

C# és WPF

A C# és a WPF (Windows Presentation Foundation) kombinációjával modern grafikus felhasználói felületeket készíthetünk asztali alkalmazásokhoz.

Fejlesztői környezet: Visual Studio 2022 (A Community verzió ingyenes)

WPF alkalmazás létrehozása: Create a new project / WPF Application (C#) template / projekt név / framework / Create gomb

A létrehozott projektben a következő fájlokat látjuk:

- MainWindow.xaml: itt található a felhasználói felület (UI) XAML-ben.
- MainWindow.xaml.cs: ez a háttérkód, ami a logikát kezeli.

XAML alapok (UI létrehozása)

A WPF az XAML nyelvet (Extensible Application Markup Language) használja a felület leírására.

A Window tag segítségével deklaráljuk az alkalmazás főablakát. A Class paraméter értéke a névtér és osztálynév egy ponttal elválasztva, a háttérkódban található osztályra mutat. A Title paraméter a programablak címsorában megjelenő címet jelenti. A Height a programablak magassága, a Width pedig a szélessége. A tagek és a paraméterek neve is nagybetűvel kezdődik, sőt a szöveges értékek is.

Egy egyszerű felület gombbal és szöveggel:

`<Grid>`

(A Grid egy alapvető elrendezés konténer, amely sorokat és oszlopokat tartalmazhat. Itt most nem használjuk. Egyszerűbb esetekben a Grid is lehet szülőkonténer az UI elemek számára.)

`<StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">`

(A StackPanel vízszintesen vagy függőlegesen rendezi el a benne lévő elemeket. Az alapértelmezett Orientation="Vertical" beállítás miatt jelen esetben függőlegesen. A panel vízszintesen és függőlegesen középre van igazítva.)

`<TextBlock Text="Hello, WPF!"`

`FontSize="20"`

`Margin="0, 0, 0, 10"`

`HorizontalAlignment="Center"/>`

(A TextBlock egy szövegmegjelenítő elem. A szöveg a Text paraméter értéke. A FontSize paraméter segítségével a betűméretet adhatjuk meg. A Margin paraméter segítségével a külső margókat, ami jelen esetben alul 10 pixel. A HorizontalAligment paraméter segítségével a vízszintes igazítást állíthatjuk be.)

```
<Button Content="Kattints ide!"  
        Width="100"  
        Click="Button_Click"/>
```

(A Button egy gomb megjelenítésére szolgáló elem. A gomb felirata a Content paraméter értéke. A Width paraméter segítségével a gomb szélességét adhatjuk meg. A Click paraméter a gomb kattintás eseményének kezelésére szolgál, értéke annak a függvénynek a neve a háttérkódban, amelyiket meg kell hívnia kattintás esetén.)

```
</StackPanel>  
</Grid>
```

A MainWindow.xaml.cs kód elején a using segítségével importáljuk a szükséges névtereket. A System.Windows tartalmazza pl. a WPF alapvető osztályait, mint pl. Window (ablakok kezeléséhez) és MessageBox (üzenetablakok kezeléséhez).

A namespace után a projekt neve alapján generált névtér szerepel. A névtér a projekt vagy alkalmazás logikai részeit csoportosítja.

A partial class után a főablakot képviselő osztály neve (MainWindow) található. A Window osztályt örökli.

Ezen belül található az osztály konstruktora, amikor az ablak létrejön, ez fut le először. Az InitializeComponent() metódus feladata az XAML-ben definiált felület inicializálása.

Eseménykezelő hozzáadása a gombhoz:

```
private void Button_Click(object sender, RoutedEventArgs e){  
(Ez az eseménykezelő metódus akkor fut le, ha a gombra kattintunk. Az első  
paraméter, a sender azt az elemet jelöli, amely kiváltotta az eseményt, jelen esetben  
a gomb. A második paraméter az eseményhez kapcsolódó információkat  
tartalmazza.)  
        MessageBox.Show("Kattintottál!");  
(A metódus egy felugró ablakot jelenít meg a megadott szöveggel.)  
}
```

A WPF-ben a felugró üzenetablak (MessageBox) automatikusan a képernyő közepére igazítja magát. Ha azt szeretnénk, hogy a felugró ablak a főablakhoz képest legyen középre igazítva, akkor a MessageBox helyett egyedi Window-t vagy Custom MessageBox-ot kell létrehozni.

