JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
PEDAGOGICKÁ FAKULTA
KATEDRA INFORMATIKY

Windows Presentation Foundation & Data Binding

BAKALÁŘSKÁ PRÁCE

Autor práce: Vilém Janda
Vedoucí práce: Ing. Václav Novák, CSc.

2010

UNIVERSITY OF SOUTH BOHEMIA
PEDAGOGICAL FACULTY
DEPARTMENT OF INFORMATION SCIENCE

Windows Presentation Foundation & Data Binding

THE BACHELOR'S THESIS

Author: Vilém Janda
Supervisor: Ing. Václav Novák, CSc.

2010

Anotace:

Cílem práce je vytvoření uceleného kurzu ve formě e-learning studijních materiálů pro výklad technologie Data Binding ve Windows Presentation Foundation (WPF). V první, převážně teoretické části, bude proveden popis a výklad jednotlivých prvků technologie WPF se zaměřením na Data Binding. V druhé, praktické části, je k dispozici metodika a vlastní výukový kurz spolu s výkladovými audiovizuálními soubory pro samostatné studium.

Výklad je doplněn, jak řešenými příklady, tak příklady k procvičení.

Abstrakt:

The aim of this work is a course in the form of e-learning study materials for the interpretation of technology Data Binding in Windows Presentation Foundation (WPF). In the first, mostly theoretical part will be done a description and interpretation of the elements of technology, focusing on WPF Data Binding. In the second part, is available methodology and training course with their own interpretive audio-visual files for self-study.

The lectures are supplemented by solved examples, and examples for practice.

Klíčová slova:

Windows Presentation Foundation, WPF, Data Binding, Visual studio 2008, C#, XAML, .Net Framework 3.5, MS Expression Blend 3, Programovací jazyk, Programming language.

Zadání:

Ve verzi 3v5.NET Framework se klade důraz na spojení grafických a řídících vlastností aplikací tvořených za pomocí Visual Studia 2008. Úkolem diplomanta je vytvoření uceleného kurzu ve formě e-learning studijních materiálů pro výklad technologie Data Binding ve Windows Presentation Foundation (WPF).

Diplomant vytvoří výkladové video soubory pro samostatné studium. Výklad musí být doplněn, jak řešenými příklady, tak příklady k procvičení.

Jazyk práce: čeština

Poděkování

Děkuji vedoucímu práce za rady a vedení při její tvorbě.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval 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 19. dubna 2010

Osnova

1 Úvod

V současné době dosáhl rozvoj informačních technologií značného rozmachu. Neustále se vyvíjí a každým dnem využívá mnoho programů všeho druhu, které se snaží mít stále přívětivější a intuitivnější uživatelské rozhraní. Již není možné nabízet běžným uživatelům software, který je ovládán pouze prostřednictvím příkazové řádky. Naopak je stále důležitější, aby grafické uživatelské rozhraní bylo jednak přívětivé a intuitivní, ale i moderně designované. Tuto snahu se snaží podpořit i stále ještě poměrně nová technologie Windows Presentation Foundation (WPF) od firmy Microsoft.

Tato práce se primárně nezabývá vzhledem programů ve WPF, ale způsobem vazby mezi daty a vlastním vzhledem aplikace. Technologie datových vazeb „Data Binding“, která je obsažena ve WPF, právě toto propojení s daty umožňuje.

Tato práce je koncipována jako popis a ukázky jednotlivých způsobů napojení grafické části programu na data v různých úložištích. Jedná se o data v databázi, v XML souboru, nebo jen o strukturu objektů, kterou chceme s GUI propojit, při zachování oddělenosti grafické a funkcionální části programu.

Tato práce ukazuje, jak lze ve WPF téměř bez znalosti kódu XAML a bez dlouhého studia propojit grafickou a funkční část.

Práce je rozdělena do tří základních částí. V první části jde o shromáždění teorie, týkající se dané problematiky do uceleného celku. V druhé části je zpracována metodika výukového kurzu a v poslední nejdůležitější části jsou podrobně popsány čtyři příklady ilustrující danou problematiku, na nichž student pochopí různé způsoby užití Data Bindingu. Všechny příklady jsou doplněny audiovizuálními výukovými soubory.

Výukové příklady využívají různé datové zdroje a ukazují jejich možnosti a způsob práce s daty v nich obsaženými. Vysvětlují tedy základní oblasti užití Data Bindingu, včetně vazby mezi jednotlivými grafickými elementy a například vazby na strukturu objektů.

1.1 Programování v systému Windows

V prvních verzích systému Windows se programy psaly prakticky jediným způsobem, který byl založen na přístupu k rozhraní Windows API pomocí programovacího jazyka C.

Během dalšího vývoje se systému přizpůsobily další jazyky, například Visual Basic, nebo C++ a C#.

V současnosti je možné použít všechny tyto přístupy:

- Jazyk C a rozhraní Windows API
- Jazyk C++ a knihovna MFC (Microsoft Foundation Class)
- Jazyk C#, nebo C++ a Windows Forms (.NET Framework 1.0 a výše)
- Jazyk C#, nebo C++ a Windows Presentation Foundation (WPF)(1)

Každý se může rozhodnout pro přístup, který mu vyhovuje a který je k řešení konkrétního problému nejvhodnější.

Lze doporučit platformu .NET a přístupy s ní spojené. Programy v této platformě se většinou kompilují do zprostředkujícího kódu (jazyka MSIL neboli Microsoft Intermediate Language), který se pak překládá do nativního kódu při spuštění aplikace. Tento postup poskytuje jistou ochranu před chybami programů a teoretickou nezávislost na platformě.

V této práci se budeme zabývat především posledním zmíněným přístupem a to WPF, který poskytuje především mnohem efektnější a pokročilejší možnosti pro zpracování grafiky programu.

2 Windows Presentation Foundation (WPF)

2.1 Co je WPF?

Jedná se o nový grafický framework od firmy Microsoft umožňující psaní aplikací pro Windows. Prvotně bylo použito pro aplikace k systému Windows Vista. Často se považuje za primární API (application programming interface)[1] již zmíněného systému Windows Vista, ale lze ho použít i v systémech starších, například Windows XP, doinstalováním platformy .NET Framework 3.0. WPF je nové grafické rozhraní k .NET Frameworku 3.0 a .NET Frameworku 3.5. (4)

Programy ve WPF jsou tedy teoreticky spustitelné všude tam, kde je předinstalované rozhraní .NET Framework 3.0, nebo 3.5. Jako minimální požadavek bývá uváděn systém Windows XP s aktualizací Service Pack 2. K vývoji pak je potřeba Visual Studio a velice usnadní práci i balíček Microsoft Expression, kterým se budeme podrobně zabývat později.

[1] Česky „aplikační programovací rozhraní“

Předchůdcem Windows Presentation Foundation byly WinForms[2], i když WPF není zatím přímo nástupcem WinForms, ale spíše jiný způsob psaní aplikací. Nabízí nové možnosti především v grafické podobě aplikace.

WPF je zaměřeno na uživatelsky a graficky bohaté aplikace. Nabízí jiný, nový vzhled, který na první pohled zaujme ve srovnání se stále stejně vypadajícími formulářovými aplikacemi.

Windows Presentation Foundation (WPF) umožňuje vývojářům sestavit graficky bohatou aplikaci s inteligentním uživatelským rozhraním i s malými uživatelskými zkušenostmi. Tato aplikace může spolupracovat s médii, dokumenty i s jinými druhy dat.(4)

Ve WPF můžeme využívat mnoho možností:

- vektorové grafiky – umožňuje graficky bezztrátovou změnu velikosti prvků, barevné gradienty, používání geometrických tvarů
- animace – časová nebo za použití framů, rotace, přechod barev, 3D animace a mnoho dalších
- multimédia – práce s audiem a videem (avi, mpeg, wmv)
- efekty – stíny, záře, rozostření, průhlednost nebo zrcadlení
- interaktivní 3D aplikace – poskytuje podmnožinu funkcí z Direct3D zaměřenou na multimédia, dokumenty a uživatelské rozhraní
- Data Binding – provázání dat v aplikaci

WPF dále umožňuje spolupráci jak s WinAPI, tak s rozhraním DirectX[3], nebo například s WinForms.

2.2 Grafická náročnost

Další velice významnou předností WPF je fakt, že grafická část aplikace se zpracovává přímo na grafické kartě.

Jinými slovy WPF aplikace, respektive jejich grafická část psaná v XAMLu, běží zcela na vrstvě DirectX. Tedy téměř minimálně zatěžuje hlavní procesor počítače (CPU), ale naopak veškeré grafické výpočty provádí procesor na grafické kartě (GPU). Vzhledem k tomu, že GPU je pro grafické výpočty přímo optimalizován a grafické karty v současnosti dosahují vysokých výkonů, můžeme dosáhnout daleko efektivněji i velice uživatelsky a graficky bohatých aplikací.

[2] Starší grafické aplikační rozhraní pod platformou .Net

[3] Aplikační rozhraní pro přímé ovládání grafického hardwaru.

Z teoretického hlediska tedy platí, že i na méně výkonných počítačích, za předpokladu dostačující grafické karty, při použití WPF, lze jednoduše vytvořit graficky bohaté aplikace, které bychom jinými způsoby vyvíjeli buď příliš složitě, nebo by na srovnatelném hardwaru korektně nefungovaly.

2.3 XAML

2.3.1 Popis

Objektově orientované programování je pro psaní uživatelského rozhraní poměrně těžkopádné, proto je jednou z novinek ve WPF značkovací jazyk XAML založený na univerzálním značkovacím jazyce XML, který slouží ke tvorbě uživatelského grafického rozhraní.

XAML, neboli Extensible Application Markup Language, čti „zamml“, nebo „zampl“, slouží k definování grafického rozhraní aplikace.

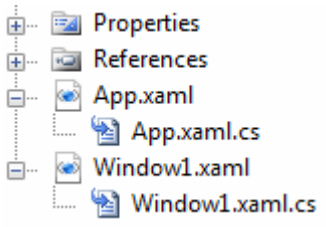
XAML je asi nejvýraznější a nejpřevratnější novinkou ve WPF a dal by se přirovnat k jazyku HTML. Pro přiklad XAML spolupracuje s logikou programu (C#, nebo VB) podobně jako HTML spolupracuje například s PHP, avšak ještě větší analogie je patrná při programování stránek v jazyce ASP.NET.

Značkovací jazyky (markup languages) jsou pro definování vzhledu ideálním řešením jak rychle, stručně, přehledně a jednoduše nadefinovat vlastnosti, umístění a spojitosti jednotlivých elementů. Mezi značkovací jazyky patří například HTML, který má obdobnou funkci v oblasti vývoje webových stránek.

2.3.2 Výhody

XAML je doplňkové programovací rozhraní grafického subsystému Windows Presentation Foundation. XAML tedy umožňuje oddělit grafickou a funkční část aplikace. Díky tomuto oddělení mohou návrháři vytvořit atraktivní grafické uživatelské rozhraní v jazyce XAML a programátoři se mohou soustředit spíše na funkční části programu a interakce mezi elementy a ovládacími prvky.

Hlavními výhodami XAMLu jsou jednoduchost, rychlost a čistota zápisu kódu, další výhodou XAMLu je pak oddělení programové části od uživatelského rozhraní. Ve WPF aplikaci se každé okno skládá z 2 souborů, jednoho „.xaml“ a připojeného „.cs“ souboru, který obsahuje logiku aplikace.(2)



Obrázek 1 : Struktura souborů ve WPF (2)

Psát aplikační logiku k programům v XAMLu je možné buď pomocí jazyku C#, nebo Visual Basic. Při použití jazyka Visual Basic budou mít soubory s logikou aplikace koncovku „.vb“. XAML soubor definuje vzhled okna, tj. jednotlivé elementy jako tlačítka, textová pole…, dále jejich umístění a vzhled. (2)

K dispozici jsou i nástroje generující kód v XAMLu, a proto není nutné tento kód psát ručně, ale narozdíl od WinForms, kde podobné generování také existuje, byl XAML navržen tak, aby vygenerovaný kód mohl programátor bez nebezpečí upravovat. Kód ne-musí být ve striktně vygenerovaném formátu, jako je tomu u WinWorms, pouze musí být zachována syntaktická správnost.

2.3.3 Zápis

Při zápisu v jazyce XAML, máme několik možností zápisu.

```
<Button HorizontalAlignment="Left" VerticalAlignment="Bottom" Width="75">
tlacitko2
</Button>
```

Obrázek 2 : XAML tlacitko2 – zápis 1

```
<Button HorizontalAlignment="Left" VerticalAlignment="Bottom"
Width="75" Content="tlacitko1" />
```

Obrázek 3 : XAML tlacitko1 – zápis 2

Jazyk XAML dále také umožňuje vkládat jednotlivé kontrolky do sebe, což nám dává více možností využití jednotlivých kontrolek, tvořením různých kombinací.

```
<Button HorizontalAlignment="Left" VerticalAlignment="Bottom" Width="75">
<TextBox>text</TextBox>
</Button>
```

Obrázek 4 : XAML vkládání kontrolek do sebe

2.3.4 Srovnání zápisu XAML vs. C#

Zde je vidět srovnání kódu v jazyce XAML (viz. XAML tlacitko1 – zápis) s jazykem C# (viz. níže). Oba tyto kódy budou dělat totéž.

```
Button btn = new Button();
btn.HorizontalAlignment = "Left" ;
btn.VerticalAlignment = "Bottom" ;
btn.Width = "75";
btn.Content = "tlacitko1";
```

Obrázek 5 : C# tlacitko1

Z obrázků je patrná jednoduchost, stručnost a přehlednost kódu v jazyce XAML, která byla popsána výše.

2.3.5 Dokument v XAML

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:src="clr-namespace:videoteka_xml.Convertor"
```

Obrázek 6 : Dokument XAML

Již zmíněné úseky kódu v XAMLU nemohou existovat samostatně, musí být součástí většího dokumentu, který ohraničuje jediný kořenový element v tomto případě element „Window“. Tyto XML dokumenty musí být jednoznačné. Z tohoto důvodu se používá tzv. obor názvů XML. Výchozí obor názvů v XML se deklaruje pomocí atributu xmlns. Tento obor se vztahuje na element, ve kterém se deklarace objevuje, ale také na všechny podřízené elementy. (2)

Pro kód XAML je obor *„http://schemas.microsoft.com/winfx/2006/presentation“*. Přestože jde o URL adresu, nenajdete na ní žádný obsah, jde pouze o obor názvů, který si společnost Microsoft zvolila pro identifikaci elementů XAML, např. *Button*, *Image*, ...

Výše uvedený obor názvů obsahoval třídy a vlastnosti, které jsou součástí grafického subsystému WPF, cože je pouze jedna z možností uplatnění XAML. Pokud chceme používat elementy a atributy specifické pro jazyk XAML, nikoliv pro WPF a to jistě chceme, musíme zahrnou ještě druhou deklaraci oboru názvů.

Tato deklarace je prakticky totožná s výše zmíněnou, pouze chybí slovo „presentation“, *„http://schemas.microsoft.com/winfx/2006“*.(1)

Obecná deklarace elementů z jazyka XAML se obvykle uvádí s prefixem „x“, ale lze použít jakýkoliv jiný prefix. Využitím tohoto prefixu je například definice třídy.

`x:Class="videoteka_xml.MainWindow"` - Atribut *Class* nepatří do oboru názvů WPF, ale XAML.

Dalším užitým prefixem je „src“. Opět lze použít jakýkoliv jiný prefix. Tento prefix spojuje kód v XAML se jmenným prostorem, ve kterém je funkční část programu. Nebo v našem případě s konkrétní třídou.

2.4 Styly

2.4.1 Popis

Velice často jsme při tvorbě grafického uživatelského rozhraní nuceni psát některé věci znovu a znovu. Je to z toho důvodu, že většinou požadujeme, aby kontrolky, které v naší aplikaci opakovaně využíváme k podobné funkci, vypadaly podobně a tím umožňovaly větší intuitivnost a přehlednost. Například pokud bychom chtěli, aby text na všech tlačítkách měl stejnou barvu. Ve WPF můžeme použít styly.

Styl si můžeme představit jako pojmenovanou skupinu vlastností, které nemusíme znovu a znovu definovat, ale můžeme je opakovaně použít na různé prvky aplikace. Styl je kolekce hodnot vlastností, které se aplikují na elementy.

Mezi hlavní výhody určitě bude patřit jednoduchost změny některé vlastnosti stylu, která se nám poté promítne do všech prvků, které styl obsahují, a není tedy nutné tuto změnu provádět u každého prvku zvlášť.

2.4.2 Setters

Styly fungují obdobně jako CSS u webových stránek. Nejdříve se mezi elementy `<Style x:Key=“..“> ... </Style>` definují vlastnosti, které mají být ve stylu obsaženy. Atribut „x:Key“ obsahuje název stylu. A pomocí elementů vložených do elementu „Style“ `<Setter Property=“...“ Value=“...“ />` nastavíme jednotlivé hodnoty vlastností ve stylu.

Třída Style je definovaná ve třídě *System.Windows*, odvozuje se ze třídy *Object*. Nejdůležitější vlastností je *Setter*, což je vlastnost obsahu třídy *Style*, proto jsou elementy *Setter* a *EventSetter* odvozeny z elementu Style. Dvě základní vlastnosti třídy *Setter* jsou *Property* (typu *DependencyProperty*, viz dále) a *Value* (typ *Object*).(1)

Celý styl se pak vloží do tzv. „Resources“, což je úložiště kam budeme ukládat různé objekty a poté k nim pomocí jejich jména (*x:Key*) můžeme přistupovat. Každá kontrolka může mít své úložiště. Hledání stylu probíhá nejprve v *Resourcech* rodičovského elementu, pokud zde není nalezen, postupuje se výše.

2.4.3 Dependency properties

Jedná se o speciální vlastnosti kontrolek, oproti standardním vlastnostem (properties) jsou rozšířeny o další funkce. Lze pro jejich nastavení použít *property trigger*. Hodnota této vlastnosti může také být závislá na hodnotách vlastností nadřazených, či jiných elementů.

2.4.4 Dědičnost

Styly mohou vzájemně dědit vlastnosti jeden od druhého. Atribut „BaseOn“ určí, z kterého stylu budou vlastnosti zděděny. Dědění lze provést i z předem definovaných tzv. obecných stylů. Tyto styly slouží k implicitnímu nastavení vlastností a chování jednotlivých kontrolek. Styl, ze kterého dědíme, musí být vždy nadefinován první.

2.5 Triggers (spoušť)

Umožňují nám reagovat na události kontrolek, datové události, nebo hodnoty vlastností. Jedná se o další důležitou vlastnost elementu Style. Na rozdíl od *Setteru* nastavují vlastnost pouze tehdy, když se něco stane. Definice *Triggeru* musí být umístěna uvnitř stylu.

2.5.1 Property triggers

Reagují na hodnoty *dependency properties*. Například jedna z těchto vlastností je *Button.isMouseOver*. V našem příkladě bude-li myš nad tlačítkem, popředí tlačítka se přebarví na zeleno (viz níže).(9)

```
<Style.Triggers>
<Trigger     Property="Button.IsMouseOver"     Value="True">
      <Setter Property="Button.Foreground" Value="Green" />
</Trigger>
</Style.Triggers>
```

Lze použít i tzv. *multitrigger*, který umožňuje zadat více podmínek pro provedení akce.

2.5.2 Event triggers

Umožňuje práci s animacemi a obecně urychluje práci s nimi. V praxi se používá ke spouštění animací, či ovládání jejich průběhu.

