Tvorba internetových aplikací pomocí technologie Microsoft Silverlight

Bakalářská práce

Jiří Kolda

Vedoucí bakalářské práce: PaedDr. Petr Pexa
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky

2009

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Anotace

Úkolem této práce je vytvořit první kompletní česky psanou referenční příručku k internetové technologii Microsoft Silverlight. Ve výkladu je vysvětlována současná verze Silverlight 2.0 ve spojení s programovacím jazykem Visual C# v Microsoft .NET Framework 3.5. Součástí práce je stručné porovnání technologie Microsoft Silverlight s technologií Adobe Flash a otestování této technologie v nejrozšířenějších webových prohlížečích.

Abstract

A goal of this work is to present the first complete refence manual for Microsoft Silverlight technology written in czech language. This essay is displaying the present-day version of Silverlight - Silverlight 2.0 along with programming language Visual C# in Microsoft.Net Framework 3.5. This paper includes a brief comparison of Microsoft Silverlight and Adobe Flash and a functionality test of the technology using the most wide-spreaded web browsers.

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce PaedDr. Petru Pexovi za to, že mi umožnil zabývat se touto problematikou, za jeho odborné vedení a za jeho cenné rady a připomínky během konzultací.

Také bych rád poděkoval své přítelkyni a rodině za podporu a pomoc se zpracováním mé práce.

Obsah

1 Úvod

K výběru tohoto tématu jsem měl několik důvodů. Prvním a myslím že nejzásadnějším důvodem pro mě byla skutečnost, že technologie Silverlight je prezentována jako konkurence technologii Adobe Flash. Jelikož se technologií Flash již dlouhou dobu zabývám, zajímalo mě, zda má Silverlight jako konkurent co nabídnout.

Druhou věcí, která mě velmi motivovala k výběru této práce, byl fakt, že Silverlight je .NET technologií. Platforma .NET je v dnešní době velmi rozšířená a hojně používaná z důvodu své velmi dobře propracované a detailní dokumentace a také možnosti výběru jednoho z hned několika programovacích jazyků, z nichž já osobně jednoznačně volím jazyk C# především kvůli jeho přehlednosti.

2 Cíle práce

Hlavním cílem mé práce je bylo sestavit první česky psanou referenční příručku popisující vytváření webových aplikací pomocí technologie Silverlight. Samozřejmě se nejedná o kompletní dokumentaci této technologie, jelikož ta je již přístupná na internetových stránkách MSDN společnosti Microsoft. Tato publikace si dává za cíl především vysvětlit základní principy, na kterých je technologie Silverlight postavena a popsat důležité prvky, které se ve spojení s touto technologii nejčastěji používají.

Součástí práce je také pět ukázkových projektů, na kterých jsou názorně ukázány možnosti technologie a postupy při řešení nejčastějších problémů, na které začínající programátor narazí. Tyto projekty jsou volně přístupné na webových stránkách vytvořených právě za účelem prezentace technologie Silverlight.

3 Současný stav problematiky

O technologii Silverlight již byla sepsána celá řada publikací, článků, online návodů a na oficiálních stránkách technologie se nachází také celá řada screencastů popisujících mnoho jejích funkcí. Bohužel však kromě jedné výjimky[13] (která se ovšem nezabývá přímo touto technologií a uvádí ji pouze v závěrečné kapitole) nevyšla na trh žádná česky psaná publikace zabývající se touto problematikou.

Při tvorbě této publikace jsem vycházel právě z anglicky psaných návodů a screencastů přístupných na webu. Přečetl jsem celou řadu návodů jak pro začátečníky, tak pro pokročilejší programátory, seznámil jsem se velmi podrobně s programovacím jazykem XAML a technologiemi Windows Presentation Foundation (WPF) a Windows Communication Foundation (WCF) a prostudoval jsem velkou část dokumentace na internetových stránkách MSDN.

Společnost Microsoft tvrdí, že technologii Silverlight používá již asi 300 tis. uživatelů na celém světě. Pravdou je, že počet Silverlight aplikací velmi rychle roste, objevilo se již i několik her, otázkou však zůstává, kolik vývojářů na tuto technologii již definitivně přešlo a pracuje v ní a kolik jich Silverlight pouze zkusilo a vrátilo se zpět k tomu co dělali dříve. Objevují se také názory, že k tak rychlému rozšíření technologie Silverlight došlo jen díky obrovské reklamní kampaní ze strany Microsoft a že si Silverlight takovou popularitu nezaslouží. Osobně si myslím, že tato technologie má zcela jistě co nabídnout, ale rozhodně to v konkurenčním boji se zaběhlou technologií Adobe Flash nebude mít lehké.

4 Metodika zpracování práce

Abych byl vůbec schopen tuto práci sepsat, bylo nejprve třeba se plně seznámit s celou technologií Silverlight, jejími principy a možnostmi, které nabízí. Za tímto účelem jsem přečetl celou řadu návodů, článků a publikací, shlédl jsem mnoho screencastů popisujících jednotlivá vzorová řešení a nastudoval jsem velkou část dokumentace na webu MSDN společnosti Microsoft.

Po nastudování všech potřebných nových znalostí jsem začal vytvářet jednoduché aplikace, které mi posloužily k pochopení zákonitostí a principů, na kterých je technologie postavena. V těchto aplikacích jsem experimentoval s jednotlivými ovládacími prvky, jejich grafickými a logickými možnostmi a postupně jsem tak získal přehled o tom, k čemu jednotlivé prvky slouží, jak je s nimi možné manipulovat a kterými cestami je asi nejpraktičtější řešit různé problémy.

Po této sérii experimentů jsem začal vytvářet praktickou část této práce. Vytvořil jsem pět vzorových projektů, z nichž každý názorně ukazuje určitou schopnost technologie Silverlight, ať už se jedná o grafické prvky, animaci či načítaní dat z databáze. Při výběru jednotlivých vzorových řešení jsem dbal na to, aby v nich byly obsaženy ty nejzákladnější principy, se kterými se začínající programátor setká nejčastěji. Všechny projekty jsem konzultoval s vedoucím mé práce PaedDr. Petrem Pexou a shodli jsme se na jejich výběru. Stejná kritéria jsem posléze aplikoval při výběru jednotlivých elementů jazyka XAML popisovaných v písemné části této práce.

Ke tvorbě praktického řešení a pro ukázky zdrojových kódů jsem použil nástroj Visual Studio Team System 2008 společnosti Microsoft. V práci je také popisován nástroj Microsoft Blend 2, který jsem rovněž během vypracovávání práce vyzkoušel.

Technologii Silverlight jsem otestoval v několika nejrozšířenějších prohlížečích. Při výběru jednotlivých prohlížečů jsem postupoval následujícím

způsobem. Na internetu jsem vyhledal servery zabývající se recenzemi softwarových produktů a na základě hodnocení a statistických údajů o rozšířenosti a kvalitě různých prohlížečů jsem vybral šest aplikací, o kterých jsem buď předpokládal, že v nich Silverlight bude podporován, nebo jsem měl v úmyslu to zjistit. K výběru také přispěly oficiální informace společnosti Microsoft o podpoře Silverlight.

5 Teoretický úvod

Společnost Microsoft je poslední dobou stále více spojována s internetovými technologiemi. Až donedávna se především jednalo o technologii ASP.NET, jakožto rozšířený a silný nástroj pro tvorbu webových aplikací. S příchodem nových technologií použitých ve Windows Vista, jako jsou Windows Presentation Foundation nebo Windows Communication Foundation, se na trhu objevila nová technologie funkčně vycházející právě z první jmenované. Touto technologií je Microsoft Silverlight.

Technologie Silverlight je společností Microsoft prezentována jako přímý konkurent pro technologii Adobe Flash a jejím hlavním cílem je zaujmout přední místa v použití na webu vedle technologií Flash či AJAX. Silverlight nabízí použití vektorové grafiky, animaci, umožňuje přehrávat videa v HD kvalitě a především vyniká možností volby použitého programovacího jazyka. Toto se týká verze 2.0 a případných dalších verzí, ve verzi 1.0 byla funkcionalita zajištěna jedině Javascriptem.

