

.NET Conf China  
2022

# 一个音乐爱好者的跨平台开发体验

—— Avalonia 跨平台开发，以 LINUX 平台为例

潘战生  
华南师范大学 网络教育学院



.NET Conf China

# 目录

1. 跨平台开发简述
2. Avalonia 环境设置
3. Avalonia UI 使用
4. P/Invoke(本机互操作性 Native Interoperability)
5. .NET 数据持久化
6. 演示





# 1. 跨平台开发简述

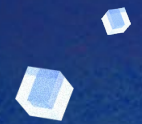
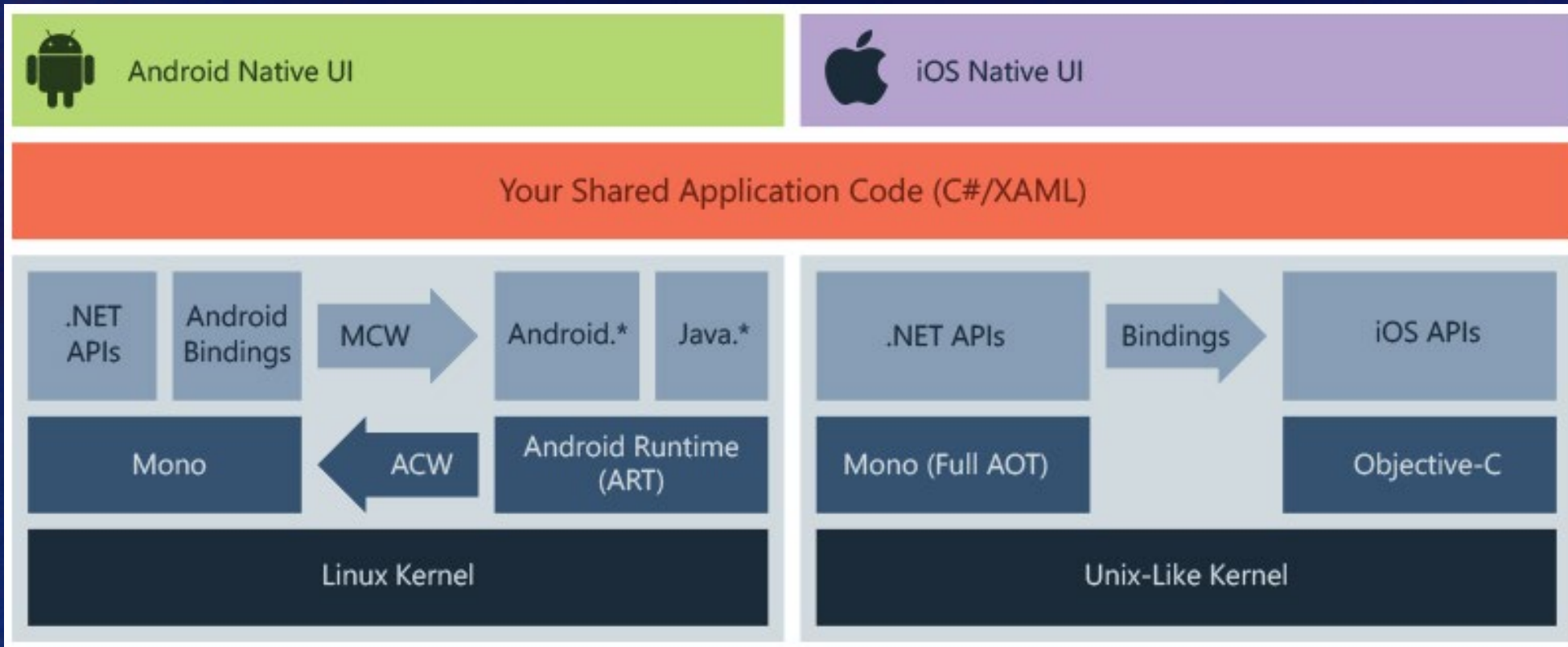


