LEKCE 4

Design aplikace, rozmístění a polohování prvků

Po úvodních pokusech s obdélníkem v příkladu v druhé lekci nastal čas na podrobnější seznámení se s grafickými prvky pro návrh prezentačního rozhraní. Jednotlivé prvky budou podrobně probrány v další lekci. Námětem této lekce je téma o jednu úroveň granularity vyšší, tedy ne design prvků, ale design uživatelského rozhraní Silverlight aplikace jako celek. Jinak řečeno – budeme se věnovat velikosti zobrazovací plochy aplikace, rozmístění prvků a definování jejich vzájemné polohy.

Rozmístění prvků prezentačního rozhraní

Dříve než se propracujete ke konkrétním možnostem a oblastem použití jednotlivých prvků, je potřeba věnovat se jejich rozmístění na ploše aplikace a způsobům, jak udržovat, nebo v případě potřeby i měnit nebo neměnit jejich velikost, případně i vzájemnou polohu, například v případě změny velikosti okna, měřítka zobrazování a podobně.

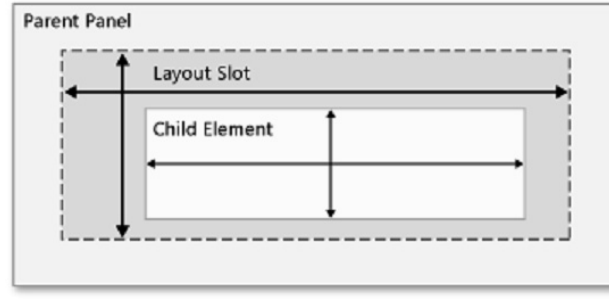
Na platformě Silverlight jsou pro tento účel k dispozici takzvané kontejnerové prvky, které zapouzdřují objekty v nich umístěné, čímž určují některé jejich vlastnosti, ale hlavně jejich

velikost, umístění a vzájemnou polohu. Abychom byli přesnější, kontejnerové prvky mohou zapouzdřovat buď přímo grafické prvky, nebo další kontejnery. Na nejvyšší úrovni je celá plocha aplikace čili objekt Panel. V něm je umístěn hlavní kontejnerový prvek pojmenovaný Layout Root.

Pro zapouzdření prvků se využívají kontejnerové objekty:

- Grid.
- Canvas.
- Stack Panel.

Tyto objekty zapouzdřují prvky nejen z pohledu objektově-orientovaného programování, ale také doslova vizuálně. To znamená, že například udržují jejich absolutní i relativní polohu i při změně rozměrů okna.



Obrázek 4.1: Hierarchie objektů Parent Panel, Layout Slot a Child Element

Základním vizuálním kontejnerem na platformě Silverlight je `Grid` čili určitá grafická forma mřížky. Znovu připomeňme XAML kód prázdné stránky vygenerované pomocí průvodce:

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SL3app.MainPage"
  Width="640" Height="480">

  <Grid x:Name="LayoutRoot" Background="White"/>
</UserControl>
```

Už tato kostra stránky prezentačního rozhraní obsahuje kontejnerový prvek `Grid`. Proč?

Určitě jste si všimli, že většina webových aplikací má pracovní plochu rozvrženou na několik samostatných oblastí. Může to být realizováno například pomocí HTML tabulky. Tato tabulka vlastně udržuje prvky v ní na absolutních nebo relativních pozicích.

Polohování prvku vůči kontejneru

Při prvních pokusech s návrhem uživatelského rozhraní ve vizuálním návrhovém prostředí jste si možná ani neuvědomili princip návrhu rozměrů prvku ani jeho polohování vůči ploše okna aplikace. Při vizuálním návrhu byl grafický prvek jednoduše tak velký, jak jste ho navrhli, a byl umístěn tam, kam jste ho vykreslili. Vzhledem k relativnímu polohování prvku vůči kontejneru, ve kterém je umístěný, je potřeba postupně probrat jednotlivé atributy pro vykreslení prvku. Jestliže byste zadali základní XAML kód obdélníka jen se dvěma parametry (barvou výplně a barvou obrysu bez specifikace rozměrů a polohy):

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Rectangle Fill="LightBlue" Stroke="Black" />
</Grid>
```

v tom případě obdélník zabere celou plochu kontejneru, ve kterém je umístěn, v tomto případě prvku `Grid`.

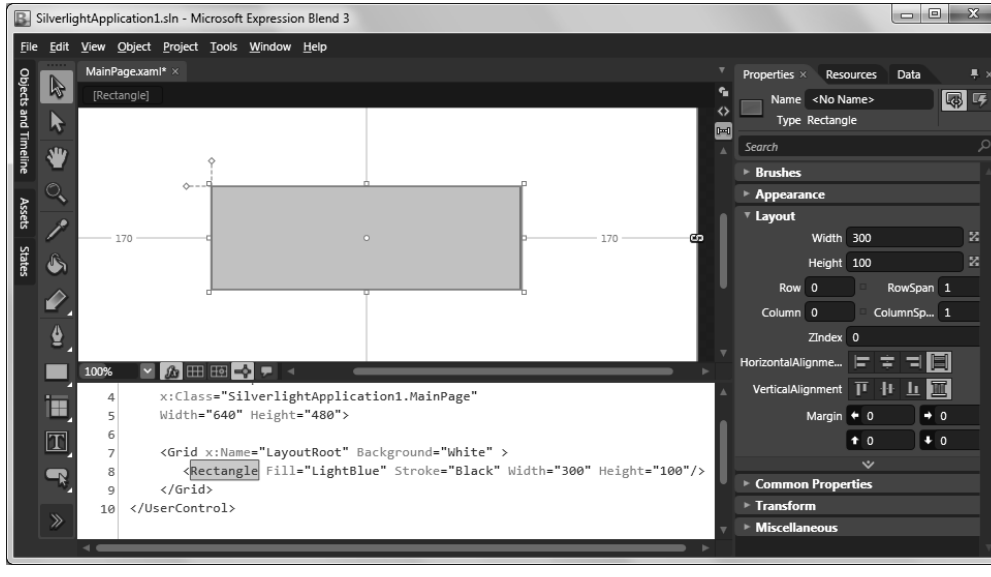
Poznámka

Barvu výplně a barvu obrysu bylo nutné zadat kvůli tomu, abyste dokázali odlišit obdélník od plochy aplikace. V opačném případě by se zobrazil bílý obdélník na bílém pozadí, nedal by se tedy vizuálně identifikovat.

Jako první krok pro přizpůsobení obdélníka můžete zkusit zadat jeho rozměry, tedy šířku a výšku.

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Rectangle Fill="LightBlue" Stroke="Black" Width="300" Height="100"/>
</Grid>
```

Obdélník bude zobrazený s danými rozměry a situovaný do středu plochy kontejneru, a to vodorovně i svisle.



Obrázek 4.2: Zobrazení obdélníka se zadanými rozměry

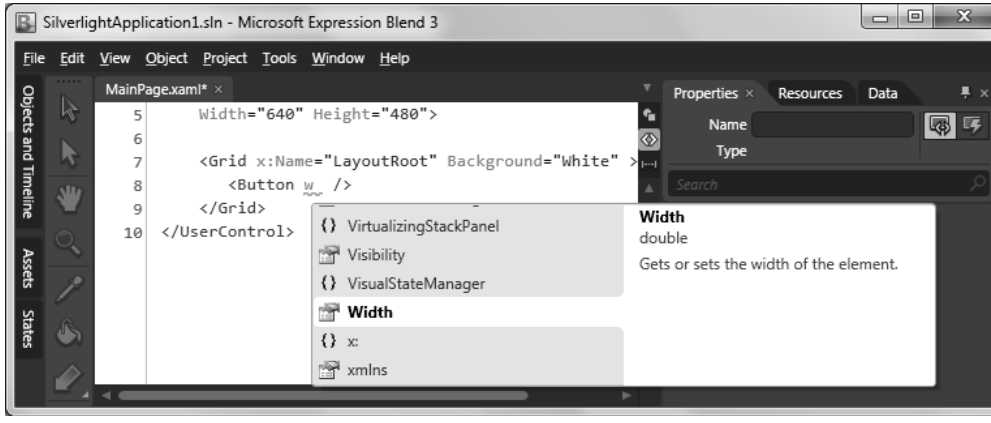
Abychom soustředili vaši pozornost na polohu a rozměry prvku, nahradíme v dalších příkladech obdélník tlačítkem. Tlačítko bez další specifikace je přece obdélník, který má už předdefinovanou barvu výplně a barvu obrysové čáry

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Button Width="300" Height="100"/>
</Grid>
```