K zobrazení Silverlight aplikací je třeba mít nainstalovaný malý (asi 4 MB) plugin, který obstarává jeho podporu ve většině nejčastěji používaných prohlížečích. Pracuje se i na verzích pluginu do mobilních zařízení (pro Windows Mobile a Symbian) a existuje také volně šiřitelná implementace pro unixové operační systémy s názvem Moonlight. Velmi pozitivní skutečností je, že k zobrazení Silverlight aplikací není nutné mít nainstalovaný příslušný .NET Framework.

Výhodou technologie je možnost propojení Silverlight aplikace s HTML stránkou, na níž je umístěna. Díky tomu je možné ovládacími prvky v aplikaci měnit obsah HTML stránky a naopak. K vývoji Silverlight aplikací slouží nejčastěji prostředí MS Visual Studio 2008 ve spojení s .Net Framework 3.5, alternativou je pak Microsoft Blend 2. Oba tyto nástroje jsou v této práci popsány a ukázány.

6 Výsledky práce

Součástí této práce (viz. příloha) je první česky psaná ucelená příručka pro začínající programátory v technologii Silverlight. V této publikaci jsou popsány základní mechanizmy a principy technologie, vysvětleny mnohé problémy, se kterými se programátoři setkají a je zde uvedeno i stručné porovnání technologie Silverlight s konkurenční technologií Adobe Flash. Hlavním výstupem mé bakalářské práce je tato publikace.

Součástí mé práce je také otestování technologie v několika nejrozšířenějších prohlížečích. Tomuto výzkumu se věnuje první kapitola publikace, v níž je přehledným způsobem zobrazena podpora Silverlight 2.0.

Praktickou část mé bakalářské práce představuje pětice projektů poukazujících na jednotlivé aspekty technologie Silverlight. Na těchto vzorových projektech jsou názorně ukázány následující prvky používané v této technologii:

- Vektorová grafika a jazyk XAML
- Komponenty umožňující rozvržení objektů na scéně
- Styly a šablony
- Animace
- Napojení na databázi pomocí LINQ

Následující část práce popisuje vzorové projekty, jejich principy a funkcionalitu zdrojových kódů. Před jejím čtením doporučuji prostudovat druhou a třetí kapitolu publikace popisující jazyk XAML a propojení s jazykem C#.

7 Popis praktického řešení

7.1 Charakteristika aplikací

1. Vektorová grafika a XAML
   - *Tento projekt ukazuje použití základních grafických prvků jazyka XAML a jejich možnosti. Na těchto prvcích je názorně předvedeno několik způsobů manipulace s výplněmi či okraji.*

2. Komponenty umožňující rozvržení objektů na scéně
   - *V tomto projektu je čtenář seznámen s několika elementy zajišťujícími rozvržení objektů na scéně. Tyto elementy mohou řadit objekty pod sebou nebo vedle sebe, rozdělovat scénu na části apod.*

3. Styly a šablony
   - *Tato vzorová aplikace představuje využití stylů a šablon pro jednoduchou parametrizaci objektů a definování jejich vizuální podoby.*

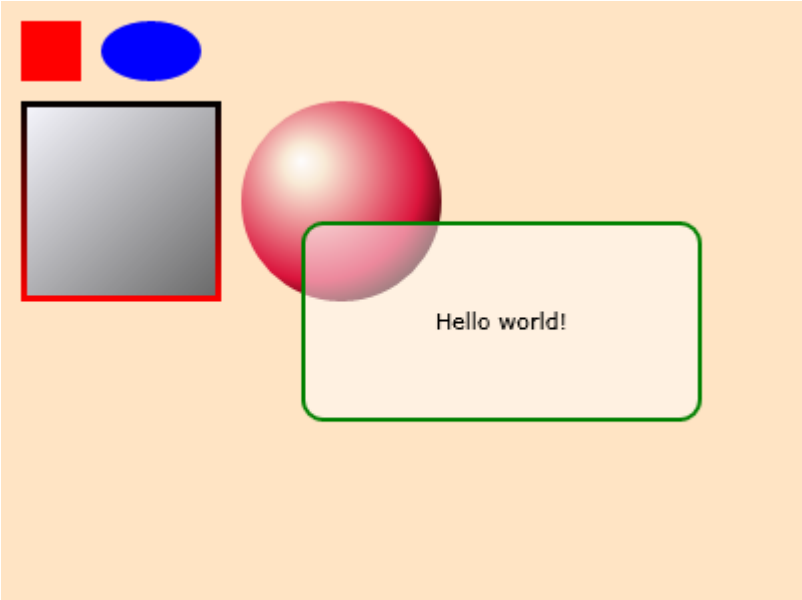
4. Animace
   - *Zde se čtenář seznámí s několika způsoby animace v Silverlight a získá přehled o tom, jaké parametry je možné u XAML elementů animovat.*

5. Napojení na databázi pomocí LINQ
   - *Tento projekt názorně ukazuje propojení Silverlight aplikace s SQL databází pomocí technologie WCF a jazyka LINQ a zobrazení načtených dat v tabulce.*

7.2 Použité technologie

Všechny ukázkové aplikace byly vytvořeny s pomocí technologie Silverlight 2.0. Jako vývojové prostředí jsem použil MS Visual Studio Team System 2008 SP1 pod platformou .NET Framework 3.5. Kompletní zdrojové kódy jsou přiloženy jako přílohy a zdrojové soubory se nacházejí na přiloženém CD.

7.3 Vektorová grafika a jazyk XAML



Tento projekt slouží k ukázání základních grafických možností jazyka XAML. Jsou v něm použity základní grafické elementy jazyka XAML, jako jsou <Rectangle>, <Ellipse> a <Border>. Tyto elementy mají nastavené různé hodnoty "barevných" parametrů (výplň, okraj) a jsou rozmístěny na plochu pomocí pevných souřadnic.

Nejvýše na scéně se nacházejí dva velmi jednoduché elementy objekty <Rectangle> a <Ellipse>, tedy obdélník a elipsa.

```
<Rectangle Height="30" Width="30" Fill="Red" Canvas.Left="10" Canvas.Top="10" />
<Ellipse Height="30" Width="50" Fill="Blue" Canvas.Left="50" Canvas.Top="10" />
```

Z ukázky zdrojového kódu vidíme, že použití oba elementy potřebují k tomuto zobrazení totožné parametry. Oba elementy mají nastavenou výšku a šířku, barvu výplně a umístění na scéně.

Pod těmito dvěma objekty jsou již složitější prvky. Opět se jedná o elementy <Rectangle> a <Ellipse>, tentokrát však mají definovanou složitější grafickou podobu. Následující zdrojový kód popisuje čtverec nalevo.

```
<Rectangle StrokeThickness="3" Height="100" Width="100"
           Canvas.Left="10" Canvas.Top="50" >
    <Rectangle.Fill>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
            <GradientStop Color="GhostWhite" Offset="0"/>
            <GradientStop Color="DimGray" Offset="1" />
        </LinearGradientBrush>
    </Rectangle.Fill>
    <Rectangle.Stroke>
        <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
            <GradientStop Color="Red" Offset="1"/>
            <GradientStop Color="Black" Offset="0" />
        </LinearGradientBrush>
    </Rectangle.Stroke>
</Rectangle>
```

Narozdíl od výše zmiňovaného obdélníku má tento objekt definovánu lineárně interpolovanou výplň a obrys. Na této ukázce je důležité představení elementu <LinearGradientBrush>, který zajišťuje lineární interpolaci barev. Stěžejními parametry tohoto elementu jsou "počáteční" a "koncový" bod, jejichž hodnoty jsou vždy dvojice reálných čísel vyjadřujících relativní pozici na vodorovné a svislé ose v rozmezí od nuly do jedné. Jednotlivé barvy jsou definovány jako subelementy <GradientStop> s parametry "Color" a "Offset", jejichž hodnoty udávají příslušnou barvu a pozici na spojnici počátečního a koncového bodu. Pozice na spojnici bodů je opět definována jako reálné číslo v rozmezí od nuly do jedné.