# 为何需要跨平台开发？

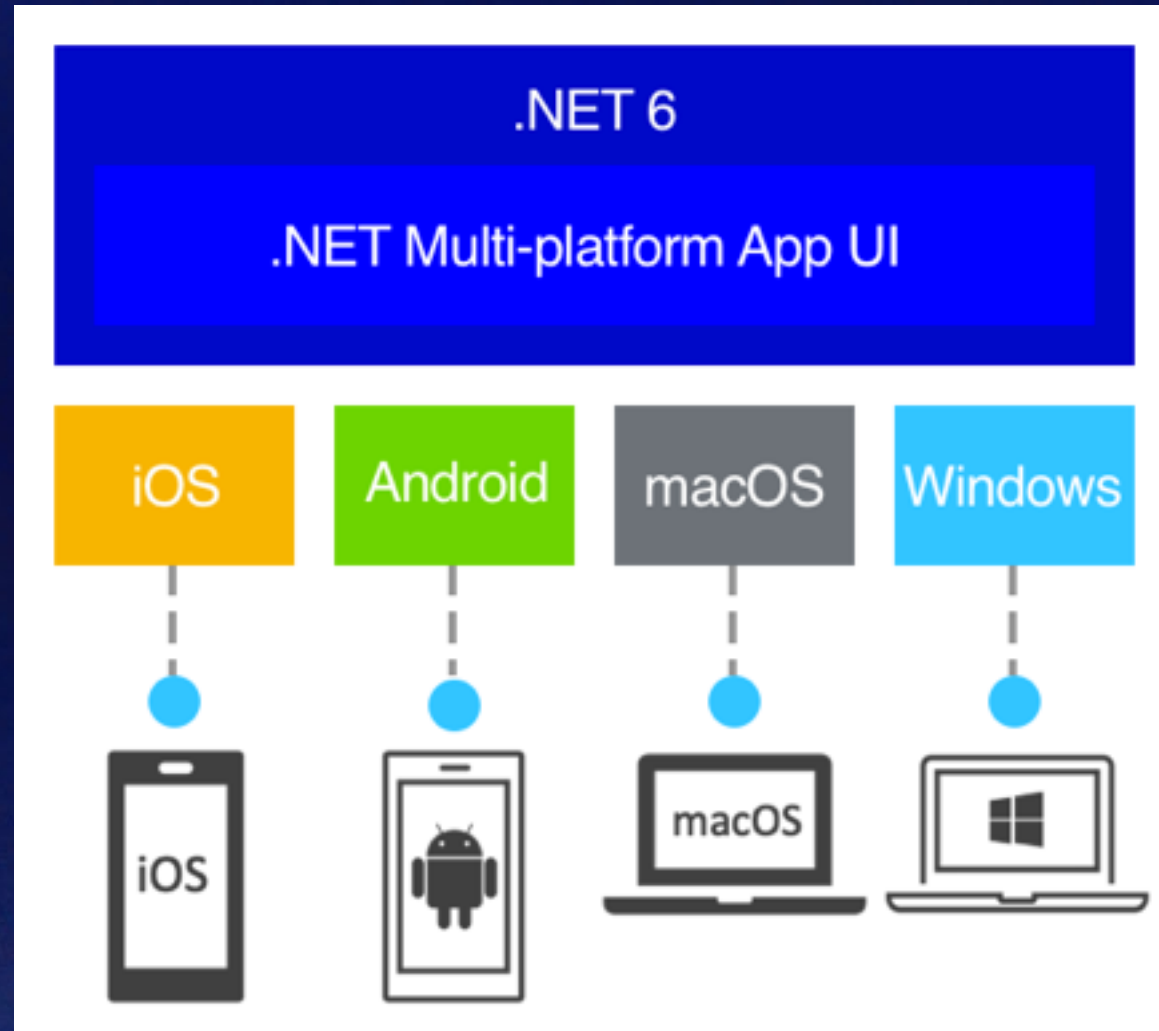
- 应用原有知识
- 减少学习成本
- 减少人力资源成本
- 易于维护
- ○ ○ ○ ○ ○ ○



# .NET Multi-platform Development -- Xamarin



# .NET Multi-platform App UI (MAUI)



# MAUI 唯独缺少了对 LINUX 的支持



LINUX 系统很少用户使用?

LINUX 桌面系统的用户逐年增加

国家政策的原因

个人爱好的原因 (MusicFans)



MusicFans(数年前) 用 Xamarin Forms 技术, 支持

macOS/Windows UWP/iOS/Android

支持的音频文件格式:

wav/flac/ape/tta/wv/dsf/dff/ogg/opus/m4a/aiff/aac/mp3

尚缺乏对 Linux 的支持

App Store Preview



MusicFans 4+

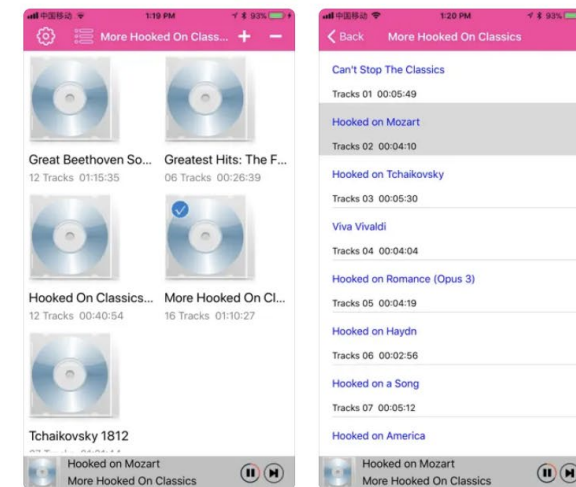
iPZS

Designed for iPad

★★★★★ 5.0 • 2 Ratings

Free

Screenshots iPad iPhone



MusicFans features:

