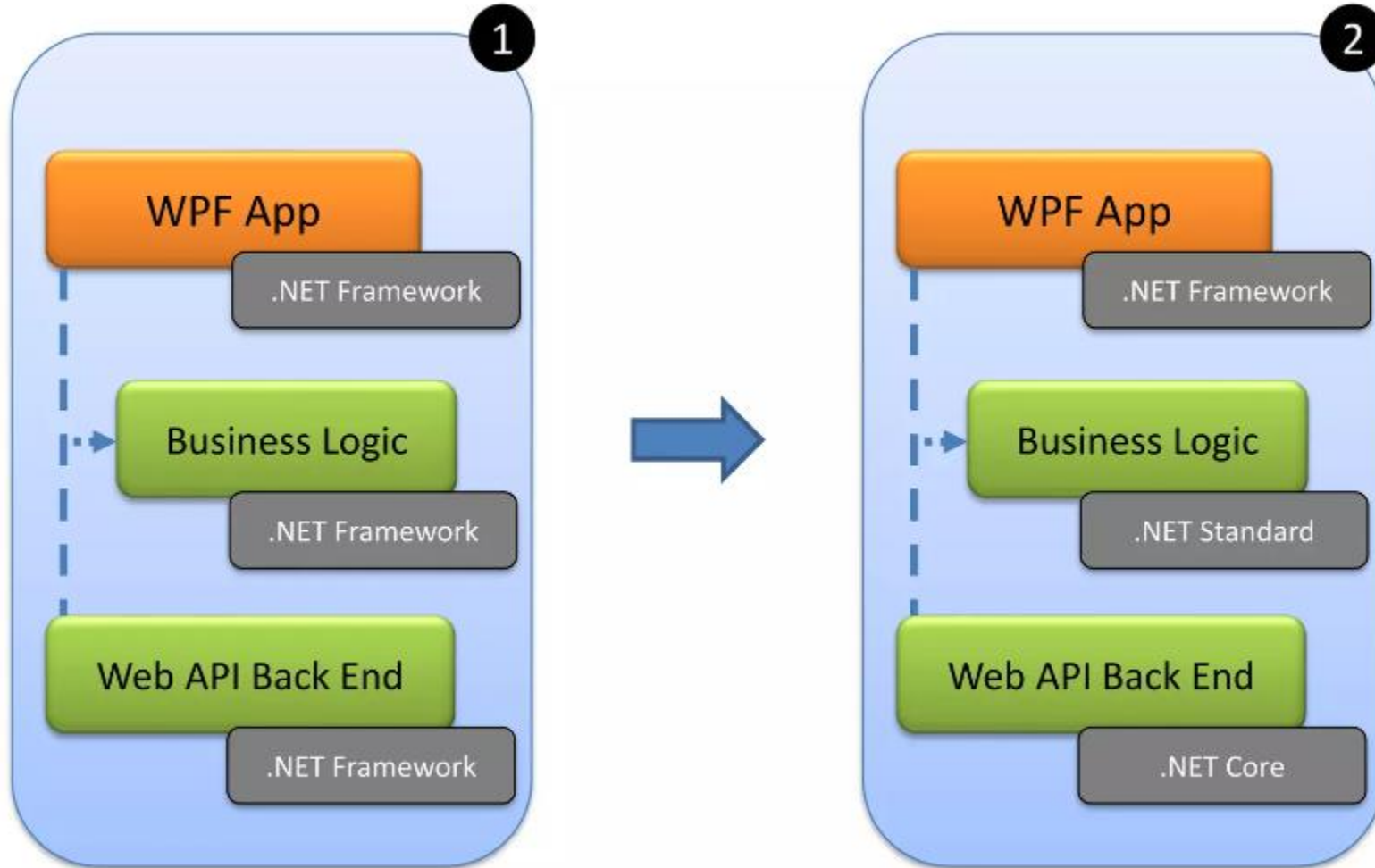
.NET Standard is also Open Source!

