Nastavení automatického zarovnání

Ve chvíli, když budeme vytvářet aplikaci, která bude využívána ke skutečně užitečné činnosti, počet prvků v aplikaci bude pravděpodobně dost vysoký. Doposavad jsme pozici prvků udávali pouze na základě použití atributů **HorizontalAlignment**, **VerticalAlignment** a **Margin** a byť v jednodušších aplikacích se jedná o způsob bez problému použitelný, v aplikacích komplexnějších by bylo nastavování prvků tímto způsobem problematické.

Pokud bude potřeba, aby se prvky zarovnávaly automaticky a třeba i dynamicky (přesun prvku na základě úpravy velikosti okna) a učinit díky tomu naši aplikaci responzivní (změna rozložení na základě velikosti okna) můžeme k tomuto účelu použít jednotlivé typy panelů.

Tím úplně základním typem panelu je element Canvas. Jedná se o element, který dovoluje nastavovat svým potomků (elementům, které budou do něj vloženy) pozici absolutním způsobem. Pozici je možné nastavit v XAMLu pomocí atributů Canvas.Left, Canvas.Right, Canvas.Left, Canvas.Bottom. Pokud bychom v aplikaci z minule vyměnily element Grid za Canvas, mohli bychom pozici popisku nastavit následovně

```
<Label Name="ClickCountLabel" Content="Label" Canvas.Left="253"
Canvas.Top="190"/>
```

Jelikož je nastavení pozice elementu závislé na rodičovském prvku, neprobíhá nastavení pomocí C# na základě načtení atributu, ale pomocí statické metody třídy Canvas. Pokud bychom například potřebovali, aby byl popisek umístěn na pozici 500 zleva a 500 ze shora, mohli bychom v C# napsat následující příkazy

```
Canvas.SetLeft(ClickCountLabel, 500);
Canvas.SetTop(ClickCountLabel, 500);
```

Další panel, který můžeme použít a který by Vám měl být už povědomí je panel Grid (mřížka). V předešlých verzích aplikace jsme pozici elementů umístěných v tomto panelu nastavovali prakticky stejně jako kdybychom je nastavovali v plátně – udávali jsme jejich odsazení zleva a ze shora a tím jim určovali souřadnice, kde budou elementy umístěny. Pomocí mřížky ale můžeme pozici nastavovat i úplně jinak. Jak již název panelu napovídá, potomky tohoto elementu můžeme zarovnávat do mřížky – nastavíme v jakém budou sloupci a v jakém řádku. Před tím, než ale elementy do sloupce či řádku umístíme, musíme je vytvořit. V základu mřížka obsahuje řádek a sloupec pouze jeden a pokud jich budeme potřebovat více budeme muset ve mřížce vytvořit element Grid.ColumnDefinitions a Grid.RowDefinitions. Do Grid.ColumnDefinitions budeme vkládat definici sloupců a do Grid.RowDefinitions definici řádků. Definice sloupce se provádí pomocí elementu ColumnDefinition a definice řádku pomocí RowDefinition. Pokud bychom potřebovali vytvořit mřížku se čtyřmi buňkami (dva řádky a dva sloupce) vypadal by její zápis v XAMLu takto

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>
```

```
</Grid>
```

U jednotlivých sloupců můžeme následovně nastavit jejich šířku pomocí atributu Width a u řádků můžeme nastavit jejich výšku. Natavení výšky i šířky může probíhat buď zadáním množství pixelů anebo zadáním poměrné části. Pokud budeme velikost zadávat absolutně (pixely) bude sloupec či řádek pořád stejně veliký i přes zvětšování a zmenšování aplikace. Pokud budeme zadávat velikost relativně (poměrná část) bude se velikost sloupce či řádku měnit se změnou velikosti aplikace. Pro využití relativního nastavení je nutné za hodnotu velikosti zapsat znak *. Pokud bychom nastavili velikost například následujícím způsobem

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="2*"></ColumnDefinition>
        <ColumnDefinition Width="1*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="4*"></RowDefinition>
        <RowDefinition Height="3*"></RowDefinition>
    </Grid.RowDefinitions>
</Grid>,
```

pak bude mřížka rozdělena v šířce na tři části. První sloupec bude zabírat 2 části šířky a druhý jednu část. V případě výšky bude mřížka rozdělena na 7 částí. První řádek bude zabírat 4 části výšky a druhý řádek bude zabírat 3 části výšky. Pokud budeme poté do mřížky vkládat jednotlivé elementy budou automaticky umístěny do první buňky a pokud je budeme potřebovat přemístit je toto možné provést pomocí nastavení atributů Grid.Row a Grid.Column u každého elementu ve mřížce. Indexace řádků a sloupců probíhá od 0. Pokud bychom například potřebovali, aby tlačítko z minulé verze programu bylo umístěno v prvním řádku a prvním sloupci mřížky a popisek v druhém řádku a sloupci mřížky bylo by možné XAML upravit následovně