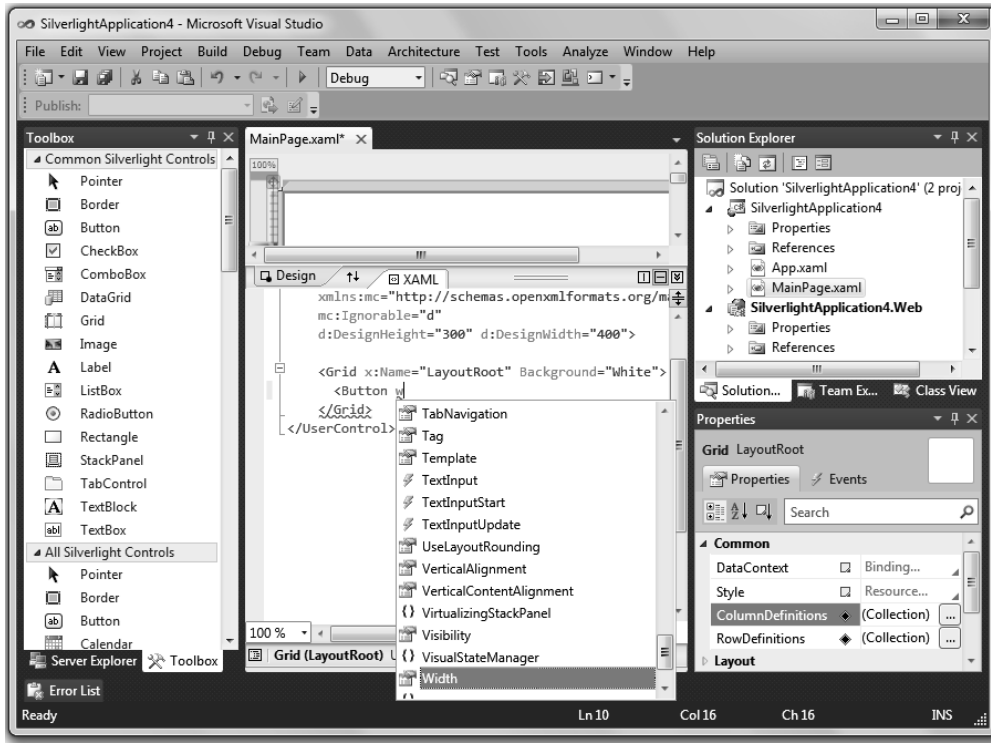
Při zadávání parametrů vám významně pomůže interaktivní nápověda nazvaná **Intellisense**, takže po napsání prvního písmena názvu parametru se vám zobrazí nabídka dostupných parametrů začínajících tímto písmenem. Název parametru takto není potřeba psát, stačí ho vybrat a potvrdit. Kromě názvu parametru se do XAML kódu vloží i znak „=“ a uvozovky označující zatím prázdny řetězec s jeho hodnotou. Například pro parametr šířky bude po napsání prvního písmena W a výběru názvu vložený řetězec:

```
Width=""
```

Interaktivní nápověda nazvaná Intellisense je k dispozici v prostředí Expression Blend i ve Visual Studiu. Její fungování je zřejmé z obrázků 4.3 a 4.4.



Obrázek 4.3: Využití Intellisense jako interaktivní nápovědy v prostředí Expression Blend



Obrázek 4.4: Využití Intellisense jako interaktivní nápovědy ve Visual Studiu 2010

Dalším parametrem, který vstupuje při polohování prvku do hry, je zarovnání, ať už horizontální nebo vertikální.

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Button Width="300" Height="100"
    HorizontalAlignment="Left" />
</Grid>
```

Zarovnání prvku přímo na okraj

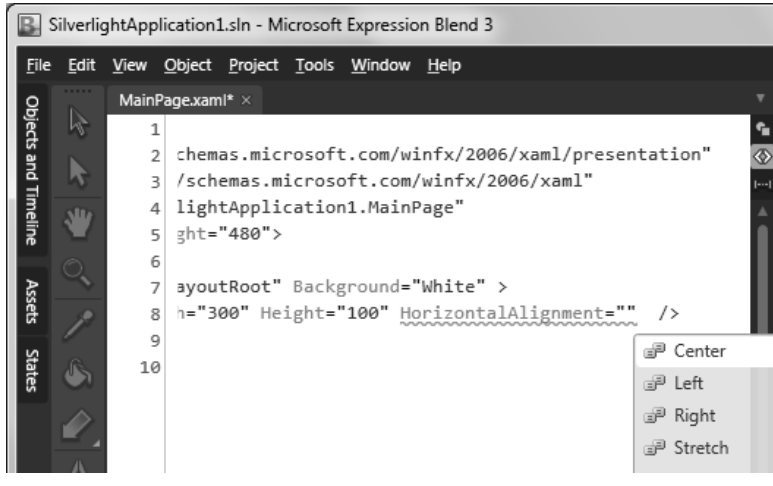
Nejdříve si ukážeme hraniční případy polohování prvku vůči kontejneru, tedy případy, kdy grafický prvek má být umístěný na některý okraj vizuálního kontejneru. Prvek může byt zarovnaný vodorovně, případně svisle. Pro definici zarovnání prvku vůči okraji vizuálního kontejneru slouží parametry:

- `HorizontalAlignment`
- `VerticalAlignment`

Také v tomto případě můžete využít interaktivní nápovědu. Parametr `HorizontalAlignment` může mít jen jednu ze čtyř předdefinovaných hodnot

- `Center`
- `Left`
- `Right`
- `Stretch`

V tomto případě není potřeba hodnotu parametru psát, stačí ji vybrat z nabídnutého seznamu, viz obrázek 4.5.



Obrázek 4.5: Využití interaktivní nápovědy pro předdefinované hodnoty parametrů

Polohu prvků je možné principiálně definovat třemi vzájemně propojenými způsoby:

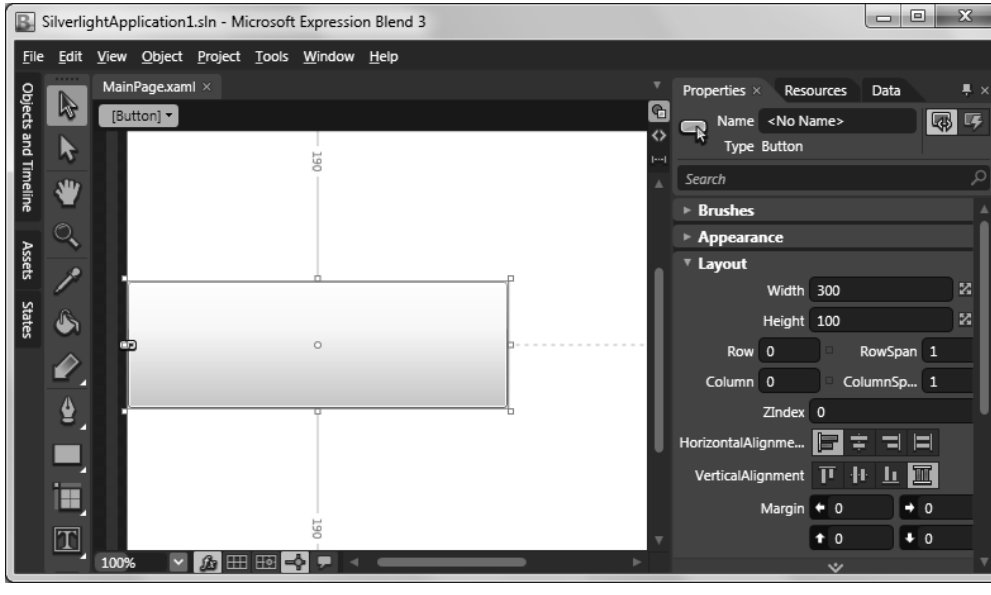
- Vizuálním návrhem, tedy vykreslením prvku v požadované velikosti na požadovanou pozici.
- Pomocí XAML kódu.
- Nastavením parametrů.

První dva způsoby už znáte, takže nastal čas zaměřit vaši pozornost na záložku **Properties** v pravém okně pracovní plochy. Záložka je podle typu vybraného prvku rozdělená na více sekcí. Předmětem našeho zájmu bude momentálně sekce **Layout**.

Pro grafický prvek Button, tedy pro tlačítko, obsahuje tato sekce pole pro zadávání číselných parametrů (obrázek 4.6):

- `Width`
- `Height`
- `RowSpan`
- `ColumnSpan`
- `ZIndex`
- `Margin` (pole pro zadání čtyř okrajů)

Sekce taktéž obsahuje ikony pro nastavení vodorovného (**Horizontal Alignment**) a svislého (**Verticall Alignment**) zarovnávání.



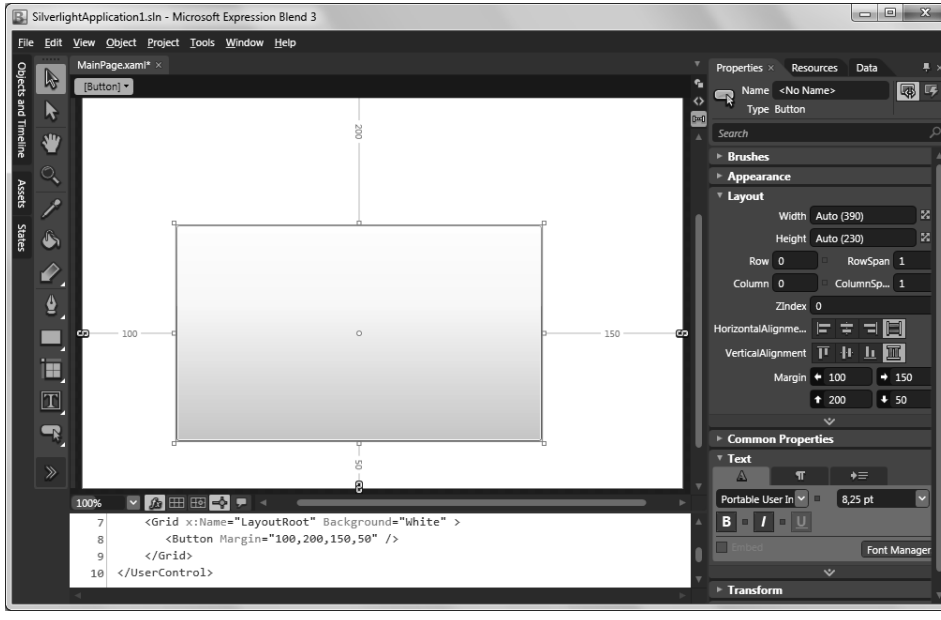
Obrázek 4.6: Skupina parametrů Layout

Jestliže chcete například příslušný prvek umístit do levého dolního rohu vizuálního kontejneru, je potřeba nastavit horizontální i vertikální zarovnávání.

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Button Width="300" Height="100"
    HorizontalAlignment="Left" VerticalAlignment="Bottom" />
</Grid>
```