Na elipse nacházející se vedle tohoto obdélníka je představen další možný styl výplně - <RadialGradientBrush>.

```
<Ellipse Height="100" Width="100" Canvas.Left="120" Canvas.Top="50" >
    <Ellipse.Fill>
        <RadialGradientBrush Center="0.4,0.4" RadiusX="0.7" RadiusY="0.7"
                             GradientOrigin="0.3,0.3">
            <GradientStop Color="White" Offset="0" />
            <GradientStop Color="AntiqueWhite" Offset="0.15" />
            <GradientStop Color="Crimson" Offset="0.75" />
            <GradientStop Color="#FF330000" Offset="0.95" />
        </RadialGradientBrush>
    </Ellipse.Fill>
</Ellipse>
```

Element <RadialGradientBrush> obstarává radiální interpolaci barev a od výše zmiňovaného elementu pro lineární interpolaci se liší svými parametry. Zřejmé jsou jistě parametry "RadiusX" a "RadiusY" představující zploštění či zúžení gradientu. Hodnota parametru "GradientOrigin" představuje bod, ze kterého gradient vychází a parametr "Center" uchovává bod, k němuž gradient směřuje. Jednotlivé barvy jsou opět definovány subelementy <GradientStop>. Na čtvrtém subelementu <GradientStop> je ukázáno použití hexadecimálního zápisu barvy ve formátu RGBA.

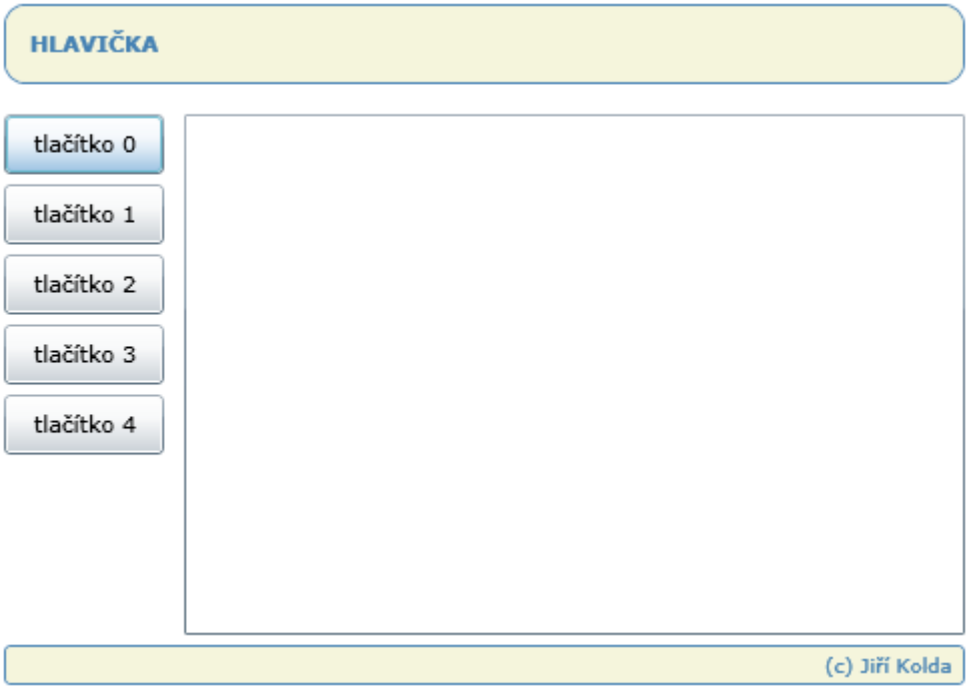
Posledním grafickým prvkem na scéně je objekt <Border>. Ten je zde nadefinován tak, že jeho výplň je poloprůhledná a jeho plocha částečně překrývá poslední zmiňovanou elipsu.

```
<Border Canvas.Left="150" Canvas.Top="110" Width="200" Height="100"
        BorderBrush="Green" BorderThickness="2" CornerRadius="10" >
    <Border.Background>
        <SolidColorBrush Color="White" Opacity="0.5" />
    </Border.Background>
    <TextBlock Text="Hello world!" HorizontalAlignment="Center"
               VerticalAlignment="Center"/>
</Border>
```

Na zdrojovém kódu vidíme, že parametry elementu <Border> jsou podobné elementům <Rectangle> a <Ellipse>. Je zde také použit parametr "CornerRadius", který udává zaoblení rohů. Průhlednost pozadí je definována pomocí elementu <SolidColorBrush> a jeho parametru "Opacity", který je nastaven na hodnotu 0.5. Uvnitř elementu <Border> se nachází ještě objekt <TextBlock> představující statické textové pole a zobrazující nápis "Hello

World!". Z celé struktury je patrná funkce elementu <Border> jako "ohraničení", které může obsahovat další objekty.

7.4 Komponenty zajišťující rozvržení objektů na scéně



V tomto projektu se blíže zabývám tzv. Layout komponenty, což jsou prvky jazyka XAML, které umožňují rozmisťovat a uspořádávat objekty na scéně.

Scéna je rozdělena na několik částí dvěma elementy <Grid>. První (s názvem "rows") rozděluje scénu na tři řádky a v jeho druhém řádku se nachází další ("cols"), vytvářející v tomto řádku dva sloupce:

```
<Grid x:Name="rows" Width="500" Height="350" Background="White"
      ShowGridLines="False">
    <Grid.RowDefinitions>
        <RowDefinition Height="60"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="20"/>
    </Grid.RowDefinitions>

    <Grid x:Name="cols" ShowGridLines="False" Grid.Row="1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
    </Grid>
</Grid>
```

Ze struktury kódu je patrné, že element "cols" je uvnitř elementu "rows", a vidíme, že jeho parametr "Grid.Row" má hodnotu 1, což znamená, že je ve druhém řádku (indexy řádků a sloupců vždy začínají nulou). Stejným principem, tedy zadáním příslušné hodnoty parametru "Grid.Row" či "Grid.Column", je možné každému objektu na scéně určit, kde přesně se bude nacházet.

```
<Border Grid.Row="0" Background="Beige" Height="40" Margin="10,0"
        BorderBrush="SteelBlue" BorderThickness="1" CornerRadius="10">
    <TextBlock Text="HLAVIČKA" VerticalAlignment="Center"
FontFamily="Verdana"
               FontWeight="bold" Margin="12,0,0,0" Foreground="SteelBlue"/>
</Border>
```

V prvním řádku, tedy nahoře na scéně, se nachází jakási hlavička aplikace. Jde pouze o element <Border> s definovanou barvou pozadí a okrajů obsahující objekt <TextBlock> s nápisem "Hlavička" a nastaveným formátem a barvou písma. Vidíme zde, že element <Border> má opět nastaven parametr "Grid.Row", tentokrát na hodnotu 0.

```
<Border Grid.Row="2" Background="Beige" Height="20" Margin="10,0"
        BorderBrush="SteelBlue" BorderThickness="1" CornerRadius="3">
    <TextBlock Text="(c) Jiří Kolda" HorizontalAlignment="Right"
               VerticalAlignment="Center"
               FontSize="10" Margin="0,0,5,0" Foreground="SteelBlue"/>
</Border>
```

Naprosto stejně jako hlavička je definována i spodní lišta. Opět jde o patřičně naformátovaný objekt <Border> obsahující objekt <TextBlock>. Hodnota 2 parametru "Grid.Row" vyjadřuje umístění prvku ve třetím, tedy spodním řádku.