Anybody can propose API additions

The review board approves the API

<https://docs.microsoft.com/en-us/dotnet/standard/net-standard>

# HANDLING SHARED CODE WHEN TARGETING MULTIPLE .NET IMPLEMENTATIONS



# NUGET PACKAGE

NuGet package manager

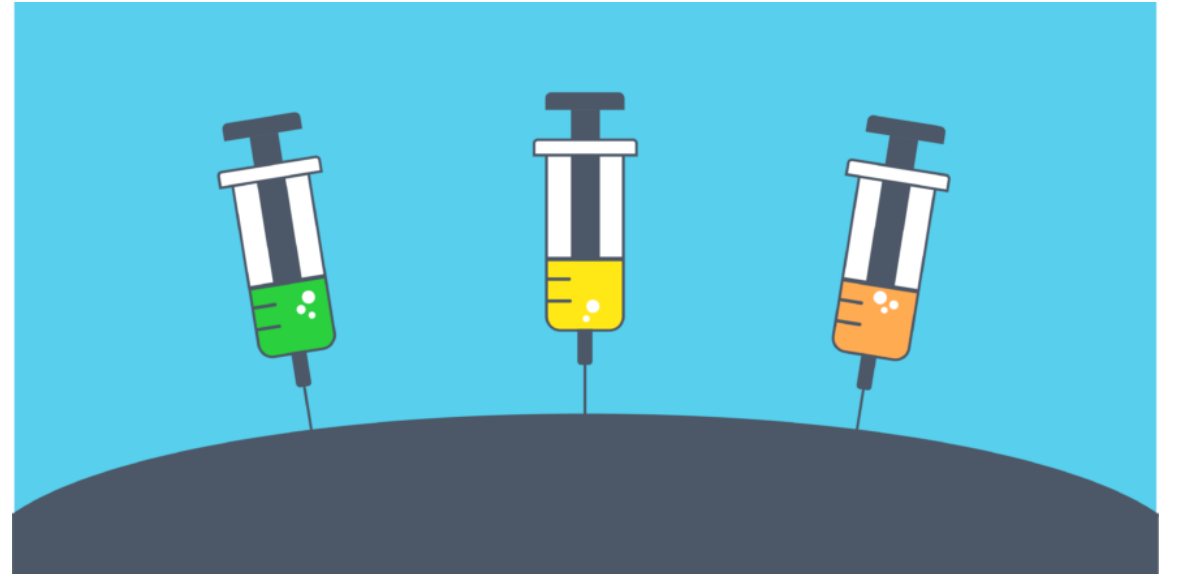
Project \*.csproj file

CLI: dotnet add package

PM> Install-Package

# DEPENDENCY INJECTION

---

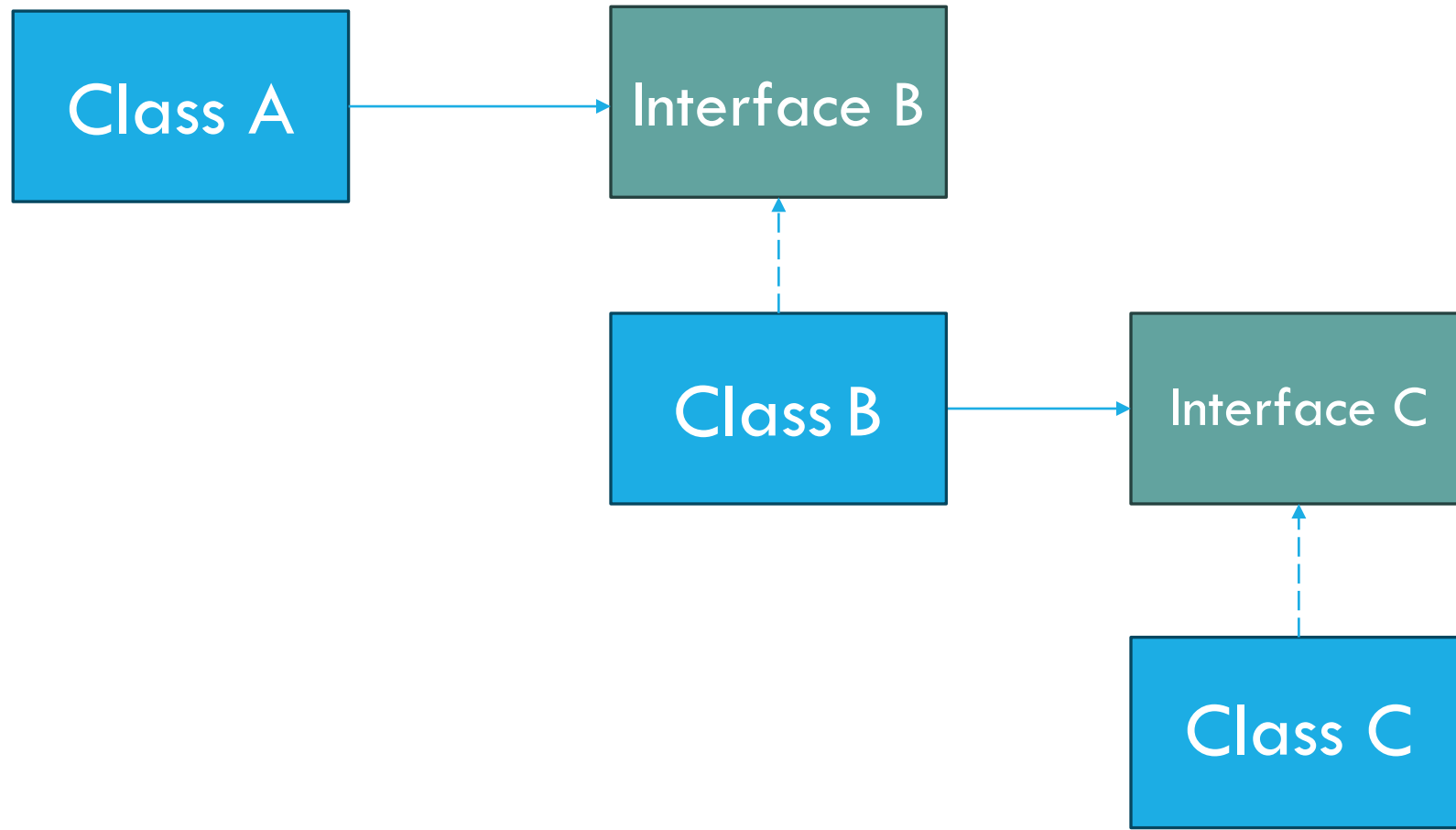




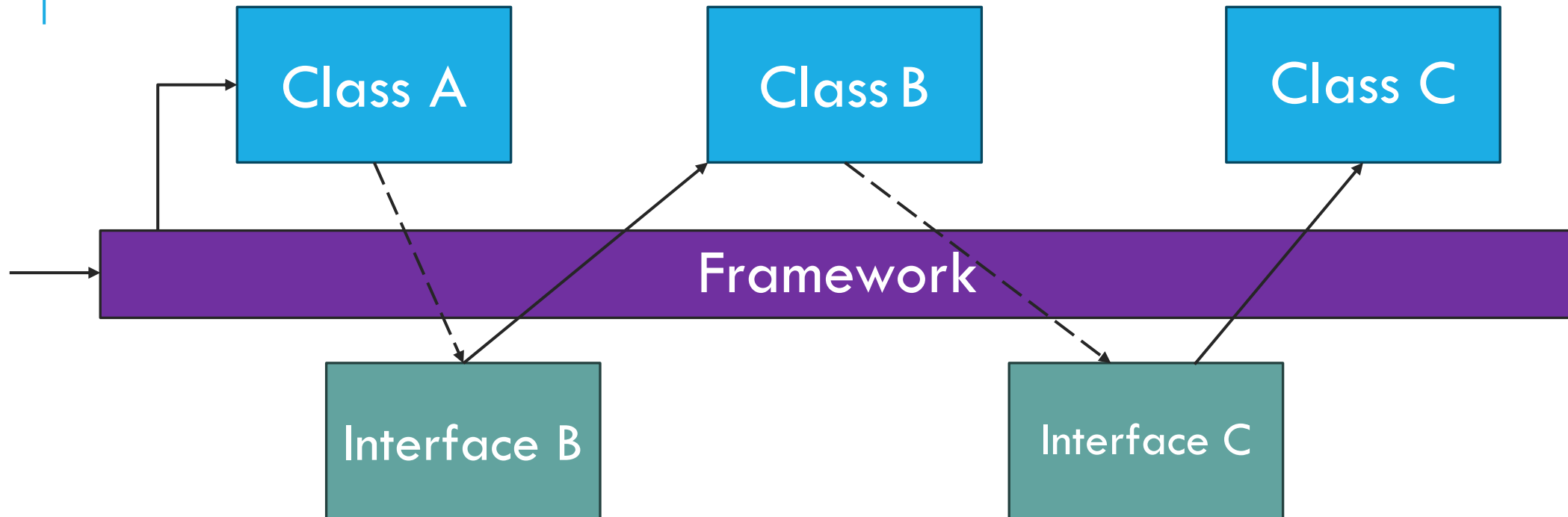
# TRADITIONAL FLOW



# DEPENDENCY INVERSION



# INVERSION OF CONTROL (IOC)



# LIFE TIME

## Service Lifetimes

### Transient

Created each time  
they are requested

### Scoped

Created once per  
request

### Singleton

Created the first  
time they are  
requested

# DEMO



# תרגיל 2

1. צור פרויקט Console דרך visual studio בשם Exercise1

2. צור מחלקה בשם StudentRepository  
- המחלקה תכלול פונקציה שמחזירה שמות סטודנטים לפי מספר בית ספר.

3. צור מחלקה בשם SchoolService  
- המחלקה תכלול פונקציה שמקבלת מספר בית ספר ומחזירה את שמות הסטודנטים.

4. הדפס את כל התלמידים בבית ספר מספר 1 בתוצאה.

5. השתמש ב IOC

6. הזרק את התלויות באמצעות  
ServiceCollection-ServiceProvider

School ID	Student name
1	Moshe Levi
1	Avi Perez
1	Galit Mizrahi
2	Ronit Chen
2	Nivi Shemesh

# ADD VS. TRYADD

## **Add{Lifetime}**

If there is an existing registration for the type, this will overwrite it

## **TryAdd{Lifetime}**

If there is an existing registration for the type, this will do nothing

# MULTIPLE REGISTRATIONS

## **Resolving directly**

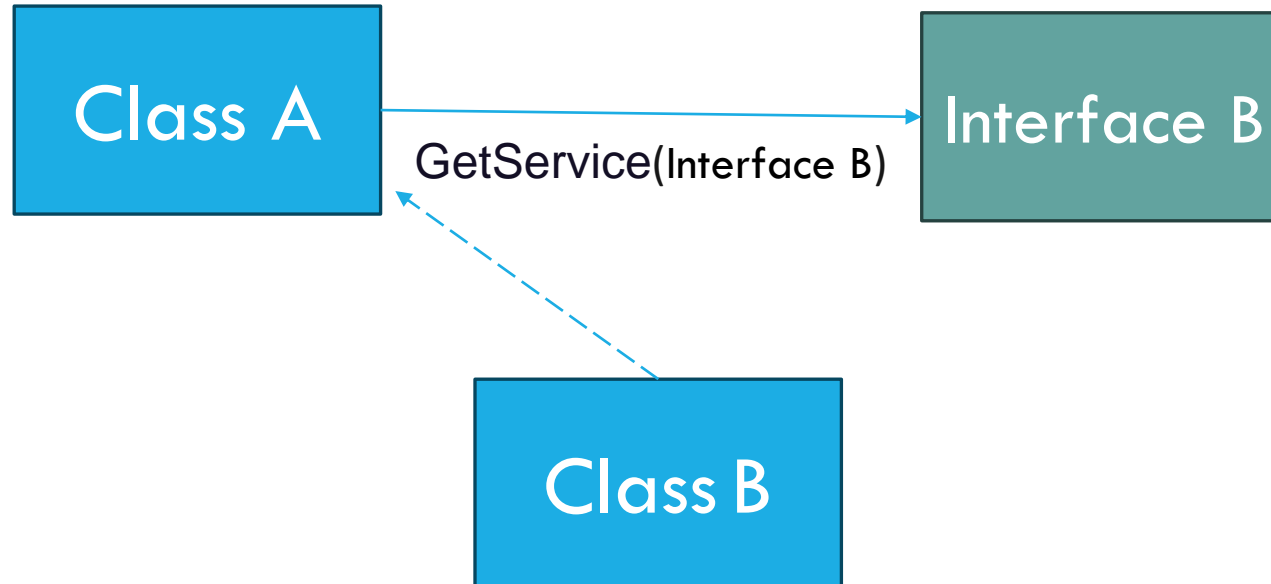
The last registration is returned

## **Resolving IEnumerable**

All registrations are returned



# SERVICE LOCATOR



```
public ClassA(IServiceProvider serviceProvider )  
{  
    _classB = serviceProvider.GetService<IClassB>();  
}
```

# DEPENDENCY INJECTION VS. SERVICE LOCATOR



## Testability

**Classes that use a Service Locator are harder to test**



## Implicit Dependencies

**Dependencies are not clearly advertised, but implicit**



# MULTI-TENANT APPLICATION

## **Problem Statement:**

In a multi-tenant application, different tenants might require different implementations of a service based on their subscription level or preferences

# BEST PRACTICES

- **Use Interfaces:** etatilicaf ot sessalc etercnoc fo daetsni secafretni esu syawlA :  
.gnitset dna gnippaws ycnedneped
- **Constructor Injection.**noitcejni dohtem ro ytreporp revo noitcejni rotcurtsnoc referP :
- **Keep Classes Slim:** Classes with too many dependencies may indicate too much responsibility. Try to refactor heavy classes into smaller, more focused ones.
- **Avoid Service Locator:** Minimize the use of ServiceProvider within your code. It breaks the Inversion of Control principle.
- **Service Lifetime Management:** Ensure to choose the appropriate service lifetime (Transient, Scoped, Singleton) based on the application needs.
- **Unit Testing:** Leverage dependency injection to write effective unit tests with mocks.