Zarovnání prvku vůči okrajům kontejneru

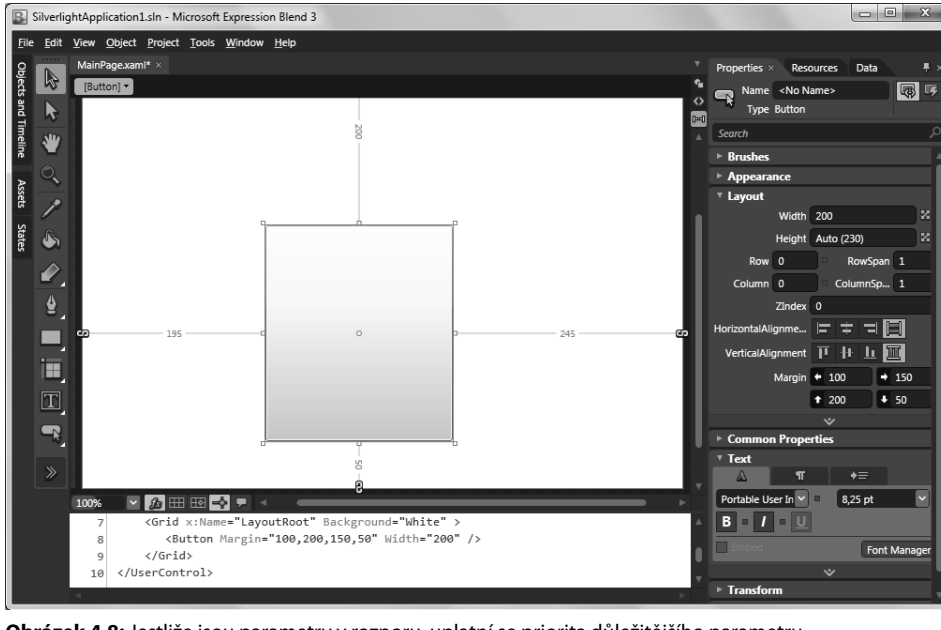
Pomocí parametru `Margin` je možné nastavit všeobecnou polohu prvku vůči okrajům kontejneru. V nejjednodušším případě, kdy má být příslušný prvek, například obrázek, klíčovým prvkem příslušné obrazovky aplikace, není nutné definovat rozměry prvku, stačí pomocí parametru Margin nastavit okraje pro umístění prvku. Podobně může být i tlačítko klíčovým prvkem dialogového okna.



Obrázek 4.7: Nastavení polohy prvku vůči okrajům vizuálního kontejneru

Například:

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Button Margin="100,200,150,50" />
</Grid>
```



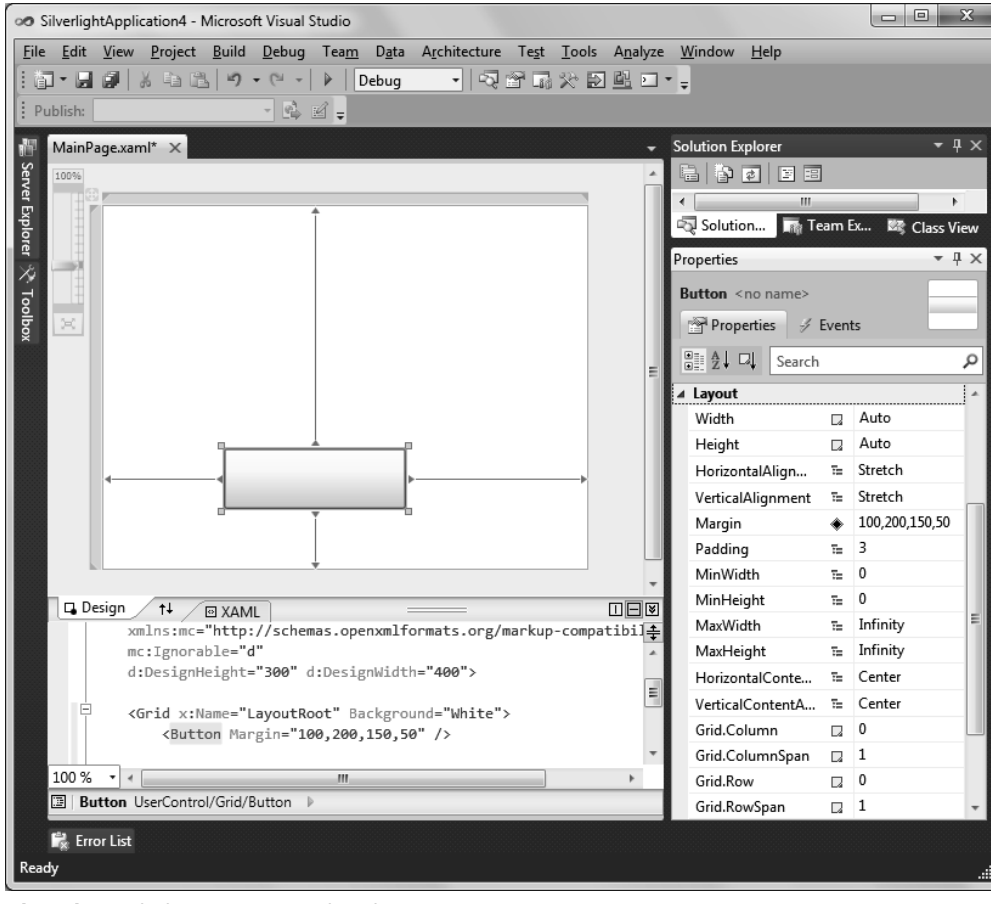
Obrázek 4.8: Jestliže jsou parametry v rozporu, uplatní se priorita důležitějšího parametru

Můžete si vyzkoušet i takové zadání parametrů, které je v rozporu, například jestliže nastavíte zároveň šířku prvku i jeho vodorovné okraje. Rozpor je v tom, že šířka plochy vizuálního kontejneru je 640, což je více než součet obou okrajů a šířky prvku.

