

27. Tipos y propiedades en .NET MAUI XAML

Extensible Application Markup Language

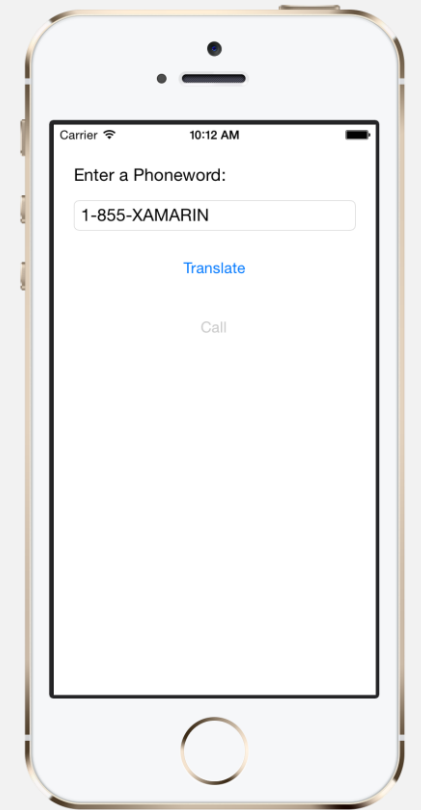
XAML fue creado por Microsoft específicamente para describir la UI.



Definiendo una pantalla en XAML

XAML se usa para construir el árbol de objetos visuales, en este caso una página visual.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```



Basado en XML: distingue entre mayúsculas y minúsculas, las etiquetas abiertas deben cerrarse, etc.

Definiendo una pantalla en XAML

XAML se usa para construir el árbol de objetos visuales, en este caso una página de contenido.

Las etiquetas
crean
elementos
visuales

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phonenumber" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```

Definiendo una pantalla en XAML

XAML se usa para construir el árbol de objetos visuales, en este caso una página de contenido.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phonenumber:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```

Los atributos establecen propiedades o eventos

Definiendo una pantalla en XAML

XAML se usa para construir el árbol de objetos visuales, en este caso una página de contenido.

Nodos
secundarios
establecen
relaciones

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContentPage ...>
    <StackLayout Padding="20" Spacing="10">
        <Label Text="Enter a Phoneword:" />
        <Entry Placeholder="Number" />
        <Button Text="Translate" />
        <Button Text="Call" IsEnabled="False" />
    </StackLayout>
</ContentPage>
```

XAML + Code Behind

El archivo XAML y el code-behind están vinculados.

```
<?xml version="1.0" encoding="UTF-8" ?>  
<ContentPage x:Class="Phoneword.MainPage" ...>
```

```
namespace Phoneword  
{  
    public partial class MainPage : ContentPage  
    {  
        ...  
    }  
}
```

x:Class Identifica el nombre completo de la clase definida en el código subyacente.

Inicialización de XAML

El código detrás del constructor tiene una llamada a **InitializeComponent**, que es responsable de cargar el XAML y crear los objetos.

```
public partial class MainPage : ContentPage
{
    public MainPage ()
    {
        InitializeComponent ();
    }
}
```

La implementación del método generado por el compilador XAML como resultado de la etiqueta x:Class – agregado al archivo oculto (misma clase parcial)

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" isVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```

Text es una cadena que se establece directamente

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" isVisible="True" Opacity="0.75"  
    FontAttributes="Bold,Italic" FontSize="Large"  
    Margin="5,20,5,0" TextColor="#fffc0d34" />
```

isVisible es un booleano que se
convierte a partir del valor usando
Boolean.TryParse

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" IsVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```

La opacidad es un doble que se convierte a partir del valor usando **Double.TryParse**

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" IsVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```

Las enumeraciones se analizan con **Enum.TryParse** y admiten [Flags] con valores separados por comas

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" IsVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```

```
[TypeConverter(typeof(ThicknessTypeConverter))]  
public struct Thickness  
{  
    ...  
}
```

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos


```
<Label Text="This is a Label" IsVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```

Margin es un objeto de **Thickness**, puede especificarlo como un solo número, dos números o cuatro números (L,T,R,B)

Propiedades

Los atributos XML solo permiten valores de **cadena**; funciona bien para tipos intrínsecos

