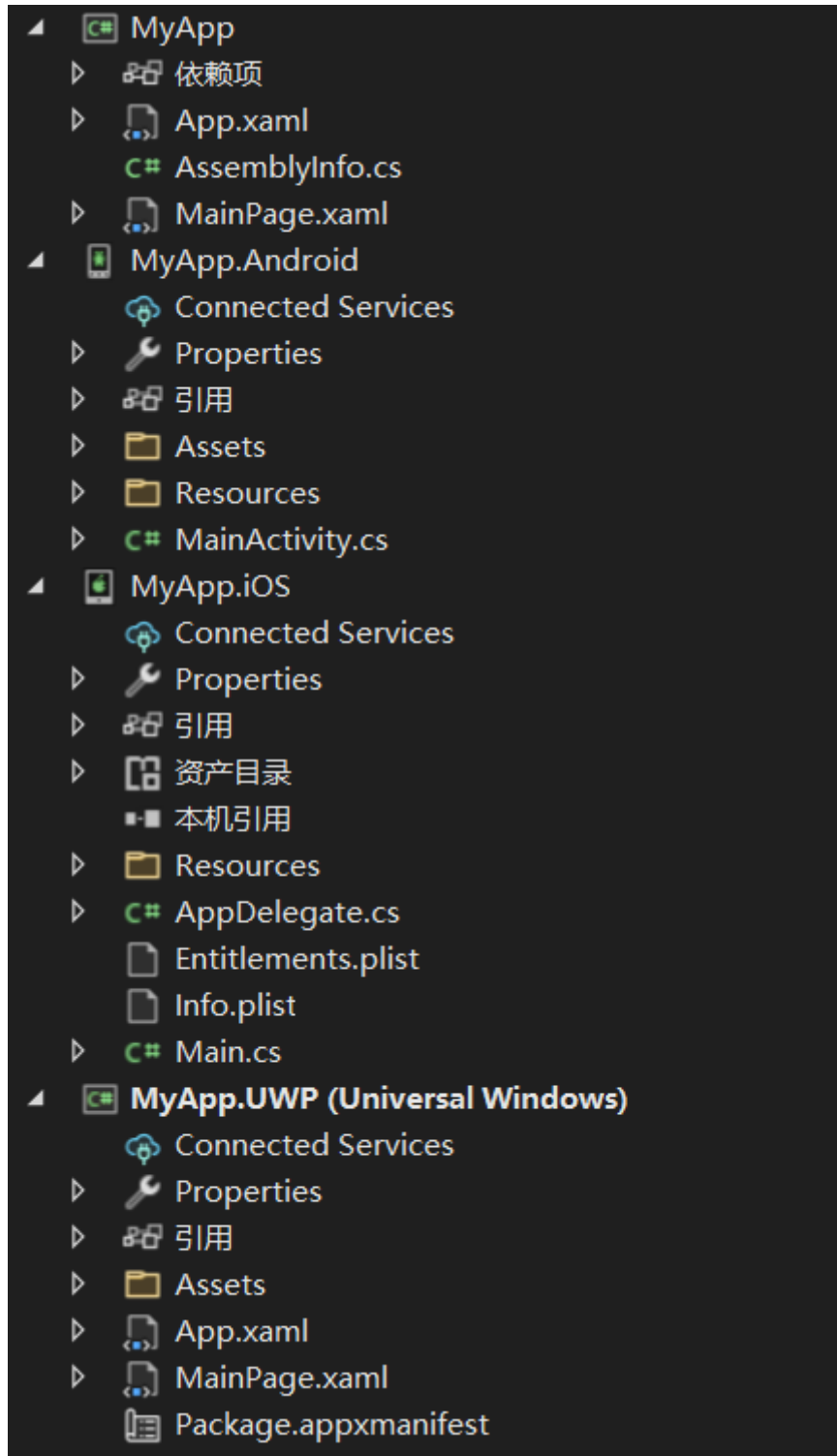


# Xamarin.Forms介绍

如下所示, 展示了一个标准的空白Xamarin.Forms项目, 同时包含了UWP、Android、IOS项目。



**MyApp**: 第一个项目是 **.NET Standard** (标准)库。该项目包含可以跨平台共享的所有代码, 此项目是您将编写应用的所有用户界面以及不需要与本机(不同的平台) API 交互的所有代码的位置。

在 Xamarin.Forms 解决方案中, .NET Standard 库包含所有跨平台代码以及支持共享代码所需的所有文件。例如, 它包含用户界面定义、处理用户界面并对其做出反应的命令性代码、必须嵌入到应用包中的图像、资源文件和配置文件。言下之意, 特定于平台的代码将不会包含在 .NET 标准库中。相反, 它将被放置在Android, iOS和UWP项目中。

