

## Wykład Menu



## Pasek statusu

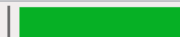
# Pasek stanu

## StatusBar

Podobnie jak w kontrolce Grid najpierw definiujemy strukturę panelu `<ItemsPanelTemplate>`, dopiero potem dodajemy do niego poszczególne elementy `<StatusBarItem>` wewnątrz których możemy umieszczać różnego typu kontrolki

```
<StatusBar>
  <StatusBar.ItemsPanel>
    <ItemsPanelTemplate>
      <Grid>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="Auto" />
          <ColumnDefinition Width="120" />
        </Grid.ColumnDefinitions>
      </Grid>
    </ItemsPanelTemplate>
  </StatusBar.ItemsPanel>
  <StatusBarItem Grid.Column="0" HorizontalContentAlignment="Center">
    <TextBlock Text="Przykładowy tekst" />
  </StatusBarItem>
  <Separator Grid.Column="1" />
  <StatusBarItem Grid.Column="2" HorizontalContentAlignment="Stretch">
    <ProgressBar Value="75" Height="16"/>
  </StatusBarItem>
</StatusBar>
```

Przykładowy tekst



# Wizualne systemy programowania

---

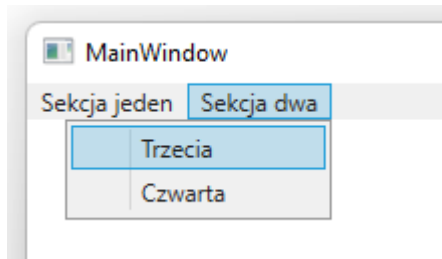
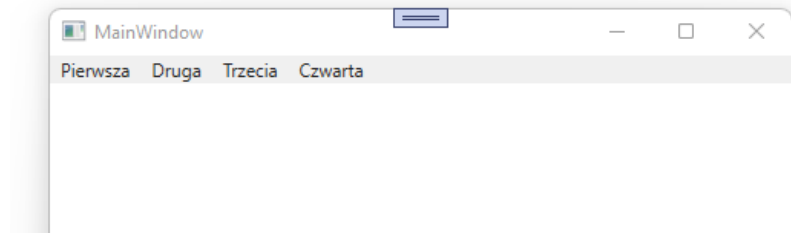


## Menu

# Menu

Jednym z przydatnych komponentów systemu Windows jest menu. W WPF-ie realizowane jest standardowo za pomocą kontrolki `<Menu></Menu>`

```
<Menu>
  <MenuItem Header="_Pierwsza"/>
  <MenuItem Header="_Druga"/>
  <MenuItem Header="_Trzecia"/>
  <MenuItem Header="_Czwarta"/>
</Menu>
```



Menu rozwijane uzyskujemy poprzez zagnieżdżanie znaczników `<MenuItem>`

```
<Menu>
  <MenuItem Header="Sekcja jeden">
    <MenuItem Header="_Pierwsza"/>
    <MenuItem Header="_Druga"/>
  </MenuItem>
  <MenuItem Header="Sekcja dwa">
    <MenuItem Header="_Trzecia"/>
    <MenuItem Header="_Czwarta"/>
  </MenuItem>
</Menu>
```



## Menu

Podkreślenie przed pierwszym znakiem każdej z etykiet tworzy skrót klawiaturowy do tej pozycji. Naciskamy **Alt**, a następnie dany znak, aby aktywować pozycję menu.

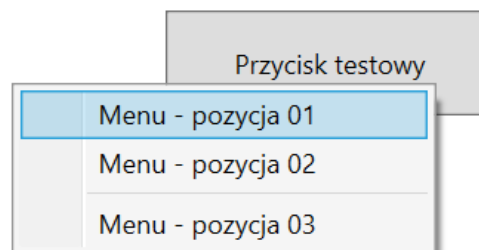
```
<Menu>  
    <MenuItem Header="_Pierwsza"/>  
    <MenuItem Header="_Druga"/>  
    <MenuItem Header="_Trzecia"/>  
    <MenuItem Header="_Czwarta"/>  
</Menu>
```

W tym przykładzie: Alt + D  
pozycja druga w menu.

