Benefícios de usar o XAML

5 minutos

O XAML é uma linguagem de marcação que você pode usar em vez do código C# para criar sua interface do usuário. Usando XAML, você pode dividir seu código de interface do usuário e comportamento para tornar ambos mais fáceis de gerenciar.

Nesta unidade, você vai comparar o uso de XAML com a definição do layout da interface do usuário usando código C#. Você aprenderá sobre alguns dos benefícios de usar XAML como uma linguagem de marcação para definir a interface do usuário.

O que é uma linguagem de marcação?

Uma linguagem de marcação é uma linguagem de computador que você pode usar para introduzir vários elementos em um documento. Você descreve os elementos usando marcas predefinidas. As marcas têm significados específicos no contexto do domínio em que o documento é usado.

Por exemplo, você pode usar a linguagem HTML para criar uma página da Web que você pode exibir em um navegador da Web. Você não precisa entender todas as marcas usadas no exemplo a seguir. O importante é ver que esse código descreve um documento que tem o texto "Olá, Mundo!" como conteúdo.

HTML

```
<!DOCTYPE html>
<html>
    <body>
        <p>Hello <b>World</b>!</p>
    </body>
</html>
```

Você provavelmente já trabalhou com uma linguagem de marcação. Talvez você tenha criado uma página da Web usando HTML ou talvez tenha modificado as definições de Extensible Markup Language (XML) em um arquivo project.csproj do Visual Studio. As ferramentas de compilação da Microsoft analisam e processam esse arquivo.

É comum que arquivos que contêm linguagem de marcação sejam processados e interpretados por outras ferramentas de software. Essa natureza interpretativa da marcação é

exatamente como XAML deve funcionar. No entanto, as ferramentas de software que o interpretam ajudam a gerar a interface do usuário do aplicativo.

O que é XAML?

XAML é uma linguagem de marcação declarativa criada pela Microsoft. XAML foi projetado para simplificar o processo de criação da interface do usuário em aplicativos.

Os documentos XAML criados contêm elementos que descrevem declarativamente os elementos de interface do usuário do aplicativo. Lembre-se de que esses elementos no XAML representam diretamente a instanciação de objetos. Uma vez definido um elemento em XAML, você pode acessá-lo em arquivos code-behind e definir o comportamento utilizando código C#.

Diferença entre XAML .NET MAUI XAML e XAML Microsoft

O XAML é baseado na especificação XAML de 2009 da Microsoft. Entretanto, essa especificação define apenas a sintaxe da linguagem. Assim como acontece com o WPF (Windows Presentation Foundation), a UWP (Plataforma Universal do Windows) e o WinUI 3, que usam XAML, os elementos que você declara no XAML serão alterados.

XAML apareceu pela primeira vez em 2006, com o WPF. Se você já trabalha com XAML da Microsoft há algum tempo, a sintaxe do XAML deve ser familiar.

Há algumas diferenças importantes entre a variante do .NET MAUI do XAML e o XAML usado por outras ferramentas de interface do usuário. A estrutura e os conceitos são semelhantes, mas alguns dos nomes das classes e propriedades são diferentes.