Egyéni modális ablak létrehozása (Window):

Adjunk hozzá a projekthez egy új WPF Window-t, ehhez kattintsunk a Solution Explorerben jobb gombbal a projekt nevére, majd válasszuk az Add / Item lehetőséget. A megjelenő ablakban válasszuk a Window (WPF) sablont, a név legyen CustomMessageBox.xaml, majd kattintsunk az Add gombra.

A címsorban az "Üzenet" szöveg jelenjen meg. Az ablak szélessége és magassága legyen 150 pixel.

A Window tagnek legyen WindowStartupLocation paramétere, melynek értéke CenterOwner. Így az ablak a tulajdonos főablak közepére igazodik.

A Window tagnek legyen ResizeMode paramétere, melynek értéke NoResize. Így az ablak nem átméretezhető.

A Window tagnek legyen WindowStyle paramétere, melynek értéke SingleBorderWindow. Az ablak így csak egyszerű kerettel rendelkezik és címsorral.

```
<Grid>
    <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
        <TextBlock Text="Kattintottál!"
            FontSize="16"
            Margin="0, 0, 0, 10"
            TextAlignment="Center"/>
        <Button Content="Bezárás"
            Width="100"
            Click="CloseButton_Click"/>
    </StackPanel>
</Grid>
```

A bezárás gomb kezelése a CustomMessageBox.xaml.cs-ben:

```
private void CloseButton_Click(object sender, RoutedEventArgs e)
{
```

```
        this.Close();  
(Az ablak bezárása.)  
    }
```

Nyissuk meg a modális ablakot a főablakból.

```
private void Button_Click(object sender, RoutedEventArgs e)  
{  
    CustomMessageBox customMessageBox = new CustomMessageBox();  
(Létrehozunk egy példányt az egyéni ablakból.)  
    customMessageBox.Owner = this;  
(A főablakot tulajdonosként beállítjuk.)  
    customMessageBox.ShowDialog();  
(Az ablakot modálisan megnyitjuk, vagyis a főablak nem elérhető közben.)  
}
```

Elrendezési konténerek

A WPF-ben a felhasználói felület elemeinek elrendezését konténerek segítségével oldjuk meg.

Leggyakoribb konténerek:

- Grid
- StackPanel
- DockPanel
- WrapPanel
- Canvas

Grid

A leggyakrabban használt konténer. Sorokat és oszlopokat hoz létre a felületen.

Hozzunk létre egy ablakot, amely három sorból és két oszlopból áll.

```
<Grid>  
    <Grid.RowDefinitions>  
(A sorok definiálása.)  
        <RowDefinition Height="Auto"/>  
(Az első sor magassága automatikus, a tartalom alapján.)  
        <RowDefinition Height="*/>  
(A második sor magassága akkora, hogy a fennmaradó helyet kitöltse.)
```

```
<RowDefinition Height="50"/>
```

(A harmadik sor magassága fixen 50 pixel.)

```
</Grid.RowDefinitions>
```

```
<Grid.ColumnDefinitions>
```

(Az oszlopok definiálása.)

```
<ColumnDefinition Width="*" />
```

(Az első oszlop egy egység széles.)

```
<ColumnDefinition Width="2*" />
```

(A második oszlop két egység széles.)

```
</Grid.ColumnDefinitions>
```

```
<TextBlock Text="Első sor, első oszlop"
```

```
Grid.Row="0" Grid.Column="0"
```

```
FontWeight="Bold"
```

```
Margin="5"
```

```
Background="Aqua"/>
```