1. Support FLAC(Free Lossless Audio Codec), APE(Monkey's Audio), PCM WAVE and support MP3 Audio file playing.
2. Support CUE playlist
3. Support background and screenlocked playing, automatically continue after phone

What's New

This app has been updated by Apple to display the Apple Watch app icon.

Bug fix

Ratings and Reviews

5.0 out of 5

2 Ratings





# C#、基于 XAML 的UI框架



Uno Platform		Avalonia
	Project	
87	@ Mentions	171
7,176	☆ Stars	15,948
2.2%	📈 Growth	3.7%
10.0	⚡ Activity	10.0
8 days ago	Latest Commit	2 days ago
C#	<> Language	C#
GNU General Public License v3.0 or later	License	MIT License

The number of **mentions** indicates the total number of mentions that we've tracked plus the number of user suggested alternatives.

**Stars** - the number of stars that a project has on GitHub. **Growth** - month over month growth in stars.

**Activity** is a relative number indicating how actively a project is being developed. Recent commits have higher weight than older ones.

For example, an activity of **9.0** indicates that a project is amongst the top 10% of the most actively developed projects that we are tracking.





## 2. Avalonia 环境设置



# Avalonia (Documentation: <https://avaloniaui.net>)



## Open Source

Avalonia UI is open source, free to use and always will be. Clone the source today to dig deeper.



## Visual Studio Extension

The free VS extension provides everything needed to start, including a XAML previewer.



## JetBrains Rider & ReSharper

JetBrains deep understanding of Avalonia UI shines through with rich Rider and ReSharper support.



## Cross Platform

Deploy to the platforms you care about with ease.



## Reuse Existing Skills

Leverage existing knowledge, code and packages to kickstart your project.

