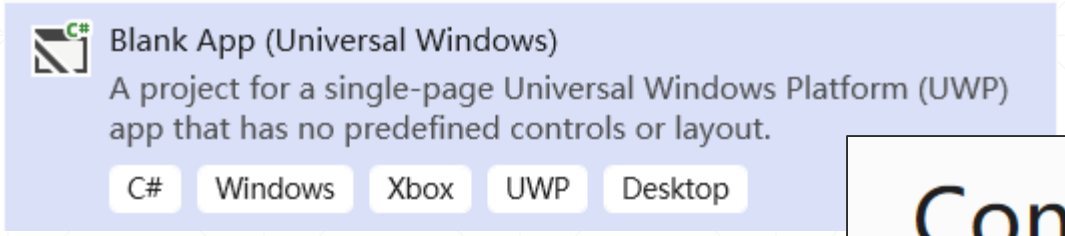
UWP系列之四

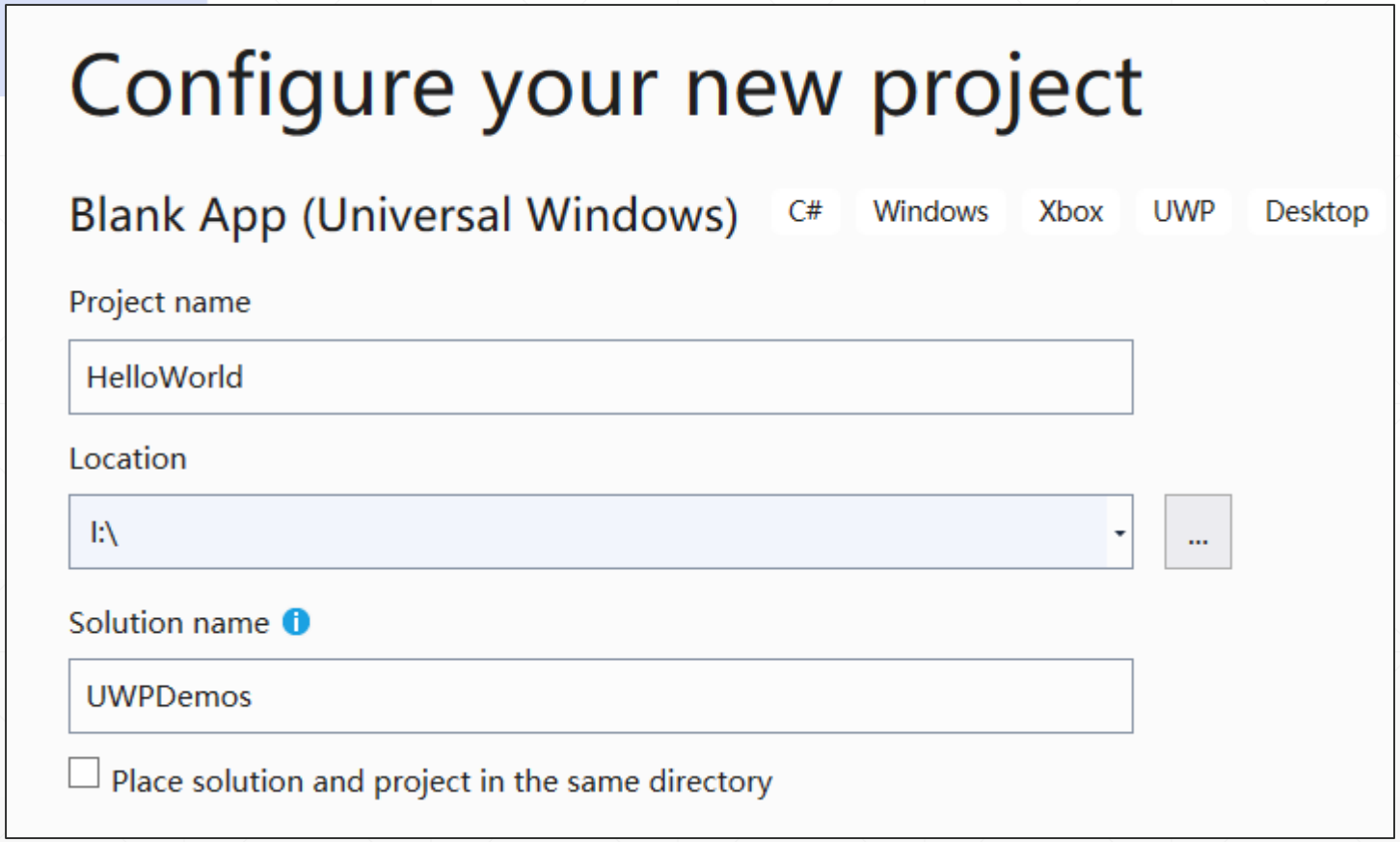
UWP核心编程模型初探

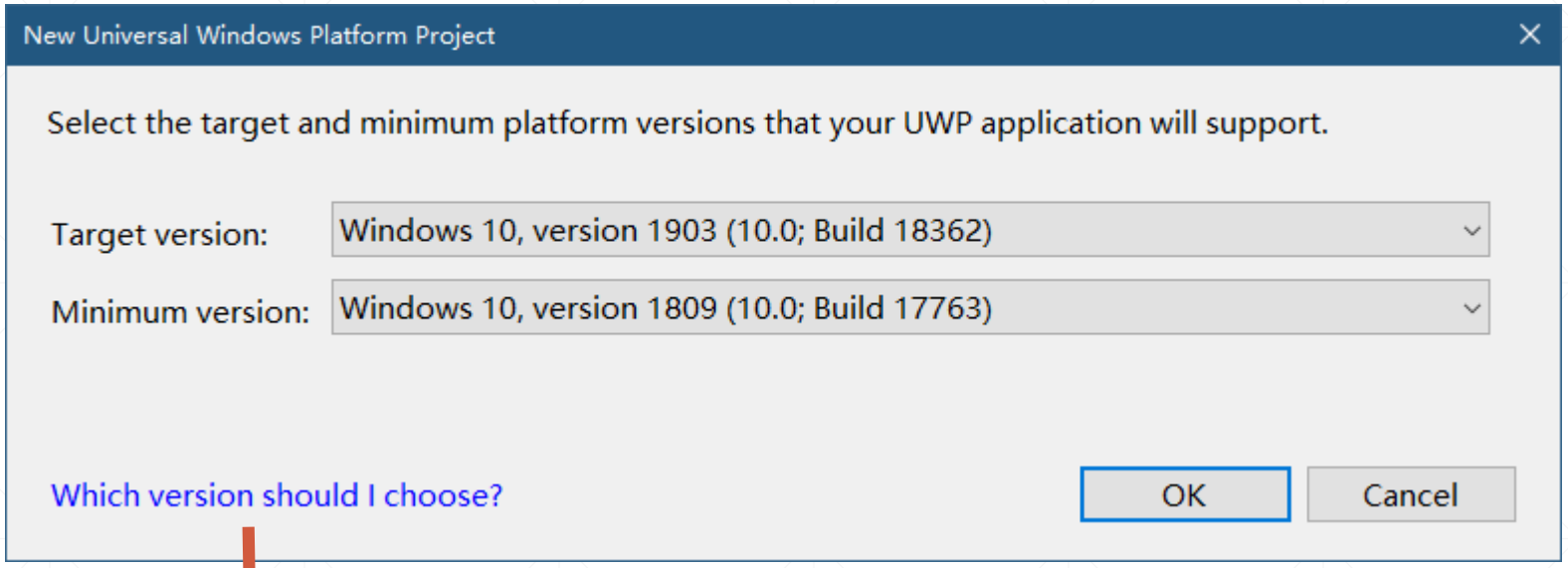
北京理工大学计算机学院
金旭亮

创建UWP项目



默认情况下，所有 UWP 应用程序在开发与调试阶段均以 CoreCLR 为目标框架，但它的发布版本则会被.NET Native提供的AOT编译器编译为Windows 10本地代码。