(Az első sor első oszlopában lévő cella szövege, a szöveg félkövér formázása, 5 pixeles margó és aqua háttérszín alkalmazása. A sorok és oszlopok indexelése 0-tól kezdődik.)

```
<TextBlock Text="Első sor, második oszlop"
```

```
Grid.Row="0" Grid.Column="1"
```

```
FontWeight="Bold"
```

```
Margin="5"
```

```
Background="Salmon"/>
```

(Az első sor második oszlopában lévő cella szövege, a szöveg félkövér formázása, 5 pixeles margó és salmon háttérszín alkalmazása. A sorok és oszlopok indexelése 0-tól kezdődik.)

```
<TextBlock Text="Második sor, teljes szélesség"
```

```
Grid.Row="1" Grid.ColumnSpan="2"
```

```
FontWeight="Bold"
```

```
Margin="5"
```

```
Background="LightGoldenrodYellow"/>
```

(A második sorban a két cella egyesítve van. A cella szövege, a szöveg félkövér formázása, 5 pixeles margó és lightgoldenrodyellow háttérszín alkalmazása. A sorok és oszlopok indexelése 0-tól kezdődik.)

```
<TextBlock Text="Harmadik sor, első oszlop"
```

```
Grid.Row="2" Grid.Column="0"
```

```
FontWeight="Bold"
```

```
Margin="5"
```

```
Background="Bisque"/>
```

(A harmadik sor első oszlopában lévő cella szövege, a szöveg félkövér formázása, 5 pixeles margó és bisque háttérszín alkalmazása. A sorok és oszlopok indexelése 0-tól kezdődik.)

```
<TextBlock Text="Harmadik sor, második oszlop"
```

```
Grid.Row="2" Grid.Column="1"
```

```
FontWeight="Bold"
Margin="5"
Background="Silver"/>
```

(A harmadik sor második oszlopában lévő cella szövege, a szöveg félkövér formázása, 5 pixeles margó és silver háttérszín alkalmazása. A sorok és oszlopok indexelése 0-tól kezdődik.)

```
</Grid>
```

A Grid.RowDefinitions és Grid.ColumnDefinitions segítségével adjuk meg a sorok és oszlopok számát és méretét.

A Grid.Row és Grid.Column segítségével adjuk meg, hogy egy elem melyik sorban és melyik oszlopban helyezkedjen el. A sorszámozás 0-tól kezdődik.

Cellaegyesítés esetén a Grid.ColumnSpan és a Grid.RowSpan használható.

Ha átméretezzük az ablakot, az oszlopok szélessége arányosan változik, az első és utolsó sor magassága nem változik (az első a tartalomtól függ, az utolsó pedig fixen 50 pixel), a második sor magassága változik, mivel ki kell töltenie a rendelkezésre álló helyet.

StackPanel

A StackPanel egy egyszerű elrendezési konténer, amely a gyermekelemeket egymás alá (vertikálisan) vagy egymás mellé (horizontálisan) rendezi az Orientation tulajdonság beállításától függően.

Egyszerű vertikális StackPanel, ami három gombot tartalmaz.

```
<StackPanel Orientation="Vertical"
Background="LightBlue"
Margin="10">
```

(Vertikális StackPanel létrehozása, a háttérszín világoskék, a külső margó 10 pixel.)

```
<Button Content="Első gomb"
Width="200"
Height="50"
Margin="5"
HorizontalAlignment="Center"/>
```

(A gomb felirata "Első gomb", a szélessége 200 pixel, a magassága 50 pixel, a külső margó 5 pixel és vízszintesen középre van igazítva.)

```
<Button Content="Második gomb"
Width="200"
```

```
        Height="50"
        Margin="5"
        HorizontalAlignment="Center"/>
<Button Content="Harmadik gomb"
        Width="200"
        Height="50"
        Margin="5"
        HorizontalAlignment="Center"/>
</StackPanel>
```

Az Orientation="Vertical" alapértelmezett érték, tehát elhagyható. A gombok a StackPanelen belül vízszintesen középre vannak igazítva. A gombok körül azért kell margót használni, hogy némi távolság köztük.