```
<Button.Triggers>
      <EventTrigger RoutedEvent="Button.Click">
            <BeginStoryboard ...>
                  ...
            </BeginStoryBoard>
      </EventTrigger>
</Button.Triggers>
```

2.5.3 Data triggers

Umožňují nám reagovat na změnu hodnoty uživatelských proměnných, podobně jako *property trigger* na vlastnost „dependency property". Je podobná třídě *Trigger*, pouze nahrazuje atribut *Property* atributem vazby *Binding*. Vazba se obecně nastavuje na jiný element. *DataTrigger* umí nastavit vlastnost, jestliže vazba bude mít konkrétní hodnotu.

```
<Style TargetType="{x:Type Button}">
<Style.Triggers>
      <DataTrigger Binding="{Binding ElementName=textBox,
      Path=Text.Length}" Value="0" />
            <Setter Property="IsEnabled" Value="False" />
      </DataTrigger>
</Style.Triggers>
</Style>
```

Příklad popisuje element s názvem *textBox*, který má vlastnost *Length* vlastnosti *Text* v prvku *TextBox*. Je-li hodnota rovna 0, tedy v *TextBoxu* není žádný text, deaktivuje se tlačítko. Tedy pokud uživatel nezadal do *TextBoxu* text, není možné stisknout např. tlačítko OK.(1)

Lze zde použít *MultiDataTrigger* pro nastavení více podmínek obdobně jako *MultiTrigger*.

2.6 Templates (šablony)

Pomocí *templates* můžeme vzhled celého prvku kompletně přepsat. Jinými slovy, můžeme zcela změnit, z jakých prvků se daná komponenta bude skládat. Všechny prvky definované ve Windows Presentation Foundation, které mají nějaký vzhled, obsahují vlastnost *Templates*, nastavenou na objekt typu *ControlTemplate*.

Například tlačítko, jehož standardní vzhled je dán obdélníkem a *labelem*, můžeme pomocí vytvoření nové *template* nadefinovat třeba jako elipsu s obrázkem uprostřed.

Obdobně se dá nastavit i chování elementu, například pomocí *Triggerů* umístěných v *ControlTemplate*.

Pro nás bude mnohem důležitější šablona typu *DataTemplate*, která primárně slouží jako datová šablona pro elementy a lze v ní tudíž využít Data Bindingu.

```
<DataTemplate x:Key="DataTemplateListBox">
    <TextBox x:Name="textBox" Text="{Binding Mode=Default, XPath=@name}" TextWrapping="Wrap"
    Background="#FFC5C5C5" IsEnabled="True"/>
    <DataTemplate.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="IsEnabled" TargetName="textBox" Value="False"/>
        </Trigger>
    </DataTemplate.Triggers>
</DataTemplate>
```

Obrázek 7 : DataTemplate

Stejně jako *ControlTemplate* i *DataTemplate* obvykle začíná stromem viditelných objektů, v našem případě jde o *TextBox*. Atribut Text je pomocí vlastnosti *XPath* navázán na proměnou *name* (v našem případě jde o atribut v XML souboru, to že jde o atribut, vyjadřuje znak - @). Dále je využit *Property Trigger*, který deaktivuje *textBox*, pokud je nad ním kurzor myši. Na dalším obrázku je definice Datového zdroje jako XML souboru, tento zdroj využívá již zmíněná vlastnost *XPath*.

```
<XmlDataProvider x:Key="videotekaDataSource" Source="videoteka.xml" d:IsDataSource="True"/>
```

Obrázek 8 : XMLDataProvider

Zdrojem dat je soubor *videoteka.xml* a název datového zdroje zní *videotekaDataSource*.

3 Data ve WPF

3.1 Pohled na data

Jedním z nejčastějších úkolů mnoha programátorů je tvorba rozhraní na vkládání, úpravy a prohlížení dat. Klasický přístup spočívá ve vytvoření formuláře, který má prvky odpovídající polím v záznamu z databáze, nebo jiného datového zdroje. Ve *WinForms* byl pro tuto činnost implementován prvek *BindingNavigator*, který umožňoval základní editaci položek v databázi. Tento přístup bylo ale velice obtížné nějak zásadně editovat, či s jeho pomocí vybudovat kvalitní pohled na celková data v databázi. Dále také grafický vzhled zmíněného prvku byl velice jednotvárný a nezajímavý.

WPF nabízí mnoho možností jak data zobrazit s využitím Data Bindingu. První se nabízí využít prvek *ListBox* a s jeho pomocí procházet databázi a vybírat položky pro pohled a editaci.

I vzhled *ListBoxu* lze jednoduše a velice výrazně změnit k nepoznání za užití již zmíněných šablon. Můžeme tím tedy efektivně docílit graficky pokročilého pohledu na data a nemusíme se nechat omezovat předdefinovanými prvky jako byl *BindingNavigator*.

3.2 Data Binding (datové vazby)

3.2.1 Co je Data Binding?

Data Binding je technika propojení prvků a elementů na data. V rámci WPF jde o techniku velice rozsáhlou, která se dá využít prakticky u kteréhokoliv prvku pro jeho napojení na datovou hodnotu. Hodnoty lze získat z XML souborů, struktury objektů, nebo třeba z jiného grafického prvku.

Základními úlohami prvků vždy bylo data zobrazovat uživateli a umožnit tato data editovat. Mnoho z těchto úloh se však v současnosti již zautomatizovalo. V minulosti by si programátor musel napsat napojení logické proměnné na *CheckBox* pomocí vlastního kódu, který by prvek inicializoval dle vlastní proměnné. Dnes již pokročilé přístupy, jimiž je například Data Binding ve WPF, umožňují nastavit vazbu mezi proměnou a grafickým prvkem.

Datová vazba může někdy nahradit správce událostí a lze tak výrazně zjednodušit kód, obzvláště pracujeme-li v XAML. Datové vazby v XAML mohou omezit používání správců událostí v souboru s kódem na pozadí a v některých případech dokonce eliminovat soubor s kódem na pozadí úplně. Výsledkem je pak kód „bez pohyblivých částí“. Vše je pak otázkou správné inicializace a mnohem menší části kódu mohou selhat.“(1)

Správci událostí se tedy stávají skrytými a zmenšuje se prostor pro chyby v jejich obsluze. U datových vazeb existuje „zdroj“ a „cíl“. Zdroj si můžeme představit jako libovolné datové hodnoty a cíl je pak prvek, na který tato data navazujeme. Toto rozdělení však není tak striktní, jak by se mohlo zdát. Často se může stát, že cíl poskytuje data dalšímu prvku a tady se z cíle stává zdroj.

3.2.2 Možnosti vazeb

Datové vazby mohou být vytvořeny mezi mnoha elementy. Toto bude podrobně rozebráno a prakticky ukázáno v poslední kapitole popisující vlastní vývoj aplikací, používajících Data Binding.

Způsoby napojování by se daly shrnout do čtyř větších oblastí. První jsou vazby mezi jednotlivými grafickými prvky, další jsou vazby na zdroje dat v podobě databáze, nebo xml souboru a poslední větší oblastí je vazba na strukturu objektů.

Z vyjmenovaných způsobů vyplývá, že v podstatě kdekoliv, kde potřebujeme zobrazit, nebo jinak editovat data, je možné tuto technologii využít.

3.2.3 Syntaxe Data Bindingu

```
<Label Content="{Binding ElementName=listBox, Path=SelectedIndex}"></Label>
```

Obrázek 9 : Data Binding zápis 1

Binding je tzv. rozšiřující značka. Definici vazby ohraničují složené závorky, *ElementName* i *Path* patří do množiny vlastností třídy *Binding*. *ElementName* vyjadřuje název elementu, v našem případě *ListBoxu*, a *Path* vyjadřuje vlastnost, na kterou se vztahujeme. V našem případě *listBox.SelectedIndex*. Vlastnost vyjádřenou v elementu *Path* lze zapsat i za užití klasické tečkové notace, proto je nazvána *Path*.

Uvnitř složených závorek platí jiná pravidla, než pro běžnou syntaxi XML. Největším rozdílem je, že uvnitř relace ohraničené složenými závorkami jsou vlastnosti oddělené čárkou a **nesmí v ní být uvozovky**.

Existuje ale i druhý způsob zápisu, kde uvozovky naopak být musí. Ve druhém případě jde sice o delší variantu, ale platí v ní klasická pravidla pro psaní XML.

```
<Label>
    <Label.Content>
        <Binding ElementName="listBox" Path="SelectedIndex" />
    </Label.Content>
</Label>
```

Obrázek 10 : Data Binding zápis 2

Další vlastností, kterou je možno použít vedle *ElementName* a *Path*, je vlastnost *Mode*. Tato vlastnost vyjadřuje režim, který chceme pro komunikaci používat.

3.2.4 Režimy datových vazeb

Režimy vyjadřují, jakým způsobem a jakým směrem bude komunikace probíhat. Režimy jsou celkem čtyři a jsou nastaveny pomocí vlastnosti *Mode*.

OneWay znamená, že komunikace probíhá pouze jednosměrně, jinými slovy aktualizuje pouze cíl ze zdroje.

OneTime je obdoba komunikace *OneWay*, jen s tím rozdílem, že cíl je inicializován ze zdroje, ale pouze jednou. Změny zdroje se pak již v cíly neprojeví.

OneWayToSource je opak běžného uvažování a opak režimu *OneWay*. Komunikace je opět jednosměrná, ale v tomto případě je cíl aktualizován ze zdroje.

TwoWay vyjadřuje komunikaci oběma směry, tedy zdroj mění cíl, ale i pokud se změní hodnota cíle, dojde k aktualizaci zdroje.(7)

3.2.5 Chování datových vazeb

Další vlastností, která se může v bloku *Binding* objevit, je U*pdateSourceTrigger*. Tato vlastnost určuje, kdy bude provedena aktualizace zdroje. Existují tři hodnoty, které lze této vlastnosti přiřadit.

LostFocus je implicitní nastavení pro prvek *TextBox*. Hodnota zdroje se aktualizuje, jakmile přestane být daný prvek aktivní.

PropertyChanged je běžná pro většinu vlastností. K aktualizaci dojde ihned po změně vázané hodnoty.

Explicit využijeme, pokud má být hodnota aktualizována až po potvrzení uživatelem. Aktualizaci je nutné v tomto případě explicitně vyvolat zavoláním metody *UpdateSource().*

3.2.6 Converter (konvertory)

Pří napojování dat může být někdy potřebné tato data převést z jednoho typu na jiný. Nemusí se jednat o pouhé přetypování tak, jak ho známe z běžných programovacích jazyků, ale převod může být definován zcela dle vůle programátora.

Třída *Binding* obsahuje vlastnost s názvem *Converter*, kde můžeme zadat vlastní třídu, která implementuje rozhraní *IValueConverter* a tedy obsahuje metody *Convert* a *ConvertBack* jež převod provedou.

```
<Window.Resources>
    <src:DurationTypeConvertor x:Key="DurationConvert"/>
    <src:StringToImageConvertor x:Key="StringToImageConvertor"/>
```

Obrázek 11 : Converter definice

Nastavení *Converteru* v XAMLu probíhá v sekci *Resources* pomocí předem definovaného prefixu src. Tento prefix odkazuje na kód v C# a blíže je vysvětlen v kapitole 2.3.5 Dokument v XAML.