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="2*"></ColumnDefinition>
        <ColumnDefinition Width="1*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="4*"></RowDefinition>
        <RowDefinition Height="3*"></RowDefinition>
    </Grid.RowDefinitions>
    <Label Grid.Column="1" Grid.Row="1" Name="ClickCountLabel" Content="Label"/>
    <Button Grid.Column="0" Grid.Row="0" Click="Button_Click" Content="TLAČÍTKO"
Width="200" Height="100" Background="Black" Foreground="White" FontSize="20"/>
</Grid>
```

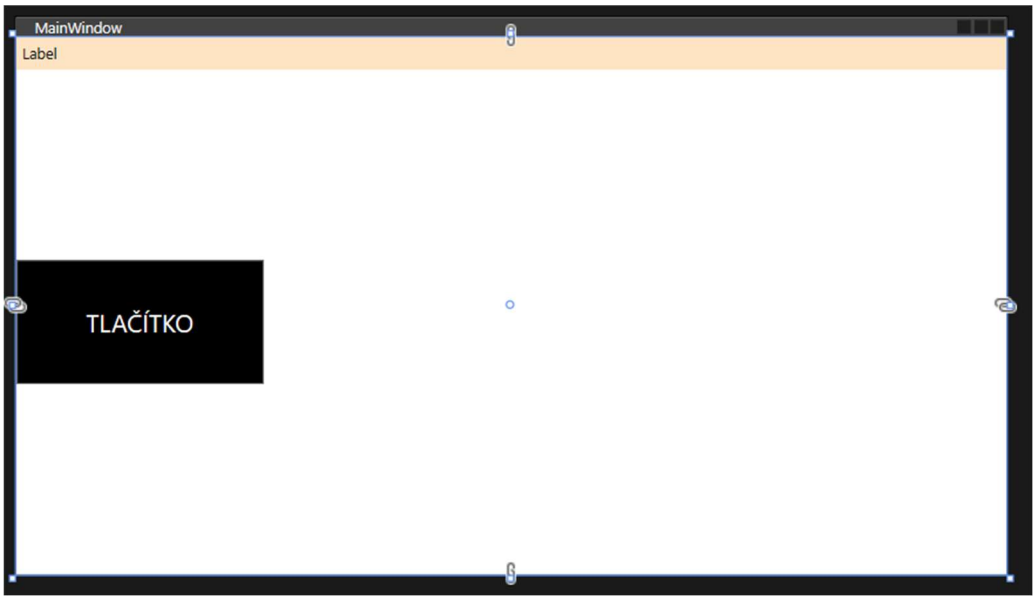
Po zápisu kódu výše uvidíte, že je tlačítko automaticky umístěno ve středu své buňky a popisek zabírá celý obsah své buňky (toto byl také důvod proč když jsme minule vytvořili tlačítko a neudali jsem žádné nastavení, zakrylo celé okno aplikace). Nastavení elementu v buňce je nadále možné provádět pomocí atributů **HorizontalAlignment**, **VerticalAlignment** a **Margin** čímž můžeme základní chování (roztažení elementu a jeho pozici) změnit.

Dalším velmi často využívaným panelem ve WPF je panel DockPanel. Jedná se o panel u kterého můžeme nastavit zarovnání ze shora, zprava, zdola a zleva, přičemž všechny elementy jsou zarovnány takovým způsobem aby nepřekrývaly elementy ostatní – každý další vložený prvek se nalepí z patřičné straně k elementům ostatním. Pokud bychom opět upravili minulou aplikaci takovým způsobem, aby místo mřížky používala DockPanel XAML by mohl vypadat následovně

```
<DockPanel>
    <Label Background="Bisque" DockPanel.Dock="Top" Grid.Column="1" Grid.Row="1"
Name="ClickCountLabel" Content="Label"/>
    <Button DockPanel.Dock="Left" Grid.Column="0" Grid.Row="0"
Click="Button_Click" Content="TLAČÍTKO" Width="200" Height="100"
Background="Black" Foreground="White" FontSize="20"/>
</DockPanel>
```

a výsledné zarovnání v aplikaci bude vypadat takto

Z obrázku a XAMLu je patrné, že popisek byl správně podle nastavení zarovnán na vrch aplikace, ale tlačítko, byť mu bylo nastaveno zarovnání zleva se nadále nachází na středu zbývajícího prostoru. Důvodem je automatické nastavení DockPanelu takovým způsobem, aby se poslední jeho potomek roztáhl na celý zbývající prostor. Pokud bychom potřebovali, aby se tlačítko (poslední potomek DockPanelu) opravdu zarovnalo nalevo bude nutné DockPanelu nastavit atribut **LastChildFill** na hodnotu **False**. Aplikace pak bude vypadat následovně



Poslední dva typy panelů, které můžeme ve WPF používat jsou panely StackPanel a WrapPanel. Zobrazení potomků ve StackPanelu funguje způsobem postupného skládání jednoho prvku za druhým – posledně vložený prvek je zobrazen na konci panelu. U StackPanelu je možné nastavovat atribut **Orientation**, který určuje směr zarovnání. Možné hodnoty tohoto atributu jsou **Horizontal** a **Vertical**. WrapPanel funguje podobně jako StackPanel akorát s tím rozdílem, že pokud se potomci nevejdou do vyhrazeného prostoru budou zalomeny na nový řádek či sloupec. Atribut **Orientation** funguje stejně jako u StackPanelu.

Pokud budeme potřebovat vlastnosti jednotlivých panelů kombinovat je možné jeden panel vložit do druhého a tím vytvořit libovolně složité zobrazení.