Horizontális elrendezés esetén az Orientation="Horizontal" értéket kell használni.

```
<StackPanel Orientation="Horizontal"
        Background="LightBlue"
        Margin="10">
    <Button Content="Első gomb"
            Width="200"
            Height="50"
            Margin="5"/>
    <Button Content="Második gomb"
            Width="200"
            Height="50"
            Margin="5"/>
    <Button Content="Harmadik gomb"
            Width="200"
            Height="50"
            Margin="5"/>
</StackPanel>
```

A StackPanel az ablak teljes szélességét elfoglalja, ha nem állítjuk be a szélességét.

Egymásba ágyazott StackPaneleket is használhatunk, például egy vertikális fő StackPanelen belül horizontális részeket hozhatunk létre.

```
<StackPanel Orientation="Vertical"
        Background="LightBlue"
```

```
Margin="10">
```

(A fő vertikális StackPanel.)

```
<TextBlock Text="Kombinált StackPanel példa"
  FontSize="18"
  Margin="5"
  HorizontalAlignment="Center"/>
```

(A fő StackPanel első eleme a cím.)

```
<StackPanel Orientation="Horizontal"
  Background="LightYellow"
  Margin="5"
  Height="100">
```

(A fő StackPanel második eleme az első horizontális StackPanel.)

```
<Button Content="Első gomb"
  Width="200"
  Height="50"
  Margin="5"/>
<Button Content="Második gomb"
  Width="200"
  Height="50"
  Margin="5"/>
<Button Content="Harmadik gomb"
  Width="200"
  Height="50"
  Margin="5"/>
```

(A horizontális StackPanel három gombot tartalmaz.)

```
</StackPanel>
<StackPanel Orientation="Horizontal"
  Background="LightGreen"
  Margin="5"
  Height="100">
```

(A fő StackPanel harmadik eleme a második horizontális StackPanel.)

```
<Button Content="Első"
  Width="200"
  Height="50"
  Margin="5"/>
<Button Content="Második gomb"
  Width="200"
  Height="50"
  Margin="5"/>
<Button Content="Harmadik gomb"
  Width="200"
  Height="50"
  Margin="5"/>
```

```
</StackPanel>
```

</StackPanel>

Természetesen egy horizontális fő StackPanelben is lehetnek vertikális StackPanelek beágyazva.

DockPanel

A DockPanel egy olyan elrendezési konténer, amely lehetővé teszi az elemek "dokkolását" (rögzítését) az ablak bal, jobb, felső és alsó széléhez. Akkor hasznos, ha az ablak különböző részeire (fejléc, lábléc, oldalsáv) szeretnénk elhelyezni elemeket.

Az elemek dokkolásának iránya a DockPanel.Dock tulajdonsággal szabályozható. Alapértelmezetten a középen fennmaradó területet foglalják el, ha nincs explicit dokkolásuk.

Egy egyszerű példa:

<DockPanel Background="LightGray">

(A DockPanel létrehozása világosszürke háttérrel.)

```
<TextBlock Text="Fejléc"
    DockPanel.Dock="Top"
    Background="LightBlue"
    FontSize="16"
    FontWeight="Bold"
    Padding="10"
    TextAlignment="Center"/>
```

(A fejléc a DockPanel tetejére van dokkolva, a betűméret 16 pixel, a betűstílus félkövér, a belső kitöltés 10 pixel, a szöveg középre van igazítva.)

```
<TextBlock Text="Lábléc"
    DockPanel.Dock="Bottom"
    Background="LightCoral"
    FontSize="14"
    FontWeight="Bold"
    Padding="10"
    TextAlignment="Center"/>
```

(A lábléc a DockPanel aljára van dokkolva.)

```
<TextBlock Text="Oldalsáv"
    DockPanel.Dock="Left"
    Background="LightGreen"
    FontSize="14"
    Padding="10"/>
```

```
TextAlignment="Center"/>
(Az oldalsáv a DockPanel bal oldalára van dokkolva.)
<TextBlock Text="Tartalom"
Background="White"
FontSize="14"
Padding="10"
TextAlignment="Center"/>
(A tartalom a fennmaradó helyett tölti ki.)
</DockPanel>
```

A tartalmi részt nem kell külön dokkolni, a fejléc, lábléc, oldalsáv dokkolása után azok az elemek, amelyekhez nincs dokkolás megadva, automatikusan a DockPanel fennmaradó helyét töltik ki.