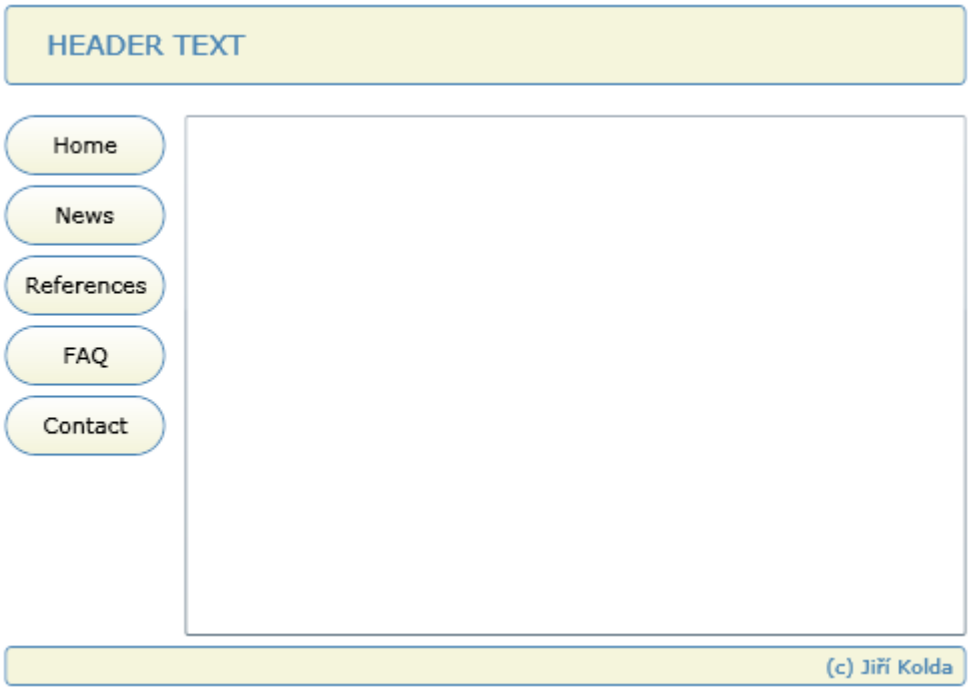
```
<StackPanel Grid.Column="0" Orientation="Vertical">
    <Button Content="tlačítko 0" Height="30" Width="80" Margin="0,5,0,0"/>
    <Button Content="tlačítko 1" Height="30" Width="80" Margin="0,5,0,0"/>
    <Button Content="tlačítko 2" Height="30" Width="80" Margin="0,5,0,0"/>
    <Button Content="tlačítko 3" Height="30" Width="80" Margin="0,5,0,0"/>
    <Button Content="tlačítko 4" Height="30" Width="80" Margin="0,5,0,0"/>
</StackPanel>

<TextBox Grid.Column="1" Margin="0,5,10,5" />
```

Jak jsem již uvedl, druhý (prostřední) řádek je rozdělen na dva sloupce. V prvním z nich, tedy na levé straně, se nachází objekt <StackPanel>

obsahující pět tlačítek. Tento element slouží k umisťování objektů pod sebou nebo vedle sebe. V našem případě jsou rozmístěny svisle (hodnota "Vertical" parametru "Orientation"), což vytváří jakýsi seznam, který je možno považovat za menu. Jednotlivá tlačítka mají pomocí parametru "Margin" definovány rozestupy, popisky jsou udávány hodnotami parametru "Content". Ve druhém sloupci se pak nachází pouze objekt <TextBox>, vyplňující celou plochu vymezenou sloupcem. Místo tohoto objektu by zde mohl být jakýkoli jiný obsah. Element <TextBox> slouží jako textové pole, do kterého uživatel může zadat nějaký text. Je zde použit pouze za účelem vyplnění volného prostoru a nemá žádnou zvláštní funkcionalitu.

7.5 Styly a šablony



Tento třetí ukázkový projekt se zabývá - jak již jeho název napovídá - styly a šablonami v jazyce XAML. Použití stylů velmi usnadňuje a urychluje práci v případech, kdy se na scéně nachází více objektů stejného vzezření. Jedná se

o jakousi analogii k technologii CSS, ve které jsou grafické definice uchovávány na určitém místě a objekty se na ně odkazují.

Šablony jsou velmi užitečným nástrojem pro navrhování grafického vzhledu jednotlivých komponent. S jejich pomocí lze od základu změnit vzhled jakéhokoli XAML prvku. Jelikož jsou styly i šablony velmi detailně popisovány v samotné příručce, ukáži a vysvětlím zde pouze jejich použití v tomto projektu a nebudu se dále zabývat teorií dopodrobna.

Rozvržení objektů na scéně je naprosto stejné jako v předchozím projektu. Jelikož je zde několik graficky shodných objektů, nabízí se využití stylů. První styl je definovány následovně:

```
<Style x:Key="borderStyle" TargetType="Border">
    <Setter Property="Background" Value="Beige"/>
    <Setter Property="Margin" Value="10,0"/>
    <Setter Property="BorderBrush" Value="SteelBlue"/>
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="CornerRadius" Value="3"/>
</Style>
```

Jedná se o definici grafického vzezření objektu <Border> (jehož název vidíme jako hodnotu parametru "TargetType" elementu <Style>) a je zde definováno pět parametrů - barva pozadí, odstup od ostatních objektů, barva a tloušťka okraje a zaoblení rohů. Stylu je přiřazen název "borderStyle" a jak vidíme na následujícím zápisu, objekty na scéně se pomocí klíčového slova "StaticResource" odkazují.

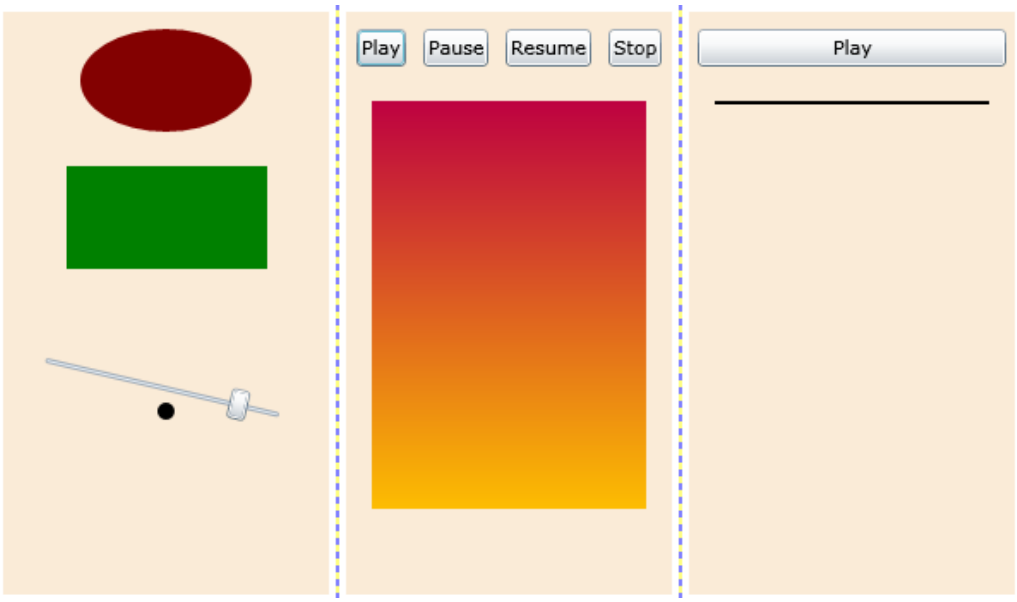
```
<Border Grid.Row="2" Height="20" Style="{StaticResource borderStyle}">
…
```

Následující část zdrojového kódu ukazuje definici šablony pro tlačítko.

```
<Style x:Key="buttonTemplate" TargetType="Button">
    <Setter Property="Height" Value="30"/>
    <Setter Property="Width" Value="80"/>
    <Setter Property="Margin" Value="0,5,0,0"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate>
                <Border CornerRadius="15" BorderBrush="SteelBlue"
                        BorderThickness="1">
                    <Border.Background>
                        <LinearGradientBrush StartPoint="0.5,0"
                                             EndPoint="0.5,1.2">
                            <GradientStop Color="Beige" Offset="0"/>
                            <GradientStop Color="SteelBlue" Offset="1"/>
                        </LinearGradientBrush>
                    </Border.Background>
                    <TextBlock Text="{TemplateBinding Content}"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Šablony jsou vytvářeny naprosto stejným způsobem jako styly, pouze je do jejich těla přidána definice hodnoty parametru "Template". V tomto případě je jako tělo tlačítka použit objekt <Border> se zaoblenými rohy a pozadím s lineární interpolací barev béžové a modré. Uvnitř tohoto prvku je objekt <TextBlock>, jehož účelem je zobrazovat popisek na tlačítku. Příslušný popisek je do šablony načítán z hodnoty parametru "Content" příslušného tlačítka pomocí klíčového slova "TemplateBinding".

7.6 Animace



Tento projekt názorně ukazuje různé způsoby animování objektů na scéně. Jsou zde použity všechny základní animační techniky, které technologie Silverlight umožňuje. Scéna je zde rozdělena na tři sloupce podle způsobu animace objektů v nich obsažených.

V prvním sloupci se nacházejí tři animované objekty. Jedná se shora o elipsu, obdélník a komponentu <Slider>. Animace na těchto třech objektech je spuštěna ihned po načtení aplikace a opakuje se neustále.