```
<Grid x:Name="LayoutRoot" Background="White" >
  <Button Margin="100,200,150,50" Width="200" />
</Grid>
```

V takovém případě vstupuje do hry priorita důležitějšího parametru. Z obrázku 4.8 je zřejmé, že prioritu má v tomto případě šířka prvku a musely se přizpůsobit okraje.

Stejné složky okna Properties má jako Expression Blend i Visual Studio. Z obrázku 4.9 je zřejmé, že parametry jsou zobrazené jen číselně bez grafických symbolů a nastavovacích prvků.



Obrázek 4.9: Složka Layout ve Visual Studiu

Rozmísťování prvků do mřížky – objekt Grid

Kontejnerový objekt `Grid` je definovaný jako mřížka vytvořená z řádků a sloupců. Umožňuje flexibilní uspořádání ostatních prvků. Velkou výhodou je i možnost zadávat rozměry mřížky nejen jako

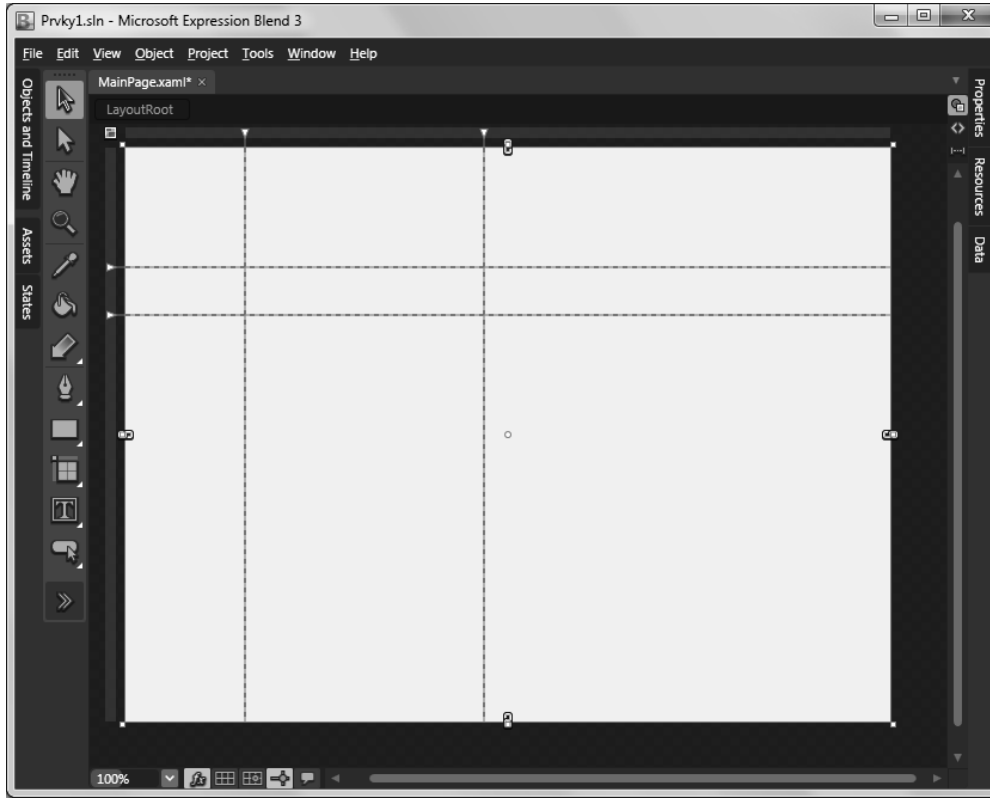
konkrétní hodnoty v pixelech, ale i relativně, případně automaticky podle obsahu. Pro tento účel se využívá takzvaná hvězdičková konvence definování rozměrů (v anglické terminologii Star sizing).

Jak vyplývá z názvu prvku `Grid` (česky mřížka), je možné tento prvek rozdělit na více polí a do těchto polí umístit prvky, u kterých se požaduje určitá vzájemná pozice. Typickým příkladem je třeba pole pro zadání hodnoty a před ním textové pole, které vysvětluje jeho význam. Tyto dva prvky je potřeba udržovat vždy v takové poloze, aby textové pole bylo vždy vlevo od pole při zadávání hodnoty.

Do hry tedy vstupují řádky a sloupce mřížky. Pro definici řádků se využívá element `Grid.RowDefinitions`, který je uvnitř elementu `Grid`. Konkrétně řádky se definují pomocí elementu `RowDefintion`. Podobné je to i u sloupců. Sekce sloupců se definuje pomocí elementu `Grid.ColumnDefinitions` a konkrétně sloupce přes elementy `ColumnDefinition`.

Mřížka s pevně definovanými rozměry

Ukážeme příklad vytvoření jednoduché mřížky s napevno definovanými rozměry. Aby byl příklad názorný, je parametr `ShowGridLine` nastavený na hodnotu `true`, takže mřížka se i vizuálně zobrazí. Výsledek návrhu je na obrázku 4.10.



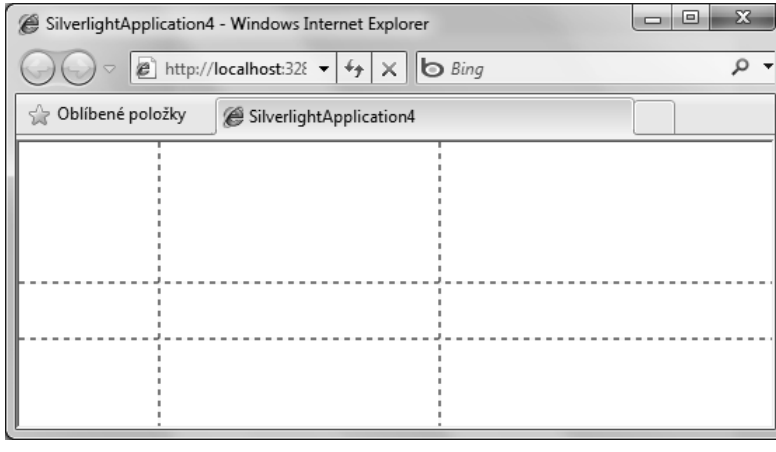
Obrázek 4.10: Definování řádků a sloupců prvku Grid

```
<Grid x:Name="LayoutRoot" Background="Beige" ShowGridLines="True" >
  <Grid.RowDefinitions>
    <RowDefinition Height="100"/>
    <RowDefinition Height="40"/>
    <RowDefinition Height="200"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100" />
    <ColumnDefinition Width="200"/>
    <ColumnDefinition Width="400"/>
  </Grid.ColumnDefinitions>
</Grid>
```

Poznámka

Parametr ShowGridLine je implicitně nastavený na hodnotu false, protože v reálných aplikacích není zpravidla potřeba mřížku zobrazovat. Je neviditelná a její hlavní úlohou je udržování vzájemné polohy prvků.

Přestože tato aplikace obsahující jen základní mřížku je tak jednoduchá, že ani nemá význam ji spouštět, doporučujeme vám, abyste si aplikaci spustili, následně měnili velikost okna prohlížeče a sledovali chování mřížky. V tomto případě zůstávají rozměry mřížky stále stejné, při zmenšování okna se prostě ořízne.



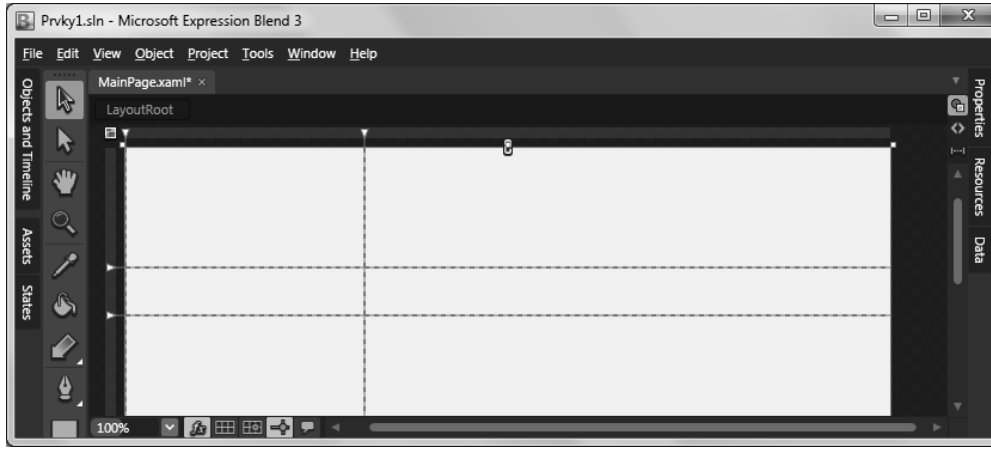
Obrázek 4.11: Rozměry mřížky zůstávají stejné i při změně velikosti okna aplikace v prohlížeči

Mřížka s flexibilně definovanými rozměry

Námětem dalšího příkladu bude rozdělení mřížky na tři sloupce. První má nastavenou šířku tak, aby se přizpůsobila obsahu. Druhý ji má pevně zadanou a třetí je roztažený na zbytek šířky prvku `Grid`. Topologii námětu přibližuje obrázek 4.12. Rozložení řádků zůstane nezměněné, prvky se budou později ukládat do středního řádku s výškou 40.

