Wprowadzenie do budowania interfejsów w .NET MAUI

Przewodnik z przykładami, podsumowaniami i ćwiczeniami

Wprowadzenie

.NET MAUI (Multi-platform App UI) to ewolucja Xamarin.Forms, umożliwiająca tworzenie nowoczesnych aplikacji mobilnych i desktopowych z jednej bazy kodu. W tym przewodniku poznasz podstawy tworzenia interfejsów użytkownika w .NET MAUI, krok po kroku, z praktycznymi przykładami.

Przygotowanie środowiska

1. Otwórz Microsoft Visual Studio 2022 (minimum wersja 17.3 lub nowsza)
2. Upewnij się, że masz zainstalowane obciążenie "Programowanie aplikacji mobilnych z pakietem .NET" (Mobile development with .NET)
3. Dla aplikacji desktopowych upewnij się, że masz zainstalowane obciążenie "Programowanie aplikacji klasycznych dla platformy .NET" (Desktop development with .NET)

> **Ważne:**
> Jeśli napotkasz problemy z długimi ścieżkami podczas kompilacji, umieść projekt w krótszej ścieżce, np. bezpośrednio na pulpicie lub dysku C:.

Tworzenie pierwszego projektu

1. Otwórz Microsoft Visual Studio 2022
2. Wybierz "Utwórz nowy projekt" (Create a new project)
3. Wybierz typ projektu ".NET MAUI App". Możesz użyć filtrowania przez C#, Multi-platform, MAUI
4. Nazwij projekt: "InterfejsMAUITest"
5. Zaakceptuj domyślne ustawienia i kliknij "Utwórz"

Struktura projektu

Po utworzeniu projektu, zauważysz następującą strukturę:

- **App.xaml/App.xaml.cs** - punkt wejściowy aplikacji
- **AppShell.xaml/AppShell.xaml.cs** - definicja głównej powłoki aplikacji
- **MainPage.xaml/MainPage.xaml.cs** - główna strona aplikacji
- **MauiProgram.cs** - konfiguracja aplikacji .NET MAUI
- **Platforms** - katalog zawierający kod specyficzny dla platform
- **Resources** - zasoby aplikacji (obrazy, fonty, style)

> **Różnica w stosunku do Xamarin.Forms:**
> W .NET MAUI kod specyficzny dla platform jest teraz umieszczony w folderze Platforms, a nie w osobnych projektach dla każdej platformy.

Wersja 1 - Podstawowy Label

Zmodyfikuj plik `MainPage.xaml` następująco:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="InterfejsMAUITest.MainPage">

    <Label Text="Jan Kowalski"
           VerticalOptions="Center"
           HorizontalOptions="Center"
           TextColor="Red"
           BackgroundColor="Blue" />

</ContentPage>
```

Czego się nauczyliśmy

- Podstawowej struktury pliku XAML w .NET MAUI
- Użycia elementu Label do wyświetlania tekstu
- Ustawiania koloru tekstu i tła za pomocą właściwości TextColor i BackgroundColor
- Pozycjonowania elementów za pomocą właściwości VerticalOptions i HorizontalOptions

Warto zapamiętać

- W MAUI zmieniły się przestrzenie nazw - zamiast "http://xamarin.com/schemas/2014/forms" używamy "http://schemas.microsoft.com/dotnet/2021/maui"
- Struktura podstawowych kontrolek jest podobna do Xamarin.Forms, ale z ulepszeniami
- Opcje wyrównania jak Center, Start, End działają tak samo jak w Xamarin.Forms

Ćwiczenia do samodzielnego wykonania

1. Zmień tekst, kolory tła i czcionki etykiety na własne wartości
2. Wypróbuj różne opcje pozycjonowania (Start, End, Fill) zarówno dla VerticalOptions jak i HorizontalOptions
3. Dodaj właściwość FontSize i ustaw rozmiar czcionki na "Large"
4. Dodaj właściwość FontAttributes="Bold" i sprawdź jak wpływa na wygląd tekstu
5. Spróbuj użyć predefiniowanych kolorów (np. Colors.BlueViolet) zamiast Color.Blue

Wersja 2 - Kontrolki definiowane w kodzie C#

Usuń zawartość pliku `MainPage.xaml` i zastąp ją podstawowym szkieletem:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="InterfejsMAUITest.MainPage">

</ContentPage>
```