```
<Ellipse x:Name="eli1" Width="100" Height="60" Margin="10">
    <Ellipse.Fill>
        <SolidColorBrush x:Name="backGr" Color="Red"/>
    </Ellipse.Fill>
</Ellipse>

<Storyboard x:Name="anim_eli1">
    <ColorAnimation Storyboard.TargetName="backGr" Storyboard.TargetProperty="Color"
                    From="#FFFF0000" To="#FF000000" Duration="0:0:1"
                    AutoReverse="True" RepeatBehavior="Forever" />
</Storyboard>
```

Elipsa má definovánu šířku, výšku a barvu pozadí. Barva je jednolitá a objekt <SolidColorBrush>, který pozadí obstarává má název "backGr".

Na tento název se odkazuje element <ColorAnimation> parametrem "StoryBoard.TargetName", který slouží k definici samotné animace. Ve zdrojovém kódu vidíme použití hexadecimálního barevného formátu u hodnot parametrů "From" a "To", které určují počáteční a koncovou hodnotu barvy pozadí. Animace trvá jednu sekundu, opakuje se neustále a je reverzní (po dosažení koncové hodnoty proběhne zpětná animace na hodnotu počáteční).

```
<Rectangle x:Name="rec1" Width="100" Height="60" Fill="Green" Margin="10"/>

<Storyboard x:Name="anim_rec1">
    <DoubleAnimation Storyboard.TargetName="rec1" Storyboard.TargetProperty="Width"
                     From="100" To="150" Duration="0:0:1.5"
                     AutoReverse="True" RepeatBehavior="Forever" />
</Storyboard>
```

V případě obdélníka slouží animace k jeho plynulému rozšíření a opětovnému zúžení. Tato animace je definována pomocí elementu <DoubleAnimation>, jehož účelem je animovat libovolnou hodnotu typu "Double", tedy reálné číslo. Ve zdrojovém kódu vidíme, že obdélník je pojmenován "rec1" a na tento jeho název se opět animace odkazuje. Jeho šířka osciluje mezi hodnotami 100 a 150 pixelů a každá tato změna trvá 1.5 sekundy. Animace je opět jako v prvním (a jako i v následujícím) případě reverzní a opakuje se neustále.

Posledním objektem nacházejícím se v tomto sloupci je <Slider>. Na tomto prvku je ukázáno, že není nutné striktně animovat jenom grafické vzezření objektu, ale také jeho další vlastnosti. V případě tohoto objektu <Slider> je to jeho parametr "Value", jenž udává polohu jezdce na liště. Zároveň je zde ukázáno použití transformace objektu.

```
<Storyboard x:Name="anim_sld1">
    <DoubleAnimation Storyboard.TargetName="sld1" Storyboard.TargetProperty="Value"
                     From="0" To="100" Duration="0:0:3"
                     AutoReverse="True" RepeatBehavior="Forever" />
    <DoubleAnimation Storyboard.TargetName="rotace" Storyboard.TargetProperty="Angle"
                     From="-20" To="20" Duration="0:0:3"
                     AutoReverse="True" RepeatBehavior="Forever" />
</Storyboard>

<Slider x:Name="sld1" Minimum="0" Maximum="100" Width="150" Margin="0,50,0,0">
    <Slider.RenderTransform>
        <RotateTransform x:Name="rotace" Angle="0" CenterX="75" CenterY="20"/>
    </Slider.RenderTransform>
</Slider>
```

V tomto zdrojovém kódu vidíme, že elementu <Slider> je přiřazen subelement <Slider.RenderTransform>, kterým je možné definovat různé transformace. V našem případě se jedná o rotaci s osou otáčení v bodě o souřadnicích [75,20] a počátečním úhlem otočení 0°.

Animace je zde zajištěna dvěma elementy <DoubleAnimation>, z nichž první animuje zmíněnou pozici jezdce a druhý rotační transformaci. V případě prvního animačního elementu vychází počáteční a koncová hodnota z minima a maxima definovaného v elementu <Slider> a jezdec tedy osciluje mezi hodnotami 0 a 100, tedy od levého konce lišty až k jejímu pravému konci. Druhý element zajišťujíce oscilaci hodnoty úhlu rotace mezi -20° a 20°. Oba druhy animace mají nastavenou stejnou dobu trvání. Výsledkem celé definice je kývavý pohyb objektu <Slider> zleva doprava, což s použitím malé černé elipsy uprostřed objektu dává iluzi vah, které se naklánějí podle pozice jezdce.

Druhý sloupec obsahuje prvky, na kterých je ukázána možnost ovládat průběh animace. Jedná se o čtyři tlačítka, která ovlivňují animaci pod nimi umístěného obdélníku s lineárním gradientem na pozadí. Jejich definice je následující:

```
<StackPanel HorizontalAlignment="Center" Orientation="Horizontal">
    <Button x:Name="butPlay" Content="Play" Margin="5,10" Click="butPlay_Click" />
    <Button x:Name="butPause" Content="Pause" Margin="5,10" Click="butPause_Click" />
    <Button x:Name="butResume" Content="Resume" Margin="5,10" Click="butResume_Click" />
    <Button x:Name="butStop" Content="Stop" Margin="5,10" Click="butStop_Click" />
</StackPanel>
```

Tlačítka jsou umístěna v objektu <StackPanel>, jenž se stará o jejich seřazení vodorovně. Každé tlačítko má pomocí handleru "Click" připojenou metodu, která je zavolána po jeho stisknutí. Zleva první tlačítko slouží ke spuštění animace, druhé k pozastavení, třetí k pokračování a poslední čtvrté k úplnému zastavení. Jednotlivé metody vypadají následovně:

```
private void butPlay_Click(object sender, RoutedEventArgs e)
{
    anim_rec2.Begin();
}

private void butPause_Click(object sender, RoutedEventArgs e)
{
    anim_rec2.Pause();
}

private void butResume_Click(object sender, RoutedEventArgs e)
{
    anim_rec2.Resume();
}

private void butStop_Click(object sender, RoutedEventArgs e)
{
    anim_rec2.Stop();
}
```

Samotná animace a animovaný obdélník jsou definovány takto:

```
<Rectangle x:Name="rec2" Width="160" Height="2" Margin="10">
    <Rectangle.Fill>
        <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
            <GradientStop x:Name="upGrad" Color="Blue" Offset="0"/>
            <GradientStop x:Name="downGrad" Color="Red" Offset="1"/>
        </LinearGradientBrush>
    </Rectangle.Fill>
</Rectangle>