```
<Label Content="{Binding Converter={StaticResource DurationConvert}, Mode=Default, XPath=duration}"
```

Obrázek 12 : Converter vazba

Na dalším obrázku je patrné, jak se definuje samotná vazba na *Converter*. Použije se vlastnost *Converter*, které se přiřadí hodnota ze *StaticResource*, kterou je název *Converteru*, který jsme definovali výše.

3.3 Modely a návrhové vzory

Pokud pracujeme s WPF poprvé a již máme nějaké zkušenosti s jinými programovacími jazyky, postupujeme většinou následujícím způsobem. Vytvoříme si strom grafických elementů, vygenerujeme *handlery* a poté v kódu funkční části aplikace, např. C# ručně naplníme nebo přečteme jednotlivé hodnoty prvků.

Tento přístup je sice možný, ale nese řadu závažných problémů. *View* a *code behind* neboli XAML a část C# jsou velice těsně svázány a přímo se odkazují na jednotlivé grafické elementy, tedy například při změně názvu elementu program okamžitě přestává fungovat. Dále se *view* stává úložištěm dat, ale k této funkci není určen, má data pouze zobrazovat. Další nevýhodou je obtížnost testování kódu na pozadí tím, že k němu přistupujeme přes grafické elementy.

Jak již z textu vyplývá, není tento postup úplně ideální, zvláště pokud máme k dispozici velice rozsáhlé a pokročile možnosti oboustranného Data Bindingu.

Řešením je použít některý z návrhových vzorů

Tato práce není zaměřená na detailní rozbor a vysvětlení jednotlivých návrhových vzorů, ale vysvětluje funkci, která je pro návrhový vzor MVVM zcela zásadní. Touto funkcí je Data Binding. Jakmile použijeme Data Binding pro navázání dat na grafickou část programu již jsme vytvořili dvě oddělené části, *View* a *Model*.

3.3.1 MVC (Model, View, Controller)

Architektura MVC se dá definovat jedním odstavcem.

„Architektura MVC dělí aplikaci na 3 logické části tak, aby je šlo upravovat samostatně a dopad změn byl na ostatní části co nejmenší. Tyto tři části jsou *Model*, *View* a *Controller*. *Model* reprezentuje data a logiku aplikace, *View* zobrazuje uživatelské rozhraní a *Controller* má na starosti tok událostí v aplikaci a obecně aplikační logiku.“(16)

V praxi je ale toto právě největším problémem při vývoji aplikací s užitím tohoto návrhového vzoru. Často není jasné, kam právě tvořená část kódu spadá, zda se jedná spíše o *model*, *view* a nebo *controller*.

Přestože se jedná o velice používaný a široce rozšířený návrhový vzor, jeho využití ve WPF není zcela vhodné, protože nevyužívá plně možností, které WPF nabízí – například velice silnou funkci oboustranného Data Bindingu.

3.3.2 MVVM (Model, View, ViewModel)

Tento návrhový vzor vychází z vzoru MVC a byl vytvořen architektem WPF/Silverlight Johnem Grossmanem. Celé prostředí MS Expression Blend je napsáno s užitím tohoto vzoru.

Smysl a účel *View* a Modelu zůstává stejný jako u vzoru MVC, jenom s tím rozdílem, že *View* využívá možnost obousměrného Data Bindingu.

ViewModel představuje prostředníka, který adaptuje *Model* pro potřeby *View*. Je zodpovědný za stav *View*, volá služby nebo model přes jejich Interface. A vystavuje veřejné vlastnosti, které jsou v XAML obousměrně vázány. Pro jednoduché vlastnosti je vhodné použít typ *DependencyProperty*, pro kolekci prvků *ObservableCollection*<T> a příkazy, které volají metody modelu nebo služeb by měli být zpřístupněny přes implementované rozhraní *ICommand*.(15)

Tento postup nám umožní zachovat čistý kód v pozadí a velkou nezávislost grafické části na části funkční.

Ještě jednou zopakujeme, že *View* je grafická část programu, *Model* je její funkční část, neboli vlastní data, která budeme zobrazovat a *ViewModel* je jakýsi prostředník mezi *View* a *Modelem*, jinými slovy konvertor, který data převádí do podoby vhodné k zobrazení.

4 Užitý software

4.1 MS Expression Blend

Microsoft Expression Blend je XAML WYSIWYG (What You See Is What You Get) [4] editor. Zjednodušeně řečeno umožňuje téměř vše, co jsme až doteď psali XAML kódem udělat za pomoci grafického editoru.

Tento program umožňuje ale mnohem více, než zde bylo uvedeno; jeho největší předností je možnost tvorby pokročilých animací a grafických prvků, a pro tuto práci bude nejdůležitější jeho schopnost vytvářet datové vazby velice intuitivní a snadno pochopitelnou metodou téměř bez nutnosti znát XAML.

Dále umožňuje kód XAML nejen vytvářet za pomoci grafického prostředí, ale i přímo editovat a ihned zobrazit provedené změny. Vývoj v Blendu se dá přirovnat k vývoji Flash aplikací.

Tento nástroj je ale také velmi složitý a to s sebou nese i určité nevýhody. Hlavně se jedná o někdy ne zcela předvídatelné chování. Pokud se například během datové vazby nezobrazí ihned hodnoty do náhledu aplikace, nemusí to nutně znamenat chybné nastavení, ale pouze to, že Blend ještě neaktualizoval všechny vazby. Zde pomůže přeložit a spustit aplikaci, v ní již vše bude fungovat a po jejím zavření se správná data zobrazí i v náhledu.

Microsoft Expression Blend je, jak už název napovídá, součástí většího balíku Microsoft Expression, který obsahuje dále nástroje (MS Silverlight) k editaci aplikací primárně určených pro web.

Nejnovější verzí je Microsoft Expression 3. Tato verze obsahuje nové pokročilé funkce, například podporuje soubory Adobe Ilustratoru a Adobe Photoshopu, které lze v prostředí MS Blend dále editovat a využívat.

Na záložce Help je odkaz na stránku s online tutoriály, kde jsou k dispozici videa popisující aplikaci Blend, ale i ostatní aplikace z balíku Expression. V záložce Samples je k dispozici několik ukázkových WPF aplikací.

[4] Česky „co vidíš, to dostaneš“.

4.1.1 Stažení

Nejnovější verzi produktu Microsoft Expression Blend je možné získat v rámci balíčku Microsoft Expression 3. Jeho trial verzi funkční 60 dní je možné stáhnout zde:

- http://www.microsoft.com/Expression/try-it/default.aspx
- http://www.microsoft.com/downloads/details.aspx?FamilyID=7e2f033b-c6b5-4565-93a5-a6048246ce28&displaylang=en

Dále existuje pro studenty možnost stáhnout plnou verzi tohoto programu. Je nutno se ovšem prokázat buď ISIC kartou, nebo být součástí školy, která patří do tohoto programu. Tuto verzi nelze používat komerčně.

- https://www.dreamspark.com/default.aspx – možnost stažení plné verze pro studenty.

4.1.2 Instalace

Instalace nástroje MS Expression 3 probíhá standardním způsobem.

Při instalaci pro vývoj WPF aplikací stačí ponechat zaškrtnutou možnost Expression Blend 3 + SketchFlow.

4.1.3 Požadavky na systém

- **Operační systém:** Windows 7, Windows Vista, Windows XP SP2
- Windows® Media Player 11 (pouze pro Windows XP)
- Procesor 1 GHz
- 1 GB RAM
- 2 GB místa na disku
- .NET Framework 3.5 Service Pack 1
- Podporu pro Microsoft DirectX® 9.0
- 1024 x 768 monitor s 24-bit barvami

4.2 MS Visual Studio

Je vývojové prostředí od Microsoftu, které může být použito jak pro vývoj konzolových aplikací, tak aplikací s grafickým rozhraním. Lze s ním vyvíjet klasické formulářové aplikace i aplikace používající WPF. Jde o asi nejpoužívanější nástroj pro vývoj programů pro .NetFramework.

Pro vývoj grafické části programu ve WPF formou WYSIWYG ale není příliš vhodný, protože neumožňuje editaci stylů, tvorbu animací a další pokročilé funkce jinak

než ruční editací kódu v XAML. Z toho důvodu je vhodné užívat kombinaci obou zmíněných nástrojů: MS Blend pro vývoj grafické části a nastavení datových vazeb, a Visual Studio pro vývoj funkční části programu a editaci kódu C#. A stejně tak budeme postupovat i v ukázkových příkladech v této práci; grafická část a napojování dat bude realizováno v MS Blendu a funkční část a tvorba objektové struktury, na kterou se budeme napojovat, bude realizována v C# pomocí MS Visual Studia 2008.

4.2.1 MS SQL Server

Při instalaci Visual Studia ponechte zaškrtnutou možnost instalace Microsoft SQL Server 2008, tím dojde i k instalaci SQL serveru. Jedná se o používaný databázový server, který využijeme ve třetím příkladu.

4.2.1.1 Dodatečné stažení

Při instalaci Visual Studia je možnost automatické instalace i SQL Serveru, tedy v tomto případě již není nutná dodatečná instalace SQL Serveru.

- http://www.microsoft.com/downloads/details.aspx?familyid=220549b5-0b07-4448-8848-dcc397514b41&displaylang=en – MS SQL Server 2005 Express
- http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=7522a683-4cb2-454e-b908-e805e9bd4e28 – MS SQL Server 2008 Express

4.2.1.2 Požadavky na systém

- **Operační systém:** Windows Server 2003 Service Pack 2, Windows Server 2008, Windows Vista, Windows Vista Service Pack 1, Windows XP Service Pack 2, Windows XP Service Pack 3
- 32-Bit Systémy: 1 GHz
- 64-Bit Systémy: 1.4 GHz
- Minimum 512 MB RAM
- 1 GB místa na disku

4.2.2 Stažení

Současnou nejnovější verzi Microsoft Visual Studio 2008 lze stáhnout ve formě express verze, která skýtá některá omezení oproti plné verzi, ale pro studijní účely je použitelná, je nutno ji do 30 dnů zdarma registrovat.

- http://www.microsoft.com/express/Downloads/#2008-Visual-CS – Visual C# 2008, Express verze, obsahuje pouze podporu jazyka C#.

Další možností je opět stažení přes systém dreamspark, kde si opět studenti mohou stáhnout plnou verzi programu. Tato verze se nesmí používat komerčně.

- https://www.dreamspark.com/default.aspx – možnost stažení plné verze pro studenty.

4.2.3 Instalace

Instalace nástroje Visual Studio 2008 probíhá standardním způsobem. Důležité je, pokud budeme pracovat s databází a to v našich příkladech budeme, při instalaci ponechat zaškrtnutou možnost instalace Microsoft SQL Server 2008. Produkt SQL Server je možné doinstalovat dodatečně.

Při použití Express verze Visual Studia během instalace dochází ke stahování instalačních dat, dále je nutné se do 30 dnů zdarma zaregistrovat.

4.2.4 Požadavky na systém

- **Operační systém:** Windows XP Service Pack 2, Windows Server 2003 Service Pack, Windows Server 2003 R2, Windows Vista, Windows Server 2008
- Minimum: 1.6 GHz CPU, 384 MB RAM, 1024x768 display, 5400 RPM
- Doporučeno: 2.2 GHz CPU, 1024 MB RAM, 1280x1024 display, 7200 RPM

5 Pedagogická část

5.1 Cíl výuky, uplatnění v praxi

Student, který projde e-learningovým výukovým kurzem, by měl být schopen ve WPF téměř bez znalosti kódu XAML propojit grafickou a funkční část programu. Díky tomu by měl později zvládnout zcela předělat grafické rozhraní nebo funkční část programu za předpokladu zachování datových vazeb.

Měl by zvládnout použít technologii datových vazeb „Data Binding“, která je obsažena ve WPF a která umožňuje napojení funkční části programu na jeho grafickou část. Ať už se jedná o data v databázi, v XML souboru, nebo jen strukturu objektů, kterou je třeba s GUI[5] propojit při zachování oddělenosti grafické a funkcionální části programu.

Studium je doplněno řešenými příklady, příklady k procvičení a výkladovými audiovizuálními soubory.

Po prostudování práce a zvládnutí příkladů bude student schopen samostatně vytvořit ve zmíněném WPF grafickou aplikaci. Dále bude tuto aplikaci schopen efektivně propojit s daty v libovolné podobě. Tato práce mu tedy umožňuje užít Data Bindingu k napojení dat z libovolného úložiště, nejedná se pouze o data jako taková, ale pokud student ještě k této problematice zvládne i objektově orientované programování, bude schopen propojit i funkční část aplikace.

Dále student pochopí klíčovou součást návrhového vzoru MVVM, kterou právě oboustranný Data Binding je.

V praxi se technologie běžně využívá při tvorbě moderně designovaných programů; například je tímto způsobem vytvořená většina grafických částí systému Windows Vista a Windows7. Absolventi tohoto e-learningového kurzu mohou využít své znalosti v oborech zaměřených na IT a vývoj softwaru.

Tuto technologii může student využít i při programování v jazyce C#, Visual Basic, C++, které je součástí aplikací ve WPF, nebo Silverlight.

Student se seznámí s grafickým nástrojem MS Blend 3 od firmy Microsoft umožňující psaní aplikací pro Windows, naučí se používat vektorovou grafiku, která umožňuje graficky bezztrátovou změnu velikosti prvků, barevné gradienty, používání geometrických tvarů.

[5] Grafické uživatelské rozhraní.

5.2 Cílová skupina studentů

Práce je určena především studentům programovacích jazyků, začínajícím a mírně až středně pokročilým, kteří zvládají základy programování například v jazyce C#, Java, Visual Basic. Ale využít ji mohou i pokročilejší studenti, kteří doposud používali C# a WinForms a chtějí rozšířit své znalosti na další oblast tvorby grafického rozhraní.

Jejich oborem studia, nebo zájmu jsou informační technologie.

Návaznost studia – absolventi tohoto studia mohou své schopnosti dále rozvíjet tvorbou vlastních aplikací, ve kterých budou postupně objevovat další možnosti tvorby designu aplikace. Dále mohou prostudovat další části technologie WPF, ať už se jedná o hlubší pochopení animací, stylů, nebo jiných možností týkajících se vzhledu aplikace, které jim umožní vytvářet stále pokročilejší a profesionálnější design aplikace. Mohou také hlubším studiem programování v jazyce C#, nebo Visual Basic rozvíjet své znalosti funkční části programu a vytvářet aplikace složitější nejen po grafické, ale i funkční stránce.

Dále tato práce může přinést užitek všem zájemcům o programování, kteří by si rádi zlepšili uživatelské prostředí některých svých programů, které si vytvořili a běžně je využívají ve své práci, denním životě či ve chvílích odpočinku, v zábavě.

5.3 Metody výuky

Sada E-learningových studijních materiálů je zpracována tak, aby vedla studenty od jednodušších příkladů k náročnějším; a zároveň klade důraz na samostatnost a kreativitu studenta. Metody výuky jsou dobře použitelné v praxi a rozvíjejí poznávací proces studenta.

E-learningové studijní materiály využívají několik osvědčených přístupů k výuce:

- *Metoda práce s textem* – Je metoda založená na zpracování textových informací, v našem případě prostudování a vyzkoušení si řešených příkladů, směřující k pochopení a postupnému rozšiřování nových poznatků, dále k jejich prohloubení a osvojení. Hlavní je zde studentovo samostatné učení, při kterém rozvíjí své myšlení, představivost, schopnost zapamatování. Při této práci je důležité, aby žák porozuměl zadáním i řešením příkladů, proto je důležité postupovat od jednoduchých a názorných příkladů ke složitějším. Důležitou součástí práce jsou jednotlivá zadání úloh, dále jejich vysvětlení krok po kroku a možné řešení.
- *Problémový přístup k výuce* – Studenti v každé oblasti lépe vstřebávají učivo, které jim je předkládáno formou, při níž musí aktivně spolupracovat a snažit se najít řešení problému samostatně. Tím je zákonitě získaní takovýchto poznatků trvalejší, student je veden k hledání samostatných řešení. Student musí projít následujícími fázemi: analýza problému, nalezení hlavního jádra problému, stanovení hypotéz, jejich ověření a vyřešení…

  I pokusy, které se nezdaří, jsou cenné z hlediska vzdělávacího. Vedou k pokusu o hledání nového řešení.
- *Algoritmus* – Z hlediska metod práce je též důležitou vyučovací metodou algoritmus – neboli systém instrukcí, pravidel, příkazů, jimiž se v určité fázi řešení problému musí student řídit. Základní instrukce musí být srozumitelné a jednoznačné, použitelné pro konkrétní fázi vyučovacího příkladu. Algoritmy mají své významné místo právě v programové výuce. Z řešení jednotlivých příkladů je znát, že je důležité si osvojit algoritmus specifický pro konkrétní model.
- *Projektové učení* – V závěru mé práce má student samostatně zpracovat rozsáhlejší příklad, ke kterému musí určité poznatky získat z internetu, z běžného denního života a též musí svoji práci prezentovat před dalšími studenty. Toto jsou specifika projektového učení, ve kterém dochází k reálnému využití získaných dovedností,

technologií. Výstup je zaměřen na prezentaci a trénink důležité schopnosti obhájit svou práci.

- *Videonahrávka* – Vlastní práce je doplněna audiovizuálními soubory, které umožní studentovy sledovat vývoj aplikace krok po kroku. Umožní mu to tedy, pokud by se nezvládl držet postupu v textové podobě, najít řešení svého problému a pokračovat v dalším studiu.
- Všechny tyto metody rozvíjející osobnost studenta, usilují o to, aby student aktivně vstupoval do procesu poznávání, zapojil se do řešení problému, aby odhaloval vztahy, srovnával je a vyvozoval z nich závěry. (17)

5.4 Formy výuky

Základní formou absolvování výuky e-learningového kurzu je samostatná práce studenta, při které se výrazně rozvíjejí klíčové kompetence k učení a k řešení problémů. Samostatná práce je nezbytný prostředek aktivizace a zapojení studenta do vyučovacího procesu, má za úkol prohloubit samostatnost a připravenost studentů pro další studium i praxi.

Jsou zde využity následující formy samostatné práce:

- Samostatná práce podle vzoru – je připraven soubor několika analogických příkladů, které jsou vzorově vyřešeny a které má student pochopit a prostudovat.
- Samostatná práce rekonstrukční, kdy má student najít vlastní, originální řešení problému podle naznačeného logického postupu. Konkrétně jsou zde připraveny částečně řešené příklady, které má student samostatně dořešit, vybrat postupy, které jsou mu známy z předchozích řešených cvičení.