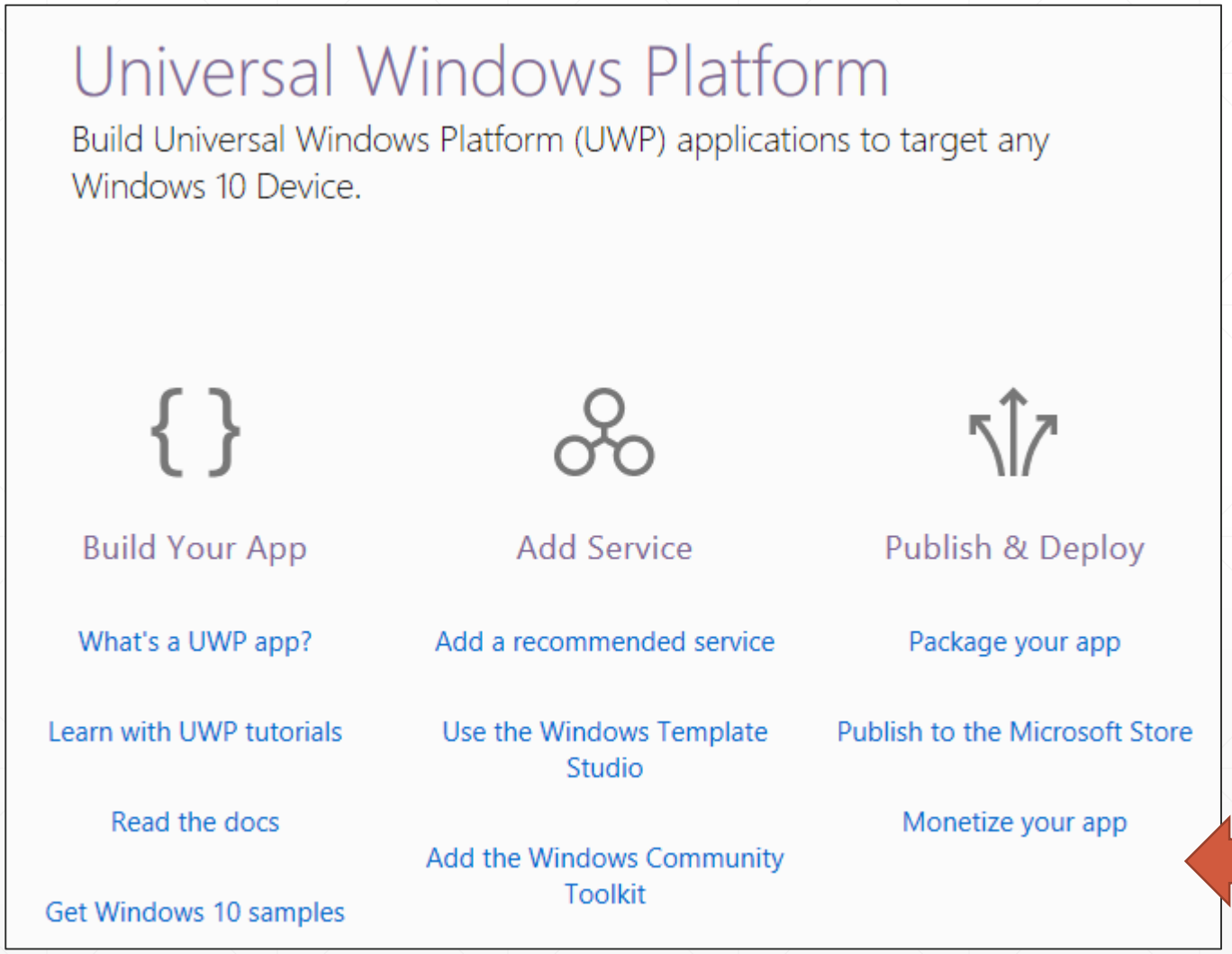


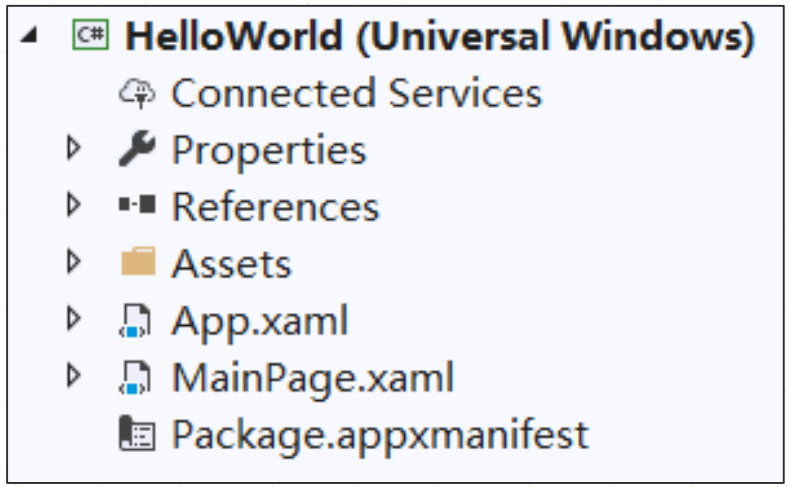
点击链接显示一个网页……

版本	描述
内部版本 18362（版本 1903）	这是最新版本的 Windows 10，已于 2019 年 4 月发布。 此版本中的一些突出功能如下： * **XAML 岛：** 现在，Windows 10 支持在非 UWP 桌面应用程序中使用 UWP 控件。 如果你正在进行 WPF、Windows 窗体或 C++ Win32 方面的开发，请查看如何将最新的 Windows 10 UI 功能添加到现有应用。 * **适用于 Linux 的 Windows 子系统：** 现在，可以直接在 Windows 中访问 Linux 文件，并使用多个新的命令行选项。 请参阅关于 WSL 中的最新信息。 有关此版本的 Windows 中添加的这些功能以及许多其他功能的信息，请访问开发人员中心，或面向开发人员的 Windows 10 中的新增功能上的详细信息页面。
内部版本 17763（版本 1809）	此版本的 Windows 10 已于 2018 年 10 月发布。 **请注意，必须使用 Visual Studio 2017 或 Visual Studio 2019 才能以此版 Windows 为目标。** 此版本中的一些突出功能如下： * **Windows 机器学习：** Windows 机器学习现已正式推出，为尖端机器学习模型提供更快的评估和支持等功能。 若要了解有关此平台的详细信息，请参阅 Windows 机器学习。 * **流畅设计：** 菜单栏、命令栏浮出控件和 XAML 属性动画等新功能已添加到 Windows 10。 请参阅流畅设计概述中的最新信息。 有关此版本的 Windows 中添加的这些功能以及许多其他功能的信息，请访问开发人员中心，或面向开发人员的 Windows 10 中的新增功能上的详细信息页面。