Criar uma interface do usuário usando XAML .NET MAUI

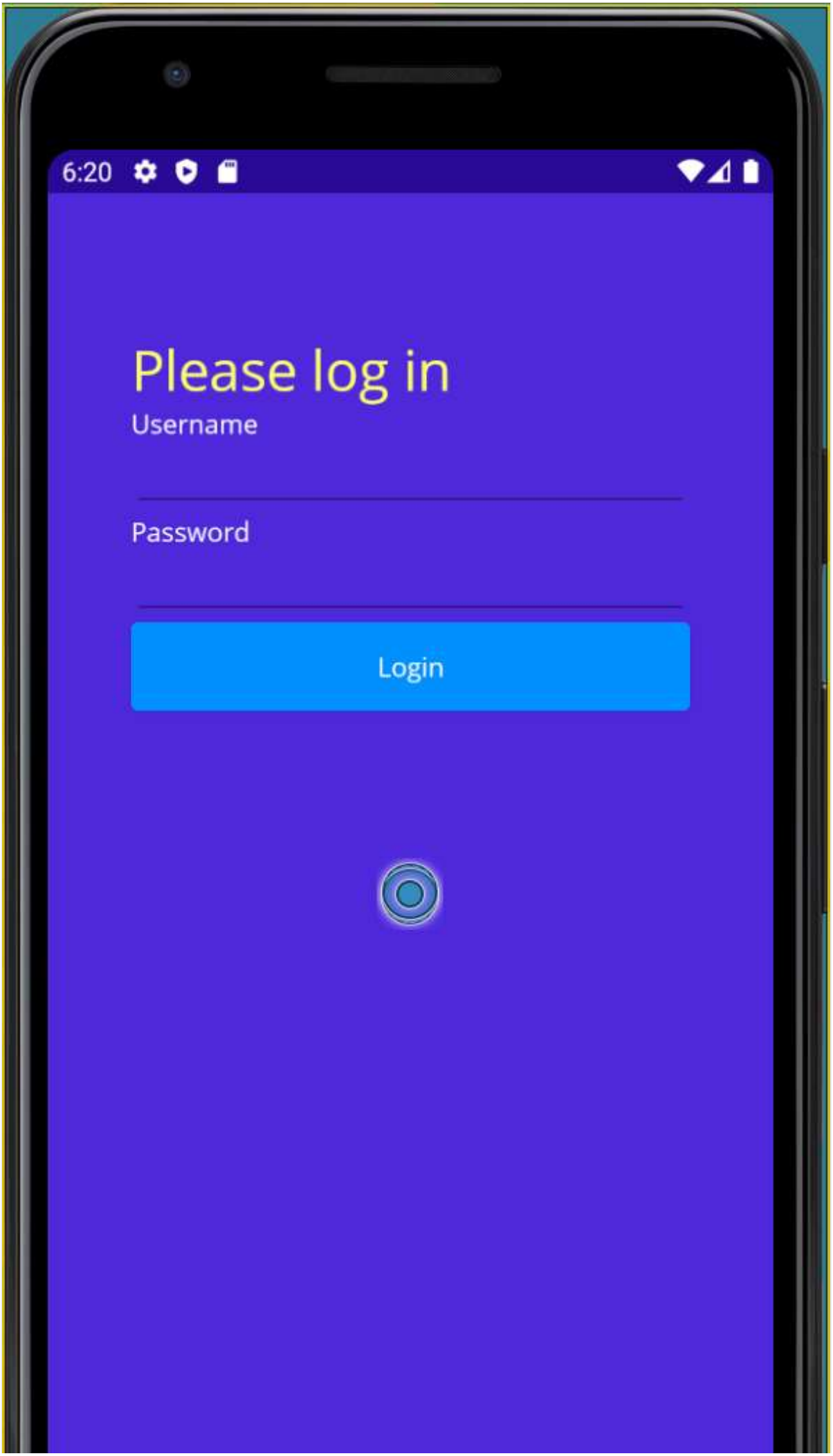
A melhor maneira de ver o XAML em ação é examinar um exemplo de um tipo de página ContentPage codificada em C# existente. Você pode compará-la com outra página que tem a mesma interface do usuário definida usando XAML.

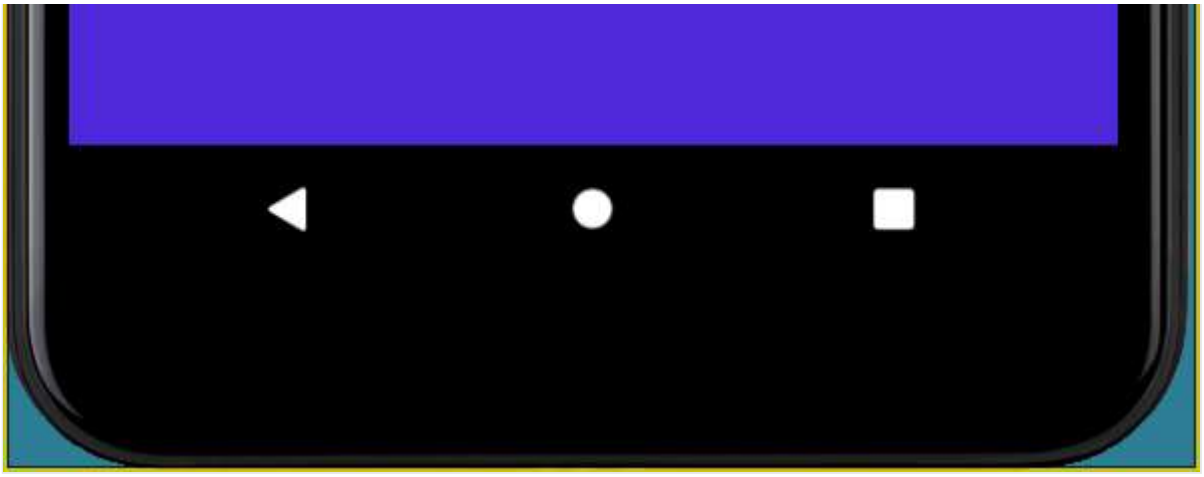
Suponha que você tenha a seguinte `ContentPage` codificada em seu aplicativo:

CSharp

```
namespace MauiCode;

public partial class MainPage : ContentPage
{
    Button loginButton;
    VerticalStackLayout layout;

    public MainPage()
    {
        this.BackgroundColor = Color.FromArgb("512bdf");

        layout = new VerticalStackLayout
        {
            Margin = new Thickness(15, 15, 15, 15),
            Padding = new Thickness(30, 60, 30, 30),
            Children =
            {
                new Label { Text = "Please log in", FontSize = 30, TextColor =
Color.FromRgb(255, 255, 100) },
                new Label { Text = "Username", TextColor = Color.FromRgb(255, 255,
255) },
                new Entry (),
                new Label { Text = "Password", TextColor = Color.FromRgb(255, 255,
255) },
                new Entry { IsPassword = true }
            }
        };

        loginButton = new Button { Text = "Login", BackgroundColor =
Color.FromRgb(0, 148, 255) };
        layout.Children.Add(loginButton);

        Content = layout;

        loginButton.Clicked += (sender, e) =>
        {
            Debug.WriteLine("Clicked !");
        };
    }
}
```

A página tem um contêiner de layout, dois rótulos, duas entradas e um botão. O código também manipula o evento Clicked para o botão. Também há apenas algumas propriedades de design definidas nos elementos na página. No runtime, em um dispositivo Android, a página tem esta aparência:


6:20
Please log in
Username
Password
Login



Embora a página tenha um design simples, há uma combinação de comportamento e design no mesmo arquivo.

O mesmo layout de página definido usando XAML tem esta aparência:

XML

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiXaml.Page1"
             BackgroundColor="#512bdf">

    <VerticalStackLayout Margin="15" Padding="30, 60, 30, 30">
        <Label Text="Please log in" FontSize="30" TextColor="AntiqueWhite"/>
        <Label Text="Username" TextColor="White" />
        <Entry />
        <Label Text="Password" TextColor="White" />
        <Entry IsPassword="True" />
        <Button Text="Log in" BackgroundColor="#0094FF"
Clicked="LoginButton_Clicked" />
    </VerticalStackLayout>
</ContentPage>
```

O código C# que inicializa a página e implementa o manipulador de eventos para o evento `Clicked` do controle `LoginButton` no arquivo code-behind tem esta aparência:

C#

```
namespace MauiXaml;

public partial class Page1 : ContentPage, IPage
{
    public Page1()
    {
        InitializeComponent();
    }

    void LoginButton_Clicked(object sender, EventArgs e)
```

```
    {
        Debug.WriteLine("Clicked !");
    }
}
```

> **Observação**
>
> O método `InitializeComponent` no construtor de página lê a descrição do XAML da página, carrega os vários controles nessa página e define suas propriedades. Você só chamará esse método se definir uma página usando a marcação XAML. O exemplo anterior mostrando como criar a interface do usuário usando código C# não invoca `InitializeComponent`.

A estrutura permite a separação do design e do comportamento. Toda a declaração da interface do usuário está contida em um único arquivo de origem dedicado. Ele é separado do comportamento da interface do usuário. Além disso, a marcação XAML fornece maior clareza para um desenvolvedor que está tentando reconhecer a aparência do aplicativo.

Benefícios de usar o XAML

O uso de XAML permite separar a lógica de comportamento do design da interface do usuário. Essa separação ajuda você a criar cada parte de modo independente e torna todo o aplicativo mais fácil de gerenciar à medida que ele cresce.

Essa abordagem também permite que um designer de interface do usuário especialista trabalhe na atualização da aparência da interface do usuário usando ferramentas de edição XAML separadamente de um desenvolvedor que atualiza a lógica da interface do usuário.

Verificação de conhecimentos

1. Qual método você precisa chamar no construtor de uma página para ler a descrição XAML da página, carregar os vários controles nessa página e definir suas propriedades? *

- ◯ InitializeComponent

  ✔ **Esta é a resposta correta.**

- ◯ MainPage
- ◯ ConfigureApp

2. Qual desses é um grande benefício de usar XAML para definir sua interface do usuário? *

- ○ O XAML é executado com mais eficiência do que o código C# para construir a interface do usuário em runtime.
- ◉ Você pode separar a definição da interface do usuário da lógica do aplicativo.

✔ **Esta é a resposta correta.**

- ○ O XAML é mais portátil do que o código C#.

Unidade seguinte: Tipos e propriedades de XAML no .NET MAUI

Continuar >