- Samostatná práce variantní, kdy student musí vyřešit s použitím vlastních postupů složitější úlohy, se kterými se ještě nesetkal, musí aplikovat své nabyté znalosti na problémovou úlohu.
- Samostatná práce tvořivá, ve které musí student na závěr prokázat, že zvládne aplikovat vědomosti v kombinaci s poznatky z jiných oborů.
- Zpočátku jsou formulovány úkoly jednoduché, návody na řešení jsou jasné a srozumitelné. Později jsou návody na řešení stručnější, student by měl již na základě svých zkušeností porozumět i zkratkovitému naznačení problému.

5.5 Délka trvání studia

Studium kurzu je rozloženo do několika úvodních kapitol:

- Teoretický úvod do Windows Presentation Foundation
- Způsoby práce s daty ve WPF
- Přehled použitého softwaru a možnost jeho stažení

Zběžné prostudování teoretického úvodu by mělo studentovi trvat zhruba 60 minut. Nastudování a pochopení způsobu práce s daty také přibližně 60 minut. Opatření použitého softwaru v závislosti na rychlosti a zatížení internetového připojení kolem 60 minut.

Nejdůležitější část výuky probíhá formou řešených příkladů, které jsou doplněny o odpovídající teoretickou část a výuka je rozdělena do těchto kapitol.

- Data Binding na XML a na Controls
- Data Binding na XML a užití Konvertoru
- Data Binding na databázi
- Data Binding na strukturu objektů

Důležitou součástí e-learning studijních materiálů jsou i ukázková videa, mapující vývoj řešených příkladů a vlastní vyřešené příklady včetně zdrojových kódů.

Samostatné příklady je nutné procvičit v prostředí MS Expression Blend a Visual Studio a studentovi by mělo stačit několik desítek minut na seznámení se s nim..

Prostudování prvního řešeného příkladu by mělo začínajícímu programátorovi trvat přibližně 90 minut, pokročilému studentovi zhruba o 15 minut méně.

Druhá kapitola je přibližně stejně časově náročná, k celkovému prostudování včetně řešení příkladu by mělo opět postačit 90 minut.

Kapitola třetí a čtvrtá je asi o 30 minut časově náročnější.

Nejnáročnější na čas je závěrečná samostatná práce studenta, která obnáší cca 5 hodin samostatného studia a řešení problému.

Celkovou dobu studia potřebnou ke zvládnutí daného problému odhaduji na 1 až 2 týdny intenzivního samostudia.

5.6 Řešené příklady – metodika

Příklady postupují od jednodušších ke složitějším z hlediska složitosti aplikace. Přesto příklady zpracovávají jedno téma různými způsoby. Příklady se liší především užitím jiných datových zdrojů. Student tak bude schopen porovnat výhody a nevýhody jednotlivých postupů a bude tyto postupy schopen kombinovat. Všechny příklady jsou

doplněny audiovizuálními soubory pro samostatné studium a vyřešenými ukázkovými příklady včetně zdrojových kódů.

V prvních dvou příkladech student získá základní orientaci v prostředí a naučí se využívat základních grafických prostředků, které WPF nabízí. Dále se v těchto příkladech seznámí se základy datových vazeb, naučí se je tvořit, upravovat a vázat data na grafické prvky, ale také vázat tyto grafické prvky mezi sebou a použít konvertor pro převod dat do vhodných podob.

První příklad je vysvětlen velice podrobným způsobem, aby student získal již zmíněné základní dovednosti a byl schopen samostatné práce. V dalších příkladech je již detailně popisovaná pouze nová problematika a opakující se problémy řeší již student sám, za pomoci několika opěrných bodů.

Třetím příkladem student pochopí další oblast užití Data Bindingu a tou je napojení na databázi ve formě SQL databáze. Tento problém je studentovi vysvětlen a je veden aby vytvořil datový zdroj z již zmíněné databáze, ale další řešení již provádí samostatně. K dispozici má video a vzorové řešení.

Ve čtvrtém případě je jako zdroj dat využita struktura objektů, student tedy pochopí poslední velkou část užití datových vazeb. Získá tedy celkový přehled využití této technologie a je schopen tuto technologii v praxi sám efektivně využívat.

5.6.1 Příklad 1, XML1

V prvním příkladu si student zopakuje některé teoretické znalosti uvedené v teoretické části práce a srovná užití v konkrétním případě. Jde o jazyk XAML, který již byl vysvětlen na začátku práce a v tomto případě má pochopit jeho konkrétní použití, přestože kód v tomto jazyce je generován za užití nástroje MS Blend.

Nejprve zde student získá informace o jazyku XML a jeho základních pravidlech. Dále pak vytvoří, nebo použije vzorový XML soubor s ukázkovými daty. Z těchto dat za využití nástrojů, které MS Blend nabízí, vytvoří datový zdroj.

Student se také naučí základní orientaci a práci v již zmíněném prostředí MS Blend. Seznámí se také s jednotlivými základním prvky, které lze použít ke zobrazování dat. S jeho pomocí také navrhne a vytvoří design aplikace.

Jakmile se dostatečně seznámí s jednotlivými základními grafickými prvky, provede vlastní napojení dat z datového zdroje na právě definované grafické prvky za užití Data Bindingu. Dále provede datové napojení jednotlivých grafických prvků mezi sebou a naučí se toto využívat.

Seznámí se s tvorbou a užitím datových šablon a stylů.

Student je v tomto příkladě veden krok po kroku, aby získal základní orientaci v prostředí a pochopil funkci jednotlivých prvků. Není ale nucen se jimi zcela přesně řídit má prostor k vlastní tvorbě a to především, pokud jde o vzhled aplikace.

Získá tedy základní znalosti pro vlastní samostatnou práci, které jsou postupně rozvíjeny v dalších ukázkových příkladech, kde jsou již opakující se části vysvětleny pouze stručně a student musí na jejich řešení přijít sám.

5.6.2 Příklad 2, XML2

V dalším příkladu student doplní své znalosti a možnost tvorby animací. Rozšíří tedy své znalosti týkající se užití stylů a triggerů a získá hlubší pochopení grafických možností ve WPF.

Dále se student naučí pokročilejší práci s Data Bindingem, tedy navazování dat za užití konvertoru, který je uzpůsobí pro potřeby náhledu. Umožní mu to převod dat na vhodný formát ke zobrazení.

V tomto příkladu tedy získá základní dovednosti pro tvorbu pokročilého grafického rozhraní ve WPF. A již pochopené základní principy navazování dat obohatí o používání konvertorů dat.

5.6.3 Příklad 3, SQL

Student v tomto příkladu získá zkušenosti s použitím tříd sloužících k navázání dat z SQL databáze. Naučí se vytvořit mezitřídu, která provede převod dat z databáze do vhodné podoby k zobrazení. Pochopí základní principy práce s databází za pomocí předdefinovaných nástrojů. Naučí se využít objektově relační mapování za pomoci nástroje *Linq to SQL class*.

Seznámí se s základní prací s objekty a navazování těchto objektů na grafickou část aplikace.

5.6.4 Příklad 4, Objekty

Tento příklad vysvětluje poslední větší chybějící oblast využití Data Bindingu. Touto oblastí je napojení vzhledu aplikace přímo na strukturu objektů. Student tím získá základní znalosti týkající se práce s objekty. A naučí se data reprezentující tyto objekty napojit na grafickou část aplikace a to oběma směry.

Dále se naučí základní vstupně výstupní operace, které využije k získání zobrazitelných dat z textového souboru. Také využije možnosti prohledat složku na disku a získat z ní další video soubory.

5.7 Další poznámky ke zpracování didaktické části

Při tvorbě příkladů se zadání ve smyslu řešeného problému nemění, ale mění se způsob realizace, jak zdroje dat, tak užitého grafického řešení.

Student postupuje od jednodušších příkladů, jak po stránce grafické, tak po datové a postupně se učí nové postupy a možnosti tvorby.

Součástí každého příkladu je popis, detailní zadání, teoretický rozbor problematiky v něm využité a vlastní řešení krok po kroku.

Část zadání se v příkladech vždy opakuje (tvorba projektu, základní design, napojení) a studenti se již tuto pro ně známou část učí dělat sami. V textu je už popsaná jen bodově. Ve videích o něco obsáhleji.

Zadání – videotéka. Bylo zvoleno proto, že je řešeno relativně často a toto studentovi umožní srovnávat jednotlivé přístupy a také problematika zadání je snadno pochopitelná.

Téma je vhodné začlenit do výuky jak pro středoškolské studenty škol zaměřených na informatiku, tak pro vysokoškolské studenty, kteří zvládli základy programování.

Nejaktuálnější informace o probírané problematice lze získat na webových stránkách společnosti Microsoft a v dokumentaci MSDN.

Pro vypracování bakalářské práce bylo nutné nastudovat prostředí MS Visual Studio 2008 a MS Expression Blend 3.

Text je v příloze doplněn o audiovizuální soubory a řešené příklady včetně zdrojových kódů.

6 Řešené příklady

V této části jsou k dispozici čtyři řešené příklady k procvičení a pochopení čtyřech základních užití Data Bindingu. Postup tvorby příkladů je podrobně rozepsán níže a také je možno ho krok po kroku sledovat pomocí přiložených videí. K dispozici jsou také zdrojové kódy k hotovým příkladům.

Příklady postupně zvyšují svou obtížnost a používají stále složitější grafické operace, které WPF poskytuje. Metodika tvorby příkladů je popsána v předchozí kapitole.

Všechny příklady řeší jeden problém různými způsoby. Problémem je vytvořit „videotéku“, neboli kolekci filmů, videí a dalších audiovizuálních souborů v různých úložištích, včetně vybraných souborů na pevném disku. A tuto kolekci efektivně a zajímavě zobrazit. Příklady se liší jak stále složitější grafickou částí, tak především způsobem uložení dat, která budeme zobrazovat. Datové vazby lze použít, jak již bylo zmíněno, na různá datová úložiště, nejsme již tedy omezeni nevzhlednou formulářovou aplikací, kterou lze efektivně navázat jen k databázi.

Data tedy budou uložena v prvních dvou příkladech v takzvaném XML (eXtensible Markup Language) souboru, což je obecný značkovací jazyk, podobný jazyku HTML, s tím rozdílem, že autor si může definovat vlastní *tagy* a tím dát datům svůj vlastní význam. Ve třetím případě budou data uložena v databázi, můžeme si představit tabulku dat, ve které každý sloupec má svůj název, typ a v řádcích jsou uložena data. V posledním případě budou data dynamicky shromažďována, tedy část dat bude v textovém souboru a část dat bude získána prohledáním zvolené části disku, vlastní data, která budeme napojovat, budou reprezentována strukturou objektů.

Příklady jsou doplněny o audiovizuální soubory ilustrující podrobně jejich vývoj. Videa jsou uloženy ve složce „Videa“ a nazvány stejně jako názvy projektů.

Ukázkové příklady byly vytvořeny plnou verzí MS Expression 3 a Visual Studio 2008 pod licencí Academic Alliance. Na systému MS Windows XP. Dále za užití VirtualBoxu byly odzkoušeny i v ostatních uvedených verzích (MS Expression 3 trial, Visual C# expression).

6.1 Společné zadání - videotéka

Vytvořte aplikaci, která bude představovat kolekci video souborů (uložených na pevném disku) a filmů, či jiných audiovizuálních médií v jiných úložištích (DVD, videokazeta, audio…).

U souborů přímo dostupných v počítači bude k dispozici fyzická adresa umístění, u jiného úložiště bude k dispozici přibližné umístění (např.: skříň na DVD)

Datová reprezentace tedy bude obsahovat:

- *Name* – představuje zobrazený název média, může být doplněn o *source*, což odkazuje na zdroj, ze kterého bylo médium získáno, což může být například webová adresa.
- *Format* (např. video soubor, DVD, videokazeta) – představuje typ média.
- *Destination* – umístění média, může být fyzická adresa na disku, nebo přibližné umístění v reálném světě.
- *Author* – autor média.
- *Description* – popis média.
- *Duration* – délka, neboli doba trvání média, udaná v minutách.
- *Language* – jazyk, nebo jazyky média.

6.2 Příklad 1, XML, Templates.

6.2.1 Popis

Příklad napojuje grafickou část aplikace na XML soubor. Zároveň vytváří datové vazby mezi jednotlivými elementy typu *Controls*. Dále ukazuje jednoduché úpravy vzhledu elementů a jejich chování za pomocí jednoduchých animací.

Video s podrobným vývojem programu je ve složce *Videa\videoteka_1.wmv*.

6.2.2 Zadání

- Vytvořte aplikaci „Videoteka_1“ typu WPF aplikace v nástroji MS Expression Blend3. (Kapitola: 6.2.4.1)
- Vytvořte XML soubor, který bude obsahovat konkrétní data zmíněná výše (*Name*, *Format*, *Destination*, *Author*, *Description*, *Duration*, *Language*). (Kapitola: 6.2.4.2)
- Pomocí nástroje Microsoft Expression Blend3 vytvořte vzhled aplikace, který se bude skládat z nového okna, *ListBoxu* a několika prvků typu *Label* a *TextBox*, reprezentujících vlastnosti konkrétního média. (Kapitola: 6.2.4.4)
- Na element *ListBox* za užití Data Bindingu napojte data z XML souboru, tak aby se prvky v *ListBoxu* reprezentovaly pouze názvem (*Name*). (Kapitola: 6.2.4.5)
- Po výběru jednoho prvku *ListBoxu* dojde k aktualizací prvků typu *TextBox*, neboli budou navázány na aktuálně zvolený prvek *ListBoxu* a budou zobrazovat jeho další vlastnosti (Author, Description…). (Kapitola: 6.2.4.7)

6.2.3 Teorie

6.2.3.1 Co je XML?

Jak již bylo řečeno XML, používá podobná pravidla jako jazyk HTML, ale v jazyce HTML musíme z větší části definovat, co se má zobrazit, a nikoliv význam zobrazených dat.

V jazyce XML je to spíše naopak, důležitější je význam dat. Z toho vyplývá, že XML toleruje daleko méně syntaktických chyb, než jazyk HTML, jeho mutace užité na webu jsou jazyky XHTML. Jazyk XML, ale zdaleka není určený jen pro web, využití nalezne všude, kde je třeba data prezentovat v různých formátech, například při zobrazování textu usnadňuje práci sazeči, tím že zaručuje správnost dat.

6.2.3.2 Syntaxe XML

Základním požadavkem je, aby celý XML dokument byl uzavřen v jenom elementu, tedy vše musí být mezi počátečním a ukončovacím tagem.

V XML lze používat atributy podobně, jako HTML, pouze s tím rozdílem, že musí být uzavřeny do uvozovek.

Například `<item name=“meet firefox“...>...</item>.`

V XML jsou všechna jména citlivá na velikost písmen, bývá zvykem je psát všechna malými písmeny.

Dalším důležitým prvkem dokumentu je XML deklarace `<?xml version="1.0" encoding="UTF-8"?>,` která určuje že jde o XML dokument a nastavuje použité kódování znaků, standardně UTF-8.

U dokumentu dále můžeme určit jeho typ pomocí deklarace typu dokumentu DOCTYPE.

6.2.3.3 Data Binding na XML soubor

Jako jednu z prvních věcí, které budeme dělat při navazování dat na grafickou část aplikace je vytvoření kontextu dat, který v tomto případě je přímo napojen na datový zdroj, se kterým budeme pracovat, tento *DataContext* nám pak poskytuje množinu dat, kterou můžeme přiřazovat podřízeným elementům. V našem případě navážeme *DataContext* okna na datový zdroj z XML souboru. Tedy dojde k vytvoření *DataSource* z XML souboru, definice zdroje se umístí do souboru *App.xaml* za užití elementu *XmlDataProvider*.

Vše se v programu MS Blend vygeneruje automaticky a zajistí nám to, že pro všechny podřízené elementy, jako je *ListBox* a jiné máme k dispozici data, ze kterých už jenom můžeme vybírat co a kde zobrazit.

Komunikace s užitím *XmlDataProvideru* je pouze jednosměrná, tedy v tomto příkladě nebude možné XML soubor aktualizovat, ale budeme z něj data pouze načítat.

DataSource lze vytvářet i jinými způsoby, některé si ukážeme v dalších příkladech, za zmínku stojí možnost *Add Sample Data Source*, který nám umožní vytvořit datový zdroj ke zkušebním účelům, můžeme ho definovat a naplnit daty dle našeho uvážení.

6.2.3.4 Data Binding mezi elementy

Další věcí, kterou v tomto příkladě využíváme je navazování jednotlivých elementů vzájemně na sebe. Pro příklad *TextBoxy* představující další informace o našem médiu navážeme na právě zvolený prvek *ListBoxu*, budou tedy zobrazovat informace pouze o právě vybraném audiovizuálním médiu.

Není tedy třeba psát vlastní logiku programu, pokud dojde k výběru jiného prvku seznamu, ale v našem případě stačí pouze proměnou *Text* v *TextBoxu* vhodně napojit na správný datový atribut, který jsme již přiřadili proměnné *SelectedItem* v *ListBoxu*.

Nastavíme tedy *DataContext* obdobně jako u čtení XML souboru, ale tentokrát na prvek *SelectedItem* a již můžeme zobrazit pouze požadovanou hodnotu.

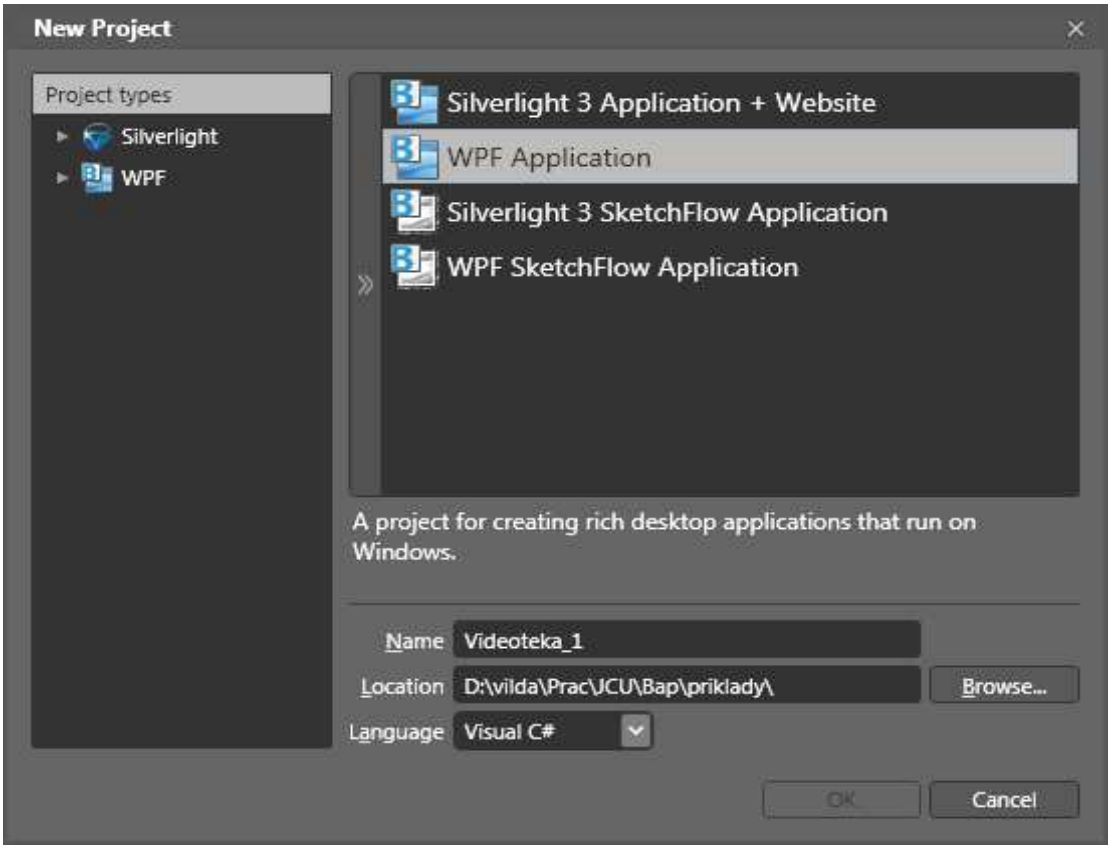
6.2.3.5 Templates

Problematika šablon již byla probrána výše, jenom zopakujme, že existují dva typy šablon, které zde využijeme. Jedná se o šablonu *ControlTemplate*, která slouží k nastavení vzhledu prvku, tento vzhled lze kompletně přepsat. *DataTemplate*, nastavuje datový obsah *Content*, který bude zobrazován, je obecně definován jako typ *Object*.

6.2.4 Řešení

6.2.4.1 Založení projektu

Vytvořte aplikaci „Videoteka_1“ typu WPF aplikace v nástroji MS Expression Blend3. Nejprve spustíme MS Blend 3 a vytvoříme nový projekt typu *WPF Application*.



Obrázek 13 : Nový WPF projekt

Dále nastavíme název projektu do pole *Name*, umístění na disku a jazyk, který budeme používat pro tvorbu aplikační logiky, ponecháme C#. V tomto případě si ale stejně vystačíme pouze s XAMLem. Další typy projektu, které máme k dispozici v plné verzi, se týkají produktu Silverlight určeného pro webové aplikace, ale s velice podobným ovládáním jako WPF.

6.2.4.2 Tvorba XML souboru

Vytvoříme XML soubor, který bude představovat prezentovaná data. Ukázkový XML soubor je přiložen na CD ve složce *Data/videoteka.xml*.

Ukázka XML dokumentu, použitého k tomuto příkladu:

```
<?xml version="1.0" encoding="UTF-8"?>
<items>
  <item name="meet firefox" source="http://www.mozilla.com ">
    <format>video</format>
    <language>english</language>
      ...
  </item>
</items>
```

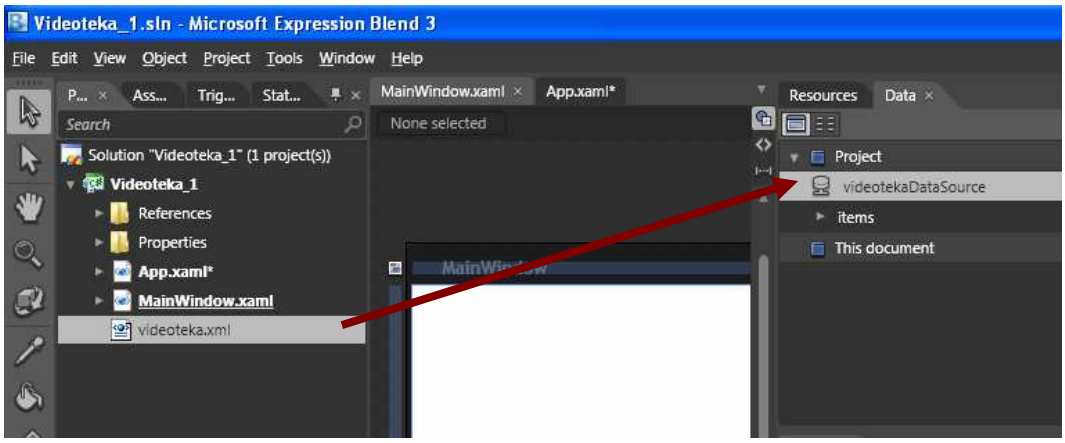
6.2.4.3 Připojení XML souboru a nastavení datového zdroje

Datový zdroj (*DataSource*) představuje kolekci všech dat, která budeme v aplikaci používat. V našem případě to bude obraz XML souboru.

Abychom mohli datový zdroj s externího souboru vytvořit, je nejprve vhodné tento soubor připojit k projektu.

Připojení provedeme na kartě *Projects* tak, že klikneme pravým tlačítkem myši na kořenovou ikonu s projektem (*Viedoteka_1*), zvolíme možnost *add existing item* a vybereme náš XML soubor.

Nyní přetažením souboru *videoteka.xml*, který se nám přidal do složky s projektem, doprava na kartu *Data*, docílíme vytvoření nového datového zdroje.



Obrázek 14 : Připojení XML souboru a tvorba datového zdroje

Nyní již máme data z XML souboru k dispozici a můžeme s nimi pracovat. Pokud na kartě *Resources* otevřeme náhled kódu XAML souboru *App.xaml*, uvidíme, že zde došlo k definici našeho datového zdroje s názvem *videotekaDataSource*.

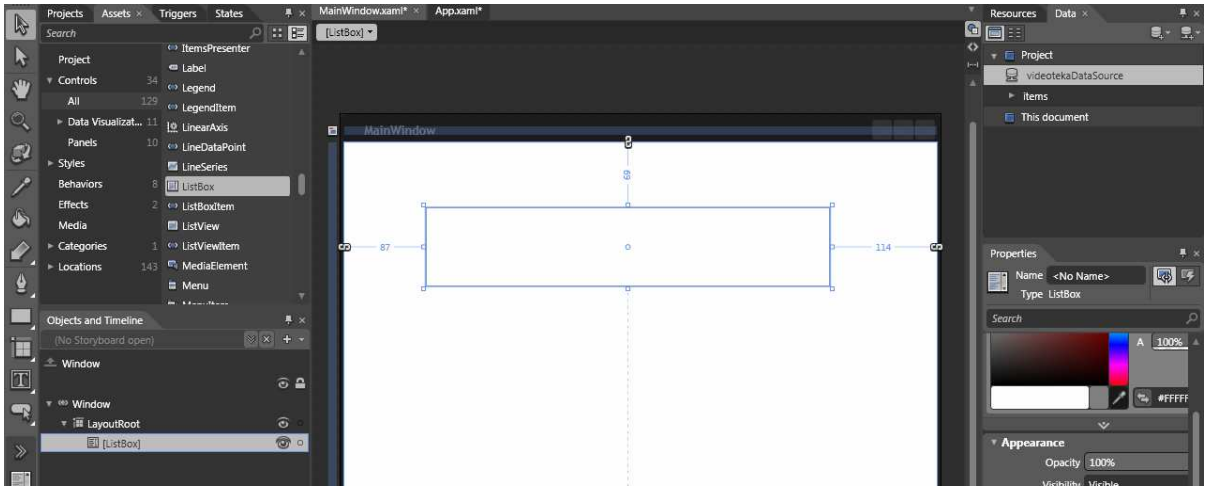
6.2.4.4 Vložení grafických prvků

Nyní již můžeme vkládat grafické prvky z karty *Assets*, položky *Controls*. Jedná se o klasické předdefinované „kontrolky“, které můžeme využívat.

Pro náš účel použijeme prvek typu *ListBox*.

Přetáhneme tedy prvek do hlavního okna a na kartě *properties* můžeme nastavit jeho vlastnosti.

Pokud použijeme nástroj *Sellection*, je jím černě vyplněná šipka, můžeme tahem měnit velikost a pozici *ListBoxu*. Dále můžeme nastavovat příchytné body, které určují, jak se bude prvek chovat při změně velikosti okna.



Obrázek 15 : Nastavení ListBoxu

Další věcí, kterou vložíme, je několik dvojic prvků typu *Label* a *TextBox*. Tyto dvojice prvků nám budou umožňovat zobrazovat další informace o videu, například autora, popis, atd.

Opět nastavíme příslušné pozice, rozměry a uchycení prvků.

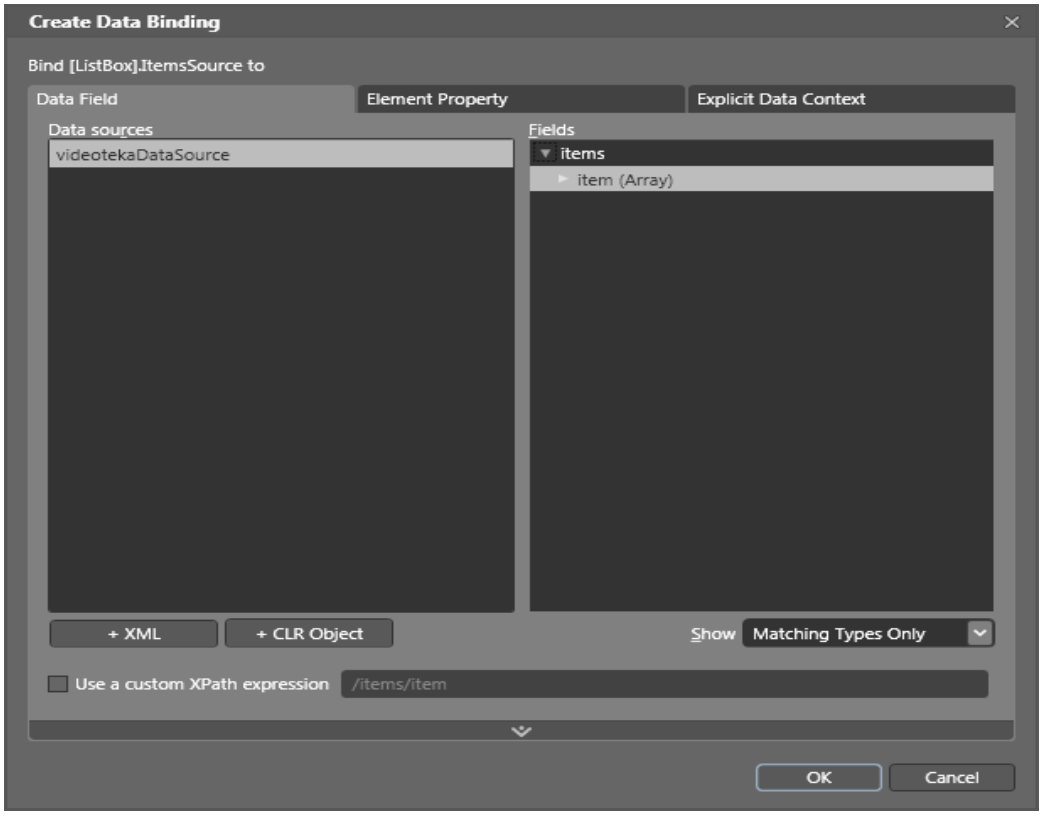
Další možností, kterou můžeme nastavit v panelu *Properties* je pozadí dokumentu, nebo jednotlivých prvků. Vybráním okna ve stromu prvků a kliknutím na *Gradient Brush* v panelu *Properties* nastavíme tedy barevný přechod pozadí okna, můžeme kliknutím na přechod barev přidat více než dvě základní barvy pro přechod. Dále nastavením položky *Title* titulek okna.

6.2.4.5 Navázání dat na ListBox

K vlastnímu navázání dat na naše komponenty použijeme Data Bindingu. V prostředí MS Blend lze mnoho věcí řešit graficky a není třeba psát vlastní kód. Klikneme tedy na *ListBox* a na panelu *Properties* v oddělení *Common Properties* najdeme volby *DataContext* a *ItemSource*. Protože máme nastavený platný datový zdroj, datový kontext se nám v tomto případě nastaví automaticky.

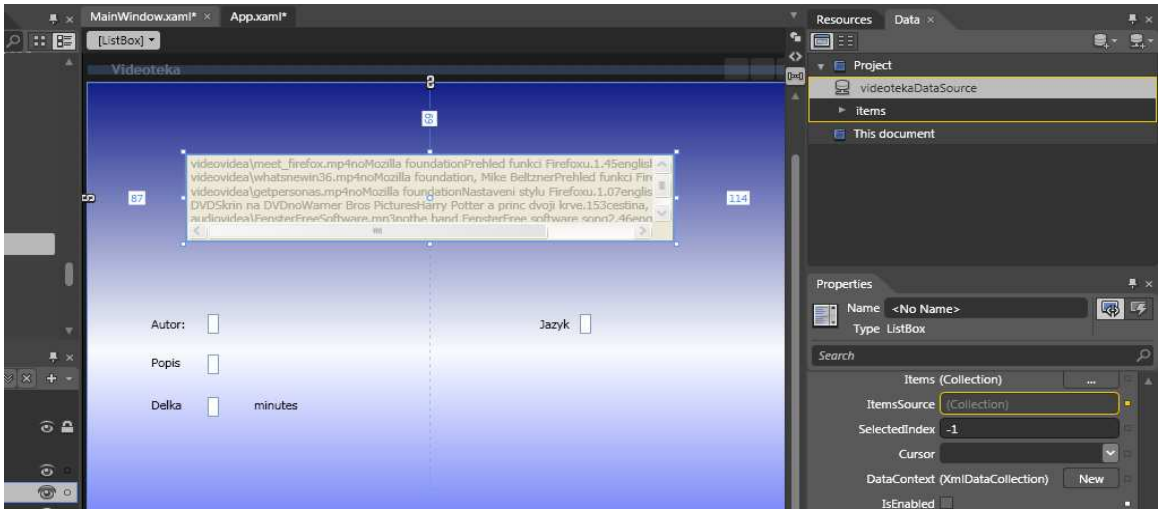
Klikneme tedy na čtvereček *Advanced property options* vedle položky *ItemSource*, která představuje zdroj dat pro naplnění *ListBoxu*, a dále klikneme na možnost *Data Binding*. Objeví se nám okno pro tvorbu datové vazby.

Zvolíme tedy na kartě *Data Field* v levé části náš datový zdroj a v pravé části kolekci položek, které jsou v něm obsažené. Jedná se o pole, které bylo vygenerováno z našeho XML souboru.



Obrázek 16 : Vytvoření datové vazby

ListBox se nám naplní požadovanými hodnotami. Protože nemáme nastavenou datovou šablonu, všechny hodnoty se spojí do dlouhých řádků.

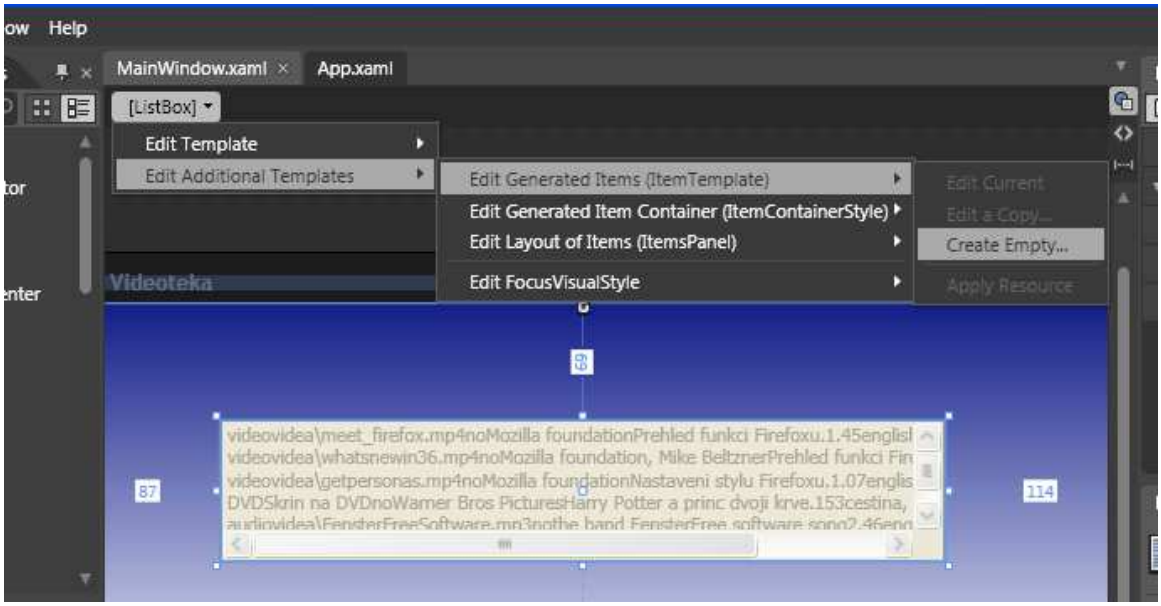


Obrázek 17 : Data Binding ListBox

6.2.4.6 Tvorba datové šablony

Kliknutím na *ListBox* provedeme jeho výběr. Dále v hlavní části okna vybereme *ListBox, Edit Additional Templates, Edit generated items.* A zvolíme možnost pro vytvoření nové prázdné šablony.

V této položce máme možnost tvořit a editovat všechny styly a šablony týkající se vybrané položky.



Obrázek 18 : Tvorba DataTemplate

Objeví se nám nabídka, ve které zadáváme název šablony, můžeme ponechat na přednastavené hodnotě. Dále zadáváme, ve kterém místě kódu chceme danou šablonu definovat. Máme na výběr v aplikaci, neboli v souboru *App.xaml*, dále pak v tomto dokumentu, dokonce můžeme zvolit i místo v dokumentu ve kterém dojde definici. A poslední možností je vytvořit slovník zdrojů, kde máme možnost šablonu uložit do specifického nového souboru a mít tak definice šablon mimo vlastní kód XAML. Pro funkci nezáleží na tom kterou možnost vybereme, jde zde spíše o přehlednost kódu.

Nyní jsme vytvořili novou prázdnou datovou šablonu, která nám bude určovat, v jaké podobě se data zobrazí. Neboli jak budou vypadat jednotlivé položky *ListBoxu*. Prozatím postačí, když vložíme *Label*. Nesmíme zapomenout nastavit *Content*, textový obsah *Labelu* na vázanou hodnotu pomocí Data Bindingu.

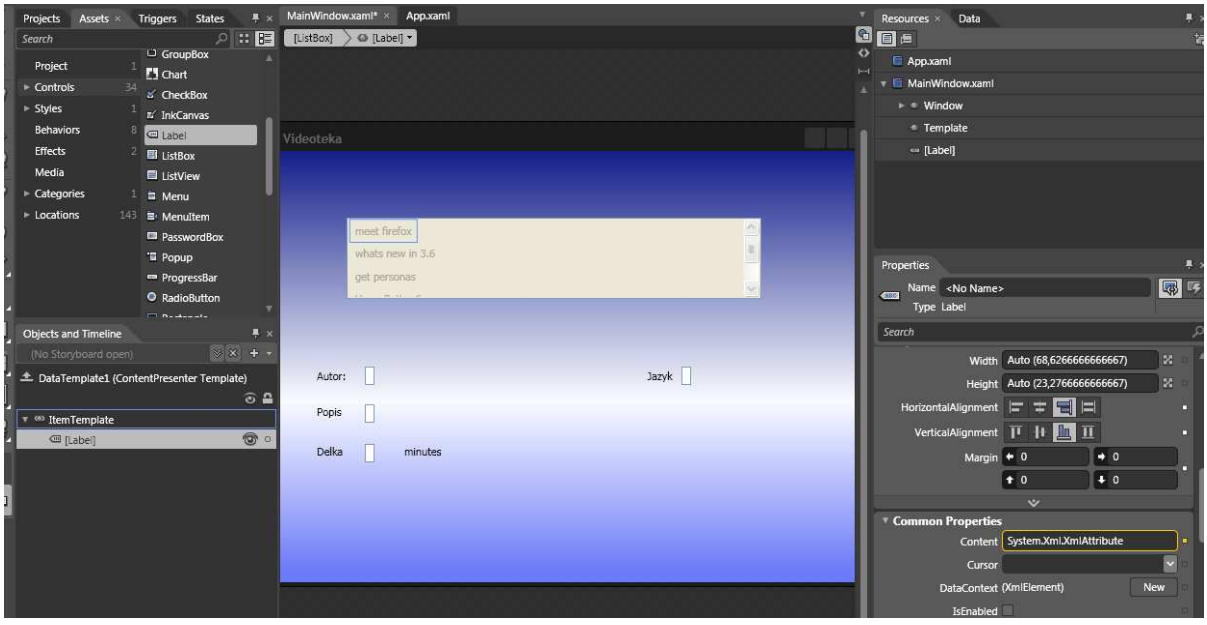
Obdobným způsobem, tedy jako jsme nastavovali *ItemSource*, nastavíme položku *Content* tentokrát, ale zvolíme *kartu Explicit Data Context* a vybereme atribut *@Name*.

Obrázek 19 : Vytvoření datové vazby 2

Pokud jsme vazbu vytvořili správně, objeví se nám v panelu nastavení u položky *Content* text *System.Xml.XmlAttribute* a místo textu *Labelu* se již objeví názvy videí definované v XML souboru.

Dále ještě můžeme buď změnit vlastnosti prvku *Grid*, který se nám vložil jako rodič před *Label*, nebo tento prvek zcela odstranit. Vlastní vzhled definujeme ve stylu.

Výsledek by mohl vypadat například takto.



Obrázek 20 : Fungující DataTemplate

Máme fungující datovou vazbu a datovou šablonu pro *ListBox*. V *ListBoxu* se nám tedy po překladu zobrazují všechny názvy videí definované v našem XML souboru. Pokud se opět podíváme do zdrojového kódu, který jsme vytvořili, uvidíme, že na námi zvolené

místo, předvolené je do souboru *MainWindow.xaml*, se vytvořil nový uzel *DataTemplate* s názvem *DataTemplate1*. A obsahuje náš *Label* s definovanou datovou vazbou.

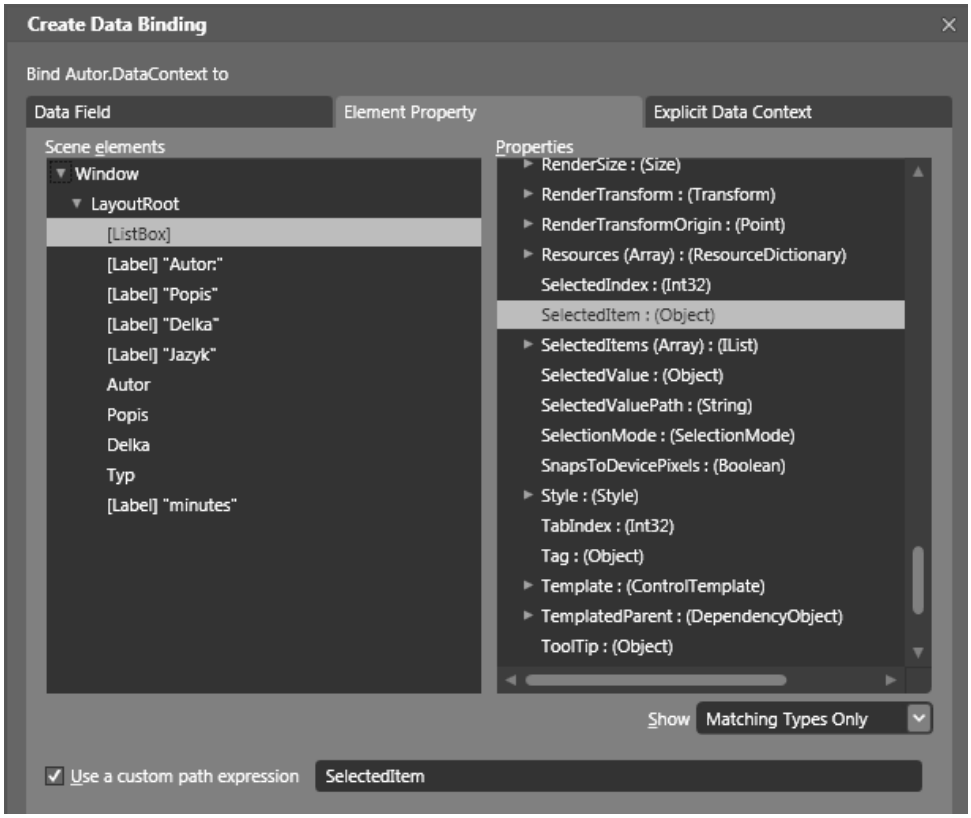
Všimneme si, že vlevo dole, ve stromu objektů jsme se vnořili do datové šablony a editujeme prvky v ní. Z editace vystoupíme kliknutím na únikovou šipku vedle názvu šablony, která nás vrátí o úroveň výše do šablony pro okno.

6.2.4.7 Tvorba vazby mezi prvky

Další důležitou možností, kterou nám WPF poskytuje je možnost vytvářet datové vazby mezi jednotlivými elementy, nebo kontrolkami. Jde o to, že nyní navážeme naše připravené *TextBoxy* přímo na aktuálně vybraný prvek *ListBoxu*.

Lze to provést kliknutím, nebo výběrem ve stromu prvků. Vybereme TextBox a položku *DataContext* navážeme na vybraný prvek *ListBoxu*. Podrobně popsáno: v panelu *Properties* klikneme na čtvereček vedle položky *DataContext* a zvolíme *DataBinding*.

Tentokrát vybereme kartu *Element Property.* Zde máme v levém panelu seznam elementů a vpravo se nám zobrazují vlastnosti, které lze napojovat. Vybereme tedy vlevo prvek *ListBox* a v pravé části položku *SelectedItem*.

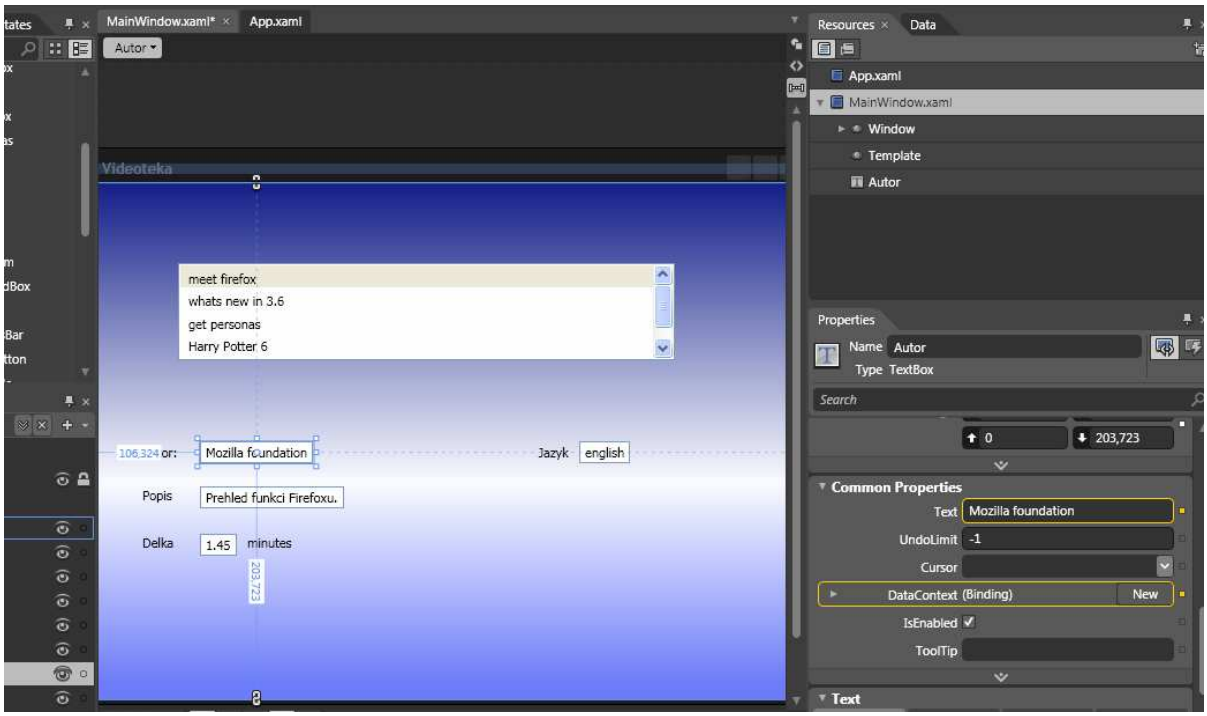


Obrázek 21 : Data Binding TextBox1

Tímto krokem jsme vytvořili kontext dat, který můžeme využít v rámci daného *TextBoxu*.Takže nyní nám jen zbývá napojit proměnnou *Text* na požadovanou hodnotu. Provedeme to stejným způsobem, jako jsme se již naučili. Nastavíme tedy novou vazbu na položku *Text*, použijeme explicitní datový kontext a vybereme uzel, který chceme zobrazovat.

Totéž zopakujeme pro všechny *TextBoxy*, které máme a všechny vlastnosti, které chceme zobrazit.

Pozor, pokud chceme vidět efekt ihned, musíme nastavit hodnotu *SelectedItem* v *ListBoxu* na platnou pozici, například na nulu. Provedeme v panelu *Properties* po výběru *listBoxu*. Ještě zkontrolujme, že máme položku *isEnabled* v nastavení *ListBoxu* zaškrtnutou jinak bychom nemohli provádět výběr položek po překladu aplikace.



Obrázek 22 : Data Binding mezi elementy