<Storyboard x:Name="anim_rec2">
    <DoubleAnimation Storyboard.TargetName="rec2" Storyboard.TargetProperty="Height"
                     From="2" To="275" Duration="0:0:2" AutoReverse="True"
                     RepeatBehavior="Forever" />
    <ColorAnimation Storyboard.TargetName="upGrad" Storyboard.TargetProperty="Color"
                    From="Blue" To="Red" Duration="0:0:5" AutoReverse="True"
                    RepeatBehavior="Forever" />
    <ColorAnimation Storyboard.TargetName="downGrad" Storyboard.TargetProperty="Color"
                    From="Red" To="Yellow" Duration="0:0:5" AutoReverse="True"
                    RepeatBehavior="Forever" />
</Storyboard>
```

Obdélník je vyplněn svislou lineární interpolací mezi barvami modrou a červenou. Tyto barvy jsou představovány dvěma elementy <GradientStop>. Oba elementy mají zadán název ("upGrad" a "downGrad"), díky čemuž mohou být animovány nezávisle na sobě. Animace se pak skládá celkem ze tří částí. Zároveň jsou animovány výška obdélníka a obě barvy v jeho výplni. Výška se pohybuje mezi hodnotami 2 a 275 pixelů, horní barva gradientu se mění z modré na červenou a spodní barva z červené na žlutou.

V posledním sloupci je vytvořena animace s využitím klíčových snímků. Nacházejí se zde dva objekty. Opět se jedná o spouštěcí tlačítko a animovaný obdélník.

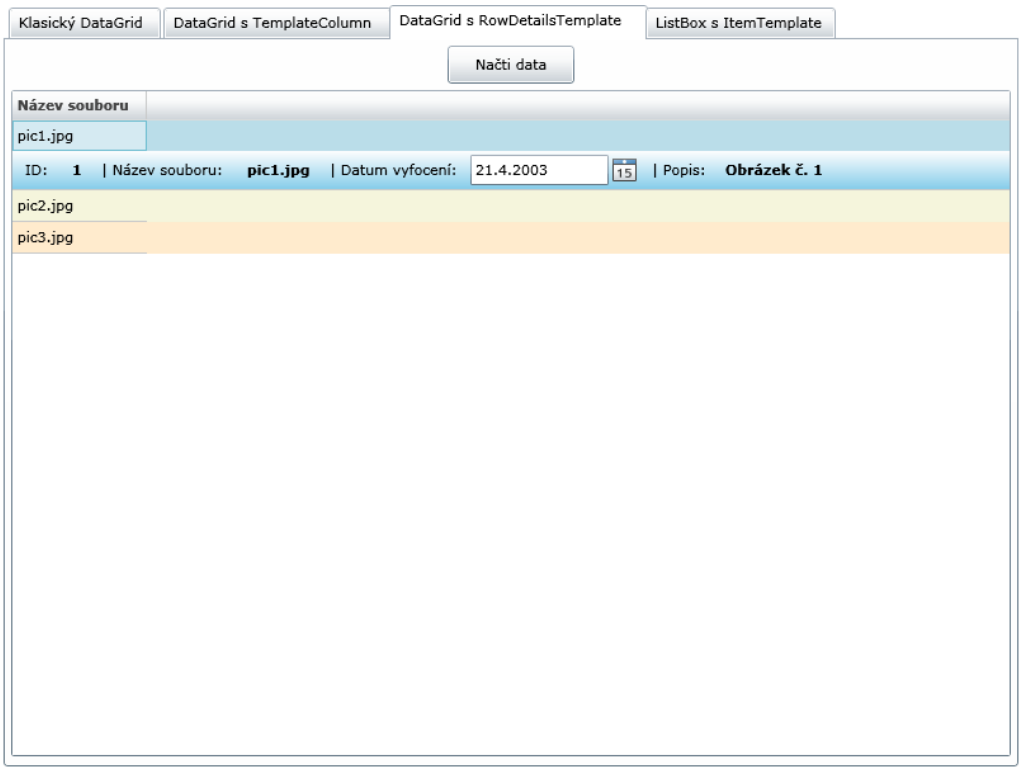
```
<Button Click="butPlay2_Click" x:Name="butPlay2" Content="Play" Margin="5,10" />
<Rectangle x:Name="rec3" Width="160" Height="2" Fill="Black" Margin="10"/>

<Storyboard x:Name="anim_rec3">
    <DoubleAnimationUsingKeyFrames Storyboard.TargetName="rec3"
                                   Storyboard.TargetProperty="Height"
                                    AutoReverse="True" >
        <LinearDoubleKeyFrame KeyTime="0:0:0" Value="2"/>
        <SplineDoubleKeyFrame KeySpline="0,0 1,0" KeyTime="0:0:1" Value="275"/>
        <SplineDoubleKeyFrame KeySpline="0.10, 0.21 0.00, 1.0" KeyTime="0:0:1.3"
                              Value="200"/>
        <SplineDoubleKeyFrame KeySpline="0,0 1,0" KeyTime="0:0:1.5" Value="275"/>
        <SplineDoubleKeyFrame KeySpline="0.10, 0.21 0.00, 1.0" KeyTime="0:0:1.7"
                              Value="250"/>
        <SplineDoubleKeyFrame KeySpline="0,0 1,0" KeyTime="0:0:1.8" Value="275"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

Stěžejní je zde element <DoubleAnimationUsingKeyFrames>. Uvnitř tohoto elementu se nacházejí definice jednotlivých klíčových snímků. První snímek v tomto případě slouží pouze jako nastavení počátečního stavu. Následuje série snímků vytvářejících animaci po křivce. Každý s těchto klíčových snímků má pevně danou dobu a hodnotu, které má animovaný parametr dosáhnout. Pomocí parametru "KeySpline" je pro každý klíčový snímek definováno buď postupné zrychlení nebo zpomalení k dané hodnotě.

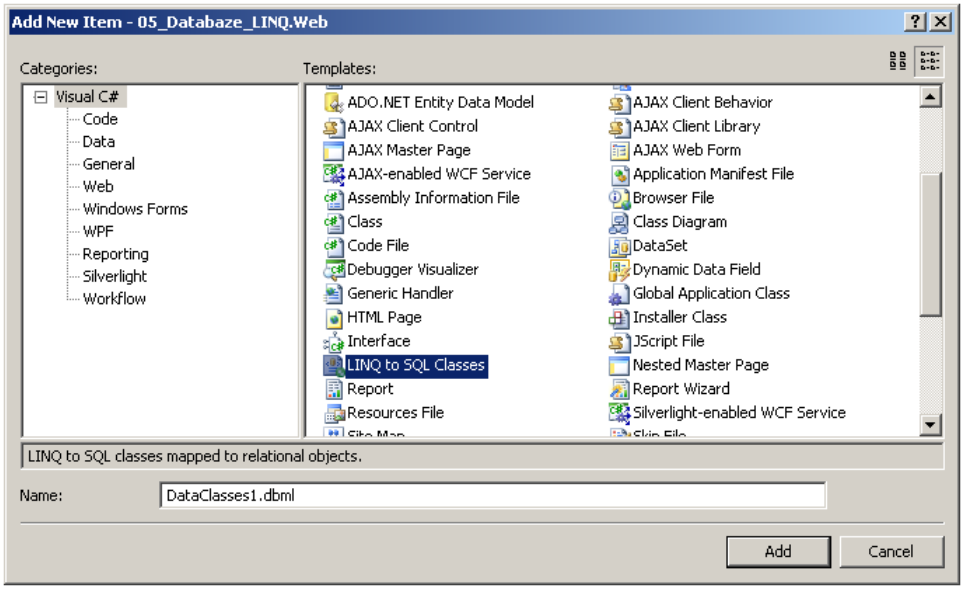
Výsledek celého zápisu můžeme považovat za animaci pádu opony. Vzhledem k nastavenému parametru "AutoReverse" se animace opět převrátí a projde všechny klíčové snímky pozpátku až k počátečnímu. Poté se animace zastaví.

7.7 Napojení na databázi pomocí LINQ

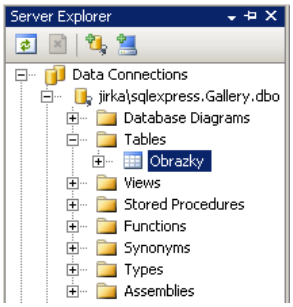


Poslední projekt ukazuje, jakým způsobem se aplikace napojuje na SQL databázi a jaké jsou možnosti zobrazení načtených dat. Jak je zmíněno v příručce, na tomto projektu popíši celý (relativně komplikovaný) postup připojení k databázi pomocí technologie WCF a jazyka LINQ. K tomuto jsem se rozhodu z toho důvodu, že tato tématika je teoreticky dosti obsáhlá a zasloužila by si samostatnou publikaci v rozsahu této práce. Nebudu zde tedy zacházet do přílišných a pro účel této publikace zbytečných detailů a vysvětlím principy na jednoduchém projektu, ve kterém představuji čtyři možné cesty k zobrazení dat načtených z databáze. Nejprve tedy vysvětlím, co všechno je třeba zařídit, aby se aplikace dokázala na databázi připojit. Budu zde popisovat postupy ve vývojovém prostředí Visual Studio 2008.

Celý proces připojování na databázi se odehrává v projektu webové stránky, která hostuje Silverlight aplikaci. První věc, kterou do tohoto projektu musíme přidat je soubor zvaný "LINQ to SQL Classes". Do projektu ho přidáme pomocí dialogu "Add New Item", který spustíme příkazem, jenž najdeme v kontextovém menu názvu projektu.