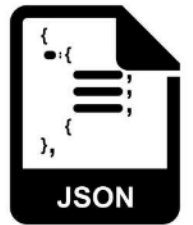
# תרגיל מסכם

# AGENDA DAY 2

- Configuration Management in .NET Core Applications
- Logging and Caching in .NET Core Applications
- Introduction to Entity Framework Core for Database Management
- Overview of UI Frameworks for Windows Desktop Applications
- Introduction to Cross-Platform UI Frameworks for Desktop Development
- Choosing a UI Framework for Your .NET Desktop Application

# CONFIGURATION

---



JSON FILE

{JSON}

# DEFINING CONFIGURATION IN JSON FILES

Is a default

Is very easy

The usual way



mastore.org/appsettings.json

```
{
  "Key1": "Hello!!!",
  "GeneralSettings": {
    "Subsection": {
      "Suboption1": "subvalue1_from_json",
      "Suboption2": 200
    },
    "KeyString": "World!!!",
    "KeyInt": 123,
    "KeyDouble": 123.456,
    "KeyBool": true
  },
  "Features": {
    "FeatureA": true,
    "FeatureB": false,
    "FeatureC": true
  }
}
```

# HOW TO USE DIRECTLY

```
internal class DemoService
{
    private readonly IConfiguration _configuration;

    0 references
    public DemoService(IConfiguration configuration)
    {
        _configuration = configuration;
    }
}
```

appsettings.json

```
_configuration.GetValue<string>("Key1");
```

```
_configuration.GetValue<string>("GeneralSettings:Subsection:Suboption1");
```

```
{
  "Key1": "Hello!!!",
  "GeneralSettings": {
    "Subsection": {
      "Suboption1": "subvalue1_from_json",
      "Suboption2": 200
    },
    "KeyString": "World!!!",
    "KeyInt": 123,
    "KeyDouble": 123.456,
    "KeyBool": true
  }
}
```

# LOGICAL CONFIGURATION STRUCTURE

MyStringKey = "This is a string value"

```
_configuration.GetValue<string>("MyStringKey")
```

MyBooleanKey = true

```
_configuration.GetValue<bool>("MyBooleanKey")
```

MyIntegerKey = 100

```
_configuration.GetValue<int>("MyIntegerKey")
```

# CONFIGURATION HIERARCHY

```
"GeneralSettings": {  
  "Subsection": {  
    "Suboption1": "subvalue1_from_json",  
    "Suboption2": 200  
  },  
  "KeyString": "World!!!",  
  "KeyInt": 123,  
  "KeyDouble": 123.456,  
  "KeyBool": true  
},
```

appsettings.json

```
_configuration.GetValue<string>("GeneralSettings:Subsection:Suboption1");
```

# CONFIGURATION HIERARCHY BY SECTION

```
"GeneralSettings": {  
  "Subsection": {  
    "Suboption1": "subvalue1_from_json",  
    "Suboption2": 200  
  },  
  "KeyString": "World!!!",  
  "KeyInt": 123,  
  "KeyDouble": 123.456,  
  "KeyBool": true  
},
```

appsettings.json

```
_configuration.GetValue<int>("GeneralSettings:Subsection:Suboption2");
```

# DEFINE CONNECTION STRING

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TennisBookingDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
}
```

appsettings.json

```
"ConnectionStrings": {
  "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=TennisBookings;Trusted_Connection=true;TrustServerCertificate=true;"
},
```

```
"Logging": {
  "LogLevel": {
    "Default": "Warning"
  }
},
```

```
"AllowedHosts": "*"
}
```

# PROBLEMS WITH GETTING VALUES

- Repetitive code
- Fragile naming
- Can lead to bugs

# STRONGLY TYPED - FIRST OPTION

## BIND CLASS

```
public class GeneralSettings
{
    0 references
    public SubsectionSettings Subsection { get; set; }
    0 references
    public string KeyString { get; set; }
    0 references
    public int KeyInt { get; set; }
    0 references
    public double KeyDouble { get; set; }
    0 references
    public bool KeyBool { get; set; }

    1 reference
    public class SubsectionSettings
    {
        0 references
        public string Suboption1 { get; set; }
        0 references
        public int Suboption2 { get; set; }
    }
}
```

```
"GeneralSettings": {
  "Subsection": {
    "Suboption1": "subvalue1_from_json",
    "Suboption2": 200
  },
  "KeyString": "World!!!",
  "KeyInt": 123,
  "KeyDouble": 123.456,
  "KeyBool": true
},
```

```
var generalSettings = new GeneralSettings();
Configuration.Bind("GeneralSettings", generalSettings);
```



# STRONGLY TYPED - SECOND OPTION APPLYING THE OPTIONS PATTERN

Injecting options with `IOptions<T>`

On class Startup

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddOptions();

    services.Configure<Features>(Configuration.GetSection("Features:HomePage"));
}
```

On class

```
0 references | 0 exceptions
public IndexModel(IOptions<Features> featureSetting, Weather
```

# STRONGLY TYPED - THIRD OPTION

## APPLYING THE OPTIONS PATTERN WITH SINGLETON

```
services.Configure<FeaturesSettings>(Configuration.GetSection("Features"));  
services.AddSingleton(p=>  
    p.GetRequiredService<IOptions<FeaturesSettings>>().Value);
```

On class

```
0 references  
public DemoService(FeaturesSettings features)  
{
```

## IOPTIONS<T>

- Does not support options reloading
- Registered as a singleton in D.I. container
- Values bound when first used
- Can be injected into all service lifetimes
- Does not support named options

## OTHER OPTIONS

### **IOPTIONS SNAPSHOT<T>**

- Supports reloading of configuration
- Registered as **scoped** in D.I. container
- Values may reload per request
- Can not be injected into singleton services
- Supports named options

## OTHER OPTIONS

### **IOPTIONS MONITOR<T>**

- Supports reloading of configuration
- Registered as a **singleton** in D.I. container
- Values are reloaded immediately
- Can be injected into all service lifetimes
- Supports named options

# CHOOSING AN OPTIONS INTERFACE

	Use in singletons	Supports reloading	Named options
IOptions	✓	✗	✗
IOptionsSnapshot	✗	✓	✓
IOptionsMonitor	✓	✓	✓

# DEMO

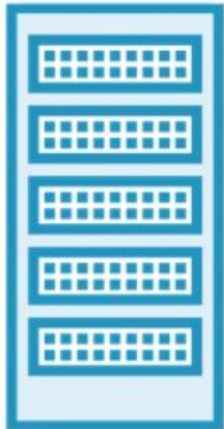


# ENVIRONMENTS SETTINGS

Launch Profiles

Environments

**Microsoft.Extensions.Configuration.EnvironmentVariables**



ConfigureDevelopment()

ConfigureServicesDevelopment()



ConfigureStaging()

ConfigureServicesStaging()



ConfigureProduction()

ConfigureServiceProduction()



# ON VISUAL STUDIO

environment values : ASPNETCORE\_ENVIRONMENT

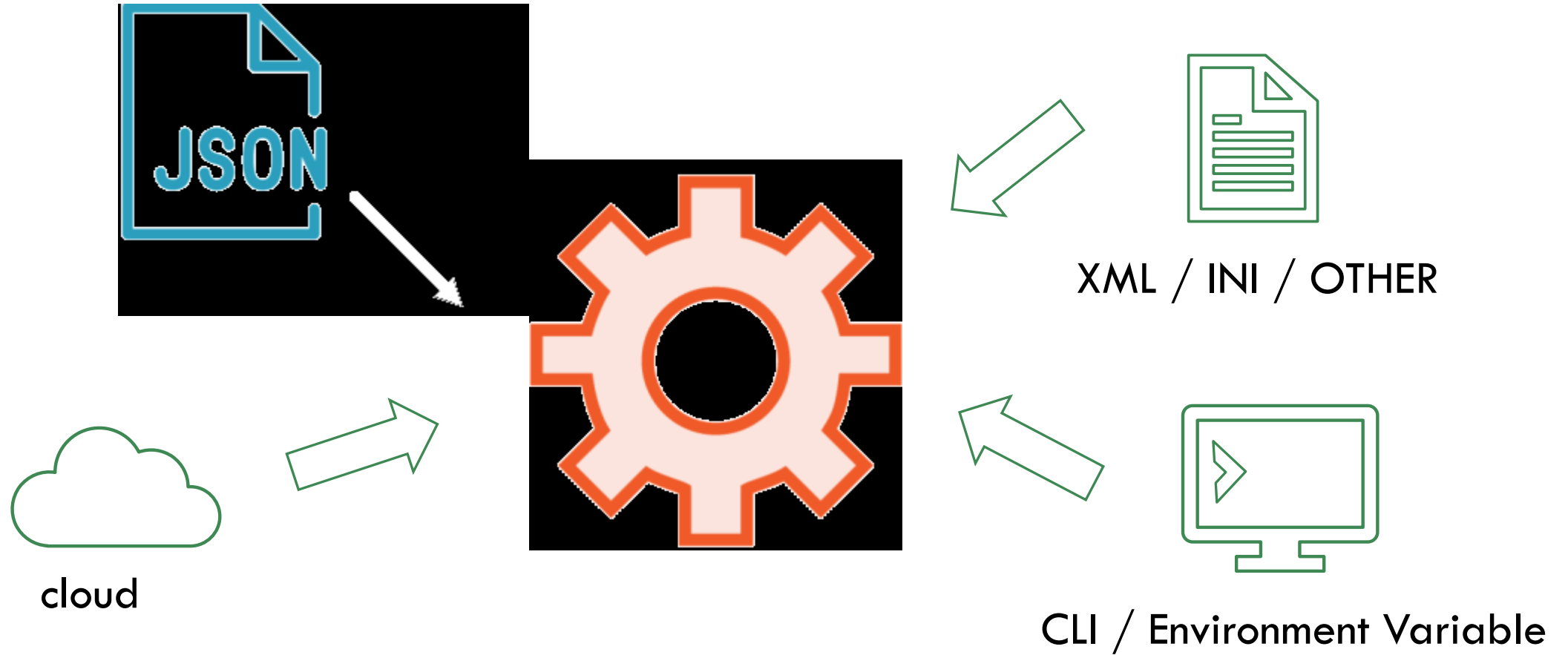
Environment variables:

Name	Value
ASPNETCORE_ENVIRONMENT	Development

- appsettings.json
  - appsettings.Development.json
  - appsettings.Staging.json

# CONFIGURATION PROVIDERS



```
public static IHostBuilder CreateDefaultBuilder(string[] args)
```

```
builder.ConfigureAppConfiguration((hostingContext, config) =>
{
    var env = hostingContext.HostingEnvironment;

    config.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true, reloadOnChange: true);


    if (env.IsDevelopment() && !string.IsNullOrEmpty(env.ApplicationName))
    {
        var appAssembly = Assembly.Load(new AssemblyName(env.ApplicationName));
        if (appAssembly != null)
        {
            config.AddUserSecrets(appAssembly, optional: true);
        }
    }
    config.AddEnvironmentVariables();
    if (args != null)
    {
        config.AddCommandLine(args);
    }
})
```

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-8.0#ini-configuration-provider>