Nyní se nám již po překladu budou zobrazovat platná data načtená z XML souboru.

Získali jsme tedy základní funkci programu, data se zobrazují na grafické prvky, propojili jsme *View* a *Model*. Na první pohled se zdá, že vzhled se zatím nijak neliší od starších přístupů a podobnou funkci bychom byli schopni realizovat i za užití jiných starších nástrojů, například formulářové aplikace v C#. Jeden z hlavních rozdílů je ve způsobu napojení dat pomocí Data Bindingu, kdy jsme dosáhli volné vazby mezi daty a vzhledem aplikace a můžeme nyní vzhled téměř libovolně editovat bez nebezpečí ztráty funkčnosti aplikace.

Proto si nyní nastíníme největší sílu nástroje MS Blend a celého WPF a tou je tvorba vzhledu prvků. V tomto příkladě ještě nebudeme využívat animací, k nim se dostaneme v dalším příkladu, který bude na tento navazovat.

6.2.4.8 Nastavení stylu

Nyní přetvoříme vzhled prvku *ListBox*. Prozatím pouze bez použití animací. Postupujeme podobně, jako když jsme vytvářeli datovou šablonu.

Tedy obdobně jako jsme vytvářeli datovou šablonu, vytvoříme styl jednotlivých prvků *ListBoxu*. Zvolíme *ListBox, Edit additional Templates, Edit generated item container*. Tentokrát můžeme využít volbu *Edit a Copy*, která nám zkopíruje již existující a právě používanou šablonu a umožní nám ji editovat.

Šablona obsahuje dva prvky. Prvním je prvek *border* s názvem *Bd,* který představuje možnost ohraničení jednotlivých prvků, v podstatě jde o jakýsi rámeček. Druhý je prvek *Content Presenter,* který je velmi důležitý, protože do něj se promítají data, která jsme definovali v datové šabloně.

Všimneme si, že na kartě *Assets*, nejsou zdaleka jen kontrolky, které můžeme používat jako grafické elementy. Je zde mnohem více například položka *Styles* a v ní různé základní styly, které jsou implicitně použity. Zkopírovali jsme tedy a nyní upravujeme jeden z těchto implicitních stylů.

Takže zpět k vlastní úpravě vzhledu; důležité je pochopit, že nyní neupravujeme *Label*, který jsme si definovali v datové šabloně a který nám zobrazuje data. Nyní upravujeme jeho okolí a samotný *Label* nebo cokoliv co je definováno v datové šabloně se zobrazí pomocí prvku *ContentPresenter*. Nejjednodušší co můžeme nyní udělat je použít prvek, který má podobné chování jako prvek *Border*. Je jím například *GroupBox*. Přetáhneme ho tedy z karty *Assets* položky *Controls*, nad prvek s názvem *Bd*. Tím dojde k jeho vnoření. Struktura ve stromu elementů je tedy taková, že *Content Presenter* je na stejné úrovni s prvkem *Header* – hlavičkou *GroupBoxu*, jejich rodičem je prvek *GroupBox*, jeho rodičem je *Bd* a jeho rodičem vlastní šablona – *Template*.

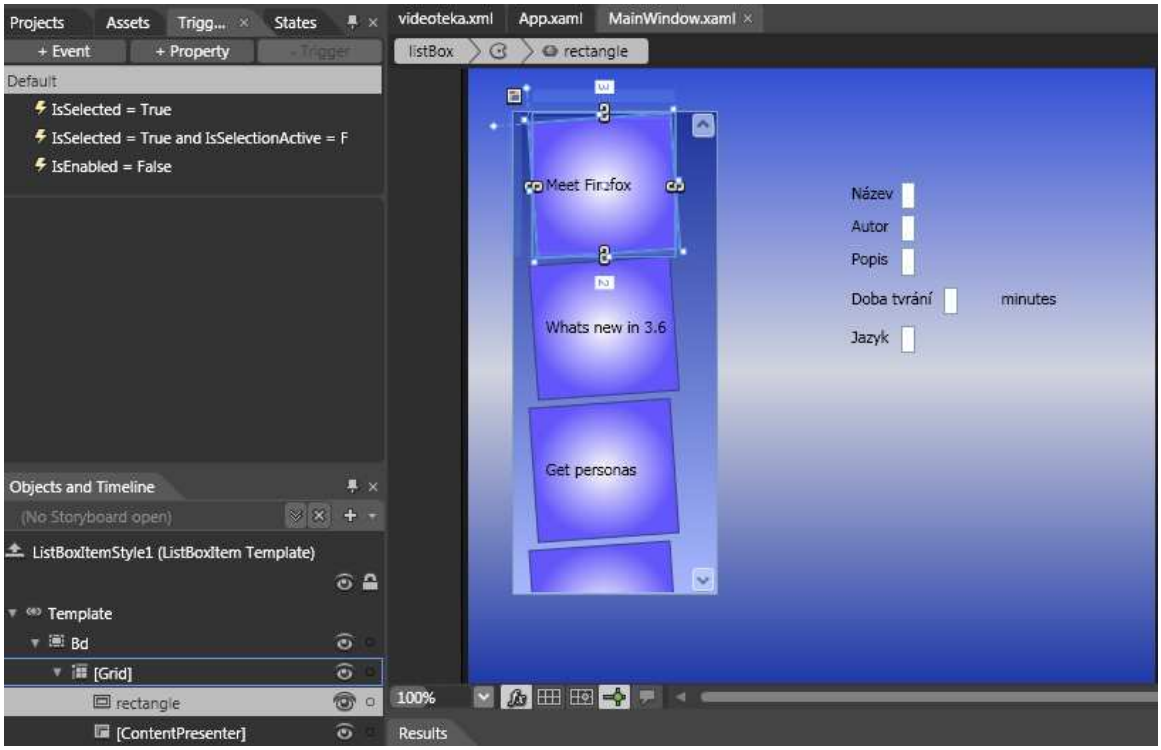
Nejsme, ale omezeni prvkem s vlastnostmi *GropuBoxu*. Lze využít jakýkoliv prvek, například obdelník – *Rectangle*. Vytvoříme tedy jinou strukturu stromu elementů. *Content Presenter* je na stejné úrovni s prvkem *Rectangle*, jejich rodičem je prvek *Grid*, který zajišťuje rozmístění více prvků na stejné úrovni, jeho rodičem je *Bd* a jeho rodičem vlastní šablona – *Template*.

Nyní když se ještě nacházíme v editaci stylu jednotlivých prvků *ListBoxu* se podíváme ještě na jednu věc, která byla v teorii zmíněna a to jsou *triggery* neboli spouště. Tyto nám umožňují vyvolat podmíněné chování elementů.

Zvolíme-li kartu *Triggers*, která se nachází hned vedle karty *Assets* objeví se několik nových možností a již předpřipravených *triggrů*. Nejprve možnosti: máme možnost přidat reakci na událost *+Event*, nebo na hodnotu v proměnné *+Property*.

Další položky v panelu obsahují nastavení jednotlivých spouští. Položka *default* určuje výchozí nastavení prvku, položka *isSelected = true* je *property trigger*, v tomto konkrétním případě se spustí v okamžiku kdy je položka *ListBoxu* vybrána. Další *Trigger*, který se nám vygeneroval, odpovídá dvěma podmínkám, které musí platit současně, aby došlo k vyvolání akce.

Při kliknutí na jednotlivé již připravené *Triggery*, máme možnost je editovat, nastavit akce, které mají vyvolat, nebo je zcela smazat.



Obrázek 23 : Triggers

Dále se nám nabízí další funkce, kterou je toto prostředí poměrně ojedinělé. Pokud na některý *Trigger* klikneme, objeví se vedle něj červené kolečko pro „nahrávání“. V tomto módu se nahrávají a ukládají všechna nastavení, která provedeme, a právě tyto změny se provedou při spuštění *Triggeru*.

Vybereme první *trigger*, který se nám nabízí, tento je aktivní, pokud je vybrán prvek. A změníme pozadí *GroupBoxu*, které přísluší tomuto stavu. Můžeme opět zvolit možnost *Gradient Brush* a tentokrát pro změnu *Radial Gradient*, které vytváří barevný přechod směrem ke středu.

Podobným způsobem nastavíme chování nevybraného prvku, v našem případě to bude odpovídat stavu *default*, dále pak prvku, který je sice vybrán, ale výběr není již aktivní, například, jsme mezitím kliknuli na prvek zcela mimo *ListBox* a nakonec chování neaktivního prvku.

Můžeme stiskem klávesy F5 aplikaci přeložit, spustit a ověřit si chování. Nyní již máme vzhled prvků vyladěn.

Obrázek 24 : Finální program 1

6.3 Příklad 2, XML, Convertor, Animace

6.3.1 Popis

V tomto příkladě si zopakujeme napojení na XML soubor a napojení mezi *Controls* a přidáme další funkci, kterou v sobě WPF obsahuje a tou je *Convertor*. Tato funkce nám umožňuje konvertovat (převádět) zobrazovaná data do jiné podoby a formátu. Dále využijeme další formy animací a grafických prvků, které se tu nabízí.

6.3.2 Zadání

- Vytvořte aplikaci „Videoteka_2“ typu WPF aplikace v nástroji MS Expression Blend3. Vytvořte XML soubor, který bude obsahovat konkrétní data zmíněná výše (*Name*, *Format*, *Destination*, *Author*, *Description*, *Duration*, *Language*). (Kapitola: 6.3.4.1)
- Pomocí nástroje Microsoft Expression Blend3 vytvořte vlastní vzhled aplikace, který se bude skládat z nového okna, *ListBoxu* a několika prvků typu *Label* a *TextBox*, reprezentujících vlastnosti konkrétního média. (Kapitola: 6.3.4.2)
- Vzhled aplikace vytvořte dle svého uvážení, odlišně od prvního příkladu. Můžete vytvořit jiný vzhled prvků *ListBoxu* a jinak tyto prvku rozmístit. (Kapitola: 6.3.4.4)
- Na element *ListBox* za užití Data Bindingu napojte data z XML souboru, tak aby se prvky v *ListBoxu* reprezentovaly pouze názvem (*Name*). (Kapitola: 6.3.4.3)
- Po výběru jednoho prvku *ListBoxu* dojde k aktualizaci prvků typu *TextBox*, neboli budou navázány na aktuálně zvolený prvek *ListBoxu* a budou zobrazovat jeho další vlastnosti (*Author*, *Description*…).(Kapitola: 6.3.4.3)
- V programu využijte animace, které budou reagovat na výběr nové položky. (Kapitola: 6.3.4.5)
- Vložte prvek typu *Image*, který bude za pomoci konvertoru zobrazovat ikonu představující formát souboru. (Kapitola: 6.3.4.6)

6.3.3 Teorie

6.3.3.1 Convertor

Jedná se o funkci, která nám umožňuje změnit formát dat. Nejedná se pouze o klasický převod dat mezi primitivními formáty, lze podobu dat zcela změnit námi definovaným způsobem. Lze také využít několik předdefinovaných konvertorů.

6.3.3.2 Animace

Ve WPF je tvorba animací překvapivě jednoduchá. Vystačíme si pouze s XAMLem a prostředí MS Blend nám poskytuje efektivní nástroje jak animace tvořit i bez explicitní znalosti XAMLu.

Princip animací ve WPF je odlišný od většiny ostatních přístupů. Nevytváříme časové osy, neanimujeme jednotlivé kontrolky, ale animujeme vlastnosti. Například animace zvětšujícího se tlačítka, znamená, aby se postupně měnily vlastnosti *width*

a *height* a přesně to dělají animace ve WPF. Animace je generátor hodnot, kterými se mění požadované vlastnosti, ale animovat můžeme pouze vlastnosti typu *DependencyProperty*.

Lze nastavit způsob generování hodnot (lineární, křivka…) a také dobu trvání a rychlost animace.

6.3.4 Řešení

6.3.4.1 Založení projektu

Založíme nový projekt typu *WPF application*, obdobně jako v předchozím příkladě. Nazveme ho „Videoteka_2“.

