Tipos e propriedades de XAML no .NET MAUI

5 minutos

O XAML é uma linguagem de marcação declarativa. Foi projetado para simplificar o processo de criação de sua interface do usuário. Os elementos em XAML representam diretamente a instanciação de objetos que você acessa nos arquivos code-behind.

Nesta unidade, você aprenderá a usar os tipos disponíveis em XAML e como definir e ler propriedades desses tipos.

Onde os tipos são definidos?

O .NET MAUI implementa um analisador XAML que verifica os elementos XAML declarados e instancia cada elemento como um tipo .NET. O dialeto XAML que o analisador do .NET MAUI entende é específico do .NET MAUI, embora seja semelhante ao XAML usado por outras estruturas, como o Windows Presentation Foundation.

Os tipos .NET que implementam os itens que o código XAML identifica são implementados pelo código em vários assemblies .NET. Muitos desses assemblies estão incluídos como parte dos modelos do .NET MAUI. Você também pode utilizar outros tipos personalizados carregando os assemblies apropriados como parte do seu projeto. Muitos assemblies estão disponíveis como pacotes NuGet. A maioria dos tipos comuns usados por um aplicativo MAUI está nos pacotes **Microsoft.Maui.Dependencies** e **Microsoft.Maui.Extensions**.

Cada tipo é definido em um namespace. Em seu código XAML, você deve especificar os namespaces para os tipos aos quais fez referência. A maioria dos controles MAUI está localizada no namespace **Microsoft.Maui.Controls**, enquanto o namespace **Microsoft.Maui** define tipos de utilitários como `Thickness`, e o namespace **Microsoft.Maui.Graphics** inclui tipos generalizados como `Color`. A opção de apresentar tipos dessa maneira destaca a extensibilidade do XAML. O XAML permite que você crie a interface do usuário do seu aplicativo com a liberdade de incluir elementos de XAML no .NET MAUI, tipos .NET e tipos personalizados. Na maioria dos casos, não é necessário se preocupar com esses namespaces, pois eles são trazidos usando o recurso `usings` implícito do C# que os adiciona automaticamente em todo o aplicativo.

Como instanciar tipos em XAML

A primeira etapa no uso de XAML para criar uma interface do usuário é instanciar os tipos de controle de interface do usuário. Em XAML, você pode criar um objeto de um tipo especificado usando a sintaxe de elemento de objeto. A sintaxe de elemento de objeto é uma sintaxe XML padrão e bem formada para declarar um elemento. Por exemplo, se você quiser criar um rótulo com uma cor específica, seu elemento XAML será semelhante ao seguinte código:

XML

```xml
<Label TextColor="AntiqueWhite"/>
```

O analisador XAML do .NET MAUI analisará esse elemento XAML para criar uma instância do objeto na memória. Com efeito, o rótulo XAML analisado é o mesmo que o seguinte código C#:

C#

```csharp
var myLabel = new Label
{
  TextColor = Color.FromRgb(255, 255, 100)
};
```

O que é um namespace XAML?

Lembre-se de que, para o analisador XAML verificar com êxito a definição XAML de um controle em uma página, ele deve ter acesso ao código que implementa o controle e define suas propriedades. Os controles disponíveis para uma página .NET MAUI são implementados em uma coleção de assemblies que são instalados como parte do pacote NuGet **Microsoft.Maui**. Os controles estão localizados em um namespace .NET nesses assemblies. No código C#, você coloca um namespace no escopo com a diretiva `using`. Em uma página XAML, você faz referência a um namespace com o atributo `xmlns` da página. O código a seguir mostra os namespaces que a página XAML criada na unidade anterior usa:

XML

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

    ...
</ContentPage>
```

O primeiro namespace, `http://schemas.microsoft.com/dotnet/2021/maui`, é o namespace padrão da página. Essa forma de URI do namespace é típica de XML e é um pouco diferente daquelas com as quais você pode estar familiarizado em C#. No entanto, esse URI é simplesmente um alias para um ou mais namespaces definidos pelos assemblies no pacote NuGet **Microsoft.Maui**, portanto, especificar esse namespace no início da página traz todos os tipos e controles do .NET MAUI para o escopo. Se você omitir esse namespace, não poderá usar controles como `Button`, `Label` `Entry` ou `StackLayout`.

O segundo namespace, `http://schemas.microsoft.com/winfx/2009/xaml`, faz referência aos assemblies que contêm os vários tipos intrínsecos do .NET, como cadeias de caracteres, números e propriedades. No código XAML anterior, esse namespace recebe o alias **x**. No código XAML dessa página, você faz referência aos tipos nesse namespace prefixando-os com **x:**. Por exemplo, cada página XAML é compilada em uma classe, e você especifica o nome da classe que é gerada com o atributo **x:Class** da página:

XML

```
<ContentPage ...
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiXaml.Page1"
            ...>

    ...
</ContentPage>
```

É possível fazer referência a tipos em seus próprios assemblies no código XAML por meio de um namespace XAML. Por exemplo, se você tiver tipos e métodos que quer usar em seu código XAML definidos em um namespace chamado **Utils** em seu projeto, poderá adicionar o namespace **Utils** à página, conforme mostrado no código a seguir. Neste exemplo, você acessa os tipos nesse namespace prefixando-os com o alias **mycode**.

XML

```
<ContentPage ...
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:mycode="clr-namespace:Utils"
            ...>

    ...
</ContentPage>
```

> ⓘ **Observação**
>
> Você verá mais exemplos dessa técnica mais adiante neste módulo.

Como especificar valores de propriedade em XAML

Em XML, você usa atributos para descrever ou fornecer informações sobre um elemento. Em XAML, você usa atributos para definir propriedades no tipo subjacente. Por exemplo, considerando o seguinte código C#:

C#

```
var label = new Label { Text = "Username", TextColor = Color.Black };
```

Essa instrução cria um novo objeto de `Label` e define as propriedades `Text` e `TextColor`. Para definir propriedades em XAML, usa-se atributos. O código XAML correspondente tem a seguinte aparência:

XML

```
<Label Text="Username" TextColor="Black" />
```

Algo que você pode notar que é diferente no código XAML em relação ao código C# são os valores das propriedades. Por exemplo, no código C#, usa-se o tipo `Color` para a propriedade `TextColor`. No entanto, na definição de XAML, define-se `TextColor` com um valor de cadeia de caracteres. Isso ocorre porque uma cadeia de caracteres é o único elemento válido que você pode usar para um valor de atributo XML. Portanto, deve haver uma maneira de converter cada valor de cadeia de caracteres para seu tipo correto. No XAML, você pode fazer essa conversão usando um **Conversor de Tipo**.

O que é um conversor de tipo?

Um conversor de tipo converte um atributo XML especificado como um valor de cadeia de caracteres em seu tipo correto. Para entender melhor esse conceito, observe o seguinte exemplo:

XML