# XML FILE

- section0:key0
- section0:key1
- section1:key0
- section1:key1

```
public static IHostBuilder CreateHostBuilder(string[] args) =  
    Host.CreateDefaultBuilder(args)  
        .ConfigureWebHostDefaults(webBuilder =>  
        {  
            webBuilder.ConfigureAppConfiguration((hostingContext, config) =>  
            {  
                config.AddXmlFile(  
                    "config.xml", optional: true, reloadOnChange: true);  
            });  
            webBuilder.UseStartup<Startup>();  
        });
```



```
<?xml version="1.0" encoding="UTF-8"?>  
<configuration>  
  <section0>  
    <key0>value</key0>  
    <key1>value</key1>  
  </section0>  
  <section1>
```

# INI FILE

```
[section0]
```

```
key0=value
```

```
key1=value
```

```
[section1]
```

```
subsection:key=value
```

```
[section2:subsection0]
```

```
key=value
```

```
[section2:subsection1]
```

```
key=value
```

```
});
```

```
createHostBuilder(string[] args) =>
```

```
    r(args)
```

```
    .UseDefaults(web
```

```
    .ConfigureAppCo
```

```
    .IniFile(
```

```
        "g.ini", o
```

```
    .Startup<Startup>());
```

- section0:key0
- section0:key1
- section1:subsection:key
- section2:subsection0:key
- section2:subsection1:key

=>

;

# ENVIRONMENT VARIABLE

HomePage:ShowGallery      ➔    HomePage\_\_showGallery=true

```
s.Web>set Features__Greeting__GreetingColour=#00FF00
```

# תרגיל 3

קובץ תרגיל מצורף

# LOGGING

---

.NET





# WHAT'S "IN THE BOX" FOR LOGGING

## Nuget:**Microsoft.Extensions.Logging**

- Provides a set of abstractions for logging.
- Supports different logging providers, including Console, Debug, and EventSource.
- Allows developers to create and use custom logging providers.
- Integrates seamlessly with .NET Core applications.
- Configurable via code or configuration files (e.g., appsettings.json)
- Facilitates structured logging with various log levels (Trace, Debug, Information, Warning, Error, Critical).

# USING

```
var serviceProvider = new ServiceCollection()
    .AddLogging(configure =>
    {
        configure.AddConsole();
    })
    .BuildServiceProvider();

// Get logger
var logger = serviceProvider.GetService<ILogger<Program>>();

// Use logger

logger.LogTrace("This is an trace log.");
logger.LogDebug("This is an debug log.");
logger.LogInformation("This is an information log.");
logger.LogWarning("This is a warning log.");
logger.LogError("This is an error log.");
```

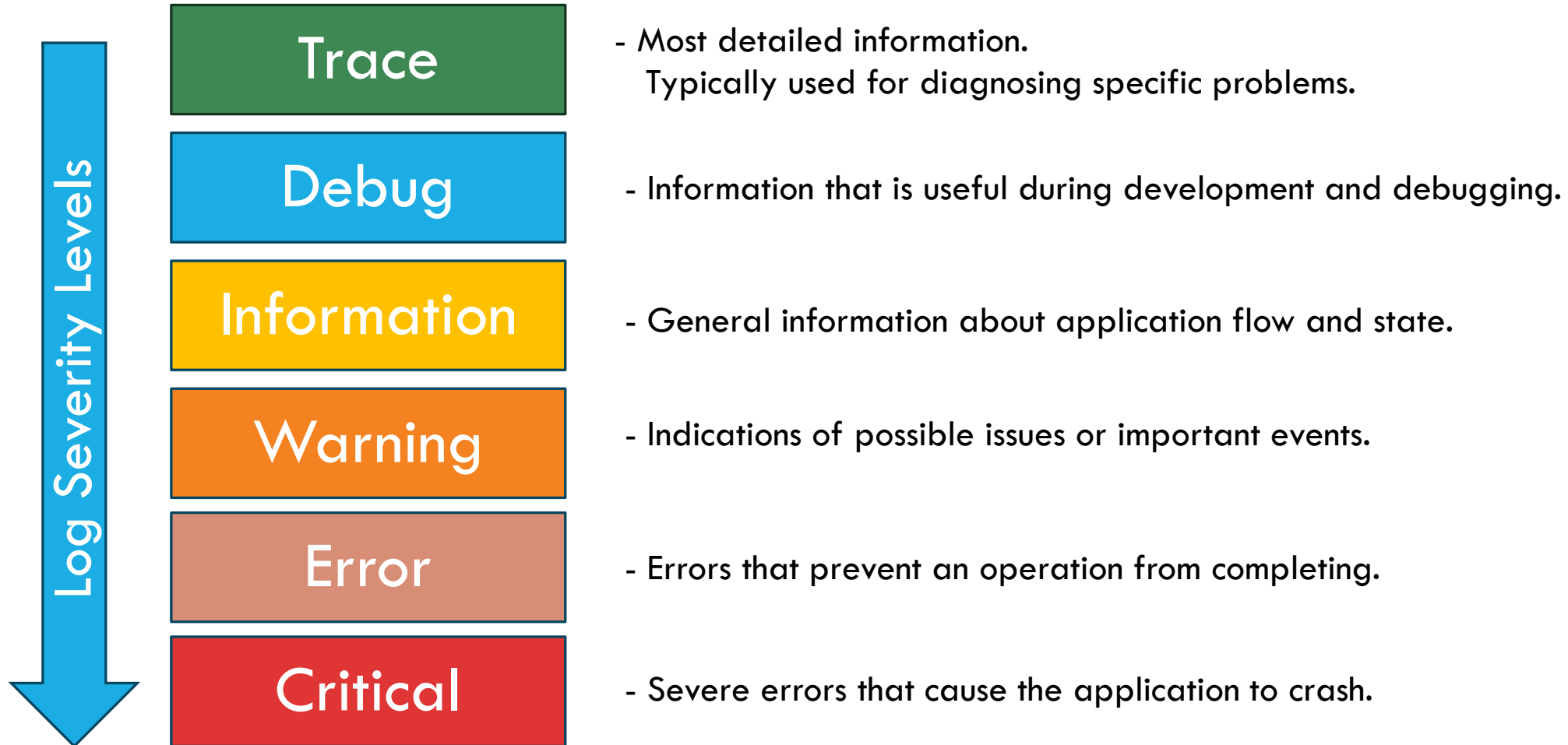
# BUILT-IN LOGGING PROVIDER

Nuget: **Microsoft.Extensions.Logging.Console**

Nuget: **Microsoft.Extensions.Logging.EventSource**

```
.AddLogging(options =>
{
    options.AddConsole();
    options.AddEventSourceLogger();
})
```

# LOG LEVELS



# EXTERNAL LOGGING PROVIDER

## - Serilog

- Simple and efficient logging library for structured logging.
- Supports various sinks (Console, File, Elasticsearch, etc.).

## - NLog

- Flexible and free logging platform.
- Supports a wide range of targets (Console, File, Database, etc.).

## - Log4Net

- Popular logging framework from the Apache Software Foundation.
- Highly configurable and extensible.

## - Seq

- Real-time structured log viewer.
- Integrates with Serilog for easy log management and visualization.

# USING SCOPE WITH PARAMETERS IN NLOG

- Scopes allow grouping of log messages.
- Useful for context-specific logging  
(e.g., within a specific request or operation).
- Parameters provide additional context in logs.

```
// Use logger with scope
using (logger.BeginScope("OperationId: {OperationId}", Guid.NewGuid()))
{
    logger.LogTrace("This is a trace log.");
    logger.LogDebug("This is a debug log.");
    logger.LogInformation("This is an information log.");
    logger.LogWarning("This is a warning log.");
    logger.LogError("This is an error log.");
    logger.LogCritical("This is a critical log.");
}
```

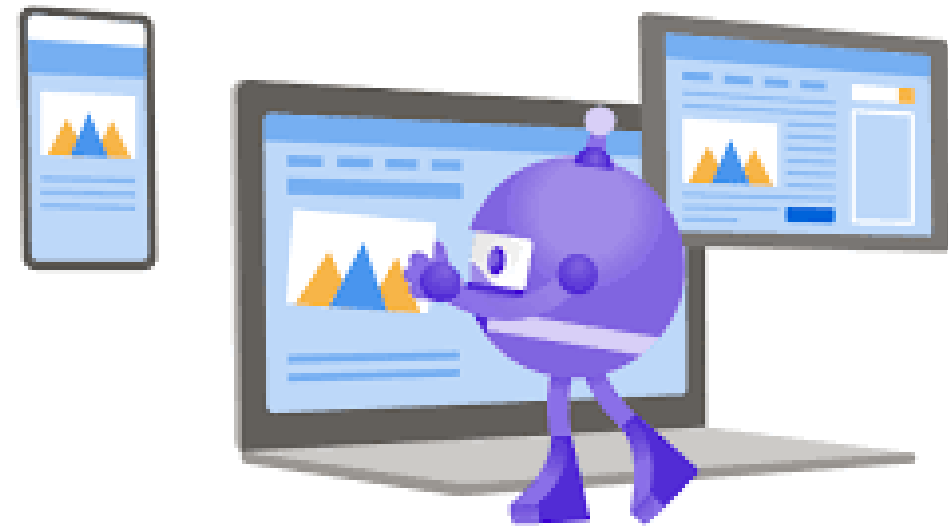
```
using (logger.BeginScope(new Dictionary<string, object>
{
    ["ScopeId"] = Guid.NewGuid(),
    ["UserId"] = 12345,
    ["Operation"] = "DataProcessing"
}))
{
    // Use logger with parameters
    logger.LogTrace("This is a trace log with parameters {Parameters}");
    logger.LogDebug("This is a debug log with parameters {Parameters}");
    logger.LogInformation("This is an information log with parameters {Parameters}");
    logger.LogWarning("This is a warning log with parameters {Parameters}");
    logger.LogError("This is an error log with parameters {Parameters}");
    logger.LogCritical("This is a critical log with parameters {Parameters}");
}
```