UWP应用需要指定目标版本和最低版本。

生成的“空白”UWP项目

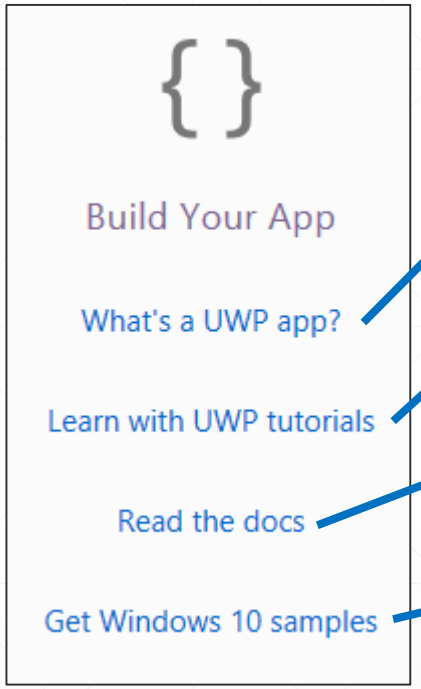




项目文件清单

给出的一些有用链接

有用的链接

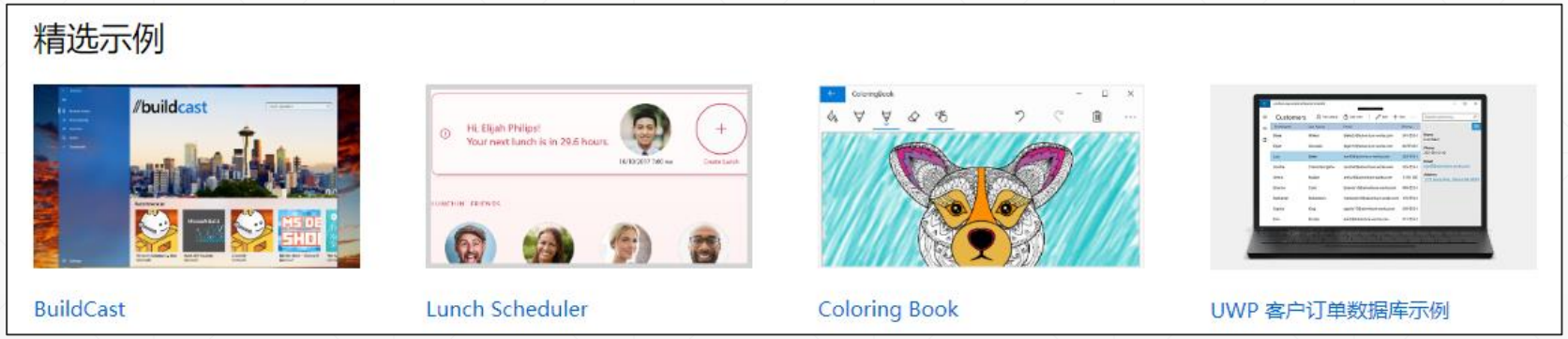


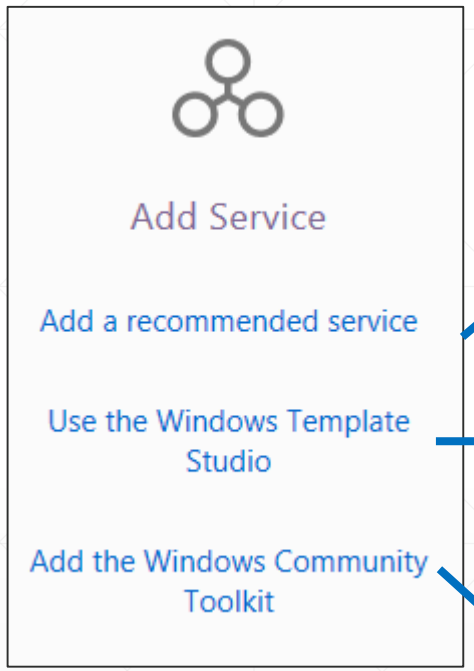
导向UWP官方文档中的“概述”部分

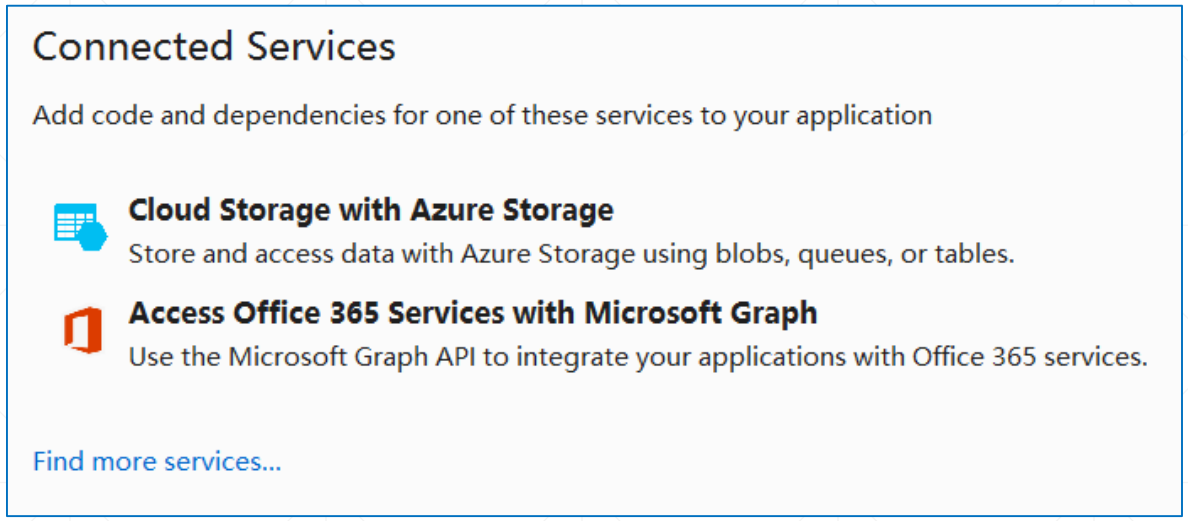
UWP官方文档提供的向导（写得一般）

UWP官方文档入口

放在GitHub上的官方示例（推荐）

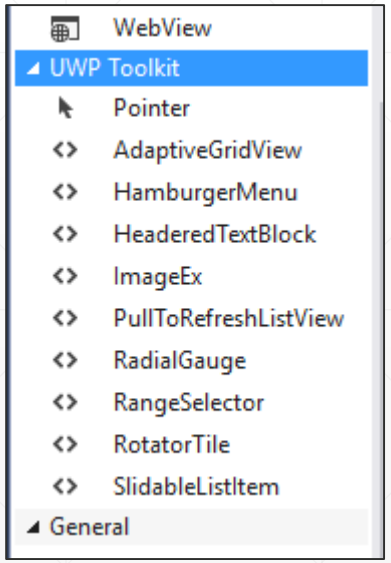






社区开发的一个适用于Visual Studio的UWP应用生成器（非常Cool）

社区维护的UWP应用工具箱，可以使用NuGet安装，功能非常全面与强大

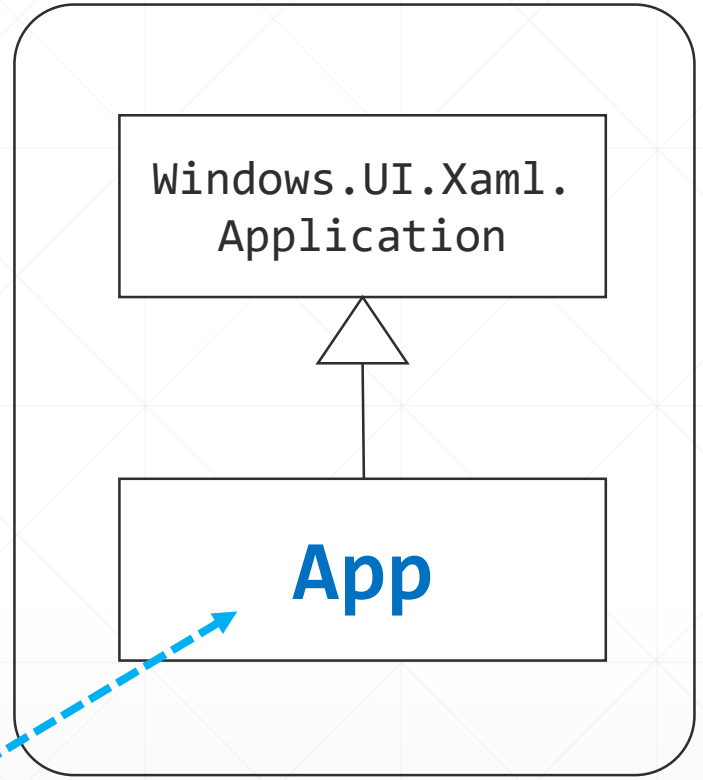


UWP程序的入口点

App.xaml

```
<Application
    x:Class="HelloWorld.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorld">

</Application>
```

