UNIDAD 6

XAML

¿Qué es XAML?

- Extensible Application Markup Language.
- Lenguaje de marcas basado en XML.
- Fácil de entender por el ser humano.
- Desarrollable bajo IDE´s como Visual Studio o como Expression Blend. Esto implica que podemos desarrollar por separado el código y la interface de usuario de una misma aplicación.

¿Qué hacemos con XAML?

- Podemos realizar las interfaces de usuario de:
  - WPF
  - SilverLight
  - UWP
  - Xamarin y Xamarin Forms
  - MAUI

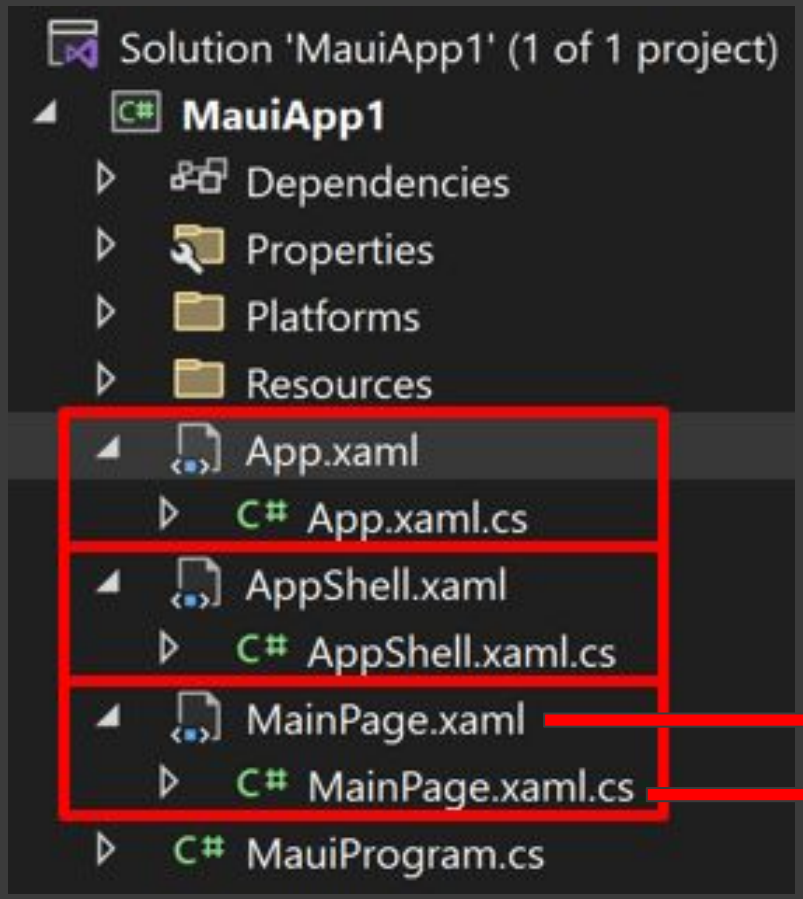
Ventajas de XAML

- Lenguaje de marcado: claro y legible. Mucho más que hacerlo mediante código.
- Jerarquía padre-hijo: claridad visual y potencia a la hora de crear la UI.
- Parecido a XML de Android.

Desventajas de XAML

- No admite código C# dentro.
- Todo el código C# va en el “Code Behind” del archivo o en el ViewModel.
- No puede contener ni bucles ni condicionales.

Ficheros XAML



Código XAML

Código C#
(Code-behind)

Sintaxis de XAML

- Su sintaxis comienza con “<“ seguido del nombre de la clase
- Todo elemento de un documento XAML pertenece a una clase. Por ejemplo el tag <Button> es de la clase Button.
- Se convertirá en objeto una vez compilemos

  Elementos = Objetos
- Se pueden anidar elementos unos dentro de otros, como cualquier otro documento XML.