# תרגיל 4

קובץ תרגיל מצורף

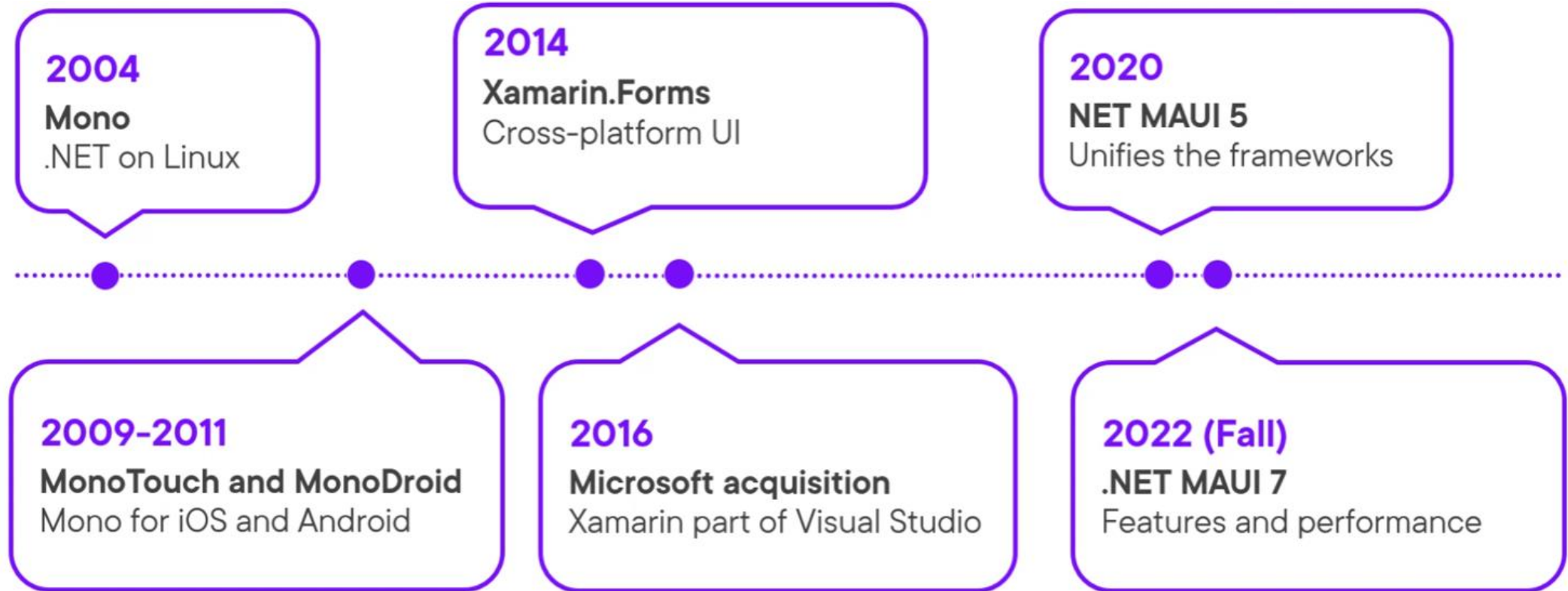
# DESKTOP APP

---





# A BRIEF HISTORY



# THE UI FRAMEWORKS FOR .NET DESKTOP APPS

**Windows Forms**

**WPF**

**UWP**

**WinUI**

**Xamarin**

**.NET MAUI**

# THE UI FRAMEWORKS FOR .NET DESKTOP APPS

## Windows desktop apps

Windows Forms

WPF

UWP

WinUI

## Cross-platform desktop apps

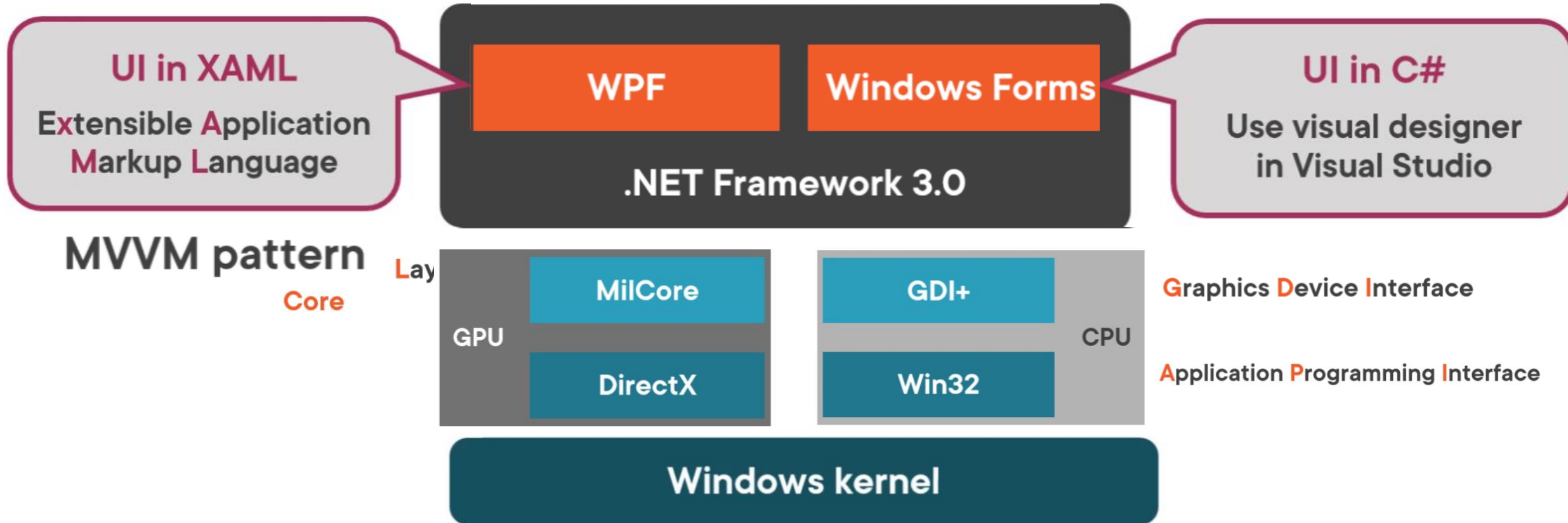
Xamarin

.NET MAUI

Blazor

Open source

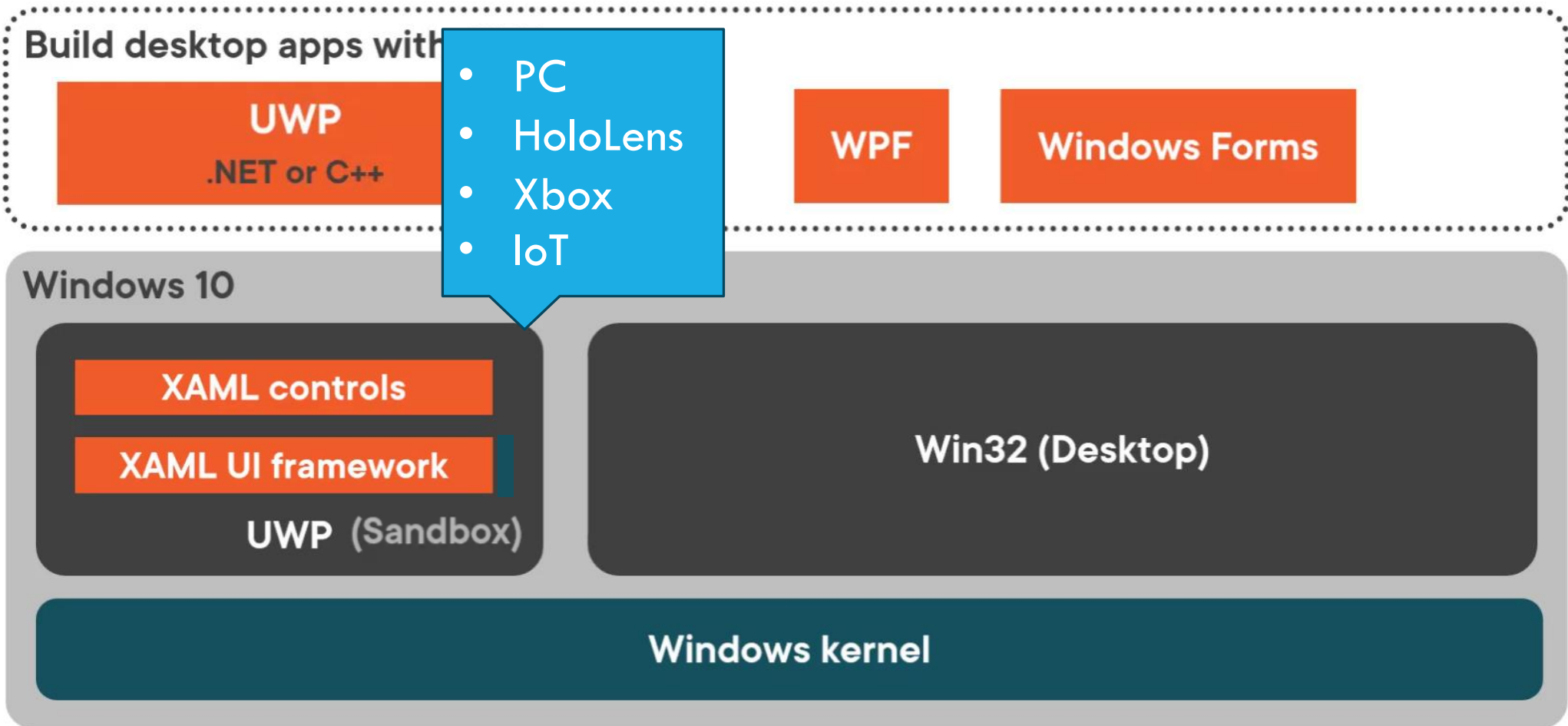
# .NET & WINDOWS FORMS & WPF



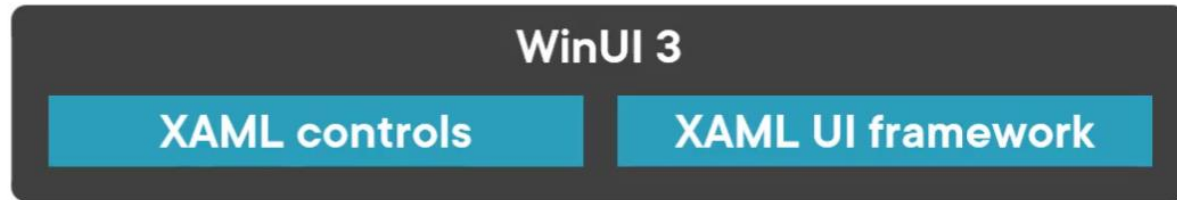