6.3.4.2 Tvorba vzhledu aplikace

Vložíme všechny prvky, které budou představovat vzhled aplikace. Postupujeme obdobně jako v předchozím příkladu, vytvoříme si nový prvek typu *ListBox*, nastavíme jeho vzhled. Vytvoříme potřebnou datovou šablonu pro prvky *ListBoxu*. Můžeme upravit také styl prvků *ListBoxu* (*ItemContainerStyle*), lze například v panelu *properties* přidat ke *Content Presenteru* efekt *Drop and Shadow*.

Lze využít postupu v předchozím příkladě, ale můžeme vzhled zcela předělat.

Vytvoříme si také další elementy, ve kterých budeme zobrazovat další informace. Tedy *Labely* a *TextBoxu*, pro výpis informací a také jeden prvek typu *Image*, který nám bude, po vytvoření konvertoru, zobrazovat typ souboru v podobě ikonky.

6.3.4.3 Napojení grafické části programu na XML soubor

Provedeme vytvoření datového zdroje z XML souboru, můžeme využít XML soubor z předchozího příkladu.

Na již vytvořený *ListBox* navážeme data z datového zdroje. Vytvoříme vazbu mezi právě vybraným prvkem *ListBoxu* a proměnnou *Text* v *TextBoxu*.

Dotvoříme vzhled jednotlivých prvků a odladíme pomocí již připojených dat.

6.3.4.4 Nastavení Layoutu

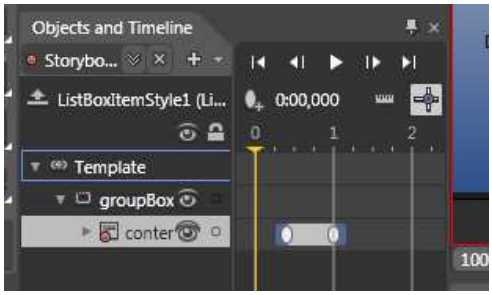
Nyní je ještě vhodné nastavit, způsob výpisu jednotlivých prvků *ListBoxu* neboli *layout*. V tomto příkladě bylo by vhodnější, kdyby data byla zobrazena vedle sebe namísto pod sebou.

Nastavení provedeme obdobným způsobem jako nastavení stylu nebo datové šablony. Budeme tedy editovat *Layout of items*. Použijeme opět možnost *Copy and Edit*. Nyní již se nám zobrazí ve stromu prvků *VirtualizingStackPanel*, u něj pouze změníme proměnnou *Orientation* na hodnotu *horizontal*.

Nyní se může stát, že se nám provedené změny nezobrazí, nebo budou nedostatečné mezery mezi jednotlivými prvky. Vhodné je tedy program přeložit a nedostatky doladit.

6.3.4.5 Přidání animací

Nyní přidáme animace do našeho projektu. Máme tedy program přibližně podobné funkčnosti jako minulý příklad. V minulém případě jsme nastavovali *Triggery* pro jednotlivé prvky *ListBoxu* v *ItemContainerStyle*. Nastavíme je tentokrát také, ale pokud vybereme některý *Trigger*, například *IsSelected = true* všimneme si možnosti *Actions when activating*. Tato volba slouží k přidávání animací.

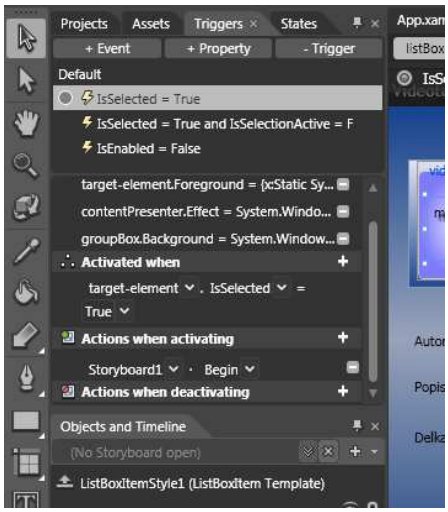


Obrázek 25 : Tvorba animace

Klikneme tedy na malé plus vedle této volby a vytvoříme novou animaci, neboli *StoryBoard*. Zadáme její název a všimneme si, že se nám objevila časová osa vedle stromu prvků.

Nyní vlastní animaci vytvoříme tak, že klikneme do požadovaného místa časové osy, například jedna vteřina, vybereme prvek ze stromu prvků, který má měnit své vlastnosti. A změníme požadovanou vlastnost v panelu *Properties*, například můžeme prvek pootočit.

Pak klikneme na nulu v časové ose a vrátíme prvek do původní pozice. A již máme animaci vytvořenou. Můžeme ji přehrát, nebo dále upravovat.



Obrázek 26 : Nastavení animace

Nyní jen zkontrolujeme, zda se nám animace přiřadila ke správnému *Triggeru*. Mělo by se po jeho vybrání v oddílu *Actions when activating*, zobrazit možnost *Storyboard1.begin*. Což nám animaci spustí. Můžeme přeložit a odzkoušet zda animace funguje.

6.3.4.6 Vytvoření Convertoru

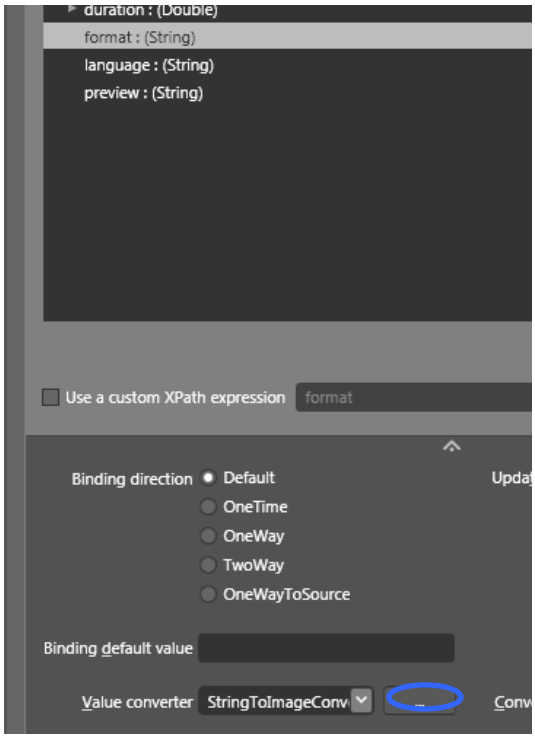
Vložíme novou položku typu *Class* s názvem *Convertor*, která bude součástí projektu. Nebo ji přidáme pomocí volby *Add existing Item*. Jde o obdobný postup jako je připojení XML souboru. Již vytvořenou třídu *Convertor* najdeme ve složce *Data*.

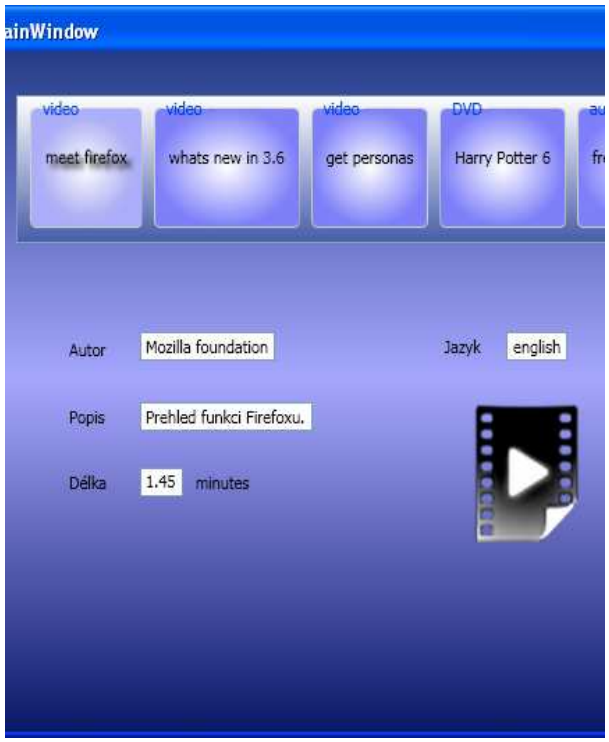
Tato třída představuje *Convertor* v jazyce C#. Pokud ji dvoj-klikem otevřeme, uvidíme třídu s názvem *StringToImageConvertor*, která dědí rozhraní *IValueConvertor* a obsahuje metody *Convert* a *ConvertBack*, které představují vlastní konverze. Stejným způsobem, přidáme také soubory *DVD.png* a *video.png*.

Nyní pokud již máme třídu připojenou, zkontrolujeme zda u položky *NameSpace* ve zdrojovém C# kódu je název projektu, poté tečka a název *Convertor*.

Nyní se vrátíme do editace souboru *MainWindow.xaml* a nastavíme vazbu na již vložený prvek *Image*. V položce *DataContext* mu nastavíme jako u *TextBoxů* hodnotu *ListBox, SelectedItem*. A nyní navážeme položku *Source* na explicitní datový kontext a prvek *format*.

V nabídce pro tvorbu datové vazby rozbalíme dole šipku s dalším nastavením a zvolíme možnost *Add Value Convertor*. Vybereme náš *StringToImageConvertor* a vazbu realizujeme. Pokud se nám volba našeho konvertoru neobjevuje, pomůže projekt přeložit.





Obrázek 27 : Convertor

Obrázek 28 : Finální program 2

6.4 Příklad 3, databáze

6.4.1 Popis

Příklad využívá, vytvořenou databázi pro MS SQL server. Tuto databázi napojuje na obdobné grafické rozhraní jako u předchozích příkladů. Struktura databáze také odpovídá společnému zadání. Přidává možnost aktualizace zdroje dat.

6.4.2 Zadání

- Vytvořte aplikaci „Videoteka_3“ typu WPF aplikace v nástroji MS Expression Blend3. (Kapitola: 6.4.4.1)
- Vytvořte databázový soubor, který bude obsahovat konkrétní data zmíněná výše (*Name*, *Format*, *Destination*, *Author*, *Description*, *Duration*, *Language*). (Kapitola: 6.4.4.26.4.4.1)
- Z databáze za užití objektového modelu databáze vytvořte datový zdroj obousměrně přístupný grafickým prvkům aplikace. (Kapitola: 6.4.4.3)
- Pomocí nástroje Microsoft Expression Blend3 vytvořte vzhled aplikace, obdobně jako v předchozích příkladech. (Kapitola: 6.4.4.4)

- Po výběru jednoho prvku *ListBoxu* dojde k aktualizaci prvků typu *TextBox* a naopak při změně prvku *TextBox* se změna projeví i v *ListBoxu*, neboli budou na sebe obousměrně navázány. Při tvorbě vzhledu využijte animace. (Kapitola: 6.4.4.4)
- K dispozici bude tlačítko, které provedené změny uloží do databáze. (Kapitola: 6.4.4.5)

6.4.3 Teorie

6.4.3.1 MS SQL server

SQL server je databázový server, instaluje se jako součást Visual Studia. Slouží ke zpřístupnění dat z databázových úložišť.

Data jsou uložena v tabulkách relační databáze. Pro manipulaci s daty lze využít příkazy jazyka SQL, nebo předdefinované třídy, které umožňují provést například relačně objektové mapování a s daty dále pracovat jako se strukturou objektů.

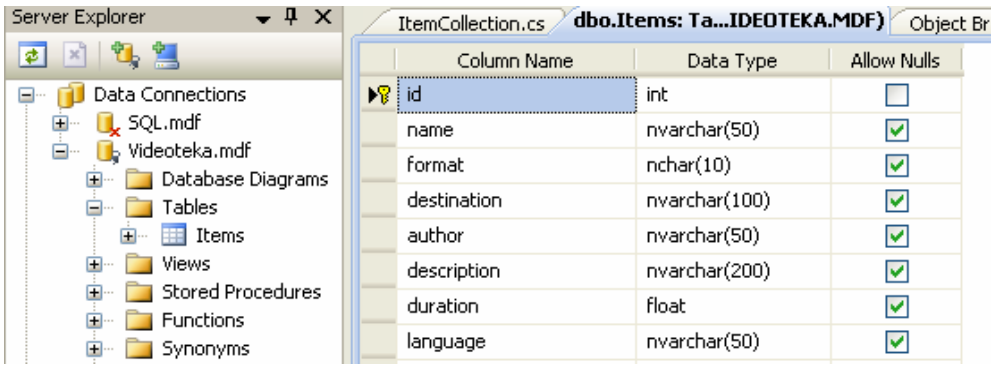
6.4.4 Řešení

6.4.4.1 Založení projektu

V prostředí MS Blend založíme novou aplikaci, stejně jako v předchozích příkladech, nazveme ji „videoteka_3“. Klikneme pravým tlačítkem na název projektu a zvolíme možnost *Edit in visual studio.* Projekt se otevře v prostředí Visual Studia.

6.4.4.2 Připojení databáze

V prostředí Visual studia přidáme do projektu, pomocí volby *Add new Item*, novou položku *Service-based database.*Po dvoj kliku ji lze editovat. Do složky *tables* tedy přidáme novou tabulku. Nastavíme položky obdobné jako u XML souboru, vypsané jsou ve společném zadání příkladu. Přidáme pole *id* typu *int*. Uložíme jako *Items*.

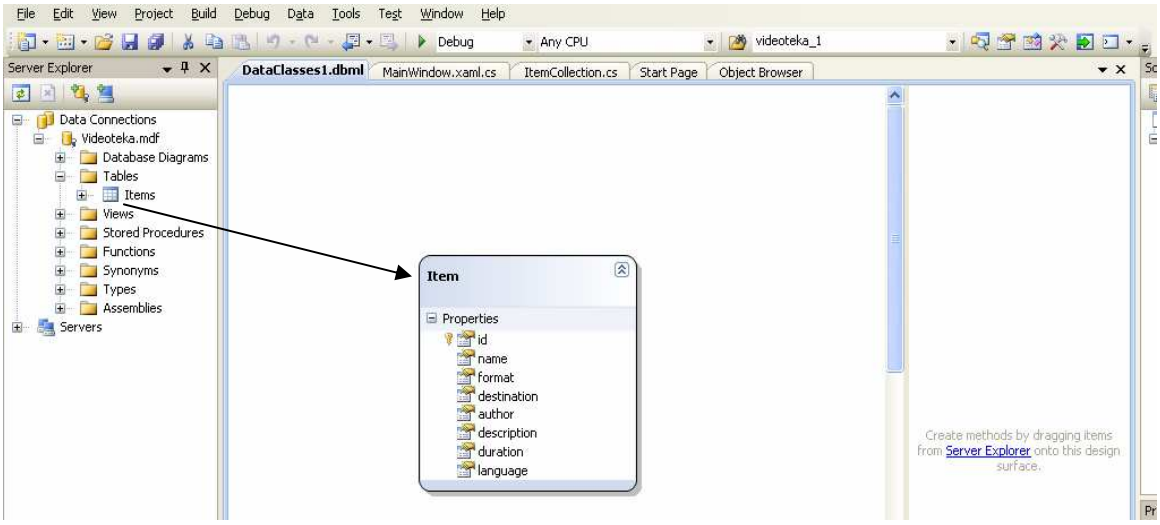


Obrázek 29 : Struktura databáze

Klikneme na tabulku pravým tlačítkem myši, zvolíme možnost *Show table data* a zadáme data, která budeme zobrazovat.

Vytvořený databázový soubor, který lze pouze připojit k projektu, je k dispozici ve složce *Data\Videoteka.mdf*.

Pokud máme k dispozici databázový soubor, přidáme k projektu další novou položku typu *LINQ to SQL Classes*. Otevře se okno pro editaci tohoto datového zdroje. Do tohoto okna přetáhneme ze *Server Exploreru* tabulku *Item* z naší databáze. V položce *DataClasses1.dbml* se nám vytvoří struktura odpovídající naší databázi.



Obrázek 30 : Vytvoření obsahu DataClasses1

Nyní vytvoříme novou třídu *ItemCollection*, nebo přidáme již vytvořenou ze složky *Data\ItemCollection.cs*. V této třídě vytvoříme instanční proměnnou typu *ObservableCollection<Item>* názvu *items*. Tuto proměnnou zpřístupníme za užití *Property*, neboli vlastnosti, s názvem Data.