Egy menüt például elhelyezhetünk a fejlécben is vízszintes menüként, de az oldalsávban is függőleges menüként.

Hosszabb szövegek esetén használjuk a TextBlock TextWrapping="Wrap" tulajdonságát, hogy a szöveg több sorba törjön, ha nem fér el egy sorban.

WrapPanel

A WrapPanel egy rugalmas elrendezési konténer, és különösen hasznos, ha dinamikusan változó elemeket szeretnénk megjeleníteni egy sorban vagy oszlopban.

Sorban vagy oszlopban helyezi el az elemeket, és ha a hely elfogy, akkor az elemeket automatikusan a következő sorba vagy oszlopba "tördeli".

Az Orientation tulajdonság határozza meg az elemek elrendezési irányát. A Horizontal érték az alapértelmezett, ebben az esetben az elemek balról jobbra lesznek elhelyezve, majd új sorba kerülnek. Vertical érték esetén pedig az elemek fentről lefelé lesznek elhelyezve, majd új oszlopba kerülnek.

Ideális választás olyan helyzetekben, amikor dinamikusan változó számú elemet kell rugalmasan megjeleníteni, hiszen ha nem tudjuk előre az elemek számát, nehéz más elrendezést használni.

Helyezzünk el WrapPanel segítségével gombokat, amelyek automatikusan átrendeződnek, ha az ablak mérete változik.

```
<WrapPanel Orientation="Horizontal"
    Background="LightGray"
    Margin="10">
```

(A WrapPanel létrehozása, amelyben vízszintesen helyezkednek el az elemek.)

```
    <Button Content="1. gomb"
        Width="100"
        Height="50"
        Margin="5"/>
```

...

```
    <Button Content="6. gomb"
        Width="100"
        Height="50"
        Margin="5"/>
```

```
</WrapPanel>
```

Függőleges elhelyezkedés: Orientation="Vertical".

UniformGrid

A UniformGrid olyan, mint a Grid, de minden cellája automatikusan azonos méretű. Az elemek egyenletesen oszlanak el. A sorok és oszlopok számát kell megadnunk.

```
<UniformGrid Rows="2"
    Columns="3"
    Background="LightGray">
```

(A UniformGrid létrehozása, amiben 2 sor és 3 oszlop van.)

```
    <Button Content="1. Gomb"
        Margin="5"
        Width="100"
        Height="50"/>
```

...

```
    <Button Content="6. Gomb"
        Margin="5"
        Width="100"
        Height="50"/>
```

```
</UniformGrid>
```

Ideális egyenletesen elosztott elemekhez, pl. számológép gombok, menüpontok.

Canvas

A Canvas egy nagyon egyszerű elrendezési konténer, amely lehetővé teszi az elemek pontos, koordináták alapú elhelyezését. Az elemek elhelyezéséhez a Canvas.Left és a Canvas.Top tulajdonságokat használjuk. Az előbbivel az elem bal szélének a távolságát adjuk meg a Canvas bal oldalához képest, utóbbival az elem tetejének a távolságát adjuk meg a Canvas felső oldalához képest. A Canvas.Right és a Canvas.Bottom alternatívaként használhatók, de ritkábban.

```
<Canvas Background="Lightyellow">
```

(A Canvas létrehozása.)

```
    <Button Content="1. gomb"
        Canvas.Left="50"
        Canvas.Top="30"
        Width="100"
        Height="50"/>
```

(A gomb létrehozása a megadott koordinátákban.)

```
    <Button Content="2. gomb"
        Canvas.Left="200"
        Canvas.Top="100"
        Width="100"
        Height="50"/>
```

```
</Canvas>
```