# .NET & WINDOWS FORMS & WPF



# UNIVERSAL WINDOWS PLATFORM (UWP)



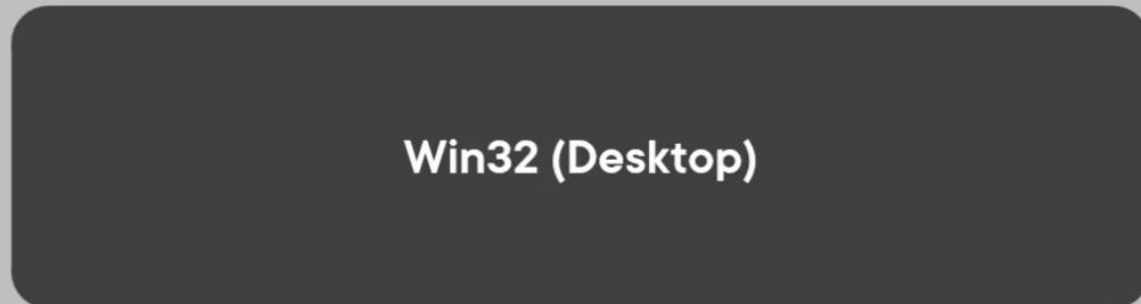
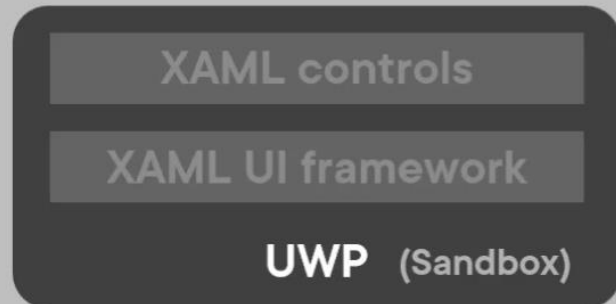
# WINDOWS UI LIBRARY (WINUI)



Build desktop apps with .NET



Windows 10



# XAMARIN AND .NET MAUI

**Before .NET 6.0**

Xamarin.Forms  
XAML and C#

Xamarin.iOS  
(MonoTouch)

Xamarin.Android  
(Mono for android)

UWP



# XAMARIN AND .NET MAUI

Since .NET 6.0

Xamarin.Forms → **.NET Multi-platform App UI (.NET MAUI)**  
XAML and C#

Xamarin.iOS →  
**.NET for iOS**

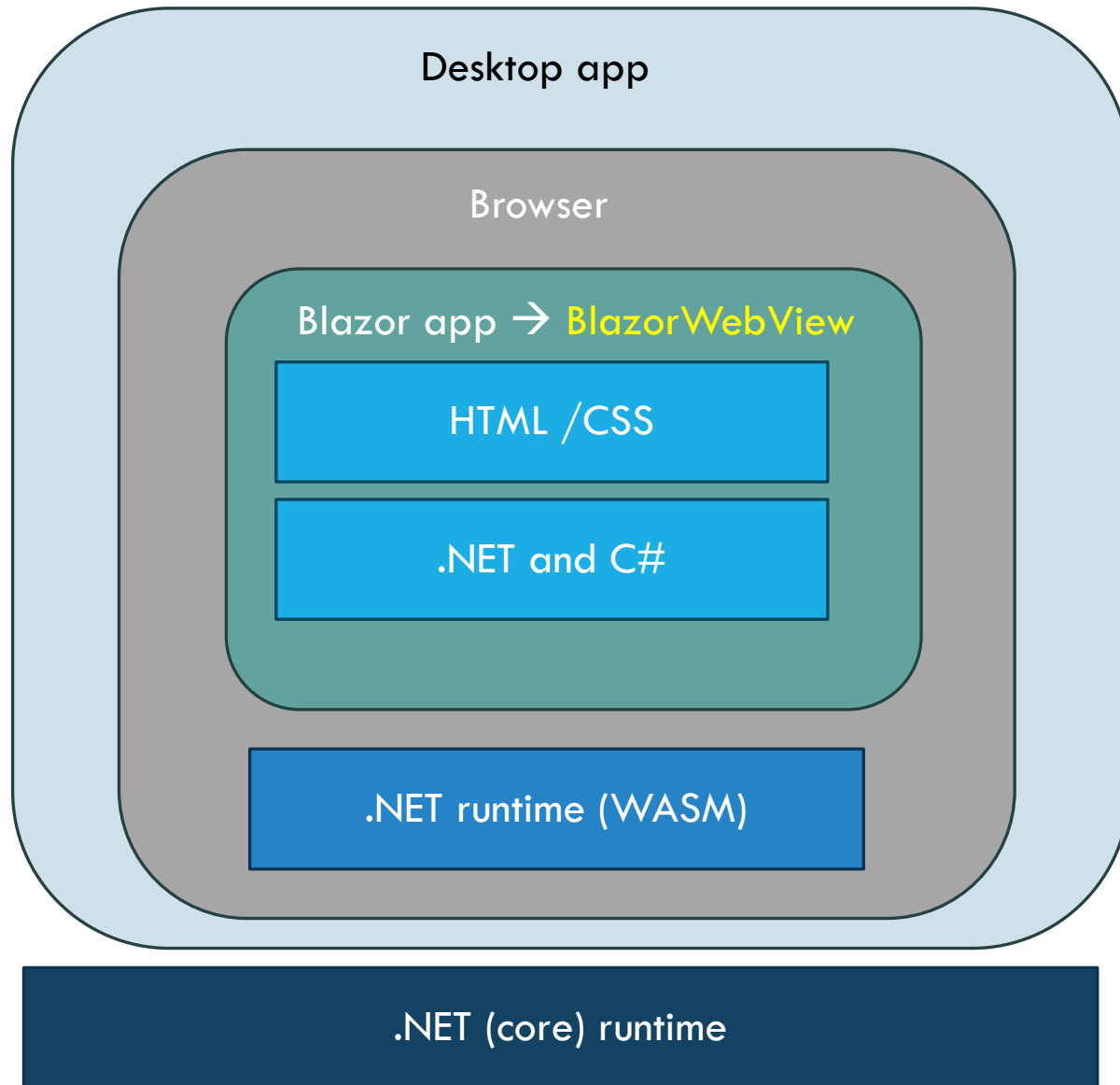
Xamarin.Android →  
**.NET for Android**