Następnie zmodyfikuj plik `MainPage.xaml.cs`:

```
using Microsoft.Maui.Controls;

namespace InterfejsMAUITest
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();

            Content = new Label
            {
                VerticalOptions = LayoutOptions.Center,
                Text = "Sterowanie z kodu"
            };

            Padding = new Thickness(50, 5, 5, 5);
        }
    }
}
```

Czego się nauczyliśmy

- Tworzenia elementów interfejsu programowo w kodzie C#
- Ustawiania wartości właściwości obiektów za pomocą inicjalizatorów obiektów
- Określania zawartości strony przez przypisanie do właściwości Content
- Ustawiania marginesów wewnętrznych za pomocą obiektu Thickness

Warto zapamiętać

- W MAUI możemy tworzyć interfejs zarówno deklaratywnie (XAML) jak i programowo (C#)
- Obiekt Thickness przyjmuje od 1 do 4 parametrów:
  Thickness(uniformSize) - ten sam rozmiar dla wszystkich stron
  Thickness(horizontalSize, verticalSize) - rozmiar poziomy i pionowy
  Thickness(left, top, right, bottom) - indywidualne wartości dla każdej strony
- Po wywołaniu InitializeComponent() możemy nadpisać ustawienia XAML kodem C#

Ćwiczenia do samodzielnego wykonania

1. Rozszerz kod C# dodając więcej właściwości do etykiety (TextColor, BackgroundColor, FontSize)
2. Zamień etykietę na przycisk (Button) z obsługą zdarzenia Clicked
3. Stwórz metodę obsługi zdarzenia Clicked, która zmienia tekst przycisku po kliknięciu
4. Stwórz interfejs programowo z dwoma kontrolkami - etykietą na górze i przyciskiem na dole
5. Wypróbuj różne wartości dla Padding i zaobserwuj ich wpływ na wygląd interfejsu

Wersja 3 - Stylizowana etykieta w kodzie

W tej wersji przedstawimy bardziej zaawansowane stylizowanie etykiety w kodzie C#:

```
using Microsoft.Maui.Controls;
using Microsoft.Maui.Graphics;

namespace InterfejsMAUITest
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();

            Content = new Label
            {
                Text = "Label 1",
                HorizontalOptions = LayoutOptions.Center,
                VerticalOptions = LayoutOptions.Center,
                BackgroundColor = Colors.Red,
                TextColor = Colors.Yellow
            };
        }
    }
}
```

Czego się nauczyliśmy

- Definiowania kolorów z użyciem klasy Colors w Microsoft.Maui.Graphics
- Łączenia różnych właściwości formatowania w jednym elemencie
- Tworzenia kompletnie stylizowanej etykiety w kodzie C#

Warto zapamiętać

- W .NET MAUI statyczna klasa Colors zastępuje klasę Color z Xamarin.Forms
- Możemy nadal używać dynamicznych kolorów: Color.FromRgb(255, 0, 0) lub Color.FromHex("#FF0000")
- Kolory można również definiować jako zasoby w słowniku zasobów (ResourceDictionary)
- W MAUI dodano lepsze wsparcie dla kolorów systemowych i trybów ciemny/jasny

Ćwiczenia do samodzielnego wykonania

1. Stwórz etykietę z gradientowym tłem za pomocą LinearGradientBrush
2. Dodaj cień do etykiety za pomocą Shadow (nowa funkcja w MAUI)
3. Stwórz etykietę z obramowaniem o zaokrąglonych rogach używając Border i StrokeShape
4. Stwórz etykietę z przezroczystym tłem (Alpha < 1.0) używając Color.FromRgba()
5. Zaimplementuj przełączanie między jasnym a ciemnym motywem aplikacji za pomocą AppThemeBinding