## 什么是.NET Standard?

[.NET Standard](#) 为 API 提供了一组规范，所有 .NET 开发平台（如 .NET Framework、.NET Core 和 Mono）都必须实现这些规范。这样可以统一 .NET 平台并避免将来的碎片化。通过创建 .NET Standard 库，可以确保代码在任何 .NET 平台上运行，而无需选择任何目标。这也解决了可移植库的一个常见问题，因为每个可移植库都可以针对一组不同的平台，这意味着库和项目之间可能存在不兼容性。Microsoft 有一[篇关于](#) .NET Standard 及其目标和实现的有趣博客文章，将解释对此规范的任何疑问。

**MyApp.Android**：第二个项目，其后缀是**Android**，是一个Xamarin.Android原生项目。它具有对共享代码和 Xamarin.Forms 的引用，并实现了应用在 Android 设备上运行所需的基础结构。

从技术上讲，Xamarin.Forms 是一个 .NET 库，它是[开源](#)的，并作为 NuGet 包提供，当你创建新解决方案时，Visual Studio 会自动安装在所有项目中。然后，Visual Studio 中的 NuGet 包管理器将通知你可用的更新。由于即使电脑处于脱机状态也应允许创建 Xamarin.Forms 解决方案，因此 Visual Studio 会在本地缓存中安装 NuGet 包的版本，这通常不是可用的最新版本。因此，建议您将解决方案中的 Xamarin.Forms 和其他 Xamarin 包升级到最新版本，并且仅升级到稳定版本。虽然 Alpha 和 beta 中间版本通常可用，但它们的唯一预期用途是试验仍在开发中的新功能。