```
<Grid x:Name="LayoutRoot" Background="Beige" ShowGridLines="True" >
  <Grid.RowDefinitions>
    <RowDefinition Height="100"/>
    <RowDefinition Height="40"/>
```

```
    <RowDefinition Height="200"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="auto" />
    <ColumnDefinition Width="200"/>
    <ColumnDefinition Width="*"/>
  </Grid.ColumnDefinitions>
</Grid>
```



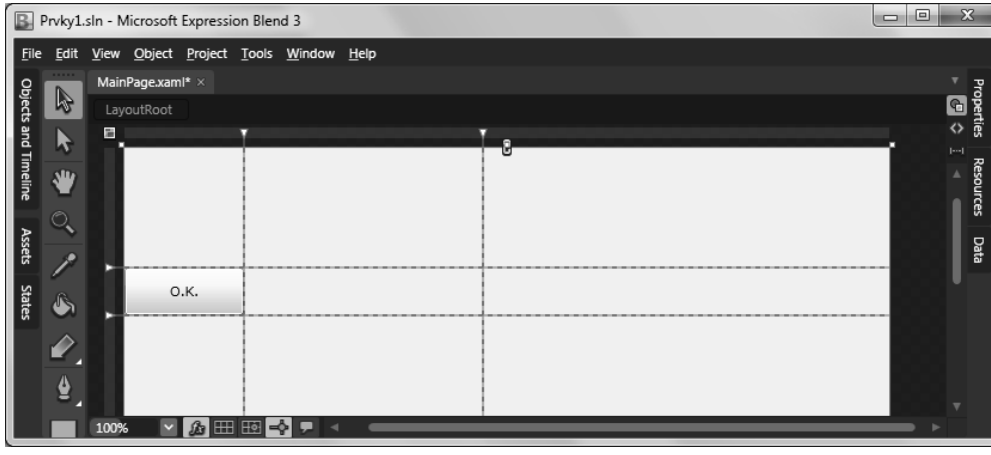
Obrázek 4.12: Mřížka s flexibilně definovanými rozměry sloupců

Určitě jste při pohledu na zobrazení mřížky v návrhovém prostředí postřehli, že tu na první pohled něco nehraje. Definovali jste tři sloupce, ale zobrazili se jen dva. Je to však jen na první pohled, při pozornějším zkoumání objevíte přerušovanou čáru mřížky na levém svislém okraji.

Stalo se přesně to, co jste definovali. V zadání bylo, že první má mít nastavenou šířku tak, aby se přizpůsobila obsahu. A v tomto případě tam není žádný obsah, takže první sloupec má nulovou délku. Abyste se přesvědčili o fungování tohoto typu flexibilního definování rozměrů mřížky, je potřeba do jednotlivých polí mřížky umístit nějaké vizuální prvky. Zatím stačí umístit do prvního sloupce druhého řádku nějaký prvek, například tlačítko, a definovat pro něj šířku (obrázek 4.13).

Umístění prvku do mřížky se určuje pomocí parametrů `Grid.Row` a `Grid.Column`.

```
<Grid x:Name="LayoutRoot" Background="Beige" ShowGridLines="True" >
  <Grid.RowDefinitions>
    <RowDefinition Height="100"/>
    <RowDefinition Height="40"/>
    <RowDefinition Height="200"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="auto" />
    <ColumnDefinition Width="200"/>
    <ColumnDefinition Width="*"/>
  </Grid.ColumnDefinitions>
  <Button Grid.Row="1" Grid.Column="0" Width="100" Content="O.K."  />
</Grid>
```



Obrázek 4.13: První sloupec mřížky se přizpůsobil rozměrům prvku, který je v něm umístěn

Nyní je už všechno podle zadání. První sloupec se přizpůsobil prvku, který je v něm umístěn, druhý sloupec má šířku pevně definovanou a třetí sloupec vyplní zbytek plochy mřížky.

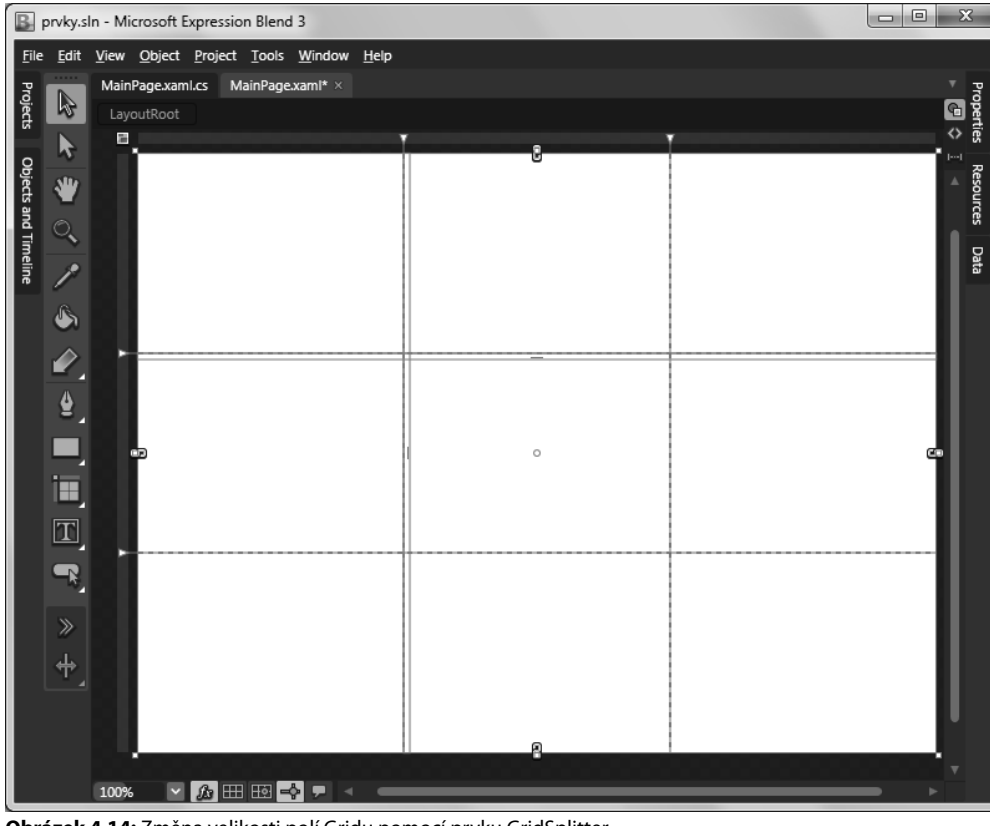
Prvek GridSplitter pro změnu velikosti polí Gridu

Prvek `GridSplitter` umožňuje uživateli podle potřeby změnit velikosti jednotlivých polí Gridu. V příkladu je vodorovné a svislé rozdělení Gridu na 3 řádky × 3 sloupce, přičemž pomocí prvků `GridSplitter` je možné měnit výšku prvního řádku a šířku prvého sloupce.

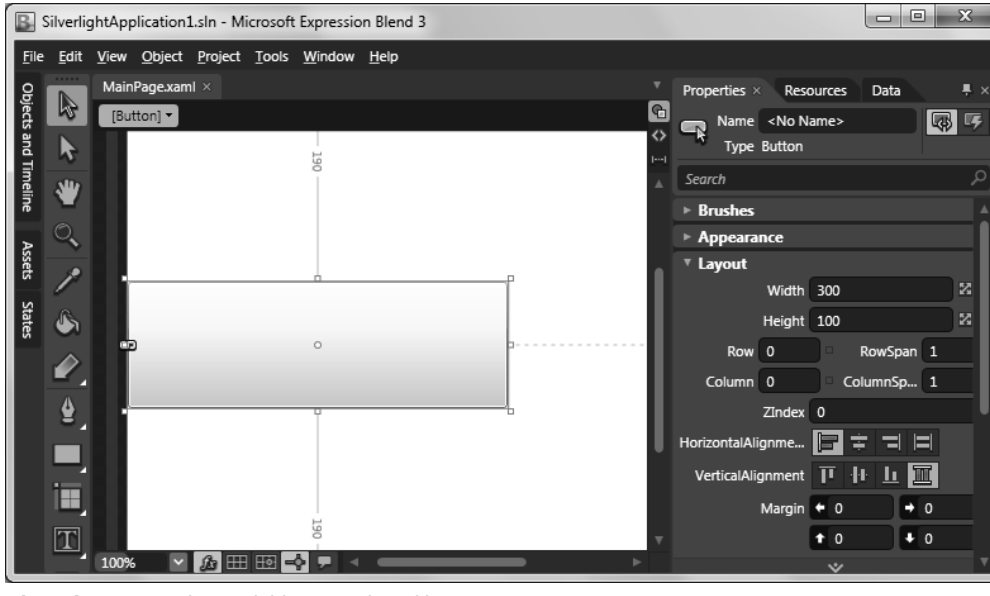
```
<Grid x:Name="LayoutRoot" Background="White" ShowGridLines="True">
  <Grid.RowDefinitions>
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition />
    <ColumnDefinition />
    <ColumnDefinition />
  </Grid.ColumnDefinitions>

  <controls:GridSplitter Grid.Row="0" Grid.Column="1"
    Width="5" Grid.RowSpan="3"
    HorizontalAlignment="Left" VerticalAlignment="Stretch"/>
  <controls:GridSplitter Grid.Row="1" Grid.Column="0"
    Height="5" Grid.ColumnSpan="3"
    HorizontalAlignment="Stretch" VerticalAlignment="Top"/>