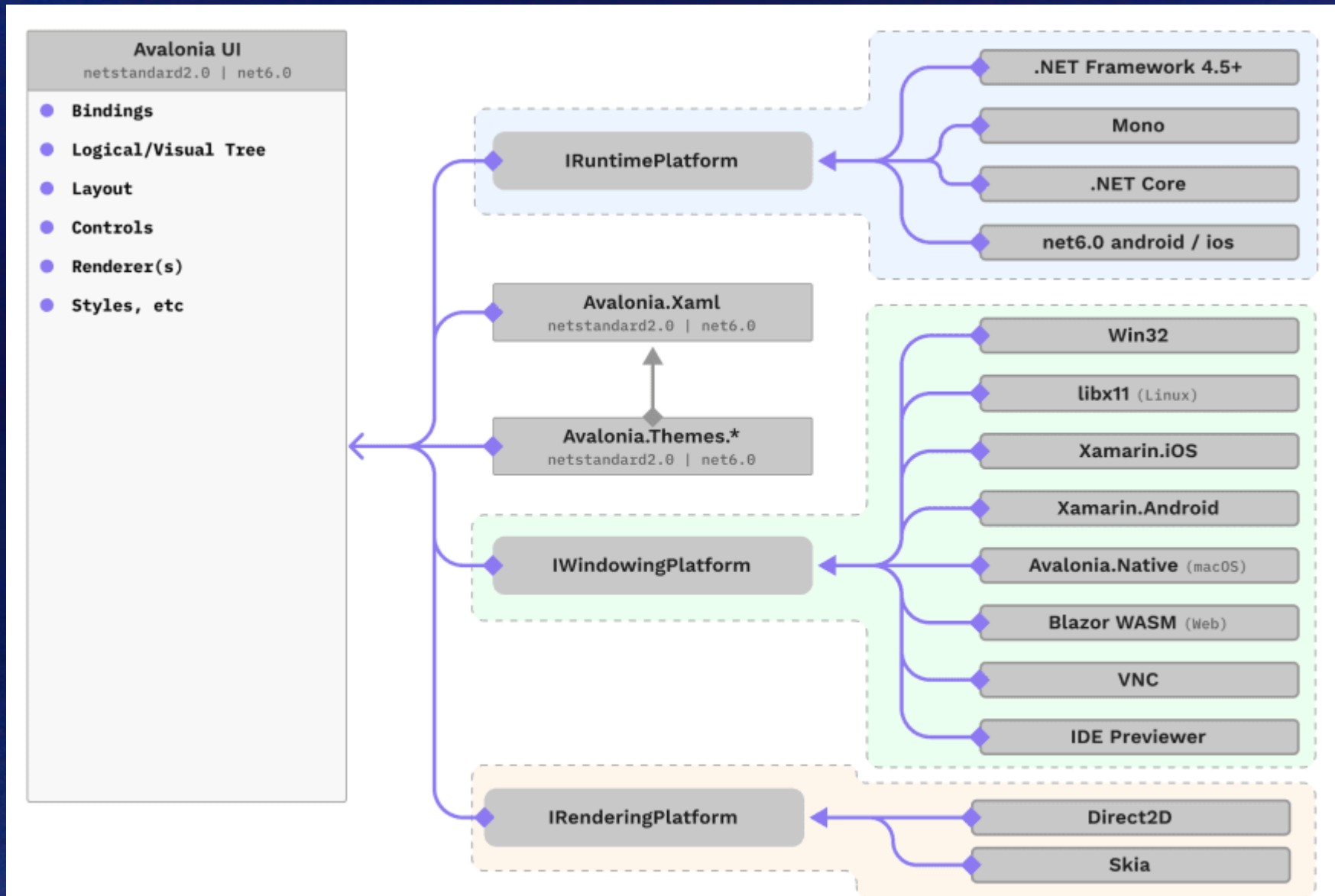


## Commercial Support





Get the support your project needs with one of our commercial support agreements. Help you can count on.



# Avalonia Architecture



# Avalonia UI

 <b>Controls</b> 
AutoCompleteBox
Border
Buttons 
Calendar
Canvas
Carousel
CheckBox
ComboBox
ContentControl
ContextMenu
Decorator
DataGrid 
DatePicker
DockPanel
Expander
Flyouts
Grid
GridSplitter
Image
ItemsControl
ItemsRepeater
LayoutTransformControl
ListBox
MaskedTextBox
Menu
NativeMenu
NumericUpDown
Panel

- Displaying images
- Date and time controls
- Menus

## Layout

Layout controls provide ability for developer to arrange child controls according to specific rules.

### Border

A control which decorates a child with a border and background.

### Canvas

A panel that displays child controls at arbitrary locations.

### DockPanel

A panel which arranges its children at the top, bottom, left, right or center.

### Expander

A control with a header that has a collapsible content section.

### Grid

A flexible grid area that consists of columns and rows.

### GridSplitter

Redistributes space between columns or rows of a Grid control.

### Panel

Base class for controls that can contain multiple children.

### RelativePanel

Defines an area within which you can position and align child objects in relation to each other or the parent panel.

### ScrollBar



# Avalonia 的前景

- Avalonia 目前的版本为 0.10.18，许多功能尚待完善
- Avalonia 的下一个发行版本为 11.0  
<https://github.com/AvaloniaUI/Avalonia/>



# Avalonia 开发环境设置(Ubuntu 22.04)



- .NET SDK 安装

```
sudo apt-get update && \  
sudo apt-get install -y dotnet-sdk-6.0
```

Ubuntu 20.04 需先添加 Microsoft package signing key 及 package repository

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O \  
packages-microsoft-prod.deb  
sudo dpkg -i packages-microsoft-prod.deb  
rm packages-microsoft-prod.deb
```

- 安装 Rider

```
sudo snap install rider --classic
```



# Avalonia 开发环境设置(Ubuntu 22.04)



## 安装 Rider 的 Avalonia Plugin

Add Plugin Repository: <https://plugins.jetbrains.com/plugins/dev/14839>

## 为 Rider 安装 Avalonia Templates