**MyApp.iOS**：第三个项目，其后缀是**iOS**，是一个Xamarin.iOS本机项目。此版本还引用了共享代码和 Xamarin.Forms，并实现了应用在 iPhone 和 iPad 上运行所需的基础结构。

**MyApp.UWP**：第四个也是最后一个项目是本机通用 Windows 项目（UWP），该项目引用了共享代码，并实现了你的应用在 Windows 10 设备（桌面和移动设备）上运行的基础结构。

## MyApp.Android

Xamarin.Android 使您的 Xamarin.Forms 解决方案能够在 Android 设备上运行。**MainActivity.cs** 文件表示 Xamarin 生成的 Android 应用的启动活动。在 Android 中，可以将活动视为具有用户界面的单个屏幕，并且每个应用都至少有一个。在此文件中，Visual Studio 添加了不应更改的启动代码，尤其是您在最后两行代码中看到的 Xamarin.Forms 的初始化代码。

```
[Activity(Label = "MyApp",
    Icon = "@mipmap/icon",
    Theme = "@style/MainTheme",
    MainLauncher = true,
    ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.KeyboardHidden)
0 个引用
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    0 个引用
    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);

        Xamarin.Essentials.Platform.Init(this, savedInstanceState);
        global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
        LoadApplication(new App());
    }
    0 个引用
    public override void OnRequestPermissionsResult(int requestCode, string[] permissions,
    {
        Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

**ConfigurationChanges**：

配置更改项代表, 设备处于不同状态的情况下,例如: 屏幕方向改变、多窗口、尺寸变化时, 应用是否会被重启, 重启行为旨在通过利用与新设备配置相匹配的备用资源来自动重新加载您的应用, 从而帮助它适应新配置。

参考: [处理配置变更](#) | [Android 开发者](#) | [Android Developers](#)

**Xamarin.Essentials.Platform.Init(this, savedInstanceState);**

注册平台生命周期事件

**global::Xamarin.Forms.Forms.Init(this, savedInstanceState);**

注册设备平台服务、自定义渲染器、自定义字体等

**LoadApplication(application);**

平台控件渲染器(Renderer)、设置首页(加载页面中的Renderer渲染到布局当中)

“**Resources**”文件夹也非常重要, 它包含子文件夹, 您可以在其中为不同的屏幕分辨率添加图标和图像。此类文件夹的名称以 **drawable** 开头, 每个文件夹表示特定的屏幕分辨率。

资源可以包含多种类型的项。以下是最常见的资源:

- 图像
- 布局文件
- 用于文本和本地化的字符串
- 样式

Xamarin [文档](#)全面解释了如何在 Android 上为不同分辨率提供图标和图像。

“解决方案资源管理器”中的“属性”元素允许您访问项目属性, 就像对任何 C# 解决方案所做的那样。对于 Xamarin, 在“**应用程序**”选项卡中, 可以指定 Visual Studio 应用于生成应用包的 Android SDK 版本。



Visual Studio 会自动选择可用的最新版本, 并提供一个下拉列表, 您可以在其中选择不同版本的 SDK。请注意, 如果开发计算机上未安装 SDK 版本, 则 Visual Studio 会使用 \* 符号标记该版本。您可以使用 Android SDK 管理器工具下载 SDK, 也可以通过**工具>选项>Xamarin > Android 设置>自动安装 Android SDK**启用自动下载。此处的 SDK 选择不会影响您要定位的 Android 的最低版本;相反, 它与 Visual Studio 将使用的生成工具的版本相关。我的建议是保持默认选择不变。

“**Android 清单**”选项卡更为重要。在这里, 可以指定应用的元数据, 例如名称、版本号、图标以及用户必须授予应用程序的权限。

MyApp.Android

应用程序

Android 清单

Android 选项

Android 包签名

生成

生成事件

引用路径

配置(C): 不可用

平台(M): 不可用

应用程序名称:

MyApp.Android

程序包名称:

com.companyname.myapp

应用程序图标:

应用程序主题:

@style/MainTheme

版本号:

1

版本名称:

1.0

安装位置:

首选内部

最低 Android 版本:

Android 5.0 (API 级别 21 - Lollipop)

目标 Android 版本:

Android 12.0 (API 级别 31 - S)

所需权限:

## MyApp.iOS

对于 Xamarin.iOS 项目，**AppDelegate.cs** 文件包含 Xamarin.Forms 初始化代码，不应更改。您可以在本项目中添加所有需要访问本机 API 和特定于平台的功能的代码，在 **Info.plist**（属性列表）中，通过每个选项卡，您可以配置应用元数据、最低目标版本、支持的设备和方向、功能（如 Game Center 和地图集成）、视觉资产（如启动图像和图标）以及其他高级功能。

**Info.plist** 文件表示 iOS 中的应用清单，因此仅与 Xamarin.iOS 相关。实际上，如果您有 Xcode 和本机 iOS 开发的经验，那么您已经知道此文件。与 Android 不同，iOS 操作系统包含自动应用于最敏感资源的限制策略，尤其是涉及安全和隐私的资源。此外，iOS 8.x、9.x、10.x 和 11.x 在操作系统处理这些选项的方式上存在差异。

## MyApp.UWP

Xamarin.Forms 解决方案中的通用 Windows 平台项目是一个普通的 UWP 项目，它引用了共享代码和 Xamarin.Forms 包。

在 **App.xaml.cs** 文件中，可以看到初始化代码。**Package.appxmanifest** 则是用于配置应用程序清单，您可以通过双击“解决方案资源管理器”中的文件来编辑它。

Visual Studio 具有用于 UWP 清单的编辑器，你至少可以配置应用元数据、图标等视觉资产以及功能。其中包括在测试和分发应用之前需要指定的权限，Windows 10 将在访问需要权限的资源之前要求用户进行确认。

Package.appxmanifest

应用程序 视觉对象资产 功能 声明 内容 URI 打包

使用此页可以设置用于标识和描述应用的属性。


显示名称: MyApp.UWP


入口点: MyApp.UWP.App


默认语言: en-US [更多信息](#)


说明: MyApp.UWP

支持的旋转: 一个指示应用的方向首选项的可选设置。

  
☐ 横向

  
☐ 纵向

  
☐ 横向翻转

  
☐ 纵向翻转

锁定屏幕通知: (未设置)

资源组:

**磁贴更新:**  
通过定期轮询 URI 更新应用磁贴。URI 模板可包含“{language}”和“{region}”令牌，在运行时将替换这些令牌以生成要轮询的 URI。  
[更多信息](#)

重复: (未设置)

URI 模板: