Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Práce s dokumenty ve formátu XML v prostředí .NET Frameworku

Michal Kůrka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Kůrka Michal

Informatika

Název práce

Práce s dokumenty ve formátu XML v prostředí .NET Frameworku

Anglický název

Working with XML documents in .NET Framework

Cíle práce

Cílem práce je analyzovat problematiku zpracování XML dokumentů s využitím prostředků .NET Frameworku v programovacím jazyce C#. V praktické části pak získané poznatky demonstrovat na ukázkové aplikaci pracující s XML dokumenty.

Metodika

Metodika bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část práce je zaměřena na vytvoření ukázkové aplikace. Na základě syntézy teoretických poznatků a výsledků zpracování ukázkové aplkace budou formulovány závěry práce.

Harmonogram zpracování

06/2012-12/2012 - analýza informačních zdrojů
01/2013 - kontrola průběhu práce (zápočet)
01/2013-02/2013 - vytvoření ukázkové aplikace
03/2013 - finalizace BP

Rozsah textové části
35-40 stran

Klíčová slova
.NET, C#, XML, tag, X-Path, LINQ, strukturovaný dokument, značkovací jazyk

Doporučené zdroje informací

XML pro každého, Jiří Kosek, Grada Publishing, 2000, ISBN 80-7169-860-1

XML technologie, Mlýnková I., Nečaský M., Pokorný J., Richta K., Toman K., Toman V., Grada Publishing, 2008, ISBN 978-80-247-2725-7

Microsoft Visual C# 2010 Krok za krokem, John Sharp, Computer Press, 2010, ISBN 978-80-251-3147-3

Vedoucí práce
Brožek Jiří, Ing.

Termín odevzdání
březen 2013

Ing. Martin Pelikán, Ph.D.
Vedoucí katedry

prof. Ing. Jan Hron, DrSc., dr.h.c.
Děkan fakulty

V Praze dne 13.9.2012

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Práce s dokumenty ve formátu XML v prostředí .NET Frameworku" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil(a) autorská práva třetích osob.

V Praze dne 15. 3. 2013 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) vedoucímu bakalářské práce panu Ing. Jiřímu Brožkovi, Ph. D. za vedení a cenné rady při vypracování bakalářské práce.

Práce s dokumenty ve formátu XML v prostředí .NET Frameworku

Working with XML documents in .NET Framework

Souhrn

Bakalářská práce se zabývá formátem XML a možnostmi jeho zpracování v prostředí .NET Framework. V úvodní části se věnuje popisu, vývoji, sytaxi a použitelnosti jazyka XML a popisuje platformu .NET Framework z hlediska vývoje, principu a knihoven určených k analýze a generování XML dokumentů.

V druhé části věnující se tvorbě ukázkové aplikace je provedena analýza požadavků, nastíněna funkčnost aplikace a popis jejího ovládání. V kapitole věnující se implementaci je vytvořeno uživatelské prostředí a popsáno použití knihoven s ukázkami kódu v programovacím jazyku C#.

Summary

The bachelor thesis deals with the XML format and the possibilities of its processing in the environment .NET Framework. The first part is focused on the description, development, syntax and usability of XML and describes the .NET Framework platform in terms of development, principle and libraries dedicated to the analysis and generating of XML documents.

In the second part focused on creating a sample application analysis requirements, outlines functionality of the application and describes its operation. In the chapter focused on the implementation there is created GUI and described using libraries with examples of code in C#.

Klíčová slova: XML, XML dokument, .NET Framework, XML DOM, C#, element, System.Xml

Keywords: XML, XML document, .NET Framework, XML DOM, C#, element, System.Xml

Obsah

Seznam diagramů

Seznam obrázků

Seznam tabulek

Seznam příkladů

1 Úvod

XML, tedy rozšiřitelný značkovací jazyk (z anglického eXtensible Markup Language), je standardem schváleným konsorciem W3C (World Wide Web Consorcium) pro značkování dokumentů. Definuje obecnou syntaxi, která se používá pro označování jednoduchými, člověku srozumitelnými značkami. Ustanovuje standardní formát pro počítačové dokumenty. Tento formát je natolik flexibilní, že jej lze dále upravit pro tak různorodé oblasti, jako jsou webové stránky, elektronická výměna dat, vektorová grafika, genealogie, aplikace katastrů nemovitostí, serializace objektů, vzdálená volání procedur, či systém hlasové pošty. [1]

XML je v současnoti přirozenou součástí všech forem programování, aby však aplikace mohly plně využívat všech výhod XML, musí být součástí operačního systému nebo dané softwarové platformy potřebná infrastuktura. Za normálních okolností má tato infrastruktura podobu nástrojů, které zajišťují syntaktickou analýzu, ověřování platnosti (validaci) dokumentů, návrh schémat a transformace dokumentů. [2, 3]

Prostředí Microsoft .NET Framework poskytuje komplexní sadu tříd pro práci s XML dokumenty a příbuznými technologiemi na různých úrovních a v souladu s nejnovějšími standardy a doporučeními W3C. Podpora XML v .NET Framework zahrnuje XML 1.0 a 1.1, jmenné prostory XML, model DOM (Document Object Model) Level 3 Core, jazyk XSD (XML Schema Definition), transformace XSLT (Extensible Stylesheet Language Transformation) a výrazy XPath. Základní třídy XML jsou navíc úzce integrovány s dalšími klíčovými součástmi prostředí .NET Framework, včetně přístupu k datům, serializaci a konfiguraci aplikací. [2]

Bakalářská práce je zaměřena na knihovny tříd .NET Frameworku, které pracují s XML dokumenty. Největší pozornost bude věnována knihovně System.Xml, která obsahuje většinu tříd pro práci s XML. Dále se práce okrajově zmíní o třídách System.Xml.Schema, System.Xml.XPath a nastíní možnosti využití knihovny System.Xml.Linq.

V praktické části práce bude vytvořena WPF aplikace pro úpravu XML dokumentu podle zadaných požadavků. Na aplikaci je předvedeno praktické využití metod a vlastností tříd výše zmiňovaných knihoven.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je představení jazyka XML – jeho historický vývoj, popis formátu XML, jeho syntaxi a použití v současné době a analyzovat možnosti zpracování XML dokumentu pomocí knihoven .NET Frameworku.

V praktické části je cílem vytvoření formulářové aplikace v prostředí Microsoft Visual Studia 2010, která bude využívat konkrétních knihoven a tříd .NET Frameworku k analýze, validaci, transformaci a editaci XML dokumentu.

2.2 Metodika

Na základě informací získaných z odborných zdrojů a praxe, budou popsány základy jazyka XML a analyzovány možnosti zpracování XML dokumentu v prostředí .NET Frameworku.

Z teoretických poznatků vychází praktická část, kde jsou získané znalosti z odborných zdrojů a praxe uplatněny k vytvoření ukázkové aplikace.

3 Přehled řešené problematiky

3.1 XML

Data jsou v XML dokumentech obsažena v textových řetězcích, které jsou uzavřeny do textových značek. Tyto značky by měly co nejstručněji popisovat obsah textu, který uzavírají. Značky mají ve specifikaci XML přesně definovanou syntaxi, kterou musí dodržovat (přípustné názvy, oddělování jednotlivých značek, umístění atd.). Konkrétní textový řetězec spolu s příslušnou značkou se označuje jako element.

XML je meta-značkovací jazyk pro textové dokumenty. To znamená, že nedefinuje pevně danou množinu značek a elementů, které by bylo nutné používat, ale umožňuje autorům dokumentů definovat si vlastní elementy, které lépe vystihují jejich data. [1]

3.1.1 Vývoj XML

XML je následníkem standardního zobecněného značkovacího jazyka SGML (Standard Generalized Markup Language). Jazyk, který se nakonec vyvinul do SGML, navrhli v 70. letech Charles Goldfarb, Ed Mosher a Ray Lorie z IBM a dále jej vyvíjelo několik set lidí po celém světě, až byl nakonec v roce 1986 přijat standard ISO č. 8879. Účelem SGML bylo vyřešit z velké části stejné problémy, jako nyní řeší XML. Byl to a je sémantický a strukturovaný značkovací jazyk, určený pro textové dokumenty.

Hlavním problémem SGML je však jeho složitost. Oficiální specifikace SGML čítá více než 150 velmi technických stránek, pokrývá mnoho zvláštních případů a nepravděpodobných situací. Jazyk je tak složitý, že neexistuje téměř žádný software, který by jej plně implementoval.

V roce 1996 Jan Bosak, Tim Bray, C. M. Sperberg-McQueen, James Clark a několik dalších lidí začalo pracovat na „odlehčené“ verzi SGML. Tato verze si zachovala většinu silných vlastností SGML, zatímco vynechala spoustu věcí, které byly nadbytečné, příliš komplikované na implementaci, nejasné koncovým uživatelům nebo se za předchozích 20 let zkušeností se SGML prostě neprokázalo, že by byly užitečné. Výsledkem byla v únoru 1998 první verze XML 1.0, která měla okamžitý úspěch. [1]

Výsledkem dalšího vývoje byly standard XML Namespaces (jmenné prostory), rozšiřitelný jazyk XSL pro stylové šablony, který se později rozdělil na aplikace XSLT

(XSL Transformation) a XSL-FO (XSL Formatting Objects), podpora kaskádových stylů CSS, standardy XLink, XPointer a XPath, dále aplikace XML Schema a další. Důležitou součástí vývoje bylo vytvoření jednotného rozhraní pro přístup k obsahu dokumentu XML z programů napsaných v jazycích Java, Javascript, C++, C# a dalších.

3.1.2 Formát XML

XML je formát pro reprezentaci a přenos obecných dokumentů. Při návrhu XML se autoři z konsorcia W3C řídili následujícími principy:

- Formát XML musí být použitelný v rámci internetu,
- formát XML by měl podporovat širokou škálu aplikací,
- formát XML musí být kompatibilní s formátem SGML,
- musí být snadné vytvářet programy, které manipulují s dokumenty v XML,
- množství variant XML by mělo být minimální – nejlépe žádné,
- XML dokumenty by měly být čitelné a pochopitelné i pro člověka.

Na základě těchto principů navhrli definici XML, která zahrnuje 2 části:

1. Definici, co je to XML dokument,
2. definici programů, které zpracovávají XML dokumenty – XML procesorů. [3]

3.1.3 XML dokument

XML dokument obsahuje vždy text (nikdy ne binární data) a lze jej tedy vytvářet, prohlížet a upravovat v jakémkoli textovém editoru (např. Poznámkový blok ve Windows apod.). Velmi jednoduchý XML dokument se může skládat jen z jednoho elementu, jak je vidět na příkladu 3-1.

Příklad 3-1 Velmi jednoduchý XML dokument

```
<urad>
  Velvyslanectví ČR
</urad>
```

Nejčastěji je XML dokument uložen v souboru s nějakým názvem, například urad.xml nebo nějakým podobným. Název souboru však není důležitý, zrovna tak by se mohl jmenovat urady.txt, aaa.ui apod. Není ani nutné, aby byl dokument uložen v nějakém souboru, může jít o záznam v databázi, odpověď serveru na dotaz prohlížeče,

či dokument vygenerovaný nějakým programem. Pro XML parsery je důležitá jeho struktura.

3.1.4 Elementy, značky a znaková data

Dokument z příkladu 3-1 se skládá z jediného elementu typu `urad`. Tento element je tvořen počáteční značkou `<urad>`, koncovou značkou `</urad>` a obsahem elementu (znakovými daty), což je vše mezi počáteční a koncovou značkou. V tomto případě je to textový řetězec `Velvyslanectví ČR`.

Značky slouží k označení určitých prvků (částí) dokumentu. Značky mají otevírací závorku (start-tag), např. `<urad>`, a zavírací závorku (end-tag), např. `</urad>`. Pokud je text mezi závorkami prázdný, lze dvojici otevírací a zavírací závorky nahradit prázdným elementem (empty-element), v našem případě např. `</urad>`. [3]

Na příkladu 3-2 je vidět možné použití závorek pro neprázdné elementy `urad` a `nazev` a pro prázdné elementy `ulice` a `mesto`.

Příklad 3-2 Párová značka a nepárová značka

```
<urad>
  <nazev>Velvyslanectví ČR v Paříži</nazev>
  <mesto></mesto>
  <ulice />
</urad>
```

Pomocí značek XML vyznačíme syntaktickou strukturu dokumentu. Sémantika značek ani význam jejich obsahu nejsou pomocí XML definovány. Význam XML spočívá v tom, že struktura dokumentu je známa, lze ji kontrolovat a zpracovat obecnými nástroji. Libovolná aplikace si může strukturu dokumentu zjistit a dle této struktury jej zpracovat. [3]

3.1.5 Atributy

Elementy v XML mohou mít atributy. Atribut je dvojice název-hodnota, připojená k počáteční značce příslušného elementu. Názvy jsou od hodnot odděleny znakem rovnítka a případně mezerami. Hodnoty jsou uzavřeny v jednoduchých nebo dvojitých uvozovkách. [1]

Atributy je možné připojit i k prázdným elementům a jejich počet není omezen. Na příkladu 3-3 je vidět možné použití atributů.

Příklad 3-3 Příklad použití atributů

```
<urad zeme='Francie' iso='fr'>
  <nazev>Velvyslanectví ČR v Paříži</nazev>
  <mesto>Paříž</mesto>
  <ulice nazev='Avenue Charles Floquet' číslo='15' />
</urad>
```

3.1.6 XML-stromy

Dokumenty XML mají strukturu stromu. XML dokument uvedený v příkladu 3-4 je složen z jednoho elementu typu `urad`. Tento element obsahuje 4 synovské elementy: `pusobnost`, `ekonom`, `telefon` a `fax`. Element `pusobnost` dále obsahuje dva vlastní synovské elementy a element `ekonom` jeden.

Příklad 3-4 XML dokument částečně popisující úřad

```
<urad id="6" zeme="alzirsko">
  <pusobnost>
    <zeme iso="DZ">alzirsko</zeme>
    <zeme iso="ML">mali</zeme>
  </pusobnost>
  <ekonom>
    <vedouci>Ing. Václav Žilka </vedouci>
  </ekonom>
  <telefon>+213-2123-0056</telefon>
  <fax>+213-2123-0133</fax>
</urad>
```

Element `urad` se nazývá rodičem elementů `pusobnost`, `ekonom`, `telefon` a `fax`. Element `pusobnost` je dále rodičem dvou elementů `zeme` a element `ekonom` je rodičem elementu `vedouci`.

V XML dokumentu může mít každý rodič více potomků, ale každý potomek (synovský element) může mít právě jednoho rodiče (výjimku tvoří kořenový element, viz níže). Každý element má právě jeden rodičovský element, to znamená, že je zcela

uzavřen v jiném elementu. Jestliže se otevírací závorka (start-tag) nachází uvnitř jiného elementu, musí být uvnitř tohoto elementu i jeho zavírací závorka (end-tag). Překrývání značek, jako je `<ekonom><vedouci>`Ing. Václav Žilka`</ekonom></vedouci>`, je v XML zakázáno.

Každý dokument XML obsahuje jeden element bez rodiče, do něhož jsou uzavřeny všechny ostatní elementy. Ve všech předchozích příkladech to byl element urad a takový element se nazývá kořenový element. Každý správně formulovaný XML dokument obsahuje právě jeden kořenový element. Protože se elementy nesmějí překrývat a protože všechny elementy, vyjma kořenového, mají právě jednoho rodiče, XML dokumenty tvoří datovou strukturu, kterou programátoři nazývají strom. [1] Tento vztah, aplikovaný na příkladu 3-4, je znázorněn na diagramu 3-1. Elementy jsou označeny zelenou barvou, znaková data červenou barvou a šipky představují vztah obsažení.

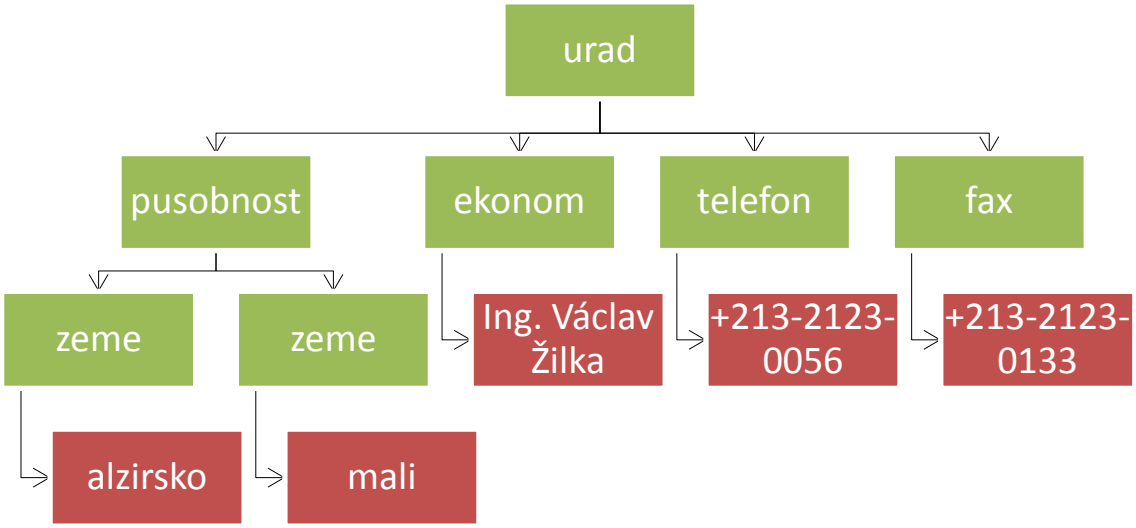


Diagram 3-1 Stromový diagram pro příklad 3-4

3.1.7 Názvy v XML

Jako názvy XML se označují názvy elementů, atributů a dalších méně běžných konstrukcí a vývojáři mohou tyto názvy vytvářet dle jejich potřeby, ale platí pro ně striktní omezení.

Element i jiné názvy v XML mohou obsahovat libovolné alfanumerické znaky. Mezi tyto znaky patří standardní anglická abeceda A až Z, a až z a číslice 0 až 9. Názvy

mohou rovněž obsahovat neanglická písmena, čísla a symboly, jako například ž, € nebo ¥. Mohou navíc obsahovat interpunkční znaky podtržítko "_", pomlčka "-" a tečka ".".

Názvy v XML nesmí obsahovat jiné interpunkční znaky, jako třeba uvozovka, apostrof, znak pro dolar, stříška, symbol procenta nebo středník. Dvojtečka je povolena, její použití je však vyhrazeno pro jmenné prostory. Názvy v XML dále nesmí obsahovat prázdné znaky jako mezera, nový řádek, tabulátor nebo pevná mezera.

Názvy v XML mohou začínat výhradně písmeny, symboly nebo znakem podtržítka. Nesmí začínat číslicí pomlčkou nebo tečkou. Co se týká délky elementu nebo jiného názvu XML, není stanoveno žádné omezení. [1]

3.1.8 Validace

XML data zahrnují elementy, atributy, textová data a některé speciální znaky jako jsou komentáře, sekce CDATA, instrukce pro zpracování apod. Hlavní síla jazyka XML pak spočívá především v možnosti specifikovat sady přípustných značek a jejich konkrétní strukturu, a ty přiřazovat jednotlivým XML dokumentům. Pokud daný dokument obsahuje pouze značky z konkrétní definované sady s odpovídající strukturou, je vůči této sadě validní, resp. se jedná o její instanci. Popis této sady se nazývá XML schématem daného XML dokumentu. XML schéma tedy obsahuje popis přípustné struktury XML dokumentů. [3]

3.2 .NET Framework

.NET Framework je softwarové rozhraní vyvinuté společností Microsoft a určené primárně pro operační systémy Microsoft Windows (od verze Windows NT), ale lze jej použít i na jiných platformách.[1]

.NET Framework poskytuje služby pro vytváření, zavádění a běh desktopových, webových a mobilních aplikací a webových služeb. Klíčové komponenty rozhraní .NET Framework tvoří modul společného běhového prostředí (CLR – Common Language Runtime), které poskytuje správu paměti a dalších systémových služeb, a rozsáhlá knihovna tříd, která obsahuje prověřený a opakovaně použitelný kód pro všechny hlavní oblasti vývoje aplikací. Platforma .NET Framework poskytuje spravované prostředí pro spouštění, vývoj, nasazení a integraci s širokou škálou programovacích jazyků. [6, 7]

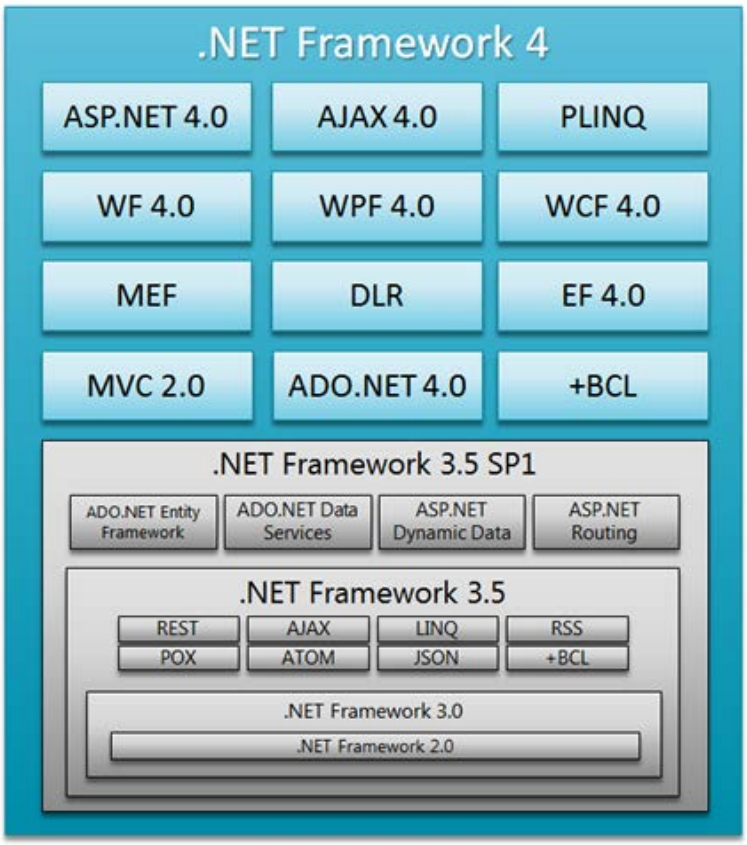
[1] Na cílové platformě musí být nainstalované příslušné knihovny. Viz například projekt Mono (http://www.mono-project.com)

3.2.1 Vývoj .NET Framework

První oficiální verze, tedy .NET Framework 1.0, byla vydána v únoru 2002 a zatím poslední verze .NET Framework 4.5 vyšla v srpnu 2012. V této práci se zaměříme na verzi .NET Framework 4 (lze doinstalovat do Windows XP a novějších), která byla vydána v dubnu 2010 současně s Visual Studiem 2010. Všechny verze .NET Framework jsou složeny ze společného běhového prostředí (CLR - Common Language Runtime), základní knihovny tříd (BCL – Basic Class Library) a dalších spravovaných knihoven. Vývoj rozhraní je popsán v tabulce 3-1 a graficky znázorněn na obrázku 3-1.

Tabulka 3-1 Nové funkce ve verzích rozhraní .NET Framework

Verze rozhraní	Verze CLR	Verze sady Visual Studio	Popis přidaných funkcí
1.0	1.0	Visual Studio .NET	První verze modulu CLR a první verze základní knihovny tříd (BCL).
1.1	1.1	Visual Studio .NET 2003	Aktualizace prostředí ASP.NET a ADO.NET. Souběžné spouštění aplikací na počítači s více verzemi CLR.
2.0	2.0	Visual Studio 2005	Nová verze CLR s dodatky k BCL, včetně druhových, rodových kolekcí a významných doplňků ASP.NET.
3.0	2.0	Visual Studio 2005	Přidáno Windows Presentation Foundation (WPF), Windows Communications Foundations (WCF), Windows Workflow Foundation (WF) a CardSpace.
3.5	2.0	Visual Studio 2008	LINQ, weby s povolenou funkcí AJAX, profil klienta .NET Framework, dynamická data a další
4.0	4	Visual Studio 2010	Nová verze CLR, rozšíření BCL, přidáno Managed Extensibility Framework (MEF), dynamic language runtime (DLR) a kód smlouvy.



Obrázek 3-1 Vývoj rozhraní .NET Framework [834]

3.2.2 Princip .NET Framework

Zdrojový kód programovacího jazyka (C#, F#, atd.) se překládá do spravovaného kódu (managed code). Spravovaný kód je mezijazykem (IL – intermediate language), protože je napůl cesty mezi jazykem vyšší úrovně (C#, F#, atd.) a jazykem nejnižší úrovně (jazykem symbolických adres nebo strojovým kódem).

Společné běhové prostředí (CLR) za běhu programu průběžně překládá zdrojový kód způsobem Just In Time (JIT). Tento proces se liší od interpretace, kterou obvykle využívají skriptovací programovací jazyky jako Perl a JScript. JIT-překladač nepřekládá nějakou funkci nebo metodu při každém jejím volání; udělá to pouze při jejím prvním volání a při tom vytvoří strojový kód nativní pro platformu, na které program běží. Za běhu aplikace se překládá pouze potřebný zdrojový kód. Pokud aplikace obsahuje například zdrojový kód pro tisk a uživatel nebude tisknout nějaký dokument, nebude tento kód nikdy potřeba, a proto jej JIT-překladač nikdy nepřeloží. [5]

3.2.3 Knihovna tříd rozhraní .NET Framework

Knihovna tříd rozhraní .NET Framework je knihovnou tříd, rozhraní a typů hodnot, které poskytují přístup k funkčnosti systému. Je to základ, na kterém jsou postaveny aplikace, komponenty a ovládací prvky rozhraní .NET Framework. Knihovna tříd je rozdělena dle jmenných prostorů (oborů názvů).

V této práci jsou využity hlavně třídy z oborů názvů System.Xml, System.Xml.Schema, System.Xml.XPath a System.Xml.Linq.

3.2.4 API platformy .NET Framework pro práci s XML

Jádrem XML v prostředí .NET Framework jsou dvě abstraktní třídy XmlReader a XmlWriter. Z těchto tříd vycházejí všechny ostatní třídy XML, včetně tříd XML DOM (XML Document Object Model), a do značné míry je využívají i jiné subsystémy pro analýzu a generování dat XML.

Readery a writery XML představují primitivní vstupní a výstupní mechanismy pro načítání a zápis dokumentů XML a slouží k programování složitějších a propracovanějších komponent. Ke zpracování dat XML se tedy může přistupovat dvojím způsobem. Může se použít některá ze specializovaných tříd, jež představují nadstavbu nad třídami XmlReader a XmlWriter, nebo se mohou použít třídy dokumentů, které zpřístupňují svůj obsah prostřednictvím známého XML DOM. [2]

3.2.5 XML readery

Prostředí .NET Framework má k dispozici řadu readerů, které slouží ke zpracování proudu dat. Reader se na zpracovávaný proud dat dívá jako na logickou posloupnost informačních jednotek a prochází jej v režimu read-only, forward-only (data pouze čte a nemůže se vracet zpět), bez ukládání do mezipaměti. Informační jednotkou může být jeden řádek databáze, jeden řádek textu, jeden bajt, jeden uzel apod.

Readery XML jsou jedním z typů readerů .NET. Třída analyzuje (separuje) obsah souboru XML a přesouvá se přitom od jednoho uzlu k dalšímu. V tomto případě je základní jednotkou informace uzel XML (element, atribut, komentář, atd.)

Reader XML vytváří externě dostupné programovací rozhraní, přes něž se může volající kód připojit k datům, které potřebuje, a načíst je. Klientským aplikacím je vrácen odkaz na instanci třídy readeru, která abstrahuje datový proud, ležící v nižší

vrstvě. Metody třídy readeru umožňují procházet obsah proudu, a to uzel po uzlu. Pro readery přestává být dokument XML textovým souborem doplněným značkami (tagy) a stává se serializovanou kolekcí uzlů. [2]

3.2.6 XML writery

XML API platformy .NET odděluje analýzu od úprav a zápisu. Při zápisu dat jsou vytvářeny nové dokumenty XML a je stanoveno, které uzly XML mají být vytvořeny (elementy, atributy, atd.). Writer pracuje s proudem informací, postupně vypisuje jeho obsah uzel po uzlu. Nemá k dispozici nástroje XML DOM pro náhodný přístup do libovolného místa proudu, díky tomu však nezabírá tolik místa v paměti.

XML writery programátorům umožňují zapisovat dokumenty stejným způsobem, jakoby je psali a ukládali v textových editorech. Umožňují použít předponu jmenného prostoru, vybrat znak pro vyplňování prázdných míst, velikost odsazení, typ uvozovek a použitý znak nového řádku a jak má být zacházeno s bílými znaky (např. mezera, odřádkování, tabulátor).[2]

3.2.7 XML DOM

Platforma .NET Framework obsahuje také třídy, které načítají a editují dokumenty XML v souladu se specifikacemi W3C DOM Level 1 a Level 2. V XML DOM je klíčová třída XmlDocument.

XML DOM nabízí programátorovi stromovou reprezentaci dokumentů XML, uloženou v paměti počítače, a podporuje změnu aktuální pozice a úpravy obsahu dokumentu.

Reprezentace XML dokumentu v XML DOM je plně editovatelná. K atributům a textu lze přistupovat v náhodném pořadí, uzly lze přidávat a odstraňovat. Aktualizace do paměti nahraného dokumentu XML DOM se provádí tak, že se nejdříve vytvoří objekt uzlu (instance třídy XmlNode) a poté se naváže na existující strom. Při použití XML DOM ke všem změnám dochází jen v paměti, je tedy nutné je před návratem zapsat na úložné médium. [2, 9]

3.2.8 LINQ

LINQ (Language Integrated Query) integrovaný dotazovací jazyk v .NET Framework zaceluje mezeru mezi světem objektově orientovaného programování a světem dat. Prostřednictvím jazyka LINQ je možné přistupovat k datům z mnoha různých zdrojů. V .NET Framework 4 jsou k dispozici implementace: LINQ pro objekty, LINQ pro SQL, LINQ pro datové sady, LINQ pro entity a LINQ pro XML. [5]

Implementace LINQ pro XML (knihovna System.Xml.Linq) umožňuje načtení a serializaci XML souboru nebo proudu, vytvoření stromu XML, dotazování do XML stromu, manipulaci s XML stromy, ověření struktury XML pomocí XSD.

4 Tvorba ukázkové aplikace

V průběhu této kapitoly bude vytvořena aplikace pro otevření, úpravu a uložení XML dokumentu, který obsahuje informace o obchodních zástupcích České republiky na zastupitelských úřadech ministerstva zahraničních věcí.

Jelikož je XML dokument využíván flashovou aplikací pro zobrazení dat v internetové prezentaci, je nutné, aby data a struktura v něm obsažená byla vždy ve správné formě, což zajistí validace XML dokumentu pomocí XSD (XML Schema Definition).

K vývoji aplikace je použit program Visual Studio 2010 Professional[2] od společnosti Microsoft a programovací jazyk C#.

4.1 Analytická část

V této části jsou shrnuty požadavky na aplikaci specifikované zadavatelem, předpokládaná funkčnost aplikace a její ovládání. Dále základní struktura XML dokumentu a představení technologie XAML využité k tvorbě formuláře.

4.1.1 Požadavky na aplikaci

Aplikace by měla splňovat tyto požadavky:

- Použitelnost na OS Windows,
- možnost stažení XML dokumentu z předem zadané webové adresy,
- zobrazení dat v přehledném formuláři,
- snadná editace dat pomocí grafického uživatelského rozhraní (GUI),
- možnost otevření a uložení rozpracovaného dokumentu z(do) lokálního úložiště,
- aplikace sama kontroluje správnou strukturu a typ dat v XML dokumentu.

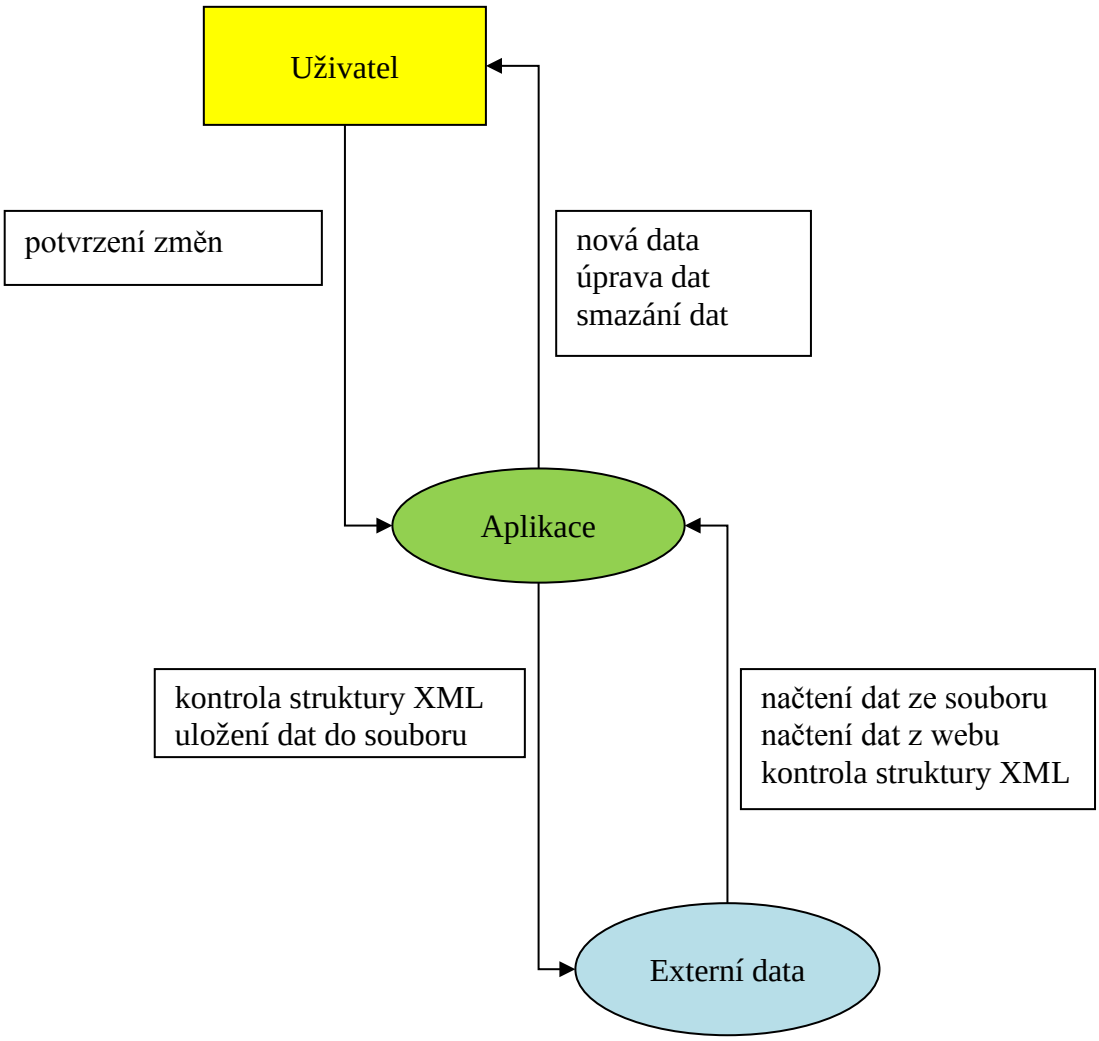
První požadavek je splněn tím, že si uživatel nainstaluje .NET Framework 4, který podporuje OS Windows od verze Windows XP SP3. K realizaci zbylých požadavků bude využita Windows Presentation Foundation (WPF)[3] aplikace vytvořena pomocí Visual Studia, dostupných knihoven .NET Frameworku a jazyka C#.

[2] Více informací o Visual Studiu 2010 Profesional (v angličtině): http://www.microsoft.com/en-us/download/details.aspx?id=16057
[3] Více informací o WPF aplikacích (v angličtině): http://msdn.microsoft.com/cs-cz/library/ms754130%28v=vs.100%29.aspx

4.1.2 Funkčnost aplikace

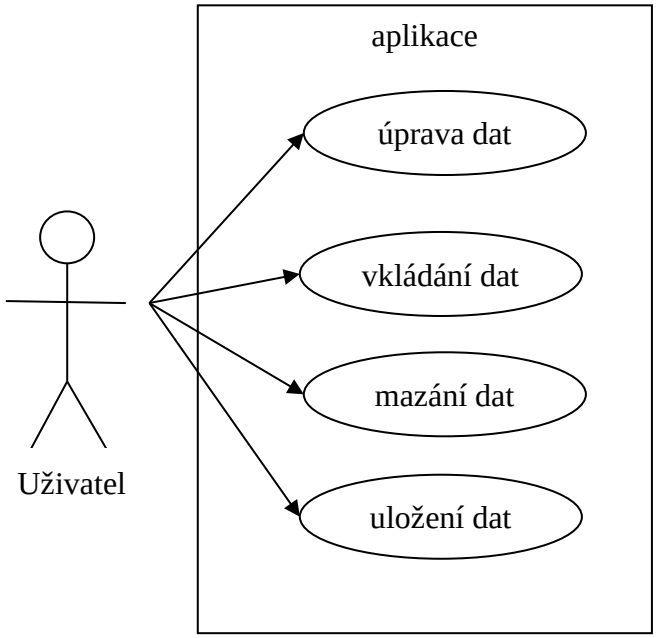
V kontextovém diagramu 4-1 je přehledně znázorněno, jak aplikace komunikuje s okolím, a kde jsou její hranice. Každá vazba vyjadřuje určitou akci, která může nastat. Jelikož je systém navržen jako jednouživatelský, je zde jen jeden terminátor (uživatel).

Diagram 4-1 Kontextový diagram



Vztahy mezi jednotlivými účastníky systému jsou popsány pomocí Use Case modelu v diagramu 4-2. Systém má pouze jednoho účastníka (uživatele), který ovládá všechny činnosti aplikace.

Diagram 4-2 Model ovládání



4.1.3 Ovládání aplikace

Uživatel spustí aplikaci, klikne na tlačítko „Stáhnout“, čímž aplikace stáhne z webové adresy soubor s XML dokumentem a jeho obsah načte do paměti pomocí XML DOM. V pravé části okna se zobrazí všechny úřady ve formě seznamu a uživatel může pomocí ovládacích prvků vkládat nové úřady (tlačítko „Nový“), upravovat údaje u současných („Upravit“), případně některé smazat („Smazat“).

Po dokončení všech úprav se vše uloží do souboru na lokální úložiště tlačítkem „Ulož XML“, odkud je možnost soubor opět načíst pomocí tlačítka „Otevřít“.

4.1.4 XML dokument

Flashová aplikace, která zobrazuje data z XML dokumentu v internetové prezentaci, předpokládá jeho přesnou strukturu a typ dat. Tato struktura je znázorněna na příkladu 4-1, kompletní XML dokument obsahující vzorová data viz příloha XML dokument.

Příklad 4-1 Struktura XML dokumentu

```
<data>
  <item id="" zeme="">
    <pusobnost>
      <zeme iso=""></zeme>
    </pusobnost>
    <ekonom>
      <vedouci></vedouci>
      <jmeno></jmeno>
    </ekonom>
    <telefon></telefon>
    <fax></fax>
    <email></email>
    <www></www>
    <urad>
      <nazev></nazev>
    </urad>
    <poznamka></poznamka>
  </item>
</data>
```

Element `<data>` je kořenový element, tudíž může být v dokumentu pouze jednou a musí uzavírat všechny ostatní uzly, kromě úvodních deklarací definujících použitou verzi XML, typ kódování dokumentu, použité jmenné prostory, cestu k validačnímu souboru a další.

Element `<item>` obsahuje data týkající se vždy jednoho konkrétního obchodního zástupce. Data obsahují informace o zemi, v které sídlí (atribut `zeme`), o zemích v kterých působí (elementy `pusobnost` a `zeme`), jméno obch. zástupce, kontakty a další. Data jsou rozdělena do podřízených elementů a atributů a musí přesně odpovídat popisu připojeného XML schématu (viz příloha XML schéma). XML schéma definuje strukturu dokumentu, možný počet opakování každého uzlu, hodnotové typy dat atd.[4]

4.1.5 XAML

Ve WPF aplikacích je grafické uživatelské rozhraní definováno jazykem XAML (eXtensible Application Markup Language), který vyvinula společnost Microsoft.

[4] Více informací o XML Schema Definition: http://www.w3.org/TR/xmlschema11-1/

XAML je deklarativní jazyk vycházející z jazyka XML, který je zahrnut v .NET Framework od verze 3.0.

Pomocí XAML lze definovat rozvržení formuláře, ovládací prvky, načítání dat, přiřazovat funkce událostem ovládacích prvků, měnit zobrazení (styly) prvků a další. Na příkladu 4-1 je znázorněn ukázkový kód z nově vytvořené aplikace.

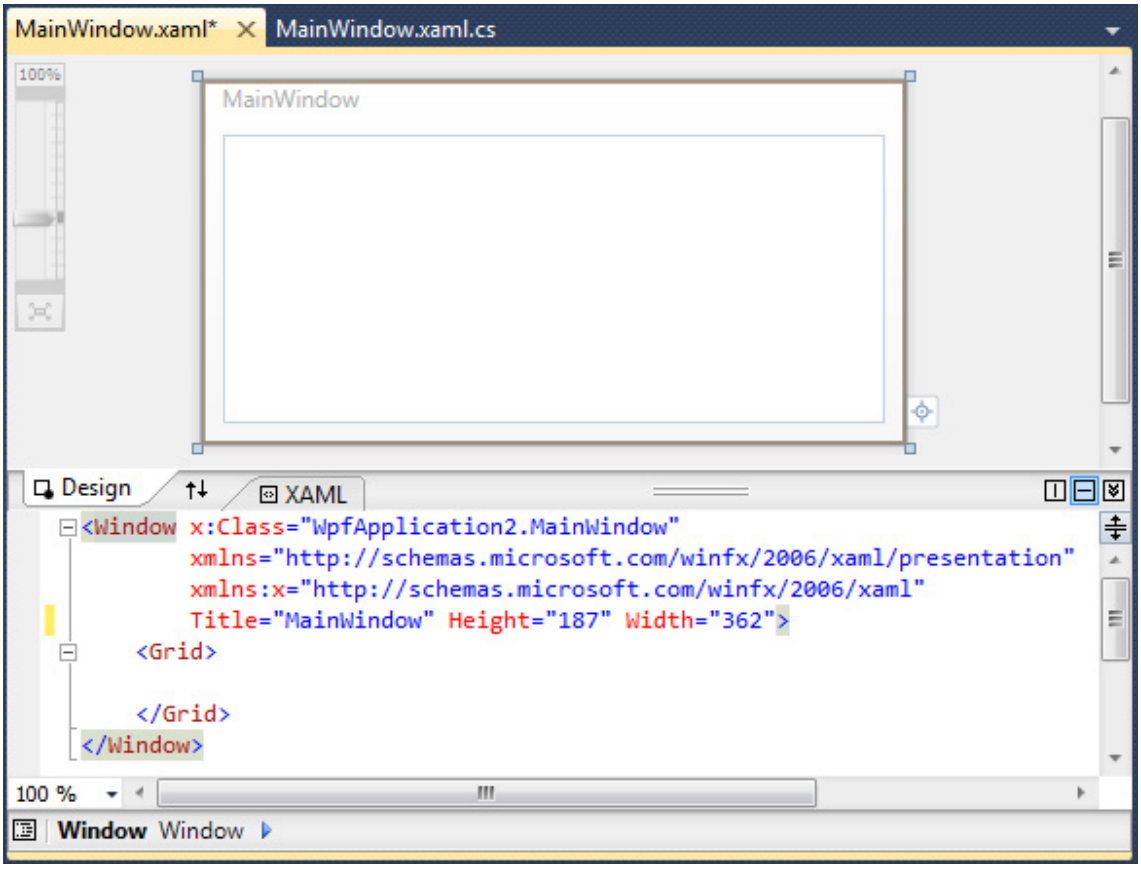
Příklad 4-2 Kód jazyka XAML definující formulářové okno a popisek Ahoj světe!

```
<Window x:Class="AhojSvete.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="197" Width="293">
    <Grid>
        <Label Content="Ahoj světe!" Height="28" HorizontalAlignment="Left"
               Margin="95,66,0,0" Name="label1" VerticalAlignment="Top"
Width="74" />
    </Grid>
</Window>
```

Kompletní kód jazyka XAML, ve kterém je definováno uživatelské rozhraní aplikace vytvářené v rámci této práce, je k nalezení v příloze **XAML dokument**. Vše, co je vytvořeno pomocí jazyka XAML, se dá naprogramovat pomocí jazyka C# (příp. Visual Basic). Není tedy nutné vytvářet uživatelské rozhraní pomocí nástrojů Visual Studia, ale je to pohodlnější a rychlejší.