## Menu kontekstowe

Menu kontekstowe dodajemy jako znacznik podrzędny do innej kontrolki (w tym przykładzie do przycisku).

```
<Button Content="Przycisk testowy">  
  <Button.ContextMenu>  
    <ContextMenu>  
      <MenuItem Header="Menu - pozycja 01" />  
      <MenuItem Header="Menu - pozycja 02" />  
      <Separator />  
      <MenuItem Header="Menu - pozycja 03" />  
    </ContextMenu>  
  </Button.ContextMenu>  
</Button>
```



## Obsługa menu w kodzie programu

---

Najprostszą (choć nie jedyną) metodą oprogramowania systemu menu jest dodanie do każdej pozycji menu jej własnej metody obsługi zdarzenia reagującej na zdarzenie Click.

```
<MenuItem Header=„Pozycja 01" Click="MenuItem_Click" />
```

```
private void MenuItem_Click(object sender, RoutedEventArgs e)
{
    // metod obsługi wybranej pozycji w menu
}
```



# Wizualne systemy programowania

---





## Timer

---

W środowisku WPF timer znajduje się w przestrzeni nazw;  
`using System.Windows.Threading;`

Utworzenie obiektu timera

```
DispatcherTimer dispatcherTimer = new DispatcherTimer();
```

Kolejnym krokiem jest ustawienie interwału timera.

Mamy wiele możliwości:

```
dispatcherTimer.Interval = TimeSpan.FromMilliseconds(10);  
dispatcherTimer.Interval = TimeSpan.FromSeconds(10);  
dispatcherTimer.Interval = TimeSpan.FromMinutes(10);  
dispatcherTimer.Interval = TimeSpan.FromHours(10);  
dispatcherTimer.Interval = TimeSpan.FromDays(10);
```

Można ustawić interwał zbiorczo (godziny, minuty, sekundy):

```
dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
```

Lub też w taktach (nie polecam) `dispatcherTimer.Interval = TimeSpan.FromTicks(1);`

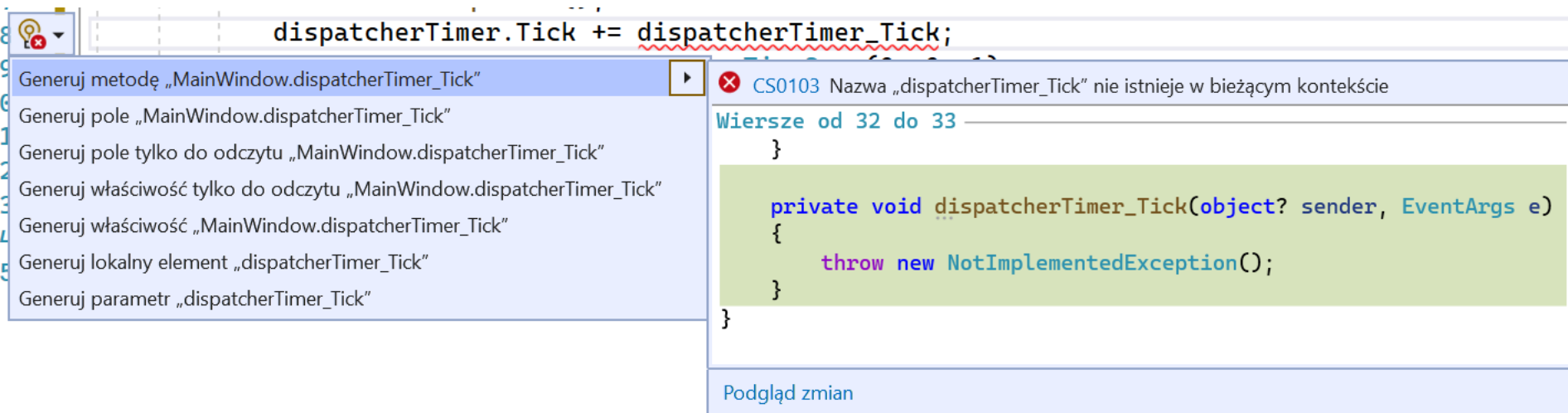
# Timer

Kolejnym krokiem jest stworzenie metody obsługi zdarzenia timera.