```
dotnet new --search ava  
dotnet new --install Avalonia.Templates
```





# Avalonia 开发环境设置(Windows 10/11)



<https://www.jetbrains.com/idea/> 下载 exe 安装 Rider

<https://www.jetbrains.com/idea/> 下载 Visual Studio 的 Rider .NET 开发扩展 ReSharper



# Visual Studio 2022 安装 Rider 扩展 ReSharper(Windows 10/11)



Create a new project

Recent project templates

- Avalonia .NET Core MVVM App (AvaloniaUI) C#
- .NET MAUI App C#

Search: avalonia

C# All platforms Desktop

No exact matches found

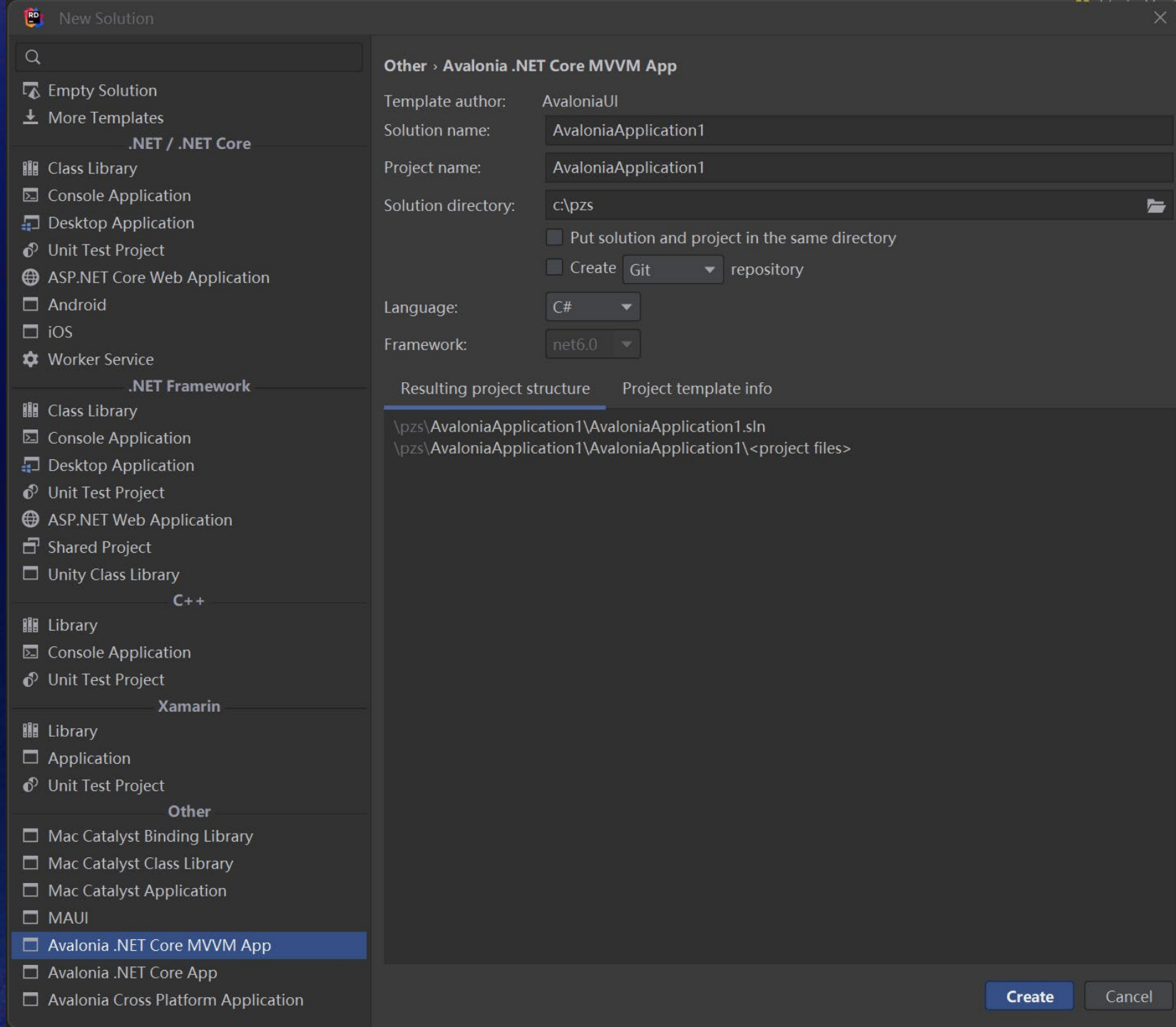
Other results based on your search

- Avalonia .NET Core App (AvaloniaUI)** [New](#)  
A cross-platform Avalonia UI Application  
C# avalonia avaloniaui ui XAML
- Avalonia .NET Core App (AvaloniaUI)** [New](#)  
A project for creating a UI application that can run on .NET Core on Windows, Linux and macOS, iOS and Android  
F# avalonia avaloniaui ui XAML
- Avalonia .NET Core MVVM App (AvaloniaUI)**  
A cross-platform Avalonia UI Application using the MVVM pattern  
C# avalonia avaloniaui ui XAML
- Avalonia .NET Core MVVM App (AvaloniaUI)** [New](#)  
A project for creating an MVVM UI application that can run on .NET Core on Windows, Linux and macOS, iOS and Android  
F# avalonia avaloniaui ui XAML
- Avalonia Cross Platform Application (AvaloniaUI)** [New](#)  
A cross-platform Avalonia UI Application using the MVVM pattern targeting

Next



# Rider 的 Avalonia 应用程序模板



Explorer

- Solution
  - AvaDemo · 1 project
    - AvaDemo
      - Dependencies
        - Imports
          - Sdk.props
          - Sdk.targets
            - Microsoft.CSharp.targets
            - Microsoft.NET.ApiCompat.targets
            - Microsoft.NET.Sdk.BeforeCommon.targets
            - Microsoft.NET.Sdk.targets
            - NuGet.Build.Tasks.Pack.targets
        - .NET 6.0
          - Assemblies
            - Implicit
            - Analyzers
          - Packages
            - Avalonia.Desktop/0.10.18
            - Avalonia.Diagnostics/0.10.18
            - Avalonia.ReactiveUI/0.10.18
            - Avalonia/0.10.18
              - XamlNameReferenceGenerator/1.3.4
          - Frameworks
          - Source Generators
        - Assets
          - avalonia-logo.ico
          - Models
        - ViewModels
          - C# MainWindowViewModel.cs
          - C# ViewModelBase.cs
        - Views
          - MainWindow.axaml
          - App.axaml
          - C# Program.cs
          - C# ViewLocator.cs

C# MainWindowViewModel.cs