关联



程序入口点是

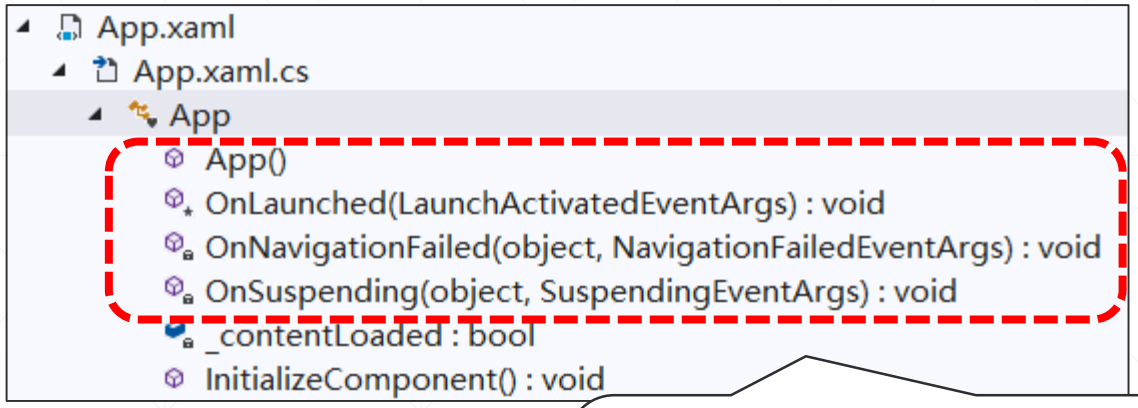
App.g.i.cs

```
public static class Program
{
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsof
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    0 references
    static void Main(string[] args)
    {
        global::Windows.UI.Xaml.Application.Start((p) => new App());
    }
}
```

App对象在Main()方法中被实例化。

App对象的职责-1

管理整个UWP应用的生命周期



可以重写特定的方法，响应UWP应用特定生命周期所触发的事件

App.xaml.cs

App对象的职责-2

保存整个App范围内可用的资源

```xaml
<Application
    x:Class="HelloWorld.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorld">
    <Application.Resources>
        <!--在此放置整个UWP项目均可使用的资源-->
    </Application.Resources>

</Application>
```

App.xaml

App对象的职责-3

处理未捕获的异常

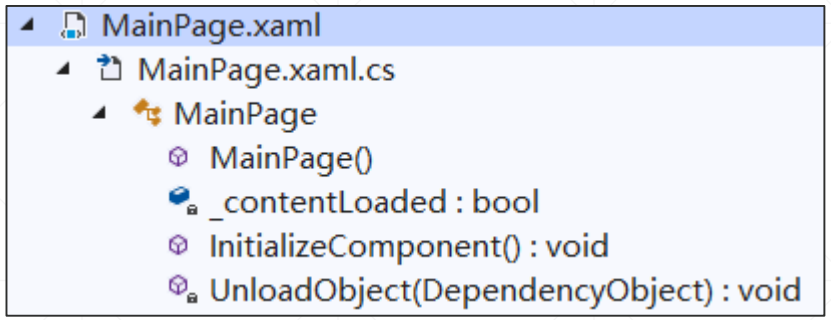
App.g.i.cs

```
3 references
partial class App : global::Windows.UI.Xaml.Application
{

    private bool _contentLoaded;

    1 reference
    public void InitializeComponent()
    {
        if (_contentLoaded)
            return;

        _contentLoaded = true;

        DebugSettings.BindingFailed += (sender, args) =>
        {
            global::System.Diagnostics.Debug.WriteLine(args.Message);
        };
        UnhandledException += (sender, e) =>
        {
            if (global::System.Diagnostics.Debugger.IsAttached)
                global::System.Diagnostics.Debugger.Break();
        };
    }
}
```