</Grid>
```

Vizuální návrh mřížky v prostředí Expression Blend

Souvislosti ohledně pevného a flexibilního členění prvku Grid nejlépe pochopíte při vizuálním návrhu mřížky. Vytvořte v prostředí Expression Blend nový projekt a všimněte si malé ikonky ovládacích prvků v okně pro design aplikace. Na obrázku 4.15 jsou tyto prvky pro lepší názornost očíslované.



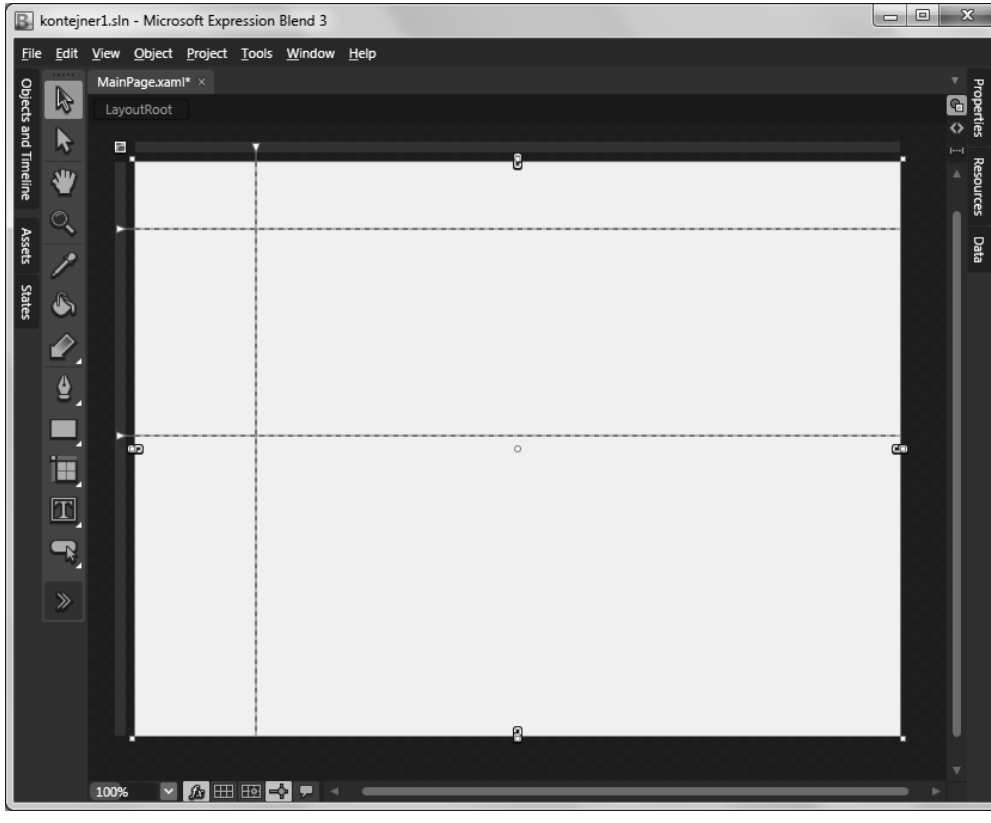
Obrázek 4.14: Změna velikosti polí Gridu pomocí prvku GridSplitter



Obrázek 4.15: Vizuální návrh hlavní mřížky aplikace

Jestliže v prostředí Expression Blend umístíte kurzor na vodorovný nebo svislý rámeček hlavního Gridu, zobrazí se vodorovné nebo svislé žluté pravítko. Pomocí posouvání a umísťování pravítek můžete rozdělit Grid kontejner pro základní pracovní plochu aplikace na několik oblastí, a to vodorovně nebo svisle. Přesněji řečeno, tímto postupem rozdělujete mřížku na řádky a sloupce.

Ukážeme příklad tabulky se záhlavím a řádky, jaké se často používají na webových stránkách. Všimněte si, že rozměry, které se budou přizpůsobovat například velikosti stránky, jsou definované pomocí znaku „*“ (hvězdička).



Obrázek 4.16: Interaktivní návrh mřížky v prostředí Expression Blend

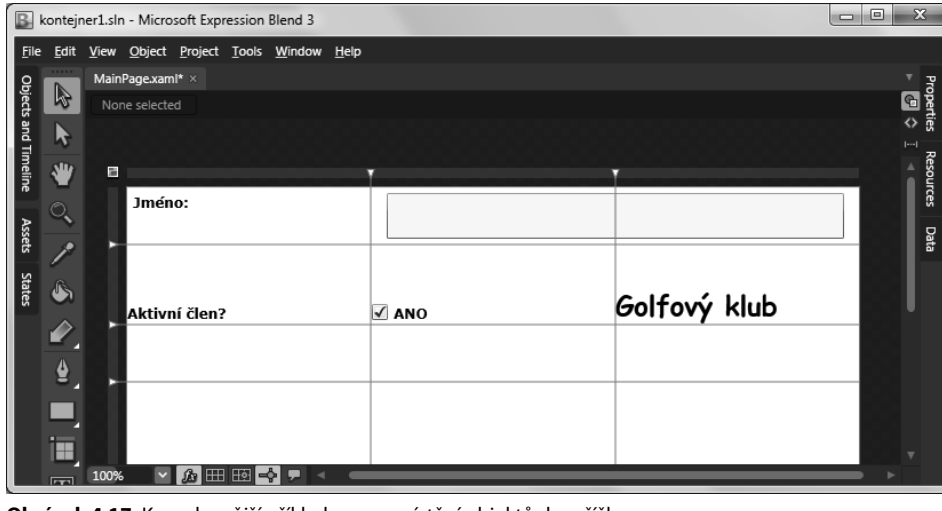
V našem případě byl výsledkem vizuálního návrhu kód:

```
<Grid x:Name="LayoutRoot" Background="White">
  <Grid.RowDefinitions>
    <RowDefinition Height="0.117*"/>
    <RowDefinition Height="0.36*"/>
    <RowDefinition Height="0.523*"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="0.158*"/>
    <ColumnDefinition Width="0.842*"/>
  </Grid.ColumnDefinitions>
</Grid>
```

Přepínáním ikony v levém horním rohu se dá měnit absolutní a relativní rozmístění buněk. Pomocí ikon visacích zámků můžete „zamknout“ fixní nastavení rozměrů vybrané oblasti, je potřeba nastavit parametr `ShowGridLines="true"`.

V dalším, trochu komplexnějším příkladu bude vytvořena mřížka obsahující i řádky s flexibilní výškou a bude ukázáno, jak se do tabulky umísťují ovládací prvky, jejichž poloha je vázaná na polohu a velikost buněk tabulky:

```
<Grid x:Name="LayoutRoot" Background="White">
  <Grid.RowDefinitions>
    <RowDefinition Height="50" />
    <RowDefinition Height="30*" MaxHeight="70"  />
    <RowDefinition Height="*" MinHeight="30" MaxHeight="50" />
    <RowDefinition Height="Auto" MinHeight="5" MaxHeight="30" />
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
    <ColumnDefinition />
    <ColumnDefinition />
    <ColumnDefinition />
  </Grid.ColumnDefinitions>

  <TextBlock Grid.Row="0" Grid.Column="0" FontSize="12"
    FontWeight="Bold" Text ="Jméno:" Margin="5"/>
  <TextBox   Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="2"
    Width="400" Background="Yellow"  Margin="5"  />

