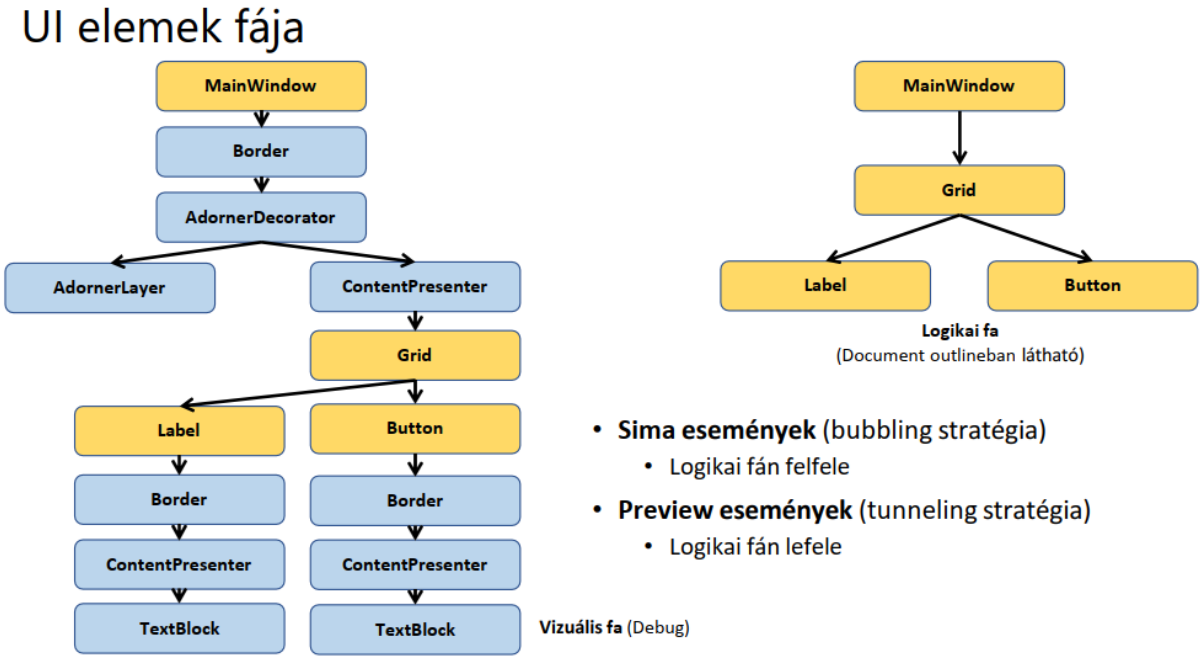
Eseménykezelés részletesebben



A **routed event** olyan esemény WPF-ben, ami **nem csak azon a vezérlőn fut le, ahol keletkezik**, hanem:

- **bejárja a vizuális fát**
- útközben **szülők (vagy gyerekek) is kezelhetik**

A *bubbling stratégiában* az esemény alulról felfelé halad a **Logikai fán**, amíg nem talál egy olyan vezérlőt, amely lekezeli. (pl. `Checked, Unchecked, Click`)

A *tunneling stratégiában* az esemény felülről lefelé halad a **Vizuális fán**, amíg nem talál egy olyan vezérlőt, amely lekezeli. Ha ez a vezérlő az eseményhez alapértelmezett vezérlő fölött van, akkor meg lehet akadályozni magának az eseménynek a kiváltását! (nem hajtódik végre!) (pl. `PreviewMouseDown, PreviewKeyDown`)

Nem minden bubbling eventnek van tunneling párja. Például létezik `Checked` (bubbling), de nem létezik `PreviewChecked` (tunneling) Csak az **INPUT** eseményeknek van tunneling párja, míg a `Checked` nem input esemény, hanem állapotváltozás!

Nézzünk pár apró példát. A vezérlőkre csak a nevükkel hivatkozok a példakódban, a XAML részleteket minimalizálom!

1. példa: Egy Grid-be belerakok sok Label-t, majd azt szeretném, hogy ha ráállok egérrel bármelyik Labelre, akkor történjen vele valami. Mindezt úgy, hogy az egérrel történő mozgást NEM egyesével figyelem a Labelen, hanem a Griden kezelem le az eseményt.

```
<Grid MouseMove="Grid_MouseMove"> XAML
```

```
private void Grid_MouseMove(object sender, MouseEventArgs e)
 {
     if (e.Source is Label l)
     {
         l.Background = Brushes.LightGreen;
             .
             .
             .
     }
 }
```

2. példa: Egy számológépet hozunk létre. Itt **NEM** a gombokat tartalmazó vezérlőn (pl. Grid) kezeljük le az eseményeket, hanem minden gombhoz hozzárendeljük ugyanazt az eseményt! Ez ebben az esetben FONTOS, hisz a Gridhez kötött logika **UI-függő**, a gombra tudunk egérrel kattintani, de beírhatjuk a billentyűzeten is stb. Azt meg már nem kezeli a Grid!!!

```
<Button Content="1" Tag="1" Margin="10" Grid.Row="2" Grid.Column="0"
Click="Button_Click" />
<Button Content="2" Tag="2" Margin="10" Grid.Row="2" Grid.Column="1"
Click="Button_Click" /> … XAML
```

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    if (sender is not Button btn)
    {
        return;
    }
    string value = btn.Tag.ToString();

    txt.Text += value;
}
```

3 példa: Az utolsó példában egy olyan vezérlőn kezelünk eseményeket, amelynél külön be kell állítani azt az adott eseményt, mert alapból az a vezérlő azt az eseményt **NEM** tudja kezelni! Itt egy GroupBox-on kezeljük le a benne lévő CheckBox-ok kijelölését! A XAML kód:

```
<GroupBox x:Name="gb" Header="Választás">
    <StackPanel>
        <CheckBox Content="Első"/>
        <CheckBox Content="Második"/>
        <CheckBox Content="Harmadik"/>
    </StackPanel>
</GroupBox>
```

Látható, hogy itt csak nevet adtam a vezérlőnek, nincs esemény hozzárendelve. Ezt a hozzárendelést a C# kódban adom meg:

```
gb.AddHandler(
    CheckBox.CheckedEvent,
    new RoutedEventHandler(CheckBox_Checked));
```

A hivatkozott metódusban pedig megadom, hogy mi történjen, ha valamelyik CheckBox-ra kattintunk:

```
private void CheckBox_Checked(object sender, RoutedEventArgs e)
{
    if (e.OriginalSource is CheckBox cb)
    {
        txt.Text += cb.Content.ToString();
    }
}
```