4.2 Implementace ukázkové aplikace

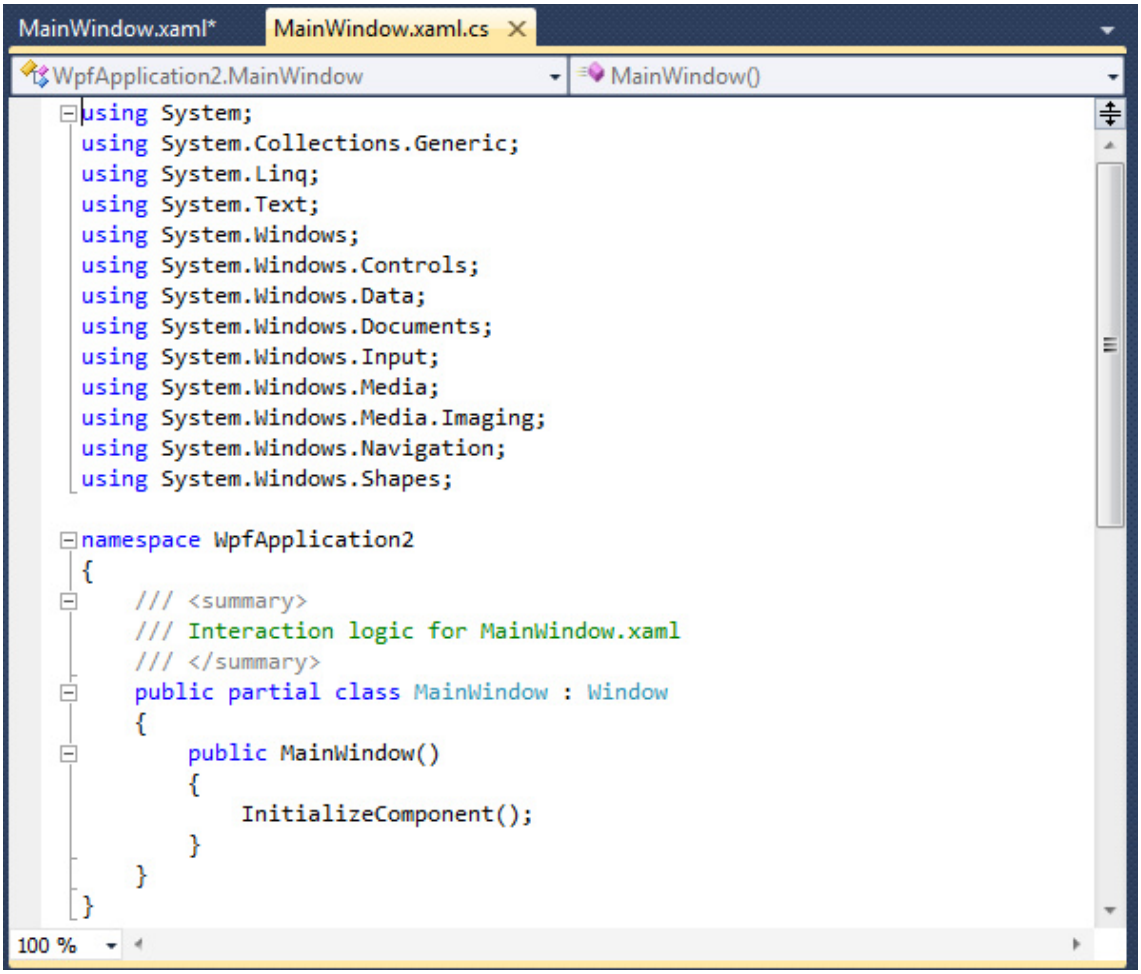
4.2.1 Vytvoření GUI



Obrázek 4-1 Založení nové WPF aplikace, zobrazení souboru MainWindow.xaml

Při tvorbě nové aplikace ve Visual Studio je dobré začít vytvořením nového projektu. Podle výběru druhu aplikace, v tomto případě WPF, Visual Studio vytvoří základní soubory potřebné pro běh aplikace. Z hlediska tvorby aplikace jsou důležité hlavně `MainWindow.xaml`, kde je popsán vzhled aplikace, a `MainWindow.xaml.cs` obsahující kód jazyka C# zajišťující funkčnost aplikace.

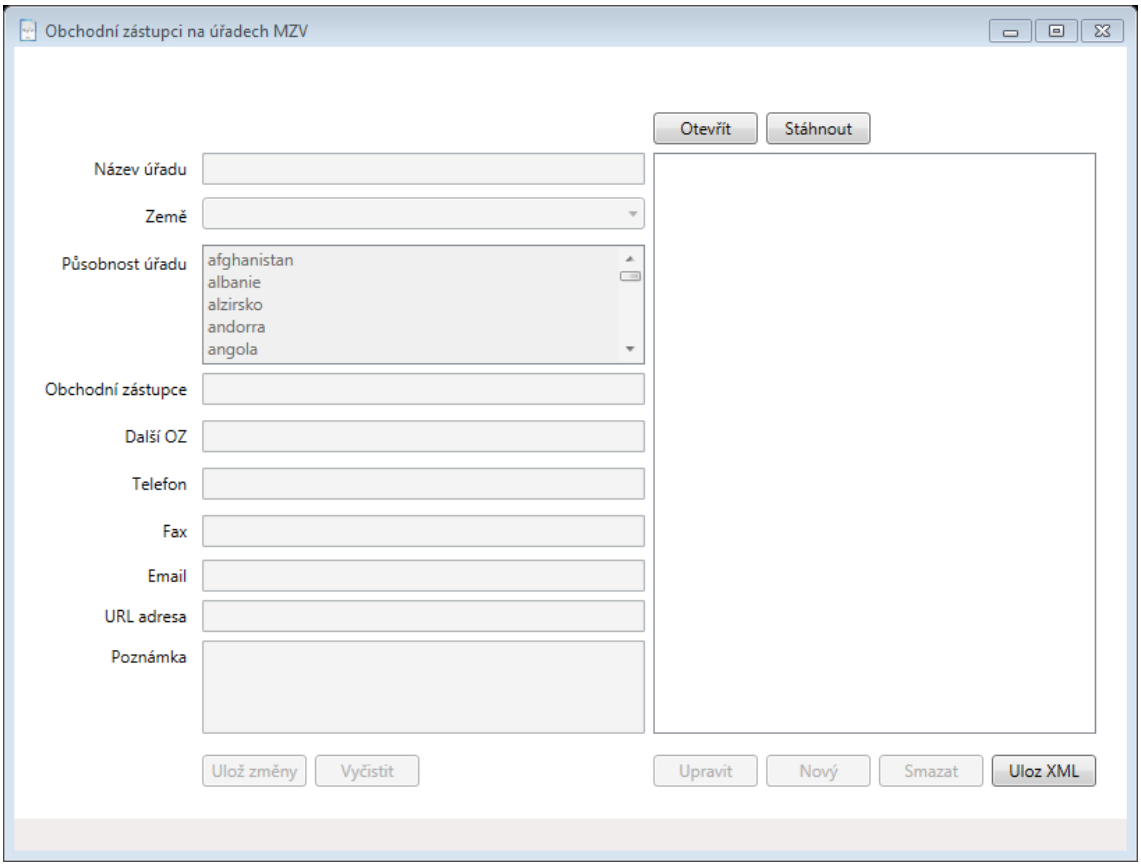
V souboru `MainWindow.xaml` se automaticky vytvoří prázdný formulář s názvem `MainWindow`, jak je vidět na obrázku 4-1. Visual Studio nabízí na tento soubor dva pohledy Design a XAML. Z jejich názvů vyplývá, že v pohledu Design je vidět vzhled formuláře a rozložení prvků a v pohledu XAML jsou formulář a jeho součásti popsány stejnojmenným jazykem.



Obrázek 4-2 Soubor MainWindow.xaml.cz

V souboru `MainWindow.xaml.cs` (obrázek 4-2) je ve výchozím stavu připojeno několik základních knihoven (jmenných prostorů) pomocí direktivy `using`. Následuje základní jmenný prostor (`namespace`), nazvaný podle názvu řešení (solution name), který obsahuje třídu s názvem `MainWindow` a její metodou `MainWindow()` inicializující komponenty pomocí metody `InitializeComponent()`.

Uživatelské prostředí je co možná nejjednodušší a přehledné. Na obrázku 4-3 je znázorněna výsledná podoba aplikace.



Obrázek 4-3 Uživatelské prostředí aplikace

V pravé části je pole pro seznam úřadů a k němu příslušející ovládací prvky – tlačítka Otevřít, Stáhnout, Upravit, Nový, Smazat a Ulož XML. V levé části jsou formulářová pole pro jednotlivé informace k vybranému úřadu ze seznamu s tlačítky Ulož změny a Vyčistit.