处理绑定异常

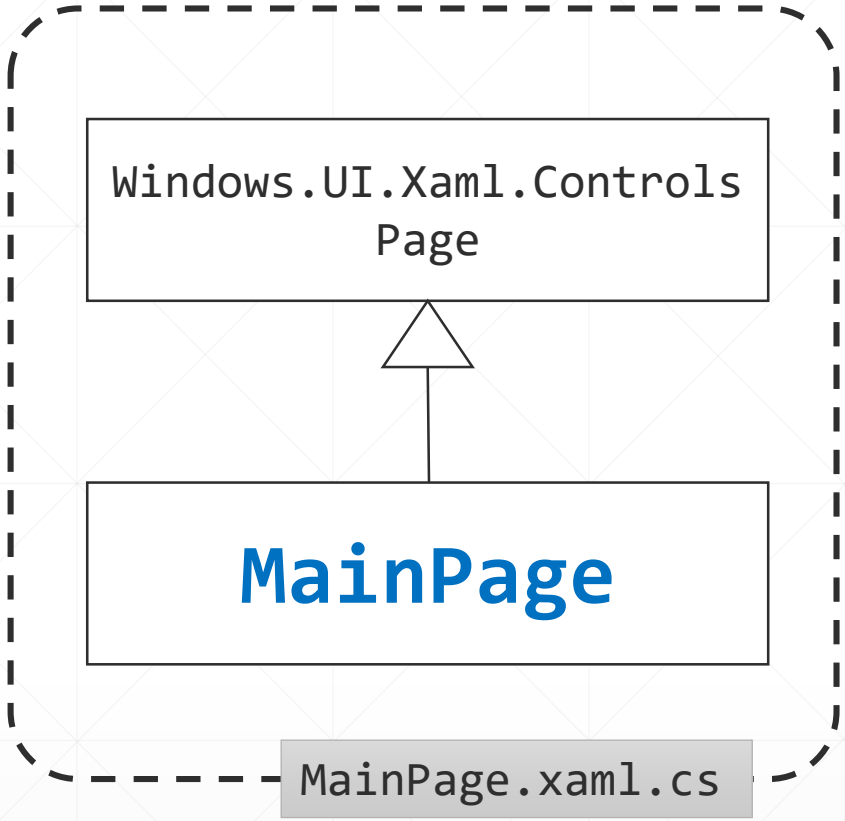
处理未捕获的异常

UWP应用的UI以页面为中心



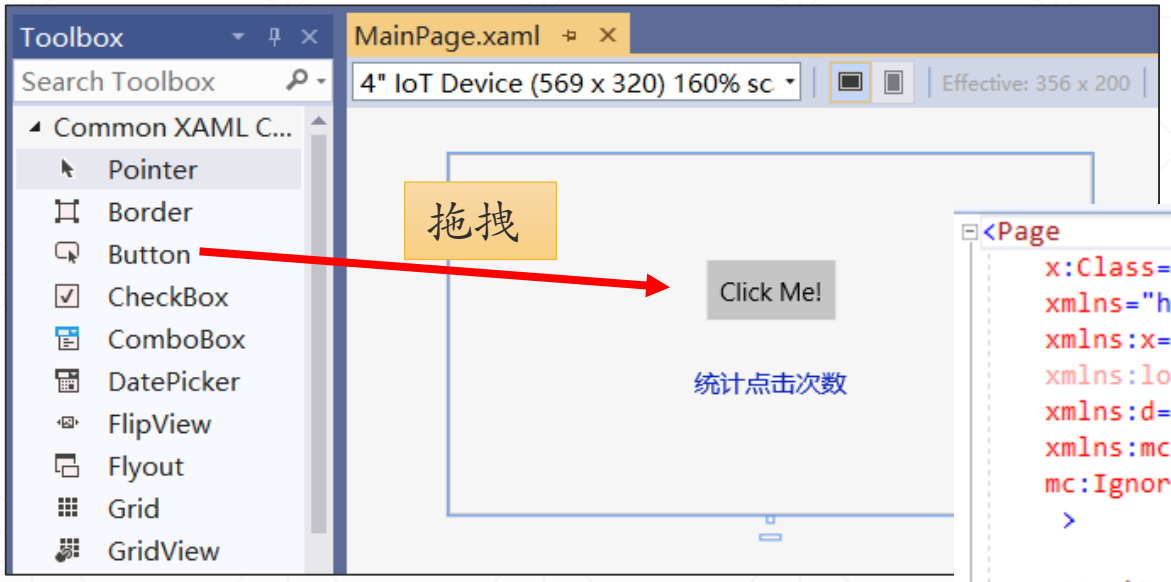

```
<Page
    x:Class="HelloWorld.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorld"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>

    </Grid>
</Page>
```

MainPage.xaml



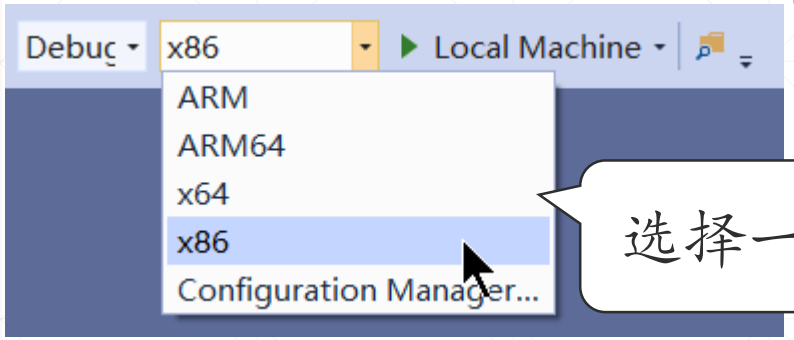
设计UI界面

每个Page通常包容有两个组成部分：Page之下是布局面板（Panel），XAML控件放在布局面板中



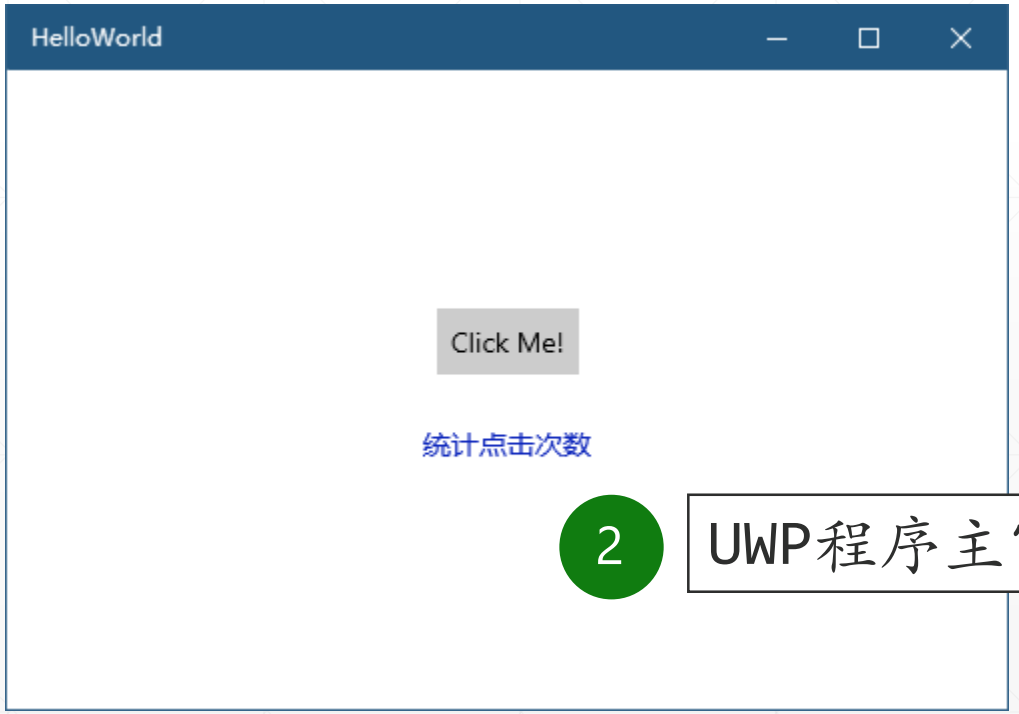

```
<Page
    x:Class="HelloWorld.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorld"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
     >

    <StackPanel Orientation="Vertical" VerticalAlignment="Center">

        <Button x:Name="btnClick" Content="Click Me!" Margin="10"
                Padding="5" HorizontalAlignment="Center" Click="BtnClick_Click"/>

        <TextBlock x:Name="tbInfo" Text="统计点击次数" TextWrapping="Wrap"
                   Margin="10"
                   Padding="5"
                   HorizontalAlignment="Stretch"
                   TextAlignment="Center" Foreground="#FF0D26BC"/>
    </StackPanel>

</Page>
```