```
public ObservableCollection<Item> Data
{
    get { return items; }
    set { items = value; }

}
```

Obrázek 31 : Property

Vytvoříme si nový objekt typu *DataClasses1DataContext*, uchováme si jej jako statickou instanční proměnnou.

V konstruktoru do proměnné *Data* předáme postupně databázová data z objektu typu *DataClasses1DataContext*, proměnné *Items*, neboli názvu tabulky v množném čísle.

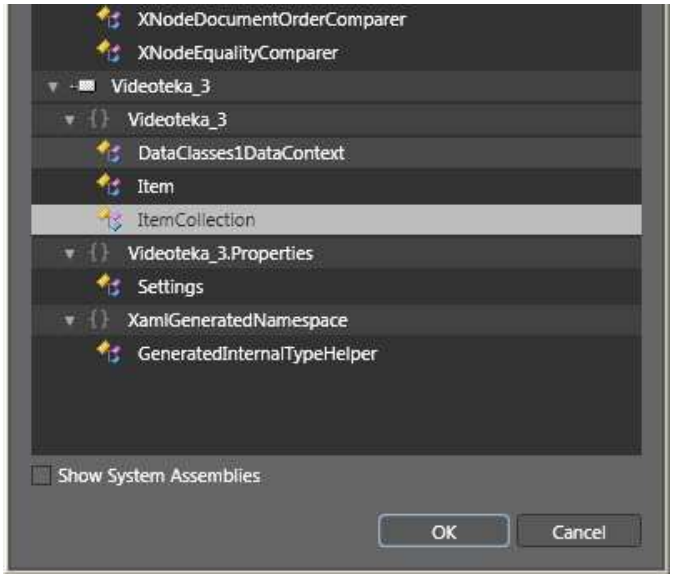
Nyní jsme převedli naše data do zobrazitelné podoby v objektu typu *ItemCollection*.

6.4.4.3 Navázání databáze na ListBox

Nyní se přepneme zpět do prostředí MS Blend. Nástroj Visual Studio je vhodné zavřít, nebo otevřený projekt ukončit a v prostředí Blendu projekt přeložit.

Nyní můžeme objekt, který reprezentuje data z databáze, připojit na položku *DataContext ListBoxu*. Pro připojení použijeme opět čtvereček vedle položky *DataContext*, *DataBinding* a kartu *DataField*, zde tlačítko *+CLR Object.* Vybereme naši třídu *ItemCollections*.

Další varianta připojení na náš objekt s databázovými daty je přidání nového datového zdroje objektového typu, který v sobě obsahuje data za databáze, jde o volbu *Define new object data source.* Další postup zůstává stejný.



Obrázek 32 : Data Binding na objekt

Pokud máme objekt napojen na datový kontext *ListBoxu*, můžeme připojit *ItemSource ListBoxu* na přístupový prvek k proměnné *items*, námi nazvaný *Data*.

Tentokrát nastavíme obousměrnou vazbu, rozbalením dalšího nastavení datové vazby a zvolením možnosti *two way*.

Ještě je nezbytné vytvořit datovou šablonu a opět použít například prvek *Label*, který navážeme na proměnnou *name*.

Data se nám nebudou zobrazovat ihned v náhledu, proto pro účely vývoje a pokusy je vhodné si přidat již zmíněný *Sample Data Source* a propojením s *ListBoxem,* místo databáze, testovat s jeho pomocí.

6.4.4.4 Nastavení grafických komponent

Opět vytvoříme design aplikace, obdobně jako v předchozích příkladech můžeme užívat stylů, *triggerů* i animací. Při použití konvertoru nesmíme zapomenout připojit potřebné soubory (Convertor.cs a obrázky).

6.4.4.5 Uložení změn

Nyní již máme design aplikace a napojení dat směrem z databáze plně funkční. Přidáme tedy možnost poslat data i opačným směrem, tedy uložit data do databáze. Ve všech datových vazbách, které se vztahují na *Text Boxy*, nebo jiné editovatelné položky zkontrolujeme, zda máme nastavenou obousměrnou vazbu, *Two way*.Obousměrná vazba se projeví tak, že změny se ukládají do datového kontextu *ListBoxu*, neboli objektu *ItemCollection*.

Přidáme tlačítko, které provede vlastní uložení dat do databáze. Nyní se můžeme přepnout do Visual Studia a do XAML kódu tlačítka přidat *handler*, odpovídající kód je `Click="metoda handleru"`. Přidání lze také provést klasicky dvoj-klikem na tlačítko.

Do automaticky vytvořené metody, pro obsluhu tlačítka přidáme kód, který zavolá metodu *CommitChanges* v naší statické proměnné typu *DataClass1DataContext*. Tím se změny, které se obousměrně ukládali do zdrojového objektu zapíšou do databáze.

Obrázek 33 : Finální program 3

Pozor, při překladu může dojít ke zkopírování původní databáze a přepsání nových dat! Databáze se zkopíruje k přeloženému souboru a do ní se uloží nová data, tedy při překladu aplikace dojde k resetování databáze původními daty. Při běžném spuštění *assembly* programu bude ukládání a načítání fungovat.

6.5 Příklad 4, Objekty

6.5.1 Popis

Napojuje vzhled aplikace na strukturu objektů. Data načítá z textového souboru a umožňuje dynamicky prohledat složku na disku pro doplnění dalších dat.

6.5.2 Zadání

- Vytvořte aplikaci „Videoteka_4“ typu WPF aplikace v nástroji MS Expression Blend3. (Kapitola 6.5.4.1)
- Vytvořte třídu *Item*, která bude obsahovat zveřejněné instanční proměnné odpovídající struktuře dat zmíněné výše (*Name*, *Format*, *Destination*, *Author*, *Description*, *Duration*, *Language*). (Kapitola 6.5.4.2)
- Dále vytvořte třídu *Items*, obsahující v sobě kolekci objektů typu *Item*. (Kapitola 6.5.4.2)
- Vlastní data načtěte do kolekce z textového souboru a do něj je i uložte. (Kapitola 6.5.4.3)
- Umožněte dynamicky prohledat složku na disku pro nalezení dalších audiovizuálních souborů. (Kapitola 6.5.4.3)
- Pomocí nástroje Microsoft Expression Blend3 vytvořte vzhled aplikace. Vzhled obousměrně napojte na kolekci dat tak aby byla umožněna editace a ukládání dat. (Kapitola 6.5.4.4)

6.5.3 Teorie

6.5.3.1 Objektově orientované programování

Objektovou strukturu jsme již použili v minulém příkladě, když jsme napojovali aplikace na kolekci dat typu *Item*. Objektová struktura, na kterou bylo napojení provedeno, ale byla vygenerována automaticky. Objektová struktura modeluje prvky z reálného světa.

Při objektově orientovaném programování využíváme strukturu jmenných prostorů a tříd. Každá třída reprezentuje objekt reálného světa, pomocí instančních proměnných reprezentuje jeho stav. Dále v sobě obsahuje metody představující chování daného objektu. Je možné třídy dědit, zapouzdřovat, skládat.

6.5.4 Řešení

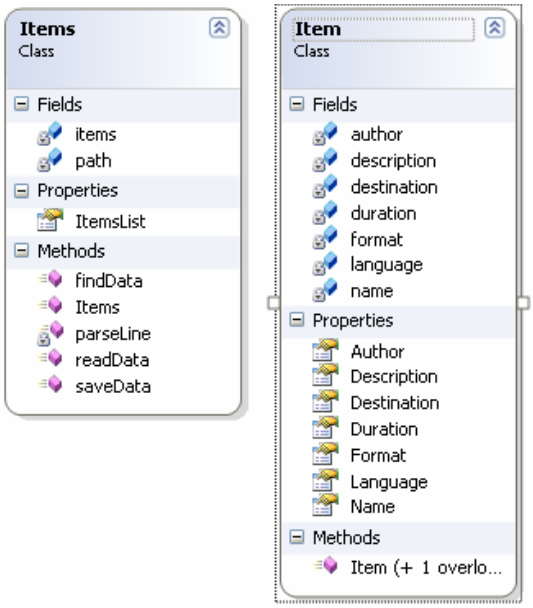
6.5.4.1 Založení projektu

V prostředí MS Blend založíme novou aplikaci, stejně jako v předchozích příkladech, s názvem „videoteka_4“. Zvolíme možnost *Edit in visual studio.* Projekt se otevře v prostředí Visual Studia.

6.5.4.2 Vytvoření struktury objektů

K projektu připojíme soubor *videoteka.txt*, který obsahuje zdrojová data. Soubor nalezneme ve složce *Data.* V panelu *Properties* nastavíme možnost *Copy to output directory* na *Copy always.*

Nyní vytvoříme strukturu objektů dle zadání. Přidáme k projektu nové třídy s názvy *Items* a *Item*. Třída *Items* bude obsahovat seznam objektů typu *Item* a metodu umožňující načítání a ukládání dat z textového souboru. Dále metodu pro dynamické načtení nových dat. Třída *Item* bude představovat jednotlivou položku videotéky. Modifikátor přístupu tříd je nutné nastavit *public*.



Obrázek 34 : Class diagram

Ve třídě Item vytvoříme instanční proměnné odpovídající datům, která budeme zobrazovat. Proměnné ponecháme jako neveřejné, ale vytvoříme k nim přístupové vlastnosti *Property*. Lze využít možnosti Visual Studia, kliknutím pravého tlačítka myší na instanční proměnnou a zvolením možnosti *Refactor\Encapsulate field*, dojde

k vytvoření přístupového bodu automaticky. Tato funkce nefunguje v express verzi Visual Studia. V této třídě dále vytvoříme parametrický konstruktor, který bude mít parametry odpovídající všem instančním proměnným.

Třída *Items* bude obsahovat kolekci prvků typu *Item*. Opět je nutné vytvořit přístupové vlastnosti k této instanční proměnné. V příkladu je vlastnost přistupující k seznamu nazvána *ItemsList*.

6.5.4.3 Tvorba metod

Vytvoříme metody *readData* a *saveData*, ve třídě *Items*, které budou načítat a ukládat data do textového souboru. Metoda pro načtení dat bude zavolána z konstruktoru této třídy.

Metody využívají objekty typu *System.IO.StreamReader* a *StreamWritter*, které umožňují načítat a ukládat data z textového souboru. Lze využít cyklus *while*, který bude opakovaně volat metodu *ReadLine*, případně *WriteLine(string text)*, dokud nebudou zpracována všechna data. Po skončení práce s textovým souborem, je nutné komunikaci ukončit zavoláním metody *Close* ve výše zmíněných objektech.

Dále vytvoříme metodu *findData*, která na zvoleném místě na disku najde existující audiovizuální soubory. Metoda využije systémovou třídu *System.Directory* a v ní obsaženou statickou metodu *GetFiles(string path)*. Pole řetězců, které nám tato metoda vrátí, poté projdeme cyklem a dle přípon určíme audiovizuální soubory.

Hotové třídy *Items* a *Item* jsou k dispozici ve složce *Data*.

6.5.4.4 Navázání dat na ListBox

Nyní vytvoříme nový objektový datový zdroj ze třídy *Items*. Pokud jsme jako cestu k textovému souboru zadali relativní adresu, data se budou zobrazovat až po překladu. Pokud byla cesta absolutní, můžeme i v náhledu vidět načtená data.

Vytvoříme také vazbu na *ListBox* a nastavíme potřebné šablony.

6.5.4.5 Nastavení grafických komponent

Vytvoříme design aplikace, obdobně jako v předchozích příkladech můžeme užívat stylů, *triggerů* i animací. Při použití konvertoru nesmíme zapomenout připojit potřebné soubory.

6.5.4.6 Nastavení aplikace

Pokud nám již funguje jednosměrná komunikace dat s aplikací, přidáme opět tlačítko umožňující ukládání dat. Toto tlačítko zavolá naší metodu *saveData*, tím dojde k uložení dat na adresu určenou instanční proměnnou *path*.

Můžeme si uvědomit, že *DataContext listBoxu* je náš zdrojový objekt. Lze ho tedy pouhým přetypováním převést na objekt typu *Items* a poté s ním jako s tímto objektem pracovat.

Vytvoříme také tlačítko, které bude načítat data z námi zvolené složky. Kód obsluhující tuto událost, bude obsahovat dialog pro nastavení složky, lze použít *Microsoft.Win32.OpenFileDialog*. A volání metody *findData*, která jako parametr dostane cestu ke zvolené složce.

Dále přidáme tlačítka plus a mínus, která budou přidávat nové prvky do kolekce záznamů, nebo naopak tyto prvky mazat. Vlastní mazání či přidání, je vhodné provést přímo na kolekci *ItemsList* představující zdroj dat *listBoxu*.

Finální program by mohl vypadat například takto.

Obrázek 35 : Finální program 4

7 Zkušební příklad

Tento příklad slouží k ověření znalostí získaných z tohoto kurzu. Studenti mají za úkol zpracovat rozsáhlejší téma. Mohou si vybrat datový zdroj, který využijí. Zadání je v tomto příkladě formulováno volně a již není doplněno podrobným postupem řešení.

7.1 Zadání

- Vytvořte a zobrazte databázi studentů ve škole.
- Vyberte si vhodný datový zdroj a naplňte jej ukázkovými daty. Datový zdroj musí obsahovat minimálně deset záznamů.
- Za užití Data Bindingu napojte tento datový zdroj na grafickou aplikaci ve WPF.
- Musí být možné vybrat jednotlivého studenta a získat o něm i další informace, jako je bydliště, datum narození atp.
- Studenty je možné dělit do větších skupin po třídách. V tomto případě se program bude chovat tak, aby po výběru jedné třídy došlo k zobrazení všech studentů do této třídy náležících. Poté bude možno o jednotlivém studentovi získat podrobnější informace.

7.2 Vstupní vzorek studentů

Zadání obdržela skupina pěti mírně pokročilých studentů.

Tato skupina respondentů jsou převážně studenti prvního ročníku středních škol, kteří prošli jeden až dvouletým zájmovým kurzem programování v jazyce C#.

7.3 Metodika příkladu

Respondenti si nejprve nastudují a zpracují řešené příklady spolu s připojenou teorií.

Poté dostanou výše uvedené zadání programu. Zadání je v některých svých částech formulováno volně, aby si studenti částečně mohli zvolit obtížnost příkladu. Pevně zadaná je funkce programu a problém, který řeší, naopak volnost je v použitém datovém zdroji a způsobu designu aplikace.

Nakonec jsou odevzdané příklady prezentovány před dalšími studenty a splnění jednotlivých bodů zadání je vyhodnoceno dle kritérií. V další části se ohodnotí použité datové vazby a schopnost designovat aplikaci.

7.4 Kritéria vyhodnocení

Vyhodnocovat se bude počet splněných bodů zadání, kde každý z pěti bodů zadání bude představovat 20 % v celkovém hodnocení.

Dále se bude hodnotit styl programování a efektivita programu.

Mimo to se ohodnotí použité datové vazby včetně výběru datového zdroje; použité grafické prvky i celkový vzhled aplikace. A také bude hodnocen vlastní přínos v těchto oblastech především ve způsobu řešení vazeb a designu aplikace.

7.5 Vyhodnocení

Testované objekty splnili zadání na 75 %, 80 %, 75 %, 80 % a 90 %.

Vyhodnocení na 80 % odpovídá splnění všech bodů zadání s výjimkou posledního. Hodnocení na 75 % představuje splnění všech bodů, kromě posledního s jednou drobnou chybou v některém z nich. A subjekt vyhodnocený na 90 % splnil všechny body a s drobnými nedostatky i bod poslední.

Tedy čtyři respondenti splnili vždy čtyři z pěti bodů zadání na srovnatelné úrovni. Jeden student se pokusil splnit i pátý bod zadání a realizovat dvě kolekce vnořené do sebe.

Studenti byli schopni realizovat zadání za užití XML souboru jako datového zdroje. S drobnými problémy byli schopni napojit data na vzhled aplikace.

Pouze jeden student realizoval zadání za užití jiného datového zdroje, konkrétně textového souboru a struktury objektů. Tohoto studenta zaujala nejvíce možnost vazby mezi grafickým rozhraním aplikace a strukturou objektů.

Hlavní přínos spatřovali v možnosti jednoduchého designování aplikace, kterému se všichni naučili ve velice krátké době.

Dva studenti projevili větší cit pro design aplikace a byli schopni vytvořit pokročilé grafické rozhraní včetně užití animací.

Největší problémy se projevily v chování aplikace MS Blend, které je někdy zcela nepředvídatelné. A studenti se s tímto obtížně vyrovnávali. Další problémy měli s připojenými datovými soubory, respektive s cestou k nim. Často používali absolutní cestu místo relativní a docházelo k problémům při přenosu na jiný počítač.

8 Závěr

8.1 Zhodnocení technologie WPF & Data Binding v MS Blend 3

Vývojový nástroj MS Blend umožňuje dosáhnout velice pokročilých výsledků. Práci vývojáře výrazně usnadňuje hlavně v oblasti designu aplikace.

Hlavní nevýhody spočívají v někdy nepředvídatelném chování tohoto nástroje, z důvodu velké složitosti a také v jeho ceně. Není k dispozici express verze jako například u Visual Studia, ale existuje pouze trial, nebo studentská verze.

Další problémy jsou při napojování designu na objekty, objektový datový zdroj pro novou třídu se objeví až po překladu. Problém vzniká také, pokud chceme jít jinou cestou, než nám předvolili autoři.

Při přepnutí do Visual Studia, vzhled a vazby ne vždy korektně fungují, někdy se ani nenačte grafický náhled XAML dokumentu.

Dalším překvapením byl jiný vzhled prvků na jiných operačních systémech, zde se jedná například o prvek *GroupBox*, jehož vzhled se lišil velice výrazně.

Několikrát se také stalo, že po výběru prvku nedošlo k zobrazení příchytných bodů a bodů pro změnu velikosti prvku. Případně se toto nastavení zobrazilo a při dalším výběru zmizelo.

Jednou se také stalo, že po korektním překladu, provedeným po tvorbě animace a ukončení aplikace nastala výjimka v editoru XAMLu, a bylo nutné se vrátit o několik kroků zpět, aby se editor stal opět funkčním.

Výhody jsou hlavně v možnosti vytvářet pokročilou a na pohled velice působivou grafiku i s minimálními znalostmi kódu. Možnost napojování dat mezi všemi způsoby šetří práci a chyby, a vytváří aplikační kód elegantnější. Kód pro vzhled aplikace, v XAMLu, až tak elegantní vždy není, ale lze pracovat bez jakýchkoliv jeho úprav.

U databáze se ukázala téměř bezproblémová práce s objektovou reprezentací dat. Obdobně se jako velice užitečná a dobře fungující ukázala práce se strukturou objektů. Možnost vazby na objekty představuje velký přínos technologie Data Binding. Práce s XML soubory je také jednoduchá a dobře fungující, problém je s pouze jednosměrnou implicitní vazbou. Dalším velkým přínosem této technologie je možnost vazby mezi jednotlivými grafickými prvky resp. mezi proměnnými těchto elementů.

8.2 Zhodnocení práce

Jak je patrné z vyhodnocení zkušebního příkladu, po nastudování této práce byli studenti schopni aplikovat získané znalosti v oblasti Data Bindingu, a také v oblasti designování aplikace.

Většina studentů testující tuto práci využila jako datový zdroj XML soubor, což dokazuje dobrou podporu tohoto typu datového zdroje v technologii WPF. Studenti byli bez předchozích znalostí XML schopni vytvořit vlastní funkční XML soubor jako zdroj dat a využít jej ve vlastní aplikaci. Student, který využil jiný datový zdroj, konkrétně objektový datový zdroj, dokazuje, že práce doplnila jeho předchozí znalosti a byl schopen je využít i v této technologii.

Další věcí, kterou se studenti z této práce byli schopni naučit, je schopnost využít možnosti, které nabízí prostředí MS Blend, hlavně v oblasti designování aplikací. A v této oblasti spatřují jeden z největších přínosů této práce i této technologie.

Jako velký přínos při studiu této technologie studenti označovali výuková videa. Tím byl úspěšně splněn cíl tvorby audiovizuálních souborů.

Z uvedených skutečností tedy vyplývá, že práce plní hlavní cíl, kterým bylo vytvořit výukový kurz na výuku Data Bindingu ve WPF.

Ověření kurzu v praxi nebylo hlavním cílem a vzorek testovaných subjektů nebyl dostatečně velký. Zhodnocení výstupů by bylo jistě objektivnější při větším vzorku studentů.

9 Použitá literatura

1. PETZOLD, Charles. *Mistrovství ve Windows Presentation Foundation*. Jakub Mikulaštík, Jiří Fadrný. [s.l.] : Computer Press, a.s., 2008. 917 s. ISBN 978-80-251-2141-2.
2. ŠTURALA, Aleš. WPF - Hello "XAML" World. *Vyvojar.cz* [online]. 2007 [cit. 2010-01-14]. Dostupný z WWW: <http://www.vyvojar.cz/Articles/447-1-wpf-hello-xaml-world.aspx>.
3. *Windows Presentation Foundation* [online]. c2010 [cit. 2010-01-14]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms754130.aspx>.
4. ŠTURALA, Aleš. WPF - úvod. *Vyvojar.cz* [online]. 2007 [cit. 2010-01-14]. Dostupný z WWW: <http://www.vyvojar.cz/Articles/445-0-wpf-uvod.aspx>.
5. *MS WindowsClient.net WPF* [online]. c2010 [cit. 2010-01-14]. Dostupný z WWW: <http://windowsclient.net/wpf/white-papers/when-to-adopt-wpf.aspx>.
6. *Data Binding Overview* [online]. 2008 [cit. 2010-01-14]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms752347.aspx>.
7. ŠTURALA, Aleš. Data Binding. *Netstudent.cz* [online]. 2007 [cit. 2010-01-14]. Dostupný z WWW: <http://www.netstudent.cz/%C4%8Cl%C3%A1nky/tabid/56/articleType/ArticleView/articleId/58/5-WPF--Data-Binding.aspx>.
8. KOUŘIL, Lukáš. DataBinding jak jej neznáte. *Netstudent.cz* [online]. 2008 [cit. 2010-01-14]. Dostupný z WWW: <http://www.netstudent.cz/%C4%8Cl%C3%A1nky/tabid/56/articleType/ArticleView/articleId/140/Default.aspx>.
9. ŠTURALA, Aleš. *Data Binding 2* [online]. [2008] [cit. 2010-01-14]. Dostupný z WWW: <http://wpfstart.cz/Tutorials/Databinding_2>.
10. NĚMEC, Luboš. Microsoft Expression Blend 3 Preview. *GRAFIKA online* [online]. 2009 [cit. 2010-01-14]. Dostupný z WWW: <http://www.grafika.cz/art/webdesign/expressionblend3preview.html>.
11. Mozilla Foundation. *Firefox Browser* [online]. 2010 [cit. 2010-03-06]. Videos. Dostupné z WWW: <http://www.mozilla.com/en-US/firefox/video/?video=personas>.
12. The band Fenster. *Open source audio* [online]. 2009 [cit. 2010-03-06]. Free software song. Dostupné z WWW: <http://www.archive.org/details/FreeSoftwareSong_131>.

13. KOSEK, Jiří. XML. *Kosek.cz* [online]. 1999, [cit. 2010-03-06]. Dostupný z WWW: <http://www.kosek.cz/clanky/xml/xml-uvod.html>.
14. ŠTURALA, Aleš. WPF Animace. *Vyvojar.cz* [online]. 2007, č. 4, [cit. 2010-03-06]. Dostupný z WWW: <http://www.vyvojar.cz/Articles/456-4-wpf-animace.aspx >.
15. JIRAVA, Jaroslav. Používáme Model-View-ViewModel – úvod. *Xaml.cz* [online]. 2010, 1, [cit. 2010-03-06]. Dostupný z WWW: <http://xaml.cz/wpf/pouzivame-model-view-viewmodel-uvod/>.
16. BERNARD, Borek. Úvod do architektury MVC. *Root.cz* [online]. 2009, 1, [cit. 2010-03-06]. Dostupný z WWW: <http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc/>.
17. Vyučovací metody. *Infogram.cz* [online]. 2009, [cit. 2010-03-09]. Dostupný z WWW: <http://www.infogram.cz/article.do?articleId=1306>.
18. MAŇÁK, Josef; ŠVEC, Vlastimil. *Výukové metody*. Brno : Paido, 2003. ISBN 80-7315-039-5.
19. Metody vyučování. *Ireferáty.cz* [online]. 2008, [cit. 2010-03-09]. Dostupný z WWW: <http://ireferaty.lidovky.cz/304/4103/Metody-vyucovani>.
20. *Organizační formy výuky* [online]. [s.l.] : Katedra chemie (Západočeská univerzita, Fakulta pedagogická), 2006 [cit. 2010-03-09]. Dostupné z WWW: <http://www.kch.zcu.cz/cz/kfs/6-Organizacni_formy_vyuky.pdf>.
21. Klíčové kompetence In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 11. 3. 2009 [cit. 2010-03-09]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Kl%C3%AD%C4%8Dov%C3%A9_kompetence#Kompetence_k_.C5.99e.C5.A1en.C3.AD_probl.C3.A9m.C5.AF>.
22. Vyučovací metody. *Infogram.cz* [online]. 2009, [cit. 2010-03-09]. Dostupný z WWW: <http://www.infogram.cz/article.do?articleId=1302>.

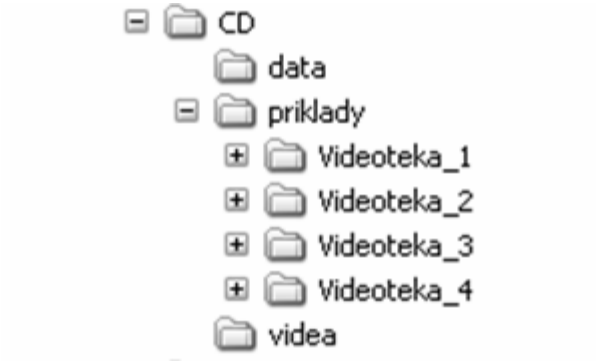
10 Seznam obrázků

11 Seznam příloh na CD

- WPF & DataBinding.pdf – Text bakalářské práce ve formátu pdf.

11.1 Data

- Audio.png
- Convertor.cs
- DVD.png
- Item.cs
- ItemCollection.cs
- Items.cs
- Video.png
- Videoteka.mdf
- Videoteka_log.ldf
- Videoteka.txt
- Videoteka.xml



11.2 Priklady

- Videoteka_1
- Videoteka_2
- Videoteka_3
- Videoteka_4

11.3 Videa

- Videoteka_1.wmv
- Videoteka_2.wmv
- Videoteka_3.wmv
- Videoteka_4.wmv
- Html rozhraní pro přehrávání videí.

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Pedagogická fakulta
Katedra informatiky
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Vilém JANDA**

Studijní program: **B1802 Aplikovaná informatika**

Studijní obor: **Výpočetní technika**

Název tématu: **Windows Presentation Foundation & Data Binding**

Zásady pro vypracování:

Ve verzi 3v5.NET Framework se klade důraz na spojení grafických a řídících vlastností aplikací tvořených za pomocí Visual Studio 2008. Úkolem diplomanta je vytvoření uceleného kurzu ve formě e-learning studijních materiálů pro výklad technologie Data Binding ve Windows Presentation Foundation (WPF).
Diplomant vytvoří výkladové video soubory pro samostatné studium a multimediálními soubory. Výklad musí být doplněn, jak řešenými příklady, tak příklady k procvičení.

Zadání bakalářské práce

Rozsah grafických prací:
Rozsah pracovní zprávy: **60**
Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Microsoft [online]. Trvale aktualizováno. Microsoft, c1980-2009, 2009* [cit. 2009-04-10]. Dostupný z WWW: <http://www.microsoft.cz>.**
2. **PETZOLD, Charles. Mistrovství ve Windows Presentation Foundation : Aplikace=Kód a Markup. Jakub Mikulaštík, Jiří Padrný. 1. vyd. PRAHA : Computer Press, a-s., 2008. 928 s., CD. ISBN 798-80-251-2141-2.**
3. **Praktické užití Silverlight 2.0: Data Binding [online]. 2009, 2009 [cit. 2009-04-10]. Dostupný z WWW: <http://zdrojak.root.cz/clanky/prakticke-uziti-silverlight-2-0-data-binding/>.**

Vedoucí bakalářské práce: **Ing. Václav Novák, CSc.**
Katedra informatiky

Datum zadání bakalářské práce: **23. dubna 2009**
Termín odevzdání bakalářské práce: **30. dubna 2010**

L.S.

doc. PhDr. Alena Hošpesová, Ph.D.
děkanka

PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 23. dubna 2009