```

1 namespace AvaDemo.ViewModels
2 {
3     public class MainWindowViewModel : ViewModelBase
4     {
5         public string Greeting => "Welcome to Avalonia!";
6     }
7 }
    
```

MainWindow.axaml

```

<Window xmlns="https://github.com/avaloniaui"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:vm="using:AvaDemo.ViewModels"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
  x:Class="AvaDemo.Views.MainWindow"
  Icon="/Assets/avalonia-logo.ico"
  Title="AvaDemo">

  <Design.DataContext>
    <vm:MainWindowViewModel/>
  </Design.DataContext>

  <TextBlock Text="{Binding Path=(MainWindowViewModel).Greeting}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Window>
    
```

Explorer

- Solution
  - AvaDemo · 1 project
    - AvaDemo
      - Dependencies
        - Imports
          - Sdk.props
          - Sdk.targets
            - Microsoft.CSharp.targets
            - Microsoft.NET.ApiCompat.targets
            - Microsoft.NET.Sdk.BeforeCommon.targets
            - Microsoft.NET.Sdk.targets
            - NuGet.Build.Tasks.Pack.targets
        - .NET 6.0
          - Assemblies
            - Implicit
            - Analyzers
            - Packages
              - Avalonia.Desktop/0.10.18
              - Avalonia.Diagnostics/0.10.18
              - Avalonia.ReactiveUI/0.10.18
              - Avalonia/0.10.18
              - XamlNameReferenceGenerator/1.3.4
            - Frameworks
            - Source Generators
          - Assets
            - avalonia-logo.ico
            - Models
            - ViewModels
              - C# MainWindowViewModel.cs
              - C# ViewModelBase.cs
            - Views
              - MainWindow.axaml
              - App.axaml
              - C# Program.cs
              - C# ViewLocator.cs

Scratches and Consoles

```

1      using ReactiveUI;
2
3      namespace AvaDemo.ViewModels
4      {
5          public class MainWindowViewModel : ViewModelBase
6          {
7              private string _Greeting = "Hello";
8              public string Greeting
9              {
10                 get
11                 {
12                     return _Greeting;
13                 }
14                 set
15                 {
16                     _Greeting = this.RaiseAndSetIfChanged( backingField: ref _Greeting, value);
17                 }
18             }
19
20             public MainWindowViewModel()
21             {
22                 Greeting = "Hello from Ava";
23             }
24
25             public void ButtonCommand()
26             {
27                 Greeting = "Press Value";
28             }
29         }
30     }
31

```



### 3. Avalonia UI 使用



# Avalonia UI Data Binding



- UI 绑定的 View Model 类

```
<Design.DataContext>  
  <vm:MainWindowViewModel/>  
</Design.DataContext>
```

- UI 绑定 命令

```
<MenuItem Header="_Preference..." Command="{Binding  
Path={MainWindowViewModel}.PreferenceCommand}" CommandParameter="menu item">  
  <MenuItem.Icon>  
    <Image Source="/Assets/gear.png"/>  
  </MenuItem.Icon>  
</MenuItem>
```

- 属性绑定:

```
<Image Name="note" Width="20" Source="{Binding Path= {AlbumItem}.note}"  
HorizontalAlignment="Left"></Image>
```

# WPF(Xamarin.Forms) 绑定UI 的自动更新



自定义类派生于 INotifyPropertyChanged:

```
public class Person : INotifyPropertyChanged
```

属性定义:

```
public string PersonName
{
    get { return name; }
    set
    {
        name = value;
        // Call OnPropertyChanged whenever the property is updated
        OnPropertyChanged();
    }
}
```





# Avalonia 绑定UI 的自动更新

自定义类派生于 ReactiveObject :