Dodajemy do timera metodę obsługi zdarzenia Tick

```
dispatcherTimer.Tick += dispatcherTimer_Tick;
```

Metodę tę należy utworzyć:



dispatcherTimer.Tick += dispatcherTimer\_Tick;

Generuj metodę „MainWindow.dispatcherTimer\_Tick”

Generuj pole „MainWindow.dispatcherTimer\_Tick”

Generuj pole tylko do odczytu „MainWindow.dispatcherTimer\_Tick”

Generuj właściwość tylko do odczytu „MainWindow.dispatcherTimer\_Tick”

Generuj właściwość „MainWindow.dispatcherTimer\_Tick”

Generuj lokalny element „dispatcherTimer\_Tick”

Generuj parametr „dispatcherTimer\_Tick”

CS0103 Nazwa „dispatcherTimer\_Tick” nie istnieje w bieżącym kontekście

Wiersze od 32 do 33

```
private void dispatcherTimer_Tick(object? sender, EventArgs e)
{
    throw new NotImplementedException();
}
```

Podgląd zmian



# Timer

---

Przykład metody obsługi zdarzenia Tick.

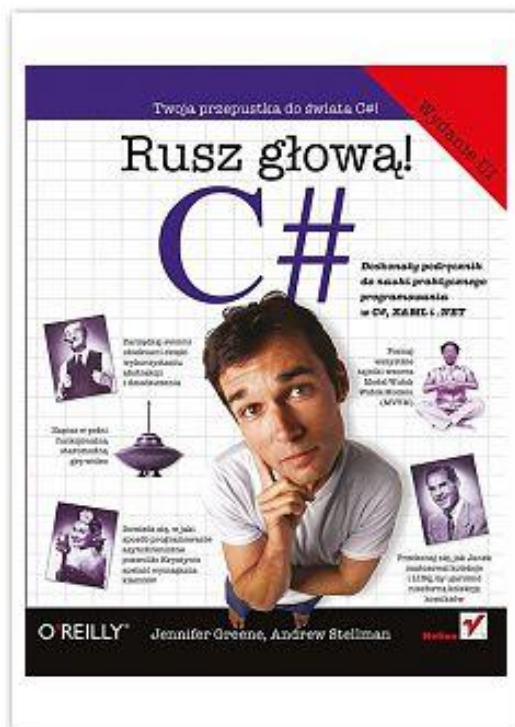
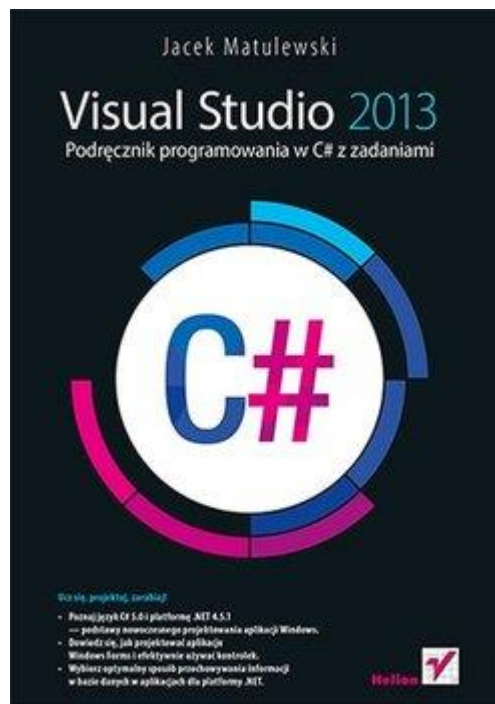
```
private void dispatcherTimer_Tick(object? sender, EventArgs e)
{
    textBlock01.Text = (++i).ToString();
}
```

Tak przygotowany timer możemy uruchomić, lub też zatrzymać.

```
dispatcherTimer.Start();
```

```
dispatcherTimer.Stop();
```

# Literatura:



Użyte w tej prezentacji tabelki pochodzą z książki: Visual Studio 2013. Podręcznik programowania w C# z zadaniami  
Autor: Matulewski Jęcek, Helion