Šablonu pro tento soubor nalezneme přímo v záložce "Visual C#", můžeme zvolit název vytvářeného souboru a stiskneme tlačítko "Add". Tento soubor se objeví uvnitř projektu v panelu "Solution Explorer". Po jeho otevření se ukáže prázdná plocha. Na této ploše budou později zobrazeny třídy, které Visual Studio automaticky vygeneruje po připojení k databázi. Každá z těchto tříd bude příslušet jedné tabulce z databáze. Pro vytvoření připojení k databázi využijeme panel s názvem "Server Explorer" na levé straně obrazovky. V tomto panelu zvolíme příkaz "Connect to Database" a následně s pomocí průvodce vytvoříme připojení k databázovému serveru, na kterém se příslušná databáze nachází. Výsledkem bude, že se struktura databáze zobrazí v panelu "Server Explorer" (viz. obrázek nalevo). Nyní již stačí "přetáhnout" vybrané tabulky z databáze na plochu otevřeného LINQ To SQL



souboru a vytvoří se jim odpovídající třídy. Následuje již pouze vytvoření WCF Servisu. Ten vložíme do projektu stejným způsobem jako předešlý soubor, tentokrát však v dialogu Add New Item zvolíme "WCF Service". Po zadání názvu souboru a stisknutí tlačítka "OK" přibudou v projektu dva soubory. Prvním je "IService1.cs" a druhým je "Service1.svc". Soubor "IService1.cs" obsahuje deklarace metod, kterými jsou načítána data z databáze a soubor "Service1.svc" (resp. jeho code-behind soubor "Service1.svc.cs") obsahuje samotná těla těchto metod. V případě našeho projektu vypadají soubory následovně:

IService.cs:

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    List<Obrazky> getPics();
}
```

Service1.svc.cs:

```
public class Service1 : IService1
{
    public List<Obrazky> getPics()
    {
        DataClasses1DataContext db = new DataClasses1DataContext();
        var query = from p in db.Obrazkies select p;

        return query.ToList();
    }
}
```

V horní části ukázky vidíme deklaraci metody "getPics()" a ve spodní části její tělo. Jediným účelem této metody je načíst z databáze všechny položky nacházející se v tabulce s názvem "Obrazky".

Poslední věcí, kterou je třeba nastavit, je typ tzv. "vazby" - [binding]. Po vytvoření WCF Servisu byl v souboru "Web.Config" automaticky vygenerován zápis o typu spojení a všech možných parametrech servisu a právě zde je třeba udělat malou úpravu. Jelikož Silverlight nepodporuje typ "wsHttpBinding", ale pouze o něco zastaralejší typ "basicHttpBinding", je třeba první jmenovaný typ nahradit tím druhým. Pokud bychom toto neudělali, servis by nefungoval, jelikož Silverlight aplikace by nedokázala vytvořit zmíněný typ vazby se serverem.

Po této změně je WCF Servis kompletní. Nachází se ale v projektu webové stránky. Abychom jej propojili se samotnou aplikací, je třeba přidat do Silverlight projektu referenci na tento servis se odkazující. Příkaz k vytvoření reference se nachází v kontextovém menu Silverlight projektu a po jeho spuštění se zobrazí dialogové okno s názvem "Add Service Reference". Zde stačí stisknout tlačítko "Discover" a Visual Studio automaticky vyhledá uvnitř solution námi vytvořený servis. Nakonec stačí pojmenovat vytvořenou referenci a vše potvrdit stisknutím tlačítka "OK".

Nyní již popíši samotný projekt a jeho funkcionalitu. Hlavním prvkem na scéně je objekt <TabControl>. Tento prvek umožňuje rozmístit obsah scény do libovolného počtu záložek, mezi nimiž je následně možné se přepínat. V tomto projektu využívám čtyř záložky. Na každé z nich se nachází tlačítko pro načtení dat z databáze a element, který načtená data určitým způsobem zobrazuje. Konkrétně na prvních třech záložkách se nachází pokaždé jinak definovaný objekt <DataGrid> a na poslední čtvrté je objekt <ListBox> s použitou šablonou určující vzhled jeho položek. Následující výňatek ze zdrojového kódu ukazuje definici první záložky:

```
<basics:TabItem Header="Klasický DataGrid" Width="120" Height="25">
    <StackPanel Orientation="Vertical">
        <Button Width="100" Height="30" Content="Načti data" Margin="0,0,0,5"
                Click="Button_Click" />
        <data:DataGrid x:Name="DataGrid1" Height="525" RowBackground="BlanchedAlmond"
                       AlternatingRowBackground="Beige" AutoGenerateColumns="True"
                       GridLinesVisibility="Horizontal" />
    </StackPanel>
</basics:TabItem>
```

Vidíme zde definovaný popisek záložky a strukturu jejího obsahu, jež je pro všechny záložky stejná. Nachází se zde tlačítko pro načtení dat a element <DataGrid>, jemuž jsou definovány pouze parametry zajišťující grafickou podobu - barvy pozadí lichých a sudých řádků a viditelnost linek oddělujících řádky. Parametr "AutoGenerateColumns" je nastaven na hodnotu "True", což zaručuje automatické vytvoření sloupců podle načítaných dat.

```
<basics:TabItem Header="DataGrid s TemplateColumn" Width="180" Height="25">
    <StackPanel Orientation="Vertical">
        <Button Width="100" Height="30" Content="Načti data" Margin="0,0,0,5"
                Click="Button_Click" />
        <data:DataGrid x:Name="DataGrid2" Height="525" RowBackground="BlanchedAlmond"
                       AlternatingRowBackground="Beige" AutoGenerateColumns="False"
                       GridLinesVisibility="Horizontal">
            <data:DataGrid.Columns>
                <data:DataGridTextColumn Binding="{Binding id}" Header="ID"/>
                <data:DataGridTextColumn Binding="{Binding fileName}"
                                         Header="Název souboru"/>
                <data:DataGridTemplateColumn Header="Datum vyfocení"
                                             SortMemberPath="dateTaken">
                    <data:DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <basics:DatePicker SelectedDate="{Binding dateTaken}" />
                        </DataTemplate>
                    </data:DataGridTemplateColumn.CellTemplate>
                </data:DataGridTemplateColumn>
                <data:DataGridTextColumn Binding="{Binding description}"
                                         Header="Popis obrázku"/>
            </data:DataGrid.Columns>
        </data:DataGrid>
    </StackPanel>
</basics:TabItem>
```

Na druhé záložce se nachází již trochu složitější verze elementu <DataGrid>. V tomto případě nejsou sloupce generovány automaticky, nýbrž jsou předem definovány a jsou jim jasně přidělená načítaná data. Je zde rovněž využita možnost vytvářet šablony sloupců. V tomto případě se jedná o sloupec zobrazující hodnoty typu "DateTime" (tedy údaj o datu a čase). Aby zde nebyly datum a čas zobrazovány jako pouhý text, je do sloupce vložen objekt <DatePicker>, což je grafický a interaktivní kalendář.

Na třetí záložce je k zobrazení načtených dat použit objekt <DataGrid> s definovaným subelementem <DataGrid.RowDetailsTemplate>. Následující zdrojový kód ukazuje definici subelementu:

```
<data:DataGrid.RowDetailsTemplate>
    <DataTemplate>
        <StackPanel Orientation="Horizontal" Height="30"
            VerticalAlignment="Center" HorizontalAlignment="Stretch">
            <StackPanel.Background>
                <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                    <GradientStop Color="SkyBlue" Offset="1" />
                    <GradientStop Color="White" Offset="0" />
                </LinearGradientBrush>
            </StackPanel.Background>
            <TextBlock Text="ID: " Margin="10,0" VerticalAlignment="Center"/>
            <TextBlock Text="{Binding id}" Margin="5,0" FontWeight="Bold"
                ...
        </StackPanel>
    </DataTemplate>
</data:DataGrid.RowDetailsTemplate>
```

Z důvodu vyšší složitosti zdrojového kódu je z ukázky vynechána definice samotného elementu <DataGrid>. Je v podstatě shodná s definicí prvku na druhé záložce, ovšem je zde jeden zásadní rozdíl. V tomto případě totiž nejsou definovány všechny sloupce, ale pouze jeden - sloupec s názvem "fileName". Právě účelem použití prvku "RowDetails" (tedy "detailů řádku") je vyhnout se zbytečně velkému počtu sloupců v tabulce. Jedná se v podstatě o panel, který se zobrazí po označení určitého řádku a poskytne všechny ostatní informace. Zde se po označení nějakého řádku objeví panel se všemi ostatními údaji načtenými z databáze, týkajícími se daného obrázku.

Na poslední záložce se nachází objekt <ListBox> s definicí šablony pro jeho položky - <ListBox.ItemTemplate>.