```
public class AlbumItem : ReactiveObject
```

属性定义:

```
private Bitmap _note = null;  
public Bitmap note  
{  
    get => _note;  
    set => this.RaiseAndSetIfChanged(ref _note, value);  
}
```



## 4. P/Invoke(Native Interoperability)



# C# Data Types Alias

C# type keyword	.NET type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32
nint	System.IntPtr
nuint	System.UIntPtr
long	System.Int64
ulong	System.UInt64
short	System.Int16
ushort	System.UInt16

# 通过 P/Invoke (Platform Invoke) 调用 C 库函数



## C# 与 C/C++ 基本类型对应

C# type	C/C++ type	Bytes	Range
bool	bool (with int fallback)	usually 1	true or false
char	wchar_t (or char if necessary)	2 (1)	Unicode <u>BMP</u>
byte	unsigned char	1	0 to 255
sbyte	char	1	-128 to 127
short	short	2	-32,768 to 32,767
ushort	unsigned short	2	0 to 65,535
int	int	4	-2 billion to 2 billion
uint	unsigned int	4	0 to 4 billion
long	__int64	8	-9 quintillion to 9 quintillion
ulong	unsigned __int64	8	0 to 18 quintillion
float	float	4	7 significant decimal digits
double	double	8	15 significant decimal digits





# 通过 P/Invoke (Platform Invoke) 调用 C 库函数

- 字符串参数:

```
[DllImport("NativeLib.dll", CharSet = CharSet.Auto)]  
// void do_something(char *str);  
private static extern void do_something(string str);
```

- 字符串返回值 :

```
[DllImport("NativeLib.dll", CharSet = CharSet.Auto)]  
// string * do_something(char *str);  
private static extern IntPtr do_something(string str);
```

再用 Marshal.PtrToString 转换为 Managed Code 的 string



# 通过 P/Invoke (Platform Invoke) 调用 C 库函数

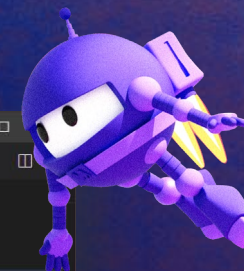


- 用户自定义类(或结构):

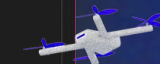
```
[DllImport("NativeLib.dll")]  
// void do_something(MyClass data);  
// 确保各成员按定义顺序保存  
[StructLayout(LayoutKind.Sequential)]  
class MyClass {  
    uint age;  
    uint grade;  
    .....  
}  
private static extern void do_something(MyClass data);
```



# 通过 P/Invoke (Platform Invoke) 调用 C 库函数



```
code sample.cs - Visual Studio Code
code sample.cs x
D: > LiveEvents > dotNetConf > 2022 > code sample.cs
1 public class Bass
2 {
3     #if LINUX
4         //x64 and aarch64 are the same for bass lib
5         public const string BassAssembly = ".libbass.so";
6     #elif WIN
7         public const string BassAssembly = ".bass.dll";
8     #elif OSX
9         public const string BassAssembly = ".libbass.dylib";
10    #endif
11    // typedef uint32_t DWORD; typedef uint64_t QWORD;
12    [DllImport(Bass.BassAssembly, CharSet = CharSet.Auto)]
13    //HSTREAM BASSDEF(BASS_StreamCreateFileUser)(DWORD system, DWORD flags, const BASS_FILEPROCS *proc, void *user);
14    public static extern uint BASS_StreamCreateFileUser (uint system, uint flags, BASS_FILEPROCS proc, IntPtr user);
15
16    [DllImport(Bass.BassAssembly, CharSet = CharSet.Auto)]
17    //HSTREAM BASSDEF(BASS_StreamCreateFile)(BOOL mem, const void *file, QWORD offset, QWORD length, DWORD flags);
18    public static extern uint BASS_StreamCreateFile(bool mem, string file, uint offset, ulong length, uint flags);
19
20
21
22
23
24
25
26
27
28
29
30
Ln 33, Col 1  Spaces: 4  UTF-8  CRLF  C#  Go Live
```



# .NET 数据持久化



- 数据持久化到数据库
- 数据持久化为 JSON 字符串/字节

System.Text.Json.JsonSerializer:

```
public static string Serialize<TValue>(TValue value, JsonSerializerOptions? options = null)
```

```
public static TValue? Deserialize<TValue>(string json, System.Text.Json.JsonSerializerOptions? options = default)
```



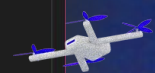


# .NET 对象序列化为字符串