Elementos raíz y espacios de nombres

- El código XAML de una página recién creado tiene este aspecto

```
<ContentPage
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="_01_HelloWorld_MAUI.MainPage">

    ……
</ContentPage>
```

Veamos cada una de las partes:

Elementos raíz

<ContentPage

……

- En este caso el elemento raíz es ContentPage, pero los posibles valores son:
  - Application
  - Shell
  - ContentPage
  - FlayoutPage
  - NavigationPage
  - TabbedPage
  - ResourceDictionary

Espacio de nombres

- XAML usa el atributo XML “xmlns” para las declaraciones de espacio de nombres

```
xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
```

- Espacio de nombres predeterminado para MAUI.
- Los elementos definidos dentro del XAML (como ContentPage, Label, etc.) hacen referencia a clases de .NET MAUI.
- Si en futuro alguna empresa saca unos elementos propios (como un label modificado) deberemos cambiar ese espacio de nombres.

Espacio de nombres

xmlns:x=http://schemas.microsoft.com/winfx/2009/xaml

- :x indica que los elementos propios de XAML ,definidos con un x: delante (x:Name, x:Key, etc.), buscarán en este espacio de nombres.
- En concreto en la especificación XAML de 2009.

Espacio de nombres

xmlns:x=http://schemas.microsoft.com/winfx/2009/xaml

- Es el espacio de nombres que contiene las clases que ofrece Microsoft para ser usadas en XAML.
- Si adquirimos un componente de un tercero en el que se define una clase ContentPage personalizada, deberemos cambiar el espacio de nombres.

Propiedades

- Se pueden asignar propiedades de cada clase a través de atributos:

```
<Button  x:Name="btnAceptar"/>
```

- La propiedad "Name" no es obligatoria, pero si no la lleva, ese elemento no será accesible desde el código C#.
- Algunas veces, si los atributos se complican o resultan difíciles de leer por ser muy largos, podemos usar otra sintaxis.

Sintaxis 1

```
<Button Name="btnAceptar"  Content="Hola mundo"/>
```

Equivale a:

Sintaxis 2

```
<Button>
    <Button.Name>btnAceptar</Button.Name>
    <Button.Content>Hola mundo</Button.Content>
</Button>
```

Propiedades

- En XAML cuando una propiedad es demasiado compleja como para expresarla en una simple cadena, podemos expresarla así.

```
<Button
    Text="Click me"
    HorizontalOptions="Center" >
    <Button.Shadow>
        <Shadow Brush="Black"
                Offset="15,5"
                Radius="20"
                Opacity="0.8" />
    </Button.Shadow>
    </Button>
```

Propiedad SIMPLE

Propiedad COMPLEJA

Propiedades

XAML

```
<Border Stroke="#C49B33"
        StrokeThickness="4"
        Background="#2B0B98"
        Padding="16,8"
        HorizontalOptions="Center">
    <Border.StrokeShape>
        <RoundRectangle CornerRadius="40,0,0,40" />
    </Border.StrokeShape>
    <Label Text=".NET MAUI"
           TextColor="White"
           FontSize="18"
           FontAttributes="Bold" />
</Border>
```

C#

```
Border border = new Border
{
    Stroke = Color.FromArgb("#C49B33"),
    Background = Color.FromArgb("#2B0B98"),
    StrokeThickness = 4,
    Padding = new Thickness(16, 8),
    HorizontalOptions = LayoutOptions.Center,
    StrokeShape = new RoundRectangle
    {
        CornerRadius = new CornerRadius(40, 0, 0, 40)
    },
    Content = new Label
    {
        Text = ".NET MAUI",
        TextColor = Colors.White,
        FontSize = 18,
        FontAttributes = FontAttributes.Bold
    }
};
```

Resultado de ambos códigos:

Propiedades

- Attached Properties o propiedades adjuntas son aquellas que pertenecen y son definidas por un tipo pero que pueden ser asignadas a cualquier elemento hijo.
- Estos elementos hijos no tienen esa propiedad por defecto, si no que las poseen porque las “heredan” de su padre.

```
<AbsoluteLayout Margin="20">
    <BoxView Color="Silver"
        AbsoluteLayout.LayoutBounds="0, 10, 200, 5" />
    <BoxView Color="Silver"
        AbsoluteLayout.LayoutBounds="0, 20, 200, 5" />
    <BoxView Color="Silver"
        AbsoluteLayout.LayoutBounds="10, 0, 5, 65" />
    <BoxView Color="Silver"
        AbsoluteLayout.LayoutBounds="20, 0, 5, 65" />
    <Label Text="Stylish Header"
        FontSize="24"
        AbsoluteLayout.LayoutBounds="30, 25" />
</AbsoluteLayout>
```