```
<ListBox x:Name="listBox" HorizontalAlignment="Stretch"
         VerticalAlignment="Stretch">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Border CornerRadius="10">
                <Border.Background>
                    <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
                        <GradientStop Color="Beige" Offset="0" />
                        <GradientStop Color="SkyBlue" Offset="1" />
                    </LinearGradientBrush>
                </Border.Background>
                <StackPanel Orientation="Horizontal" Height="30" Margin="30,0">
                    <StackPanel Orientation="Vertical" Background="SkyBlue"
                                Width="30">
                        <TextBlock Text="ID" FontWeight="Bold"
                                   HorizontalAlignment="Center"/>
                        <TextBlock Text="{Binding id}"
                                   HorizontalAlignment="Center"/>
                    </StackPanel>
                      ...
                </StackPanel>
            </Border>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Z zdrojového kódu je patrné, že samotná definice se celkem podobá definici "RowDetails" z předchozí ukázky. Jedná se o element <Border> obsahující layout komponentu <StackPanel>, která zajišťuje rozmisťování všech v ní obsažených objektů horizontálně za sebou. Mezi samotnými objekty jsou pak opět převážně prvky <TextBlock> a pro zobrazení data a času je použit objekt <DatePicker>.

7.8 Problémy při vývoji

Při vývoji projektů jsem narazil na několik problémů a komplikací. Nejčastějším problémem byla zpětná nekompatibilita pluginu pro zobrazení Silverlight aplikací v prohlížečích. Během vytváření jednotlivých projektů se několikrát objevila nová verze pluginu a automaticky se nainstalovala na můj počítač, čímž byla vždy přepsána verze, pro kterou byly jednotlivé projekty vyvíjeny. Výsledkem bylo v mnoha případech žádné nebo špatné zobrazení Silverlight aplikací. Tento problém většinou vyřešilo odinstalování pluginu a opětovné stažení nejnovější verze. Zajímavé je, že k této komplikaci nedocházelo vždy s příchodem novější verze. Několikrát se verze aktualizovala bez nejmenších problémů.

Osobně jsem se setkal i s několika problémy při používání nástroje Microsoft Blend 2. Tyto problémy se týkaly vesměs špatného, či spíše zbytečného generování XAML kódu během vytváření grafického designu scény. Jako příklad může sloužit následující zápis elementu <Rectangle>, ve kterém je zcela zbytečně použit parametr "StrokeThickness":

```
<Rectangle StrokeThickness="0" Height="100" Width="100" />
```

Občas se také objevovaly chyby týkající se propojení nástroje Blend 2 s Visual Studiem. Pokud byl tentýž projekt otevřen v obou prostředích, měla probíhat neustálá synchronizace při jakékoli změně projektu. Několikrát se mi stalo, že projekty nebyl správně synchronizován při přechodu mezi prostředími. Bylo pak třeba restartovat aplikaci se starší verzi projektu. Nutno však dodat, že tato chyba nastala velmi zřídkakdy a nelze ji považovat za zásadní a přetrvávající problém.

8 Závěrečné shrnutí

Cílem této práce bylo sestavit první česky psanou publikaci pojednávající výhradně o Microsoft Silverlight 2.0, otestovat tuto technologii v nejrozšířenějších prohlížečích a vytvořit s její pomocí sadu ukázkových projektů.

Tyto cíle jsem se snažil splnit v plném rozsahu. Sepsal jsem příručku pro začínající programátory, v níž je popisována syntaxe jazyka XAML jakožto prostředku pro vytváření grafického designu internetových aplikací a čtenář je obeznámen se základními principy, na kterých je technologie Silverlight postavena.

Sestavil jsem také sadu ukázkových projektů, v nichž představuji základní komponenty a metody, které je možné v Silverlight 2.0 používat. Tyto projekty svou náročností odpovídají tomu, co jsem si od začátku stanovil a jsou nyní volně přístupné na internetu. Součástí praktické části jsou také volně stažitelné všechny příklady použité v příručce při vysvětlování látky.

Podle mého názoru je technologie Silverlight velmi silným nástrojem pro tvorbu internetových aplikací a má velký potenciál do budoucna. Osobně jsem se touto technologií zabýval od jejích úplných začátků a byl jsem svědkem velkého počtu chyb, problémů a komplikací. Domnívám se však, že jakmile Silverlight dosáhne takové funkční stability jako Adobe Flash, bude pro něj velmi silným soupeřem. Jestliže k této stabilizaci dojde, je možné, že i já se přidám k programátorů, využívajícím technologii Silverlight.

9 Přehled použité literatury

[1] SCHWARZ, Michael. Silverlight Tutorials - Michael's Blog [online]. 2006 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://weblogs.asp.net/mschwarz/archive/2007/06/06/silverlight-tutorials.aspx>.

[2] ŠTURALA, Aleš. Aleš Šturala | Blog | česká verze » Silverlight [online]. 2007 [cit. 2009-01-22]. Text v češtině. Dostupný z WWW: <http://www.unitedstatesof.net/category/silverlight/>.

[3] SUREAU, Denis. Silverlight and XAML tutorial for Web applications and RIA [online]. 2007-2008 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://www.scriptol.com/silverlight/>.

[4] Microsoft Corporation. Tutorials : The Official Microsoft Silverlight Site [online]. 2008 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://silverlight.net/learn/tutorials.aspx>.

[5] Microsoft Corporation. Videos : The Official Microsoft Silverlight Site [online]. 2008 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://silverlight.net/Learn/videocat.aspx?cat=2>.

[6] GUTHRIE, Scott. ScottGu's Blog [online]. 2003-2009 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://weblogs.asp.net/scottgu/>.

[7] WILDERMUTH, Shawn. Shawn Wildermuth's Blog [online]. 1999-2008 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://wildermuth.com/>.

[8] Microsoft Corporation. 101 LINQ Samples [online]. 2009 [cit. 2009-01-22]. Text v angličtině. Dostupný z WWW: <http://msdn.microsoft.com/en-us/vcsharp/aa336746.aspx>.

[9] Microsoft Corporation. Silverlight - MSDN [online]. c2009 [cit. 2009-02-07]. Dostupný z WWW: <http://msdn.microsoft.com/cs-cz/library/cc838158(en-us,vs.95).aspx>.

[10] EZELL, Jesse. Silverlight vs. Flash: The Developer Story : Jesse Ezell Blog [online]. 2007 [cit. 2009-02-14]. Dostupný z WWW: <http://weblogs.asp.net/jezell/archive/2007/05/03/silverlight-vs-flash-the-developer-story.aspx>.

[11] ANDERSON, Tim. Microsoft Silverlight: 10 reasons to love it, 10 reasons to hate it [online]. 2008 [cit. 2009-02-14]. Dostupný z WWW: <http://www.theregister.co.uk/2008/08/18/silverlight_pros_and_cons/page2.html>.

[12] MACDONALD, Matthew. Pro Silverlight 2 in C# 2008. Berkeley : Apress, 2008. 607 s. ISBN 1-59059-949-7.

[13] MACDONALD, Matthew, SZPUSZTA, Mario. ASP.NET 3.5 a C# 2008. Překlad RNDr. Jan Pokorný, Jan Gregor. Brno : Zoner Press, 2008. 1584 s. ISBN 978-80-7413-008-3.

[14] CHINNATHAMBI, Kirupa. KirupaBlog : What on Earth is XAP? [online]. 2008 [cit. 2009-03-19]. Dostupný z WWW: <http://blog.kirupa.com/?p=184>.

[15] HOWER, Chad Z.. Video: Silverlight versus Flash [online]. 2007 [cit. 2009-04-18]. Text v angličtině. Dostupný z WWW: <http://www.kudzuworld.com/blogs/Tech/20070827C.en.aspx>.

10 Seznam příloh

[1] Příručka Microsoft Silverlight 2.0

[2] Zdrojové kódy všech projektů a příkladů na přiloženém CD