UWP →  
**WinUI**

macOS

.NET (Core) runtime

# BLAZOR



BlazorWebView exists for

.NET MAUI

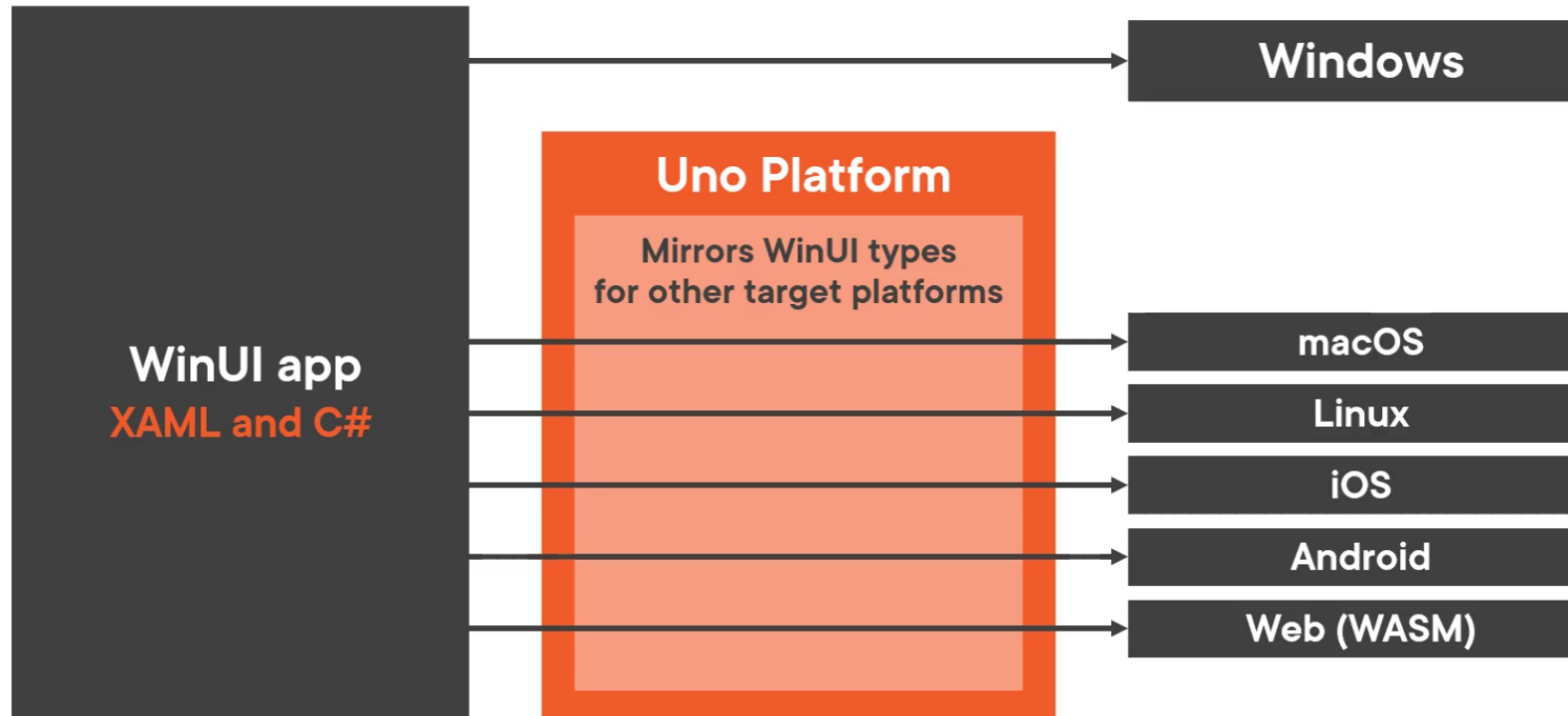
**Blazor Hybrid**

# OPENSOURCE – VALONIA PLATFORM



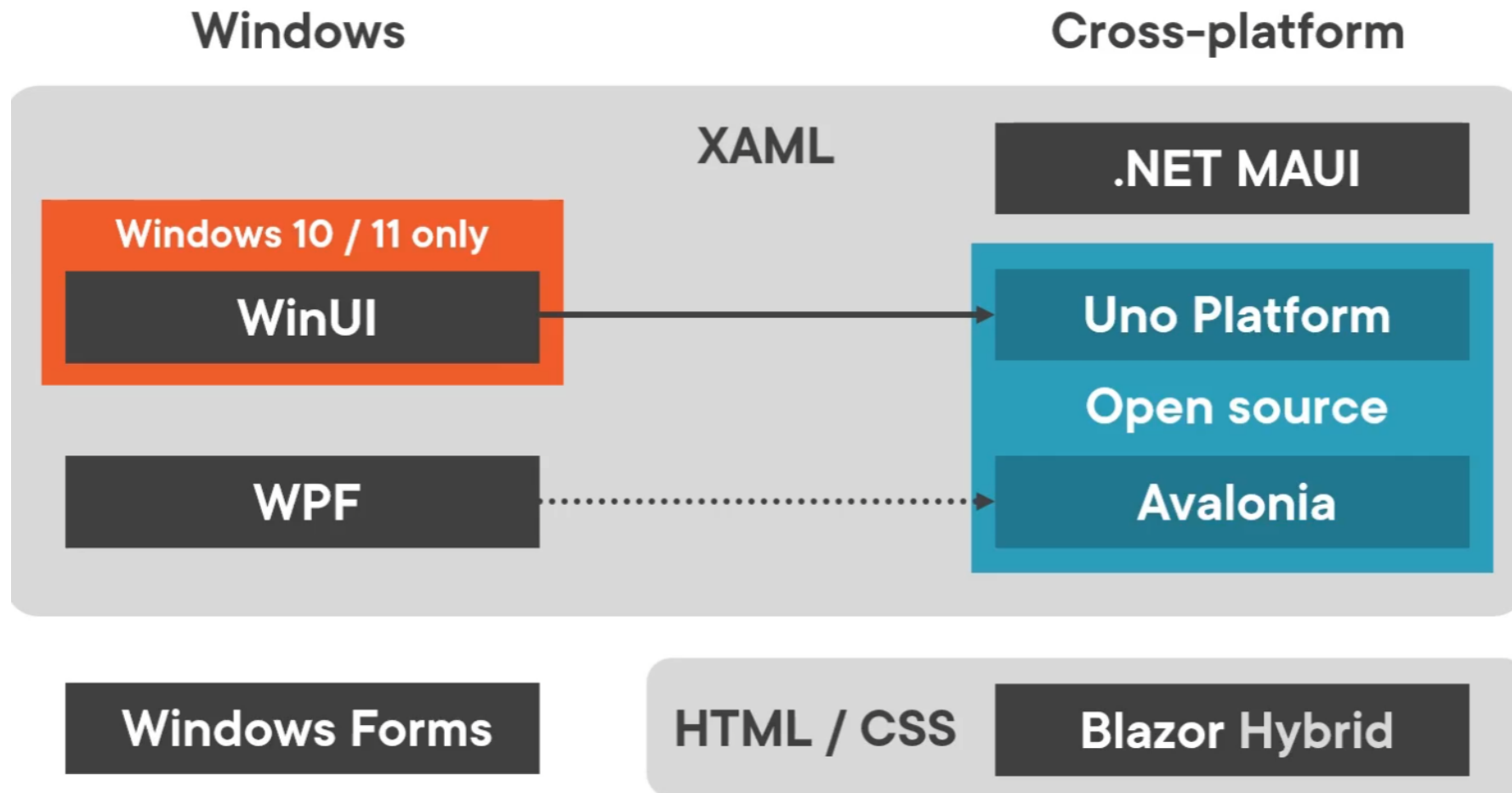
<https://github.com/AvaloniaUI/Avalonia>