  <TextBlock Grid.Row="1" Grid.Column="0" FontSize="12" FontWeight="Bold"
    Text="Aktivní člen?" VerticalAlignment="Bottom" />
  <CheckBox  Grid.Row="1" Grid.Column="1" FontSize="12" FontWeight="Bold"
    Content="ANO" IsChecked="true" VerticalAlignment="Bottom" />
  <TextBlock x:Name="klub" Text="Golfový klub"
    Grid.Row="1" Grid.Column="2" VerticalAlignment="Bottom"
    FontFamily="Comic Sans MS" FontSize="24" FontWeight="Bold"
    Margin="0,20,0,0"/>
</Grid>
```

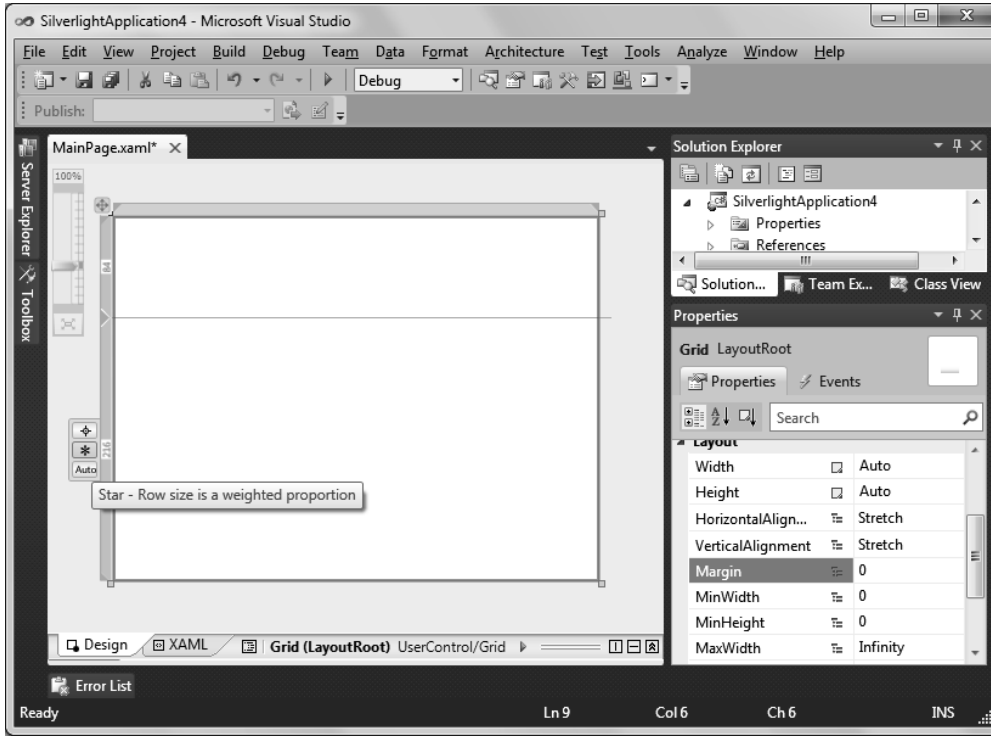


Obrázek 4.17: Komplexnější příklad pro rozmístění objektů do mřížky

Vizuální návrh mřížky v prostředí Visual Studio 2010

Také Visual Studio 2010 poskytuje komfortní možnosti pro návrh mřížky. Po označení tohoto prvku se při jeho levém a horním okraji zobrazí světle modrá pravítka. Posouváním kurzoru myši po těchto okrajových pravítkách definujete dělicí čáry mřížky, a tím vlastně její pole.

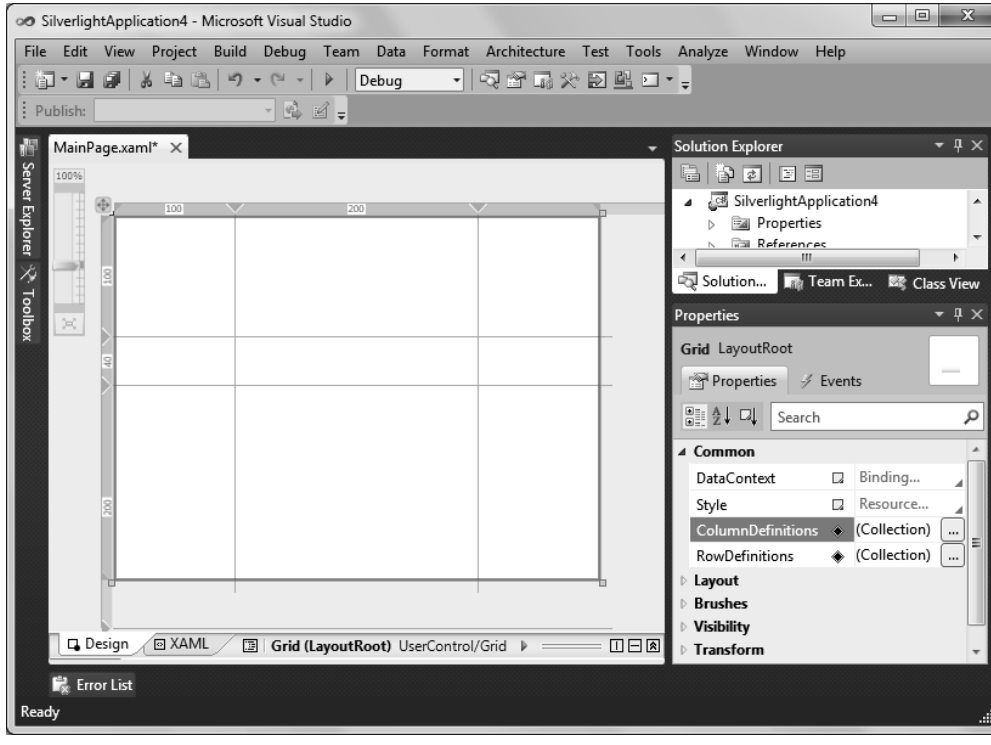
Na obrázku 4.18 je názorně vidět další velmi užitečná vlastnost, kterou Visual Studio 2010 při návrhu mřížky poskytuje. Vedle pravítka se zobrazí tři tlačítka, která se posouvají spolu s kurzorem. Pomocí nich můžete definovat, jestli bude mít příslušný řádek nebo sloupec pevnou, pohyblivou nebo automatickou šířku.



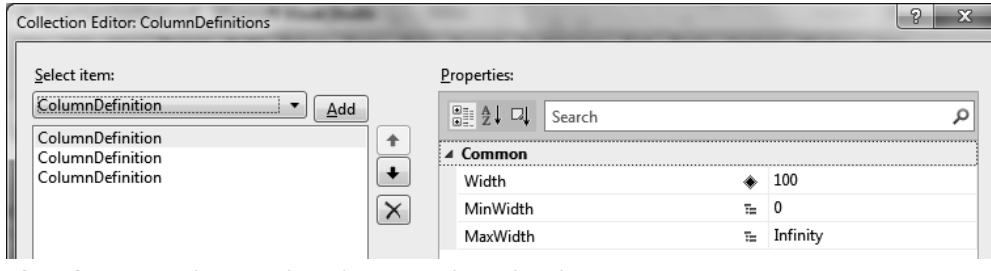
Obrázek 4.18: Interaktivní návrh mřížky v prostředí Visual Studio 2010

Na obrázku 4.19 si všimněte, že na ploše okrajových pravítek se zobrazují i kóty rozměrů příslušných řádků a sloupců. Definice řádků a sloupců mřížky jsou uloženy v parametrech `RowDefinitions` a `ColumnDefinitions`.

Kromě vizuálního návrhu je možné parametry řádků a sloupců zadávat prostřednictvím dialogového okna **Collection Editor**. Způsob přidávání řádků, případně sloupců, a definování jejich parametrů je zřejmý z obrázku 4.20.



Obrázek 4.19: Interaktivní návrh mřížky v prostředí Visual Studio 2010



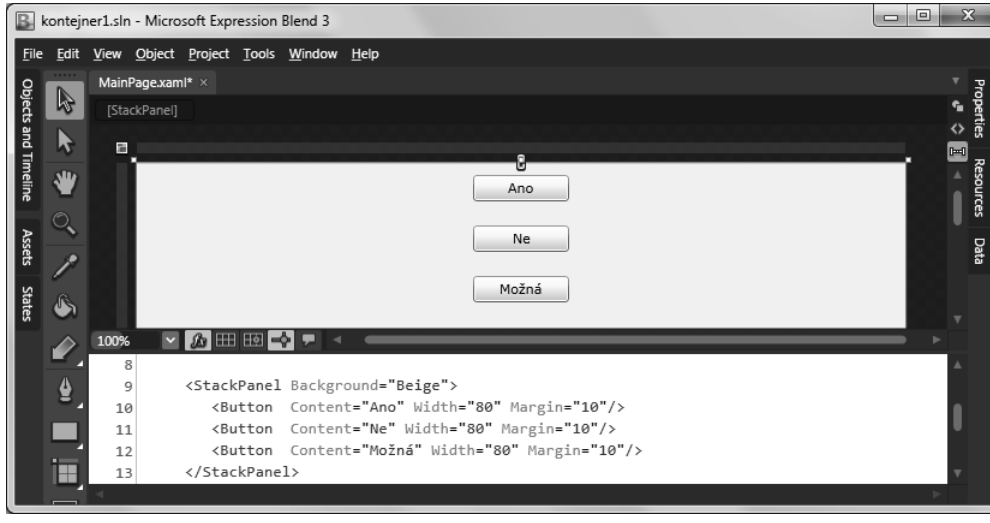
Obrázek 4.20: Interaktivní návrh mřížky v prostředí Visual Studio 2010

Ukládání prvků vodorovně nebo svisle pomocí objektu StackPanel

Kontejner `StackPanel` umísťuje prvky do jedné linie a to buď horizontálně, nebo vertikálně, takže vizuálně to znamená prvky umístěné v řade, nebo prvky umístěné nad sebou. Implicitní orientace uspořádání je vertikální. První nejjednodušší příklad na obrázku 4.21 ukazuje umístění tří tlačítek:

```
<StackPanel Background="Yellow">
  <Button  Content="Ano" Width="80" Margin="10"/>
  <Button  Content="Ne" Width="80" Margin="10"/>
```

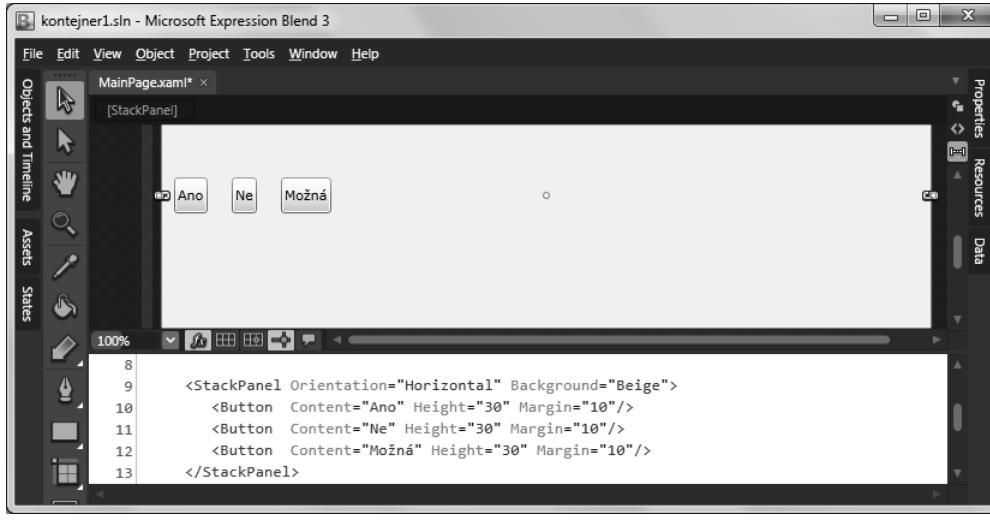
```
  <Button  Content="Možná" Width="80" Margin="10"/>
</StackPanel>
```



Obrázek 4.21: Objekt StackPanel

Jestliže změníte orientaci na horizontální, změní se uspořádání objektů zapouzdřených v StackPanelu tak, jak ukazuje obrázek 4.22.

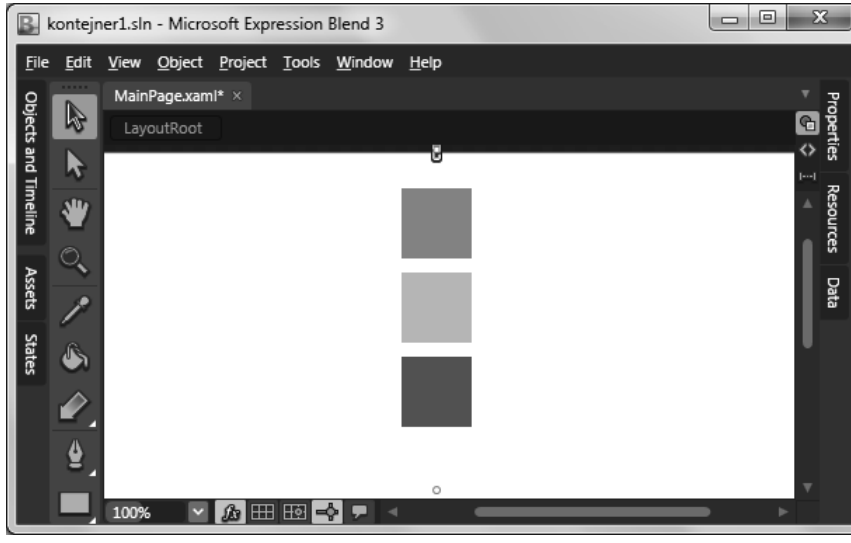
```
<StackPanel Orientation="Horizontal" Background="Beige">
  <Button  Content="Ano" Height="30" Margin="10"/>
  <Button  Content="Ne" Height="30" Margin="10"/>
  <Button  Content="Možná" Height="30" Margin="10"/>
</StackPanel>
```



Obrázek 4.22: Objekt StackPanel – horizontální orientace zapouzdřených prvků

Ukážeme ještě jeden, dalo by se říct, klasický příklad svislého uspořádání prvků – semafor (obrázek 4.23). Jednotlivá barevná pole budou vytvořena pomocí prvku `Rectangle`, tedy obdélníka.

```
<StackPanel Margin="20">
  <Rectangle Fill="Red" Width="50" Height="50" Margin="5" />
  <Rectangle Fill="Orange" Width="50" Height="50" Margin="5" />
  <Rectangle Fill="Green" Width="50" Height="50" Margin="5" />
</StackPanel>
```



Obrázek 4.23: Semafor vytvořený pomocí objektu StackPanel

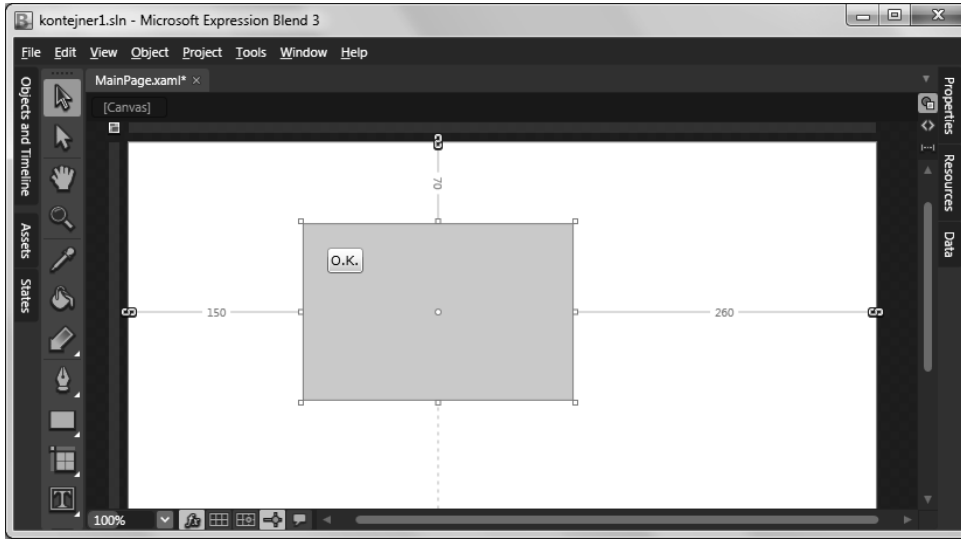
Umístění prvků do souřadnicového systému objektu Canvas

Silverlight aplikace, kde je potřeba rozmístit prvky na přesně definované místo, využívají objektový model pracovní plochy s názvem `Canvas`. Překlad tohoto pojmu do češtiny je poměrně výstižný, je to virtuální malířské, případně promítací plátno. Na tuto plochu se potom umísťují grafické objekty. Kontejnerový objekt `Canvas` definuje oblast, ve které se dají elementy umísťovat na absolutní pozice. Ideálně se hodí k přímému kreslení grafických prvků.

U grafických objektů můžete stanovit jejich polohu. Parametr `Canvas.Left` určuje vzdálenost od levého okraje prvku. Druhý parametr `Canvas.Top` zase určuje vzdálenost od horního okraje. Ve fragmentu kódu je definice polohy tlačítka vůči ploše `Canvas`.

```
<Canvas Height="150" Margin="150,70,260,0" VerticalAlignment="Top"
  Background="LightGray">
  <Button Canvas.Left="20" Canvas.Top="20" Content="O.K." ></Button>
</Canvas>
```

Všimněte si v kódu i na obrázku 4.24, že poloha objektu `Canvas` je vymezená vůči kontejneru `Grid`.



Obrázek 4.24: Relativní poloha plochy geometrického obrazce vůči ploše Canvas

Canvas jako LayoutRoot kontejner

Abychom byli přesnější, mohou nastat dva případy umístění kontejneru Canvas. Buď ponecháte kořenový vizuální LayoutRoot kontejner `Grid` a `Canvas` umístíte do něj, případně do vhodného pole mřížky.

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="kontejner1.MainPage"
  Width="640" Height="480">

  <Grid x:Name="LayoutRoot" Background="White">
    <Canvas Height="150" Margin="150,70,260,0"
      VerticalAlignment="Top" Background="LightGray">
      <Button Canvas.Left="20" Canvas.Top="20" Content="O.K." >
      </Button>
    </Canvas>
  </Grid>
</UserControl>
```

Nebo jako LayoutRoot kontejner můžete definovat přímo `Canvas`, což je potřeba deklarovat v jeho elementu pomocí atributu `x:Name="LayoutRoot"`:

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="kontejner1.MainPage"
  Width="640" Height="480">

  <Canvas x:Name="LayoutRoot" Background="LightGray">
    <Button Canvas.Left="20" Canvas.Top="20" Content="O.K." >
    </Button>
```