```
code sample.cs x
D: > LiveEvents > dotNetConf > 2022 > code sample.cs
1 // Save current Album list to disk
2 public async void SaveListCommand(string encode)
3 {
4     SaveFileDialog f = new SaveFileDialog();
5     f.InitialFileName = ".mf";
6     string? file = await f.ShowAsync(MainWindow.current);
7     if (file != null)
8     {
9         // serialize dotnet object to text file/utf8 bytes
10        if (encode != "string")
11        {
12            byte[] jsonUtf8Bytes = JsonSerializer.SerializeToUtf8Bytes(_allItems);
13            using var writer = new BinaryWriter(File.OpenWrite(file));
14            writer.Write(jsonUtf8Bytes);
15            writer.Close();
16        }
17        else
18        {
19            var options = new JsonSerializerOptions { WriteIndented = true, MaxDepth = 300};
20            string jsonString = JsonSerializer.Serialize<List<AlbumItem>>(_allItems,options);
21            File.WriteAllText(file,jsonString);
22        }
23    }
24 }
25 }
26
27
28
29
30
```

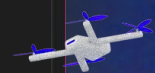
Ln 33, Col 1 Spaces: 4 UTF-8 CRLF C# Go Live



# 字符串分析为 .NET 对象



```
code sample.cs - Visual Studio Code
D: > LiveEvents > dotNetConf > 2022 > code sample.cs
1 // Load Album list from disk
2 private void GetAlbumListFromFile(string file, string encode)
3 {
4     if (encode != "string")
5     {
6         FileStream fs = new FileStream(file, FileMode.Open, FileAccess.Read);
7         BinaryReader br = new BinaryReader(fs);
8         long numBytes = new FileInfo(file).Length;
9         byte[] utf8Bytes = br.ReadBytes((int)numBytes);
10
11         var utf8Reader = new Utf8JsonReader(utf8Bytes);
12         allItems = JsonSerializer.Deserialize<List<AlbumItem>>(ref utf8Reader!);
13     }
14     else
15     {
16         // deserialize json string to object
17         string jsonString2 = File.ReadAllText(file);
18         allItems = JsonSerializer.Deserialize<List<AlbumItem>>(jsonString2!);
19     }
20 }
21 }
22
23
24
25
26
27
28
29
30
```



# Non-Windows System Drawing

Windows : System.Drawing

AvaloniaUI: Avalonia.Media.Imaging

An experimental cross-platform native graphics library:  
<https://github.com/dotnet/Microsoft.Maui.Graphics>



## ⊗ 注意

`System.Drawing` 命名空间对某些操作系统和应用程序类型有一些限制。

- 在Windows, `System.Drawing` 依赖于GDI+操作系统附带的本机库。某些Windows SKUS (Windows Server Core 或 Windows Nano) 不包含此本机库作为 OS 的一部分。如果使用此命名空间并且无法加载库, 则运行时将引发异常。
- 命名空间中的某些类型依赖于 GDI+, 而 Windows 服务以及 ASP.NET Core 和 `System.Drawing` ASP.NET 应用不支持。这些类型在 `System.Drawing.Common` NuGet 包中, 并包括 `System.Drawing.Bitmap` 和 `System.Drawing.Font`。但是, 命名空间中的基元类型 (如 `System.Drawing.Color`、`System.Drawing.Size`、`System.Drawing.Point` 和 `System.Drawing.Rectangle`) 可以在任何应用程序中使用。
- 在 .NET 5 和早期版本中, `System.Drawing.Common` NuGet 包适用于 Windows、Linux 和 macOS。但是, 存在一些平台差异。在 Linux 和 macOS 上, GDI+功能由 `libgdiplus` 库实现。默认情况下, 大多数 Linux 发行版中不会安装此库, 也不支持 GDI+ 和 macOS 上 Windows 的所有功能。还有一些平台, 其中 `libgdiplus` 完全不可用。若要在 Linux 和 macOS 上使用 `System.Drawing.Common` 包中的类型, 必须单独安装 `libgdiplus`。有关详细信息, 请参阅在 Linux 上安装 .NET 或在 macOS 上安装 .NET。
- 在 .NET 6 及更高版本中, `System.Drawing.Common` NuGet 包仅在 Windows 操作系统上受支持。有关详细信息, 请参阅 仅支持 `System.Drawing.Common` Windows。

如果无法与应用程序一同使用, 建议的替代项包括 `System.Drawing` ImageSharp、SkiaSharp、Windows 映像组件和 `Microsoft.Maui.Graphics`。



## 5. DEMO



# Thank you!

Let's build amazing apps with .NET 7  
[get.dot.net/7](https://get.dot.net/7)