运行程序



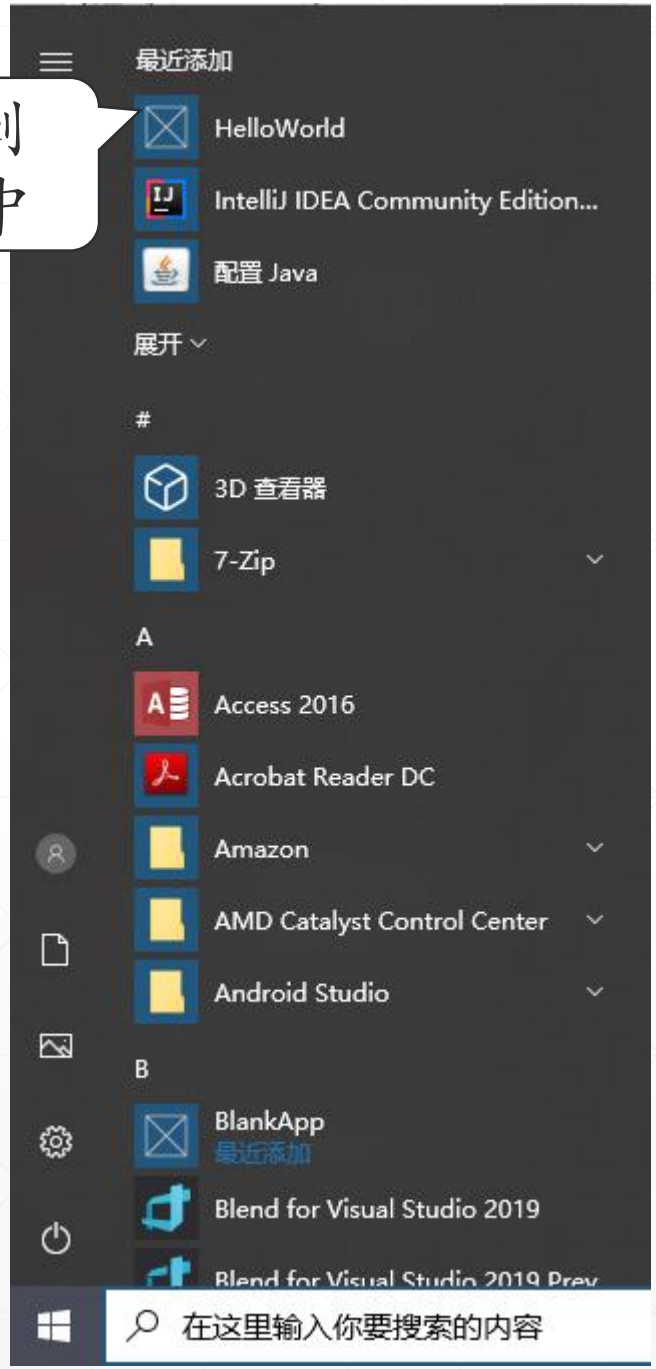
选择一种体系结构的CPU

1



2 UWP程序主窗体显示

3 UWP应用会被添加到Win10的开始菜单中



了解UWP应用UI编程模型

```
//UWP应用加载完毕时回调此方法
0 references
protected override void OnLaunched(LaunchActivatedEventArgs e)
{
    //Window.Current引用当前激活的UWP窗体对象
    //它承载一个Frame对象
    Frame rootFrame = Window.Current.Content as Frame;
    if (rootFrame == null)
    {
        //如果Frame对象没有创建，则实例化之
        rootFrame = new Frame();
        rootFrame.NavigationFailed += OnNavigationFailed;
        if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)...
        //设置UWP窗体显示这个Frame对象
        Window.Current.Content = rootFrame;
    }
    if (e.PrelaunchActivated == false)
    {
        if (rootFrame.Content == null)
        {
            //导航到主页面
            rootFrame.Navigate(typeof(MainPage), e.Arguments);
        }
        //激活当前窗体
        Window.Current.Activate();
    }
}
```

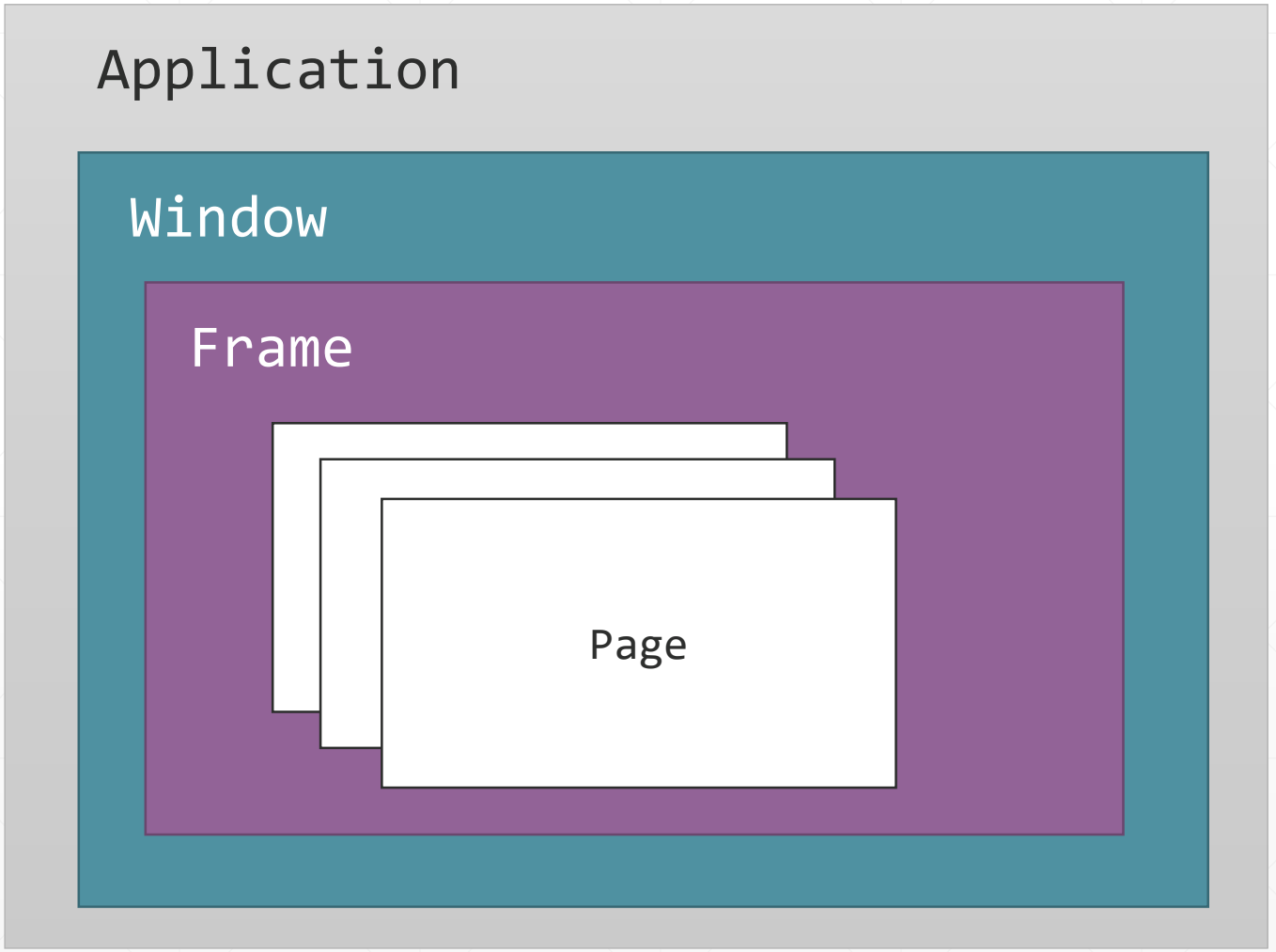
App.xaml.cs

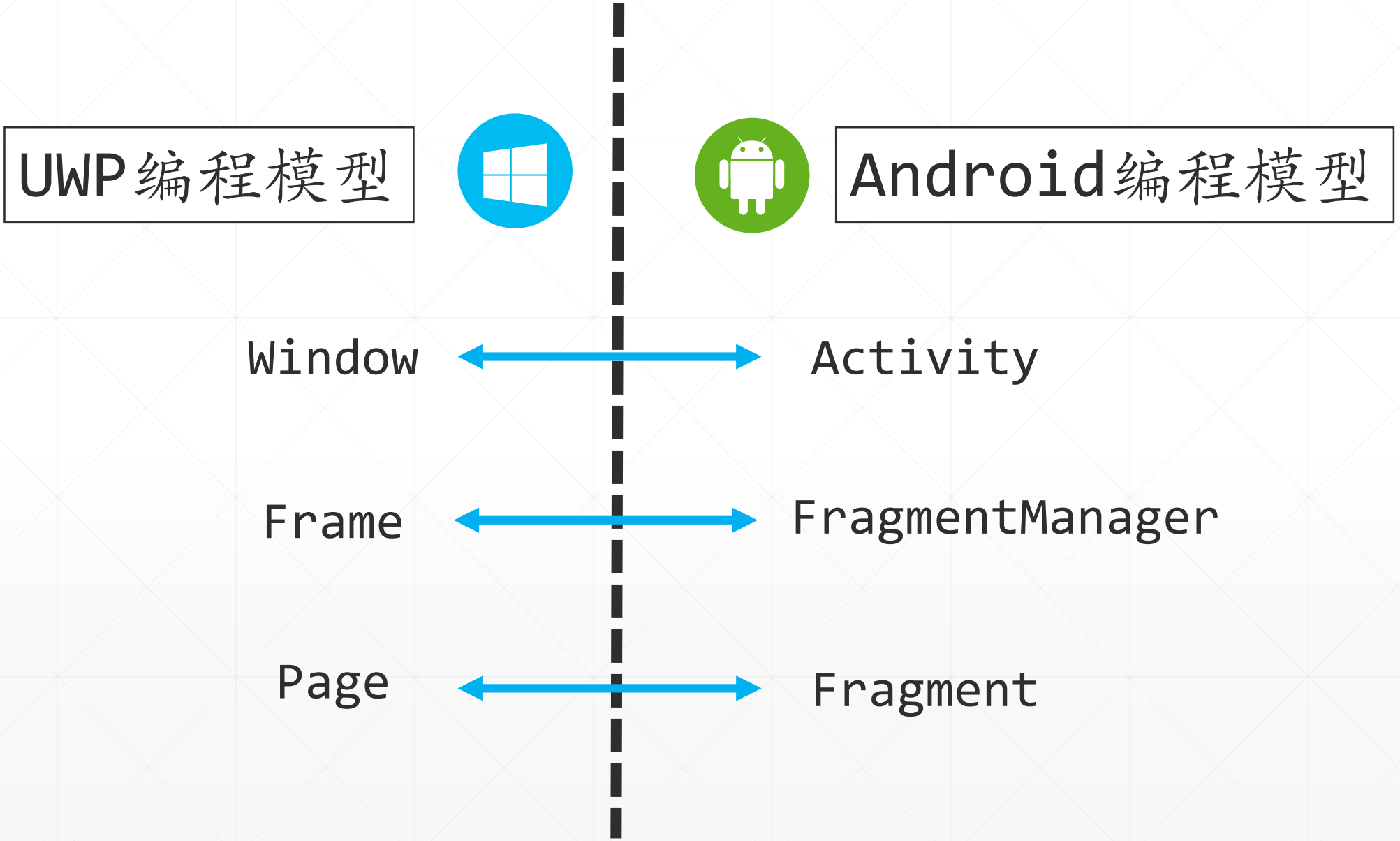
左图所示的代码为UWP项目模板自动生成的框架代码，可以看到，里面有几个核心的对象：Window，Frame和MainPage