```
<Label Text="This is a Label" IsVisible="True" Opacity="0.75"  
      FontAttributes="Bold,Italic" FontSize="Large"  
      Margin="5,20,5,0" TextColor="#fffc0d34" />
```



Los colores se pueden especificar como un valor conocido (por ejemplo, "Rojo", "Verde",...) o como un valor hexadecimal (RGB o aRGB)

Propiedades Adjuntas

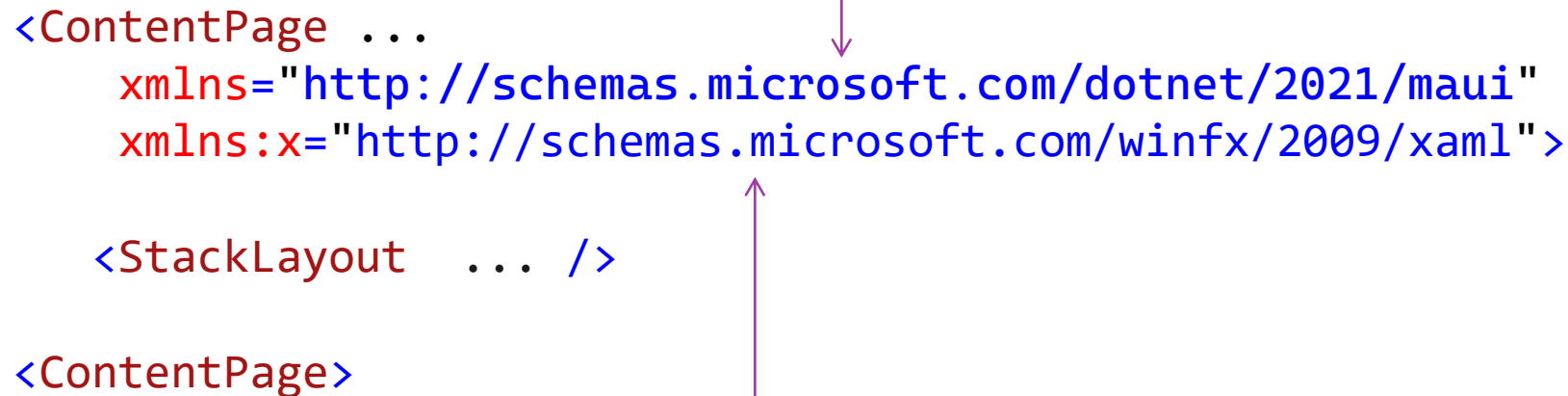
- ❖ Las propiedades adjuntas proporcionan datos "adjuntos" en tiempo de ejecución para un elemento visual
- ❖ Utilizados por Layouts para proporcionar valores específicos del contenedor en cada elemento secundario

```
<Grid>  
  <Label Text="Position" />  
  <Entry Grid.Column="1" />  
</Grid>
```


Identificando tipos

XAML crea objetos cuando encuentra una etiqueta de elemento, los espacios de nombres XML se utilizan para correlacionar los tipos de .NET con las etiquetas.

El espacio de nombres predeterminado incluye la mayoría de los tipos que .NET MAUI que usa




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

x: namespace incluye tipos XAML y tipos CLR conocidos (Int32, String, etc.)

Tipos personalizados

XAML puede crear cualquier objeto público, incluidos los que tienen constructores parametrizados; solo necesita decirle dónde "vive" el tipo.

Debe proporcionar el espacio de nombres y, posiblemente, el ensamblado donde se define el tipo.



```
<scg:List x:TypeArguments="x:String"
  xmlns:scg="clr-namespace:System.Collections.Generic;assembly=mscorlib">
  <x:String>One</x:String>
  <x:String>Two</x:String>
  <x:String>Three</x:String>
</scg:List>
```

La definición **xmlns** se puede colocar en un solo elemento, o en un elemento principal para usar con cualquier elemento secundario

Nombrando elementos en XAML

- ❖ Use x: Name para asignar el nombre del elemento que le permite hacer referencia al elemento en XAML y al código subyacente.
- ❖ Agrega un campo privado a la clase parcial generada por XAML (.g.cs)
- ❖ El nombre debe cumplir con las convenciones de nomenclatura de C# y ser único en el archivo

MainPage.xaml

```
<Entry x:Name="PhoneNumber"  
        Placeholder="Number" />
```

```
public partial class MainPage : ContentPage  
{  
    private Entry PhoneNumber;  
  
    private void InitializeComponent() {  
        this.LoadFromXaml(typeof(MainPage));  
        PhoneNumber = this.FindByName<Entry>(  
            "PhoneNumber");  
    }  
}
```

MainPage.xaml.g.cs

Trabajando con elementos nombrados

Puede trabajar con elementos con nombre como si los hubiera definido en el código, pero tenga en cuenta que el campo no se establece hasta que se llama a **InitializeComponent**

Puede
conectar
eventos,
establecer
propiedades e
incluso agregar
nuevos
elementos al
diseño

```
public partial class MainPage : ContentPage
{
    public MainPage () {
        InitializeComponent ();
        PhoneNumber.TextChanged += OnTextChanged;
    }

    void OnTextChanged(object sender, TextChangedEventArgs e) {
        ...
    }
}
```