# OPENSOURCE – UNO PLATFORM



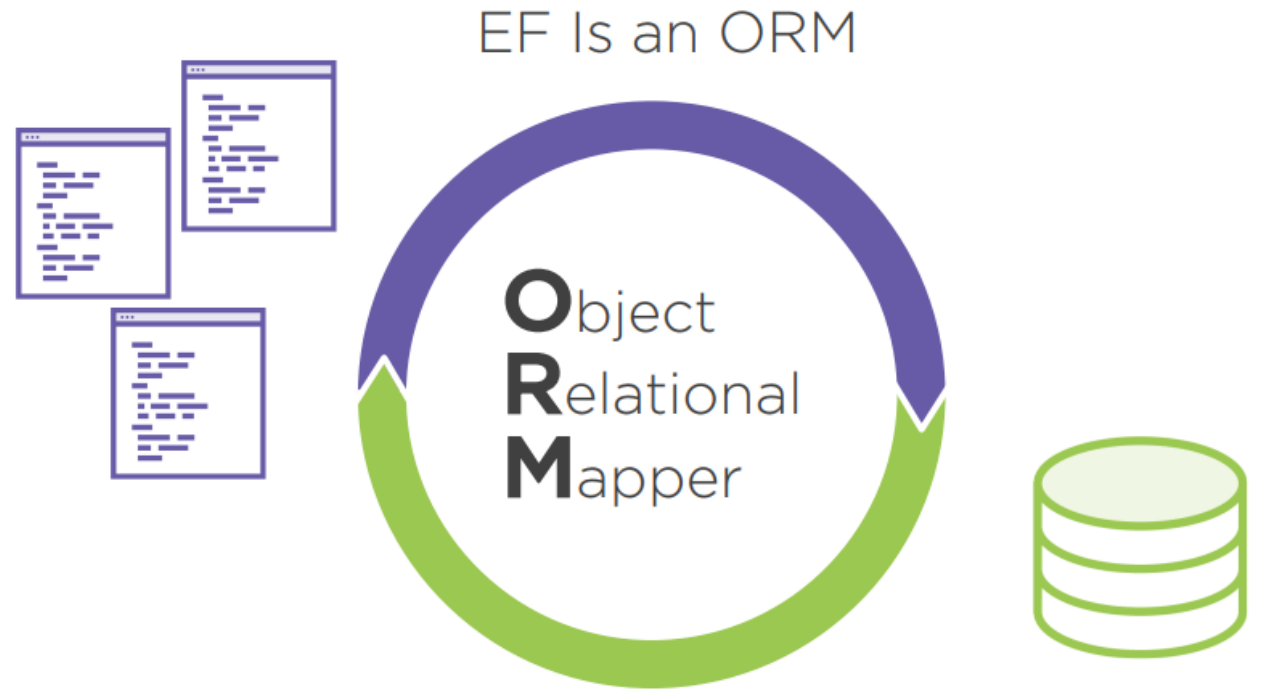
<https://github.com/UnoPlatform/Uno>

# CHOOSE A UI FRAMEWORK FOR YOUR DESKTOP APP



# ENTITY FRAMEWORK CORE

---

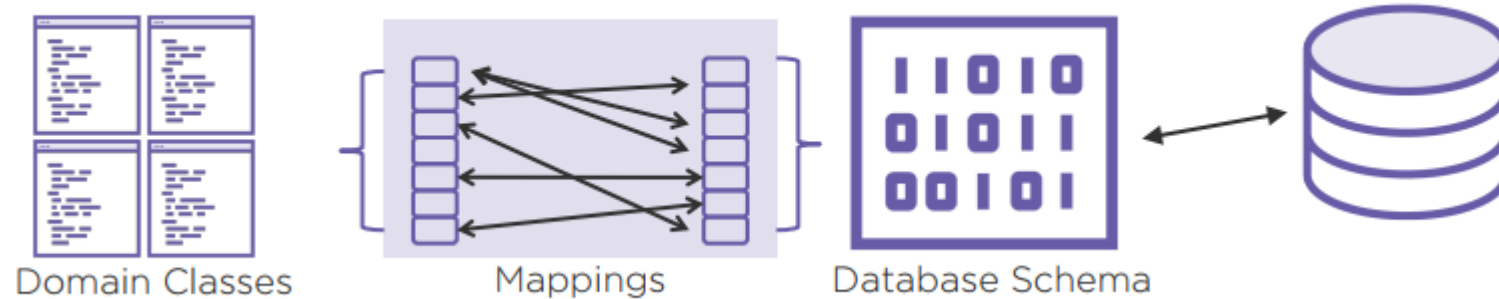


# EF MAPS DIFFERENTLY THAN MOST ORMS

Typical ORM



Entity Framework Core



# WHY THIS ORM, EF CORE?

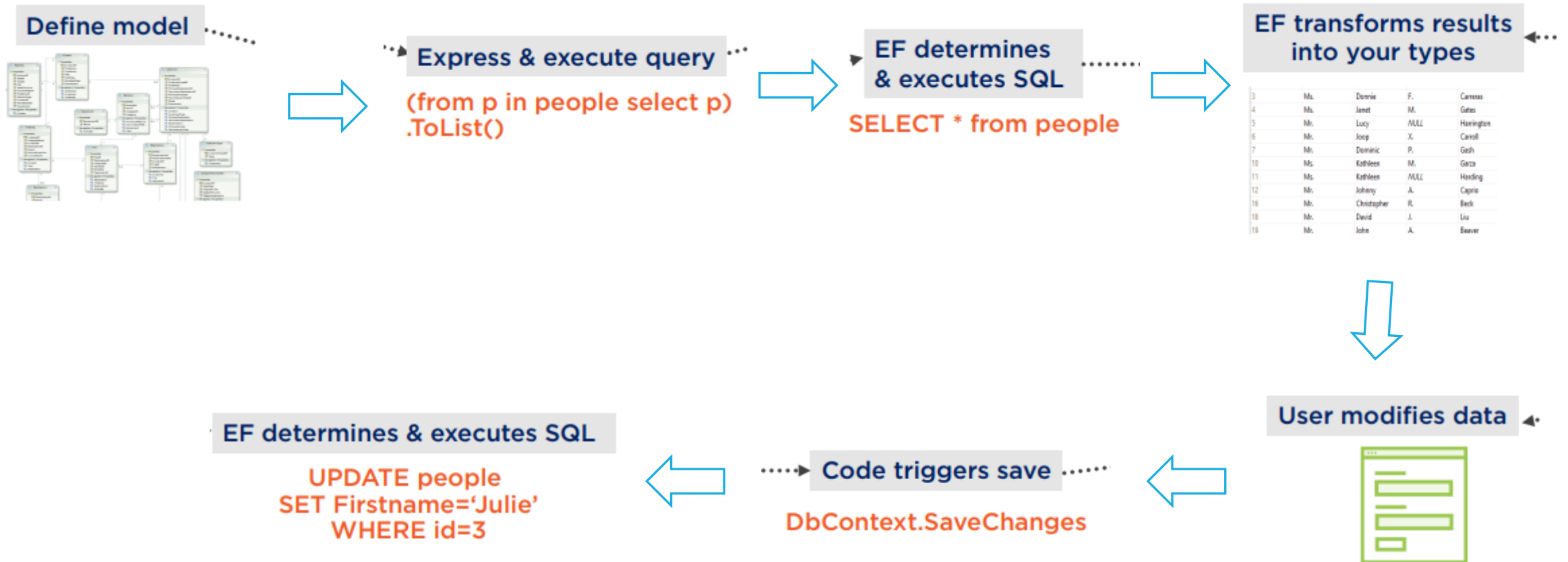
- ❑ Developer productivity
- ❑ First class member of Microsoft .NET stack
- ❑ Consistent query syntax with LINQ to Entities
- ❑ Focus on domain, not on DB, connections, commands, etc



# CURRENTLY AVAILABLE PROVIDERS FOR EF CORE

- ❑ SQL Server (Microsoft)
- ❑ SQLite (Microsoft, Devart)
- ❑ InMemory (Microsoft)
- ❑ SQL Server Compact (Erik Eilskov Jensen (MVP))
- ❑ MySQL (Oracle, Pomelo, Devart)
- ❑ Oracle (Devart)
- ❑ PostgreSQL (Npgsql/Shay Rojansky (MVP), Devart)
- ❑ IBM Data Server DB2 (IBM, Devart)
- ❑ MyCat (Pomelo)
- ❑ Firebird (Rafael Almeida)

# BASIC WORKFLOW



# INSTALL ON PROJECT

Microsoft.EntityFrameworkCore

To Manage

- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.Design
- Microsoft.EntityFrameworkCore.SqlServer

# DATABASE FIRST

1) First time: Add-Migration initial

2) Commit on DB:

**Dev**: update-database -v

**Prod**: script-migration

1) To update: Add-Migration [name\_mig]

# MIGRATIONS RECOMMENDATION

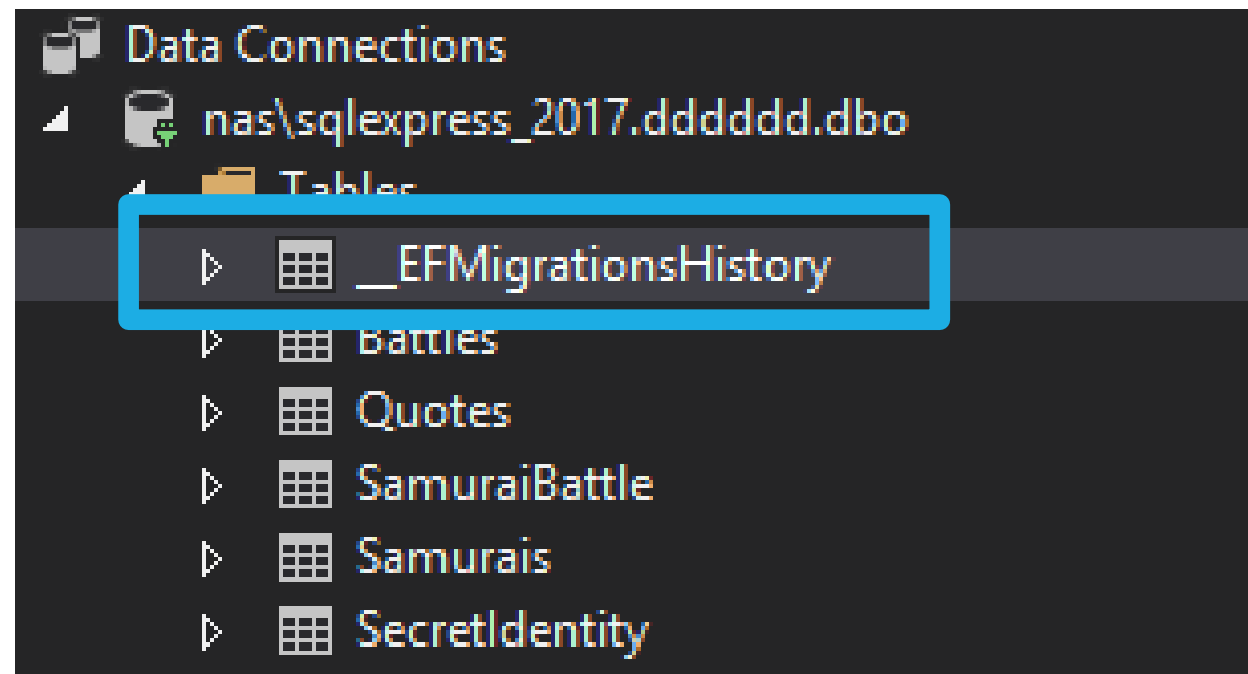
## Migrations Recommendation



**Development database**  
update-database



**Production database**  
script-migration



	MigrationId	ProductVersion
	20200116210814_initial	3.1.1
	NULL	NULL

# REVERSE ENGINEERING AN EXISTING DATABASE



Create DbContext & classes from database

Updating model is not currently supported

Transition to migrations is not pretty ... look for helpful link in resources

PowerShell command:

`scaffold-dbcontext`

# REVERSE ENGINEERING AN EXISTING DATABASE

## Command

```
Scaffold-DbContext "Data Source=localhost;Initial Catalog=dddddd;Integrated Security=True"  
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -DataAnnotations -Context MaccabiPushVoipDbContext
```



# NEW QUERYING ON EF CORE

Like

**EF.Functions.Like(property, %abc%)**

```
_context.Samurais.Where(s=>  
    EF.Functions.Like(s.Name, "%abc%")  
)
```



SQL LIKE(%abc%)

# EXECUTESQLINTERPOLATED

OLD : ~~\_DbSet.SqlQuery~~

New: \_DbSet.FromSqlInterpolated

context.Database.ExecuteSqlInterpolated(

\$"SELECT \* FROM [dbo].[SearchBlogs]({userSuppliedSearchTerm})" )

```
var e = db
```

```
    .Database
```

```
    .ExecuteSqlInterpolated($"UPDATE peoples SET Name={name} WHERE Id={id}");
```


# FROMSQLRAW

```
context.Database.ExecuteSqlRaw  
("SELECT * FROM [dbo].[SearchBlogs]({0})", userSuppliedSearchTerm)
```

```
var e = db  
    .Database  
    .ExecuteSqlRaw("UPDATE peoples SET Name={0} WHERE Id={1}", "New name", id);
```


---

## Samurais

Column Name	Data Type	Allow Nulls
 Id	int	<input type="checkbox"/>
BattleId	int	<input type="checkbox"/>
Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>


FK\_Quotes\_Samurais\_SamuraiId

## Battles

Column Name	Data Type	Allow Nulls
 Id	int	<input type="checkbox"/>
EndDate	datetime2(7)	<input type="checkbox"/>
Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
StartDate	datetime2(7)	<input type="checkbox"/>
		<input type="checkbox"/>

FK\_Samurais\_Battles\_BattleId

## Quotes

Column Name	Data Type	Allow Nulls
 Id	int	<input type="checkbox"/>
SamuraiId	int	<input type="checkbox"/>
Text	nvarchar(MAX)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>