UWP核心对象之间的关系

清单文件，包容一些元数据，这些数据所包容的信息会被应用本身及微软应用商店所使用。

Manifest




UWP编程模型
Android编程模型
Window
Activity
Frame
FragmentManager
Page
Fragment

添加事件响应

MainPage.xaml

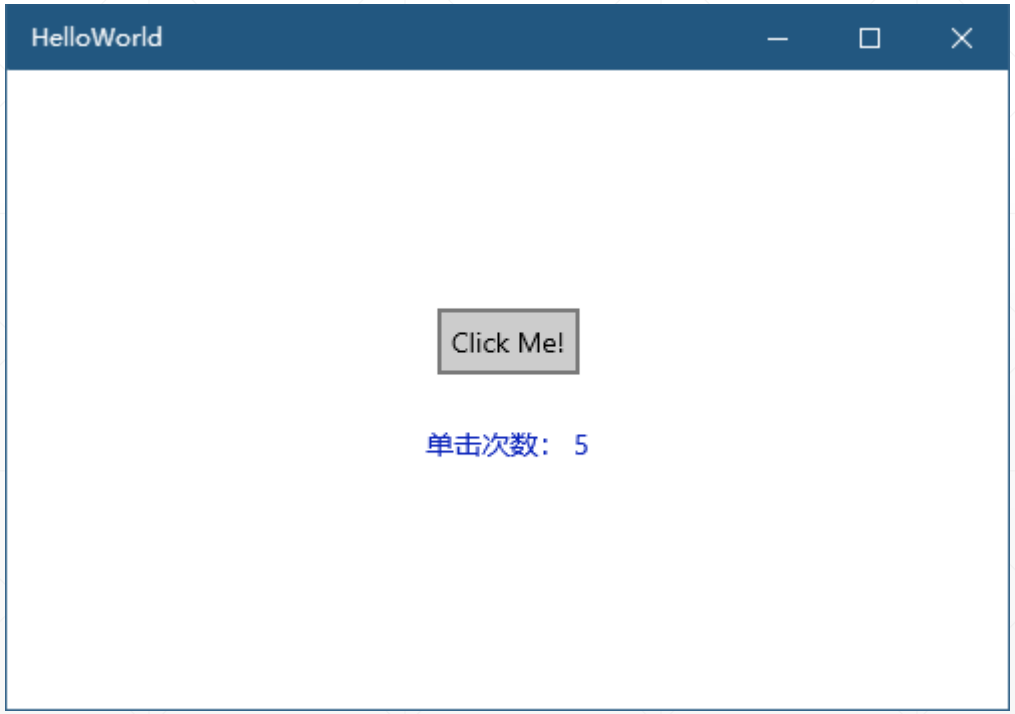
```
<Button x:Name="btnClick" Content="Click Me!" Margin="10"
        Padding="5" HorizontalAlignment="Center" Click="BtnClick_Click"/>
```

```
private int counter = 0;
0 references
private void BtnClick_Click(object sender, RoutedEventArgs e)
{
    counter++;
    tbInfo.Text = $"单击次数:  {counter}";
}
```

MainPage.xaml.cs

UWP的编程方式，与WPF高度一致。

运行结果

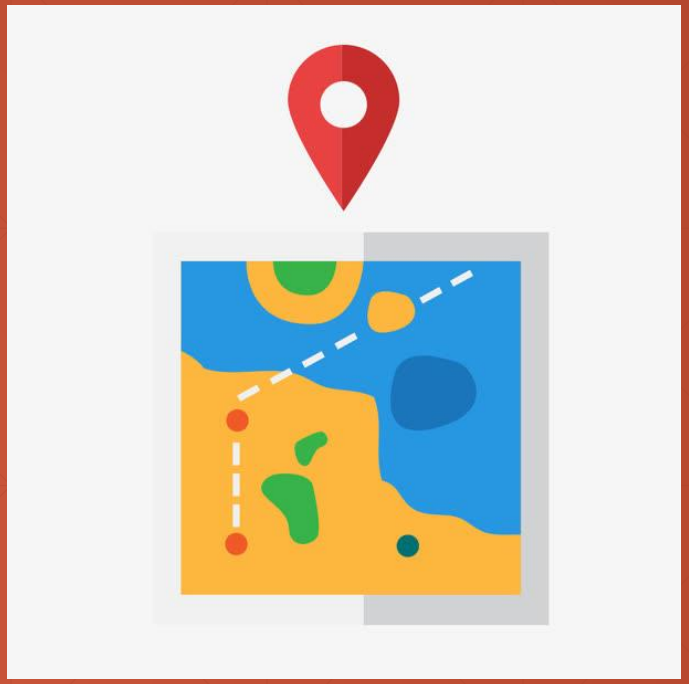


调用 Win10 API

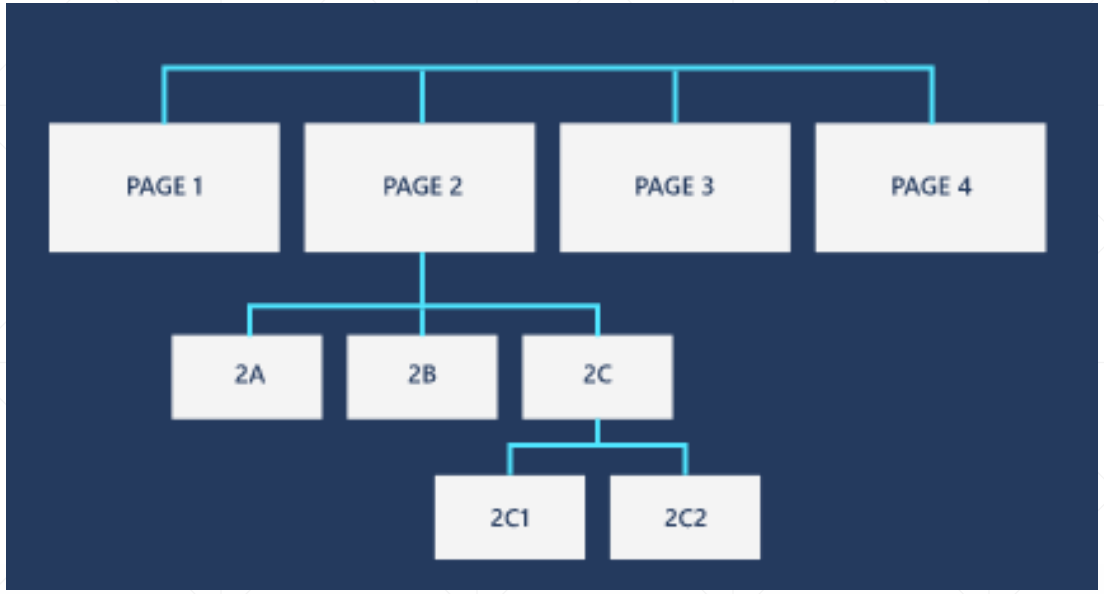
```
//调用Win10 API的语音合成引擎朗读字符串……
1 reference
private async Task Speak(string textToSpeak)
{
    MediaElement mediaElement = new MediaElement();
    var synth = new SpeechSynthesizer();
    SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync(textToSpeak);
    mediaElement.SetSource(stream, stream.ContentType);
    mediaElement.Play();
}
```

```
private int counter = 0;
//改造为异步事件响应方式
1 reference
private async void BtnClick_Click(object sender, RoutedEventArgs e)
{
    counter++;
    string message = $"单击次数:  {counter}";
    tbInfo.Text = message;
    await Speak(message);
}
```