A Canvas használata C# kódban:

```
Canvas canvas = new Canvas()
{
    Background = Brushes.LightBlue,
    Width = 400,
    Height = 300
};
```

(A Canvas létrehozása világoskék háttérrel, 400 pixeles szélességgel és 300 pixeles magassággal.)

```
this.Content = canvas;
```

(A Canvas hozzáadása az ablakhoz.)

```
Button gomb1 = new Button()
{
    Content = "1. gomb",
    Width = 100,
    Height = 50
};
```

(Az egyik gomb létrehozása.)

```
Canvas.SetLeft(gomb1, 10);
```

```
Canvas.SetTop(gomb1, 10);
```

(A gomb elhelyezkedése.)

```
canvas.Children.Add(gomb1);
```

(A gomb hozzáadása a Canvas-hoz.)

```
Button gomb2 = new Button()
```

```
{
```

```
    Content = "2. gomb",
```

```
    Width = 100,
```

```
    Height = 50
```

```
};
```

(A másik gomb létrehozása.)

```
Canvas.SetLeft(gomb2, 200);
```

```
Canvas.SetTop(gomb2, 100);
```

(A gomb elhelyezkedése.)

```
canvas.Children.Add(gomb2);
```

(A gomb hozzáadása a Canvas-hoz.)

Az elemek pontosan oda kerülnek, ahova szeretnénk. Ideális például rajzolóprogramok esetén. Hátránya, hogy nem reszponzív, ha az ablak mérete megváltozik, az elemek nem alkalmazkodnak automatikusan.

TabControl

Bár technikailag nem elrendezés, a TabControl lehetővé teszi, hogy több fülre (tab) bontsuk a felhasználói felületet. Minden fül saját tartalommal rendelkezhet, és más-más elrendezéseket használhat.

```
<TabControl>
```

```
    <TabItem Header="Fül 1">
```

(Az első fülön megjelenő szöveg.)

```
        <StackPanel>
```

(Az első fülhöz tartozó tartalom elrendezési konténere.)

```
            <TextBlock Text="Ez az első fül tartalma" Margin="10"/>
```

```
            <Button Content="Gomb 1" Margin="10"/>
```

```
        </StackPanel>
```

```
    </TabItem>
```

```
    <TabItem Header="Fül 2">
```

(A második fülön megjelenő szöveg.)

```
<Grid>
```

(A második fülhöz tartozó tartalom elrendezési konténere.)

```
<TextBlock Text="Ez a második fül tartalma"
VerticalAlignment="Center" HorizontalAlignment="Center"/>
```

```
</Grid>
```

```
</TabItem>
```

```
</TabControl>
```

Expander

Az Expander egy olyan konténer, amelynek tartalmát ki lehet nyitni és össze lehet csukni. Például részletek megjelenítése.

```
<Expander Header="Bővebben" Background="LightBlue">
    <TextBlock Text="Ez itt az Expander tartalma." Margin="10"/>
</Expander>
```

ScrollView

A ScrollView egy olyan konténer, amely görgethetővé teszi a benne lévő tartalmat. Bármilyen elrendezést alkalmazhatunk benne.

```
<ScrollView HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">
(A vízszintes és a függőleges gördítőszalag akkor jelenik meg, ha szükség van rá.)
```

```
<StackPanel>
```

```
<TextBlock Text="Első sor" FontSize="16" Margin="10"/>
```

```
<TextBlock Text="Második sor" FontSize="16" Margin="10"/>
```

```
<TextBlock Text="Harmadik sor" FontSize="16" Margin="10"/>
```

```
<TextBlock Text="Negyedik sor" FontSize="16" Margin="10"/>
```

```
<TextBlock Text="Ötödik sor" FontSize="16" Margin="10"/>
```

```
<TextBlock Text="Hatodik sor" FontSize="16" Margin="10"/>
```

```
</StackPanel>
```

```
</ScrollView>
```