```
<Label Text="Username" TextColor="Black" FontSize="42"
FontAttributes="Bold,Italic" />
```

Esse código cria um `Label` e define suas propriedades `Text`, `TextColor`, `FontSize` e `FontAttributes`.

Vamos começar com a primeira propriedade, `Text`. O texto já é uma cadeia de caracteres, o que significa que a página XAML não precisa de um conversor de tipo. Em seguida, `TextColor` usa o tipo `Color`, portanto, o XAML requer um conversor de tipo para traduzir a cadeia de caracteres para o valor correspondente `Color`. A propriedade `FontSize` é um número inteiro, portanto, o XAML requer um conversor de tipo para analisar a cadeia de caracteres em um número inteiro. Por fim, `FontAttributes` é um exemplo de um objeto complexo. Você pode combinar os valores como uma cadeia de caracteres delimitada por vírgulas: "Negrito,Itálico". A cadeia de caracteres delimitada por vírgulas é tratada como uma enumeração baseada em [Flags] e o conversor de tipo apropriado aplicará o `OR` bit a bit do valor à propriedade.

O .NET MAUI tem conversores de tipo para a maioria das classes internas e usa esses conversores de tipo automaticamente. No entanto, se um conversor específico não existir, escreva seu próprio e associe-o a seu tipo para torná-lo utilizável no XAML.

Atribuição de tipo complexo

Conversores de tipo são ótimos para as configurações de propriedade simples; no entanto, em alguns casos, você precisa criar um objeto completo com seus próprios valores de propriedade. A solução para esse problema é alterar a atribuição de propriedade para usar uma sintaxe baseada em elemento. Essa sintaxe é chamada de formulário de **Elemento de Propriedade**. Essa sintaxe envolve dividir o configurador de propriedade no formulário pai-filho, em que a propriedade é expressa em uma marca de elemento do formato **Type.PropertyName**. Suponha que você queira atribuir um reconhecedor de gestos a um rótulo para que o usuário do aplicativo possa tocar no rótulo. O reconhecedor de gestos é um objeto complexo com suas próprias propriedades. Normalmente, essas propriedades precisam ser atribuídas para garantir a funcionalidade correta:

XML

```xml
<TapGestureRecognizer NumberOfTapsRequired="2" />
```

Se você precisar atribuir esse valor a um `Label`, poderá escrever o XAML assim:

XML

```xml
<Label Text="Username" TextColor="Black" FontSize="42"
FontAttributes="Bold,Italic">
    <Label.GestureRecognizers>
        <TapGestureRecognizer NumberOfTapsRequired="2" />
    </Label.GestureRecognizers>
</Label>
```

O tipo `Label` tem uma propriedade chamada `GestureRecognizers`. Usando o formulário **Elemento de propriedade**, você pode adicionar um `TapGestureRecognizer` à lista de gestos ao arquivo `Label`.

Propriedade de conteúdo padrão

Alguns controles .NET MAUI têm uma propriedade de conteúdo padrão. A propriedade de conteúdo permite que você especifique o valor de uma propriedade em um controle sem explicitamente declará-la no XAML. Dê uma olhada no seguinte snippet de XAML:

XML

```xml
<VerticalStackLayout>
    <VerticalStackLayout.Children>
        <Label Text="Please log in" />
    </VerticalStackLayout.Children>
</VerticalStackLayout>
```

Esse código cria um `VerticalStackLayout` e adiciona um `Label` como elemento filho. Como é comum adicionar filhos a um `VerticalStackLayout`, sua propriedade `Children` é a de conteúdo padrão. Isso significa que você pode adicionar um elemento filho sem especificar explicitamente o `Children`, da seguinte maneira:

XML

```xml
<VerticalStackLayout>
    <Label Text="Please log in" />
</VerticalStackLayout>
```

Verificação de conhecimentos

1. O que um Conversor de Tipo faz? *

- ○ Um Conversor de Tipo traduz a descrição de XAML de uma página para o código C# equivalente.
- ○ Um Conversor de Tipo pode traduzir o código C# na representação de XAML de uma página.
- ○ Um conversor de tipo é usado para traduzir um valor de atributo XML especificado como uma cadeia de caracteres para seu tipo correto.

✔ Esta é a resposta correta.

2. Como se traz os próprios tipos personalizados no escopo de um arquivo XAML? *

- (○) Usa o atributo `xmlns` da página e dá ao namespace que contém seus tipos personalizados um alias que pode ser referenciado a partir dos controles XAML na página.

✔ Esta é a resposta correta.

- ( ) Adiciona uma diretiva C# `using` ao arquivo XAML.
- ( ) Aplica o atributo de namespace `x:FieldModifier` a elementos XAML nomeados.

Unidade seguinte: Tratamento de eventos no XAML

Continuar >