现在单击按钮，可以听到Windows现在用声音来报数!



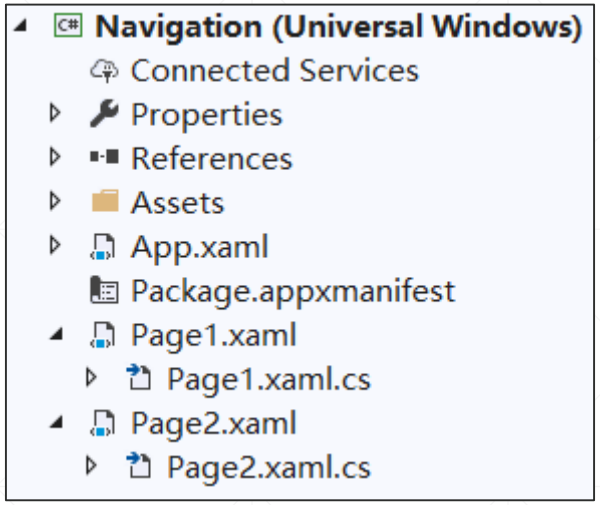
UWP页面间的导航

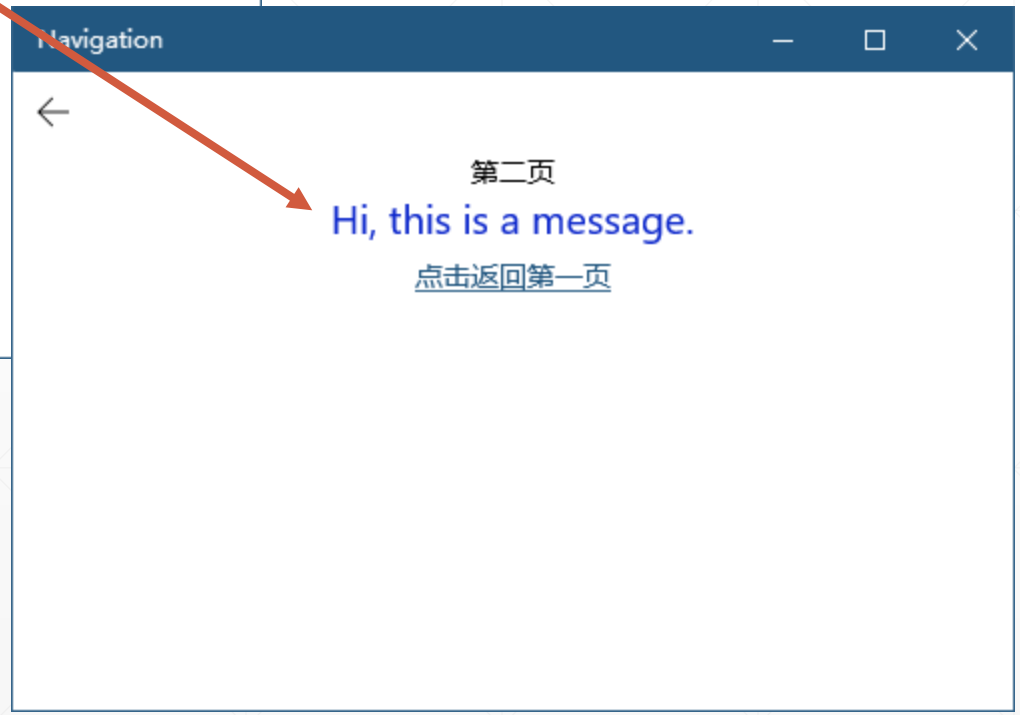


每个UWP App都可以看成是一组页面的集合，这些页面之间存在着关联，用户可以顺着这些关联关系“导航”到不同的页面。

在UWP应用中，页面间的导航是由Frame类负责的，下面通过一个实例展示UWP页面导航的基本实现方法。

示例：Navigation





Frame是实现导航的核心对象

App.xaml.cs

```
/// <summary> Invoked when the application is launched normally by th
protected override void OnLaunched(LaunchActivatedEventArgs e)
{
    Frame rootFrame = Window.Current.Content as Frame;

    // Do not repeat app initialization when the Window already has content,
    // just ensure that the window is active
    if (rootFrame == null)...

    if (e.PrelaunchActivated == false)
    {
        if (rootFrame.Content == null)
        {
            // When the navigation stack isn't restored naviga
            // configuring the new page by passing required ir
            // parameter
            rootFrame.Navigate(typeof(Page1), e.Arguments);
        }
        // Ensure the current window is active
        Window.Current.Activate();
    }
}
```

导航到Page1


```
public sealed partial class Page1 : Page
{
    public Page1()
    {
        this.InitializeComponent();
        //启用缓存
        this.NavigationCacheMode = Windows.UI.Xaml.Navigation.NavigationCacheMode.Enabled;
        //调整窗体大小
        ApplicationView.PreferredLaunchViewSize = new Size(300, 250);
        ApplicationView.PreferredLaunchWindowingMode =
            ApplicationViewWindowingMode.PreferredLaunchViewSize;
    }
    private void HyperlinkButton_Click(object sender, RoutedEventArgs e)
    {
        // this.Frame.Navigate(typeof(Page2));
        this.Frame.Navigate(typeof(Page2), name.Text);
    }
}
```

导航到Page2，注意其参数

Page1.xaml.cs

```
public sealed partial class Page2 : Page
{
    public Page2()...

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        if (this.Frame.CanGoBack)
        {
            this.Frame.GoBack();
        }
    }
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        if (e.Parameter is string && !string.IsNullOrWhiteSpace((string)e.Parameter))
        {
            greeting.Text = $"Hi, {e.Parameter.ToString()}";
        }
        else
        {
            greeting.Text = "Hi!";
        }
        base.OnNavigatedTo(e);
    }
    private void HyperlinkButton_Click(object sender, RoutedEventArgs e)
    {
        this.Frame.Navigate(typeof(Page1));
    }

}
```

Page2.xaml.cs

通过Navigate()方法的第2个参数，可以在页面之间传送信息。

小结

- 本讲通过两个示例介绍了UWP编程模型中的几个核心对象，理解它们的职责非常关键。
- 建议课后仔细阅读和运行示例源码，了解清楚示例中每条代码的含义，并查询官方文档，看看这些核心对象有哪些方法。
- 下一讲，将以本例中的HelloWorld为例，介绍如何部署UWP应用。