4.2.2 Načtení XML dokumentu

K načtení XML dokumentu do XML DOM je ve jmenném prostoru `System.Xml` připravena třída `XmlDocument`. Třída `XmlDocument` představuje vstupní bod binární struktury a centrální bod, z něhož je možné procházet uzly, načítat je a měnit obsah. Každý element v původním dokumentu XML je mapován na určitou třídu .NET Framework, která zveřejňuje své vlastnosti a metody. Ke každému elementu se lze dostat z jeho rodiče a každý element umí přistupovat ke všem svým dětem a sourozencům. Údaje specifické pro každý element (obsah, atributy) jsou dostupné formou vlastností.

Každá provedená změna je okamžitě aplikována, ale jen v paměti počítače. Třída `XmlDocument` obsahuje vstupní a výstupní rozhraní pro načítání a ukládání a zvládá přitom práci s nejrůznějšími typy úložných médií, včetně diskových souborů. Dále platí, že všechny změny základních elementů stromu XML DOM jsou obvykle trvale ukládány najednou. [2]

Dokument XML musí být celý nahrán do operační paměti a jeho uzly a atributy musí být mapovány na objekty odvozené ze třídy `XmlNode` z jmenného prostoru `System.Xml`, jinak není plný přístup k dokumentu možný. Proces tvorby stromu XML DOM je spuštěn zavoláním metody `Load` (viz příklad 4-3).

Příklad 4-3 Nažtení XML dokumentu do proměnné uradyXML

```
private XmlDocument uradyXML = new XmlDocument();
uradyXML.Load("http://www.mzv.cz/file/909103/obchodniZastupci.xml"
);
```

4.2.3 Validace dokumentu

Po načtení XML dokumentu, je možné ho validovat. Dříve se k validaci XML dokumentů používala třída `XmlValidatingReader`, která je ale v současné době zastaralá. Místo ní je podle [10] doporučována metoda `Create (XmlReader, XmlReaderSettings)` třídy `XmlReader` z jmenného prostoru `System.Xml`.

Jak je uvedeno v příkladu 4-4, XML dokument načtený do instance třídy `XmlDocument` je nutné převést na instanci třídy `XmlReaderu`, např. `XmlNodeReader`. Dále je potřeba pomocí `XmlReaderSettings` nastavit typ validace, cestu k validačnímu souboru (XSD) a ukazatel události (`ValidationEventHandler`), který odchytí a popíše případné chyby v XML dokumentu.

Pomocí metody `XmlReader.Create(XmlReader, XmlReaderSettings)` je vytvořena nová instance `XmlReaderu`, který prochází XML dokument a porovnává ho s XSD souborem. `ValidationEventHandler` pak při každé chybě v XML dokumentu volá metodu `lValReader_ValidationEventHandler`, která zobrazí chybu jako zprávu v `MessageBoxu`.

Příklad 4-4 Metoda ValidaceDokumentu(XmlDocument), která ověřuje správnost dokumentu podle XML Schema

```
public static void ValidaceDokumentu(XmlDocument XmlDok)
        {
            XmlNodeReader nodeReader = new XmlNodeReader(XmlDok);

            // validační nastavení
            XmlReaderSettings settings = new XmlReaderSettings();
            settings.ValidationType = ValidationType.Schema;
            settings.Schemas.Add("",
"http://www.mzv.cz/file/964866/obchodniZastupci.xsd");
            settings.ValidationEventHandler += new
ValidationEventHandler(lValReader_ValidationEventHandler);

            // vytvoření validačního readeru, který projde
XmlNodeReader
            XmlReader reader = XmlReader.Create(nodeReader,
settings);
            // procházení (kontrola) Xml dokumentu
            while (reader.Read()) ;
        }

        private static void
lValReader_ValidationEventHandler(object sender,
ValidationEventArgs e)
        {
            //vypsani zavaznosti a zprávy
            MessageBox.Show("Závašnost: " + e.Severity +
                            "<br />Popis: " + e.Message);
        }
```

4.2.4 Zobrazení dat ve formuláři

Když jsou data ve správném formátu načtena v XML DOM, je potřeba je zobrazit ve formuláři, aby je mohl uživatel snadno editovat. Z XML DOM (`XmlDocumentu`) jsou vyselektovány všechny uzly s názvem `item`, atribut `id` a element `nazev` a pomocí příkazu LINQ seřazeny abecedně podle názvu úřadu a transformovány do nového `XDocumentu` s jednodušší strukturou (viz příklad 4-5).

Příklad 4-5 Příklad použití dotazu LINQ pro seřazení uzlů podle abecedy

```
// LINQ dotaz pro seřazení úřadů podle abecedy
var query = from Xurad in razenySeznam.Descendants("item")
       let nazev = (string)Xurad.Descendants("urad").First()
       orderby nazev
       select new XElement("item",
           new XAttribute("id", Xurad.Attribute("id").Value),
           new XElement("nazev", nazev));
XElement result = new XElement("items", query);
```

Po seřazení jsou data opět převedena na `XmlDocument`. Z tohoto zdroje jsou vybrány uzly `item` pomocí metody `SelectNodes(XPath)` třídy `XmlDocument` a z nich je vytvořena nová instance třídy `XmlNodeList`.

Do formulářového prvku `ListBox` se data načtou pomocí iterace konstrukce `foreach` přes vytvořený `XmlNodeList` a přiřazením vybraných dat do kolekce `ListBoxu` metodou `XmlNodeList.Items.Add(Object)`. Jak vyplývá z příkladu 4-5, do seznamu se zařazují 2 údaje – jednoznačný identifikátor a název úřadu. Identifikátor slouží k jednoznačné identifikaci uzlu, aby při úpravách nedošlo ke změnám u více uzlů a název úřadu pro identifikaci uzlu uživatelem.

4.2.5 Úprava dat

Po vybrání jakékoli položky v seznamu úřadů se aktivují tlačítka `Upravit` a `Smazat`. Kliknutím na tlačítko `Smazat` se vybraný úřad odstraní z XML DOM a položku již není možno obnovit, kromě opětovného načtení všech dat z webu nebo uloženého souboru na lokálním úložišti.

Tlačítkem `Upravit` se zobrazí všechny údaje o vybraném úřadu v levé části formuláře. Výběr uzlů z XML DOM se provádí pomocí metod `SelectNode(XPath)` a `SelectSingleNode(XPath)` třídy `XmlElement`. Zobrazení hodnoty příslušného uzlu zajišťuje přiřazení vlastnosti `InnerText` u elementu nebo Value u atributu do vlastnosti `Text` (nebo jiné) příslušného formulářového prvku.

Příklad 4-6 znázorňuje výběr uzlu `item` metodou `SelectSingleNode(XPath)` podle předané hodnoty `id` a jeho přiřazení do instance třídy `XmlNode` s názvem `item`. Dále zobrazuje přiřazení názvu úřadu formulářovému prvku `TextBox` pojmenovaného `nazev`, přiřazení země, kde úřad sídlí, formulářovému prvku `ComboBox` s názvem

zemeUradu. Pomocí iterace přes kolekci XmlNodeList s názvem zemePusobnosti a přiřazením vybraných hodnot do kolekce SelectedItems metodou Add jsou označeny státy, v kterých vybraný úřad působí.

Příklad 4-6 Zobrazení hodnot z XML DOM ve formulářových prvcích

```
XmlNode item = rootUrady.SelectSingleNode("item[@id=" + id + "]");

nazev.Text = item.SelectSingleNode("urad/nazev").InnerText;
zemeUradu.SelectedItem = item.Attributes["zeme"].Value;

XmlNodeList zemePusobnosti = item.SelectNodes("pusobnost/zeme");
if (zemePusobnosti.Count > 0)
{
    seznamZemi.SelectedItems.Clear();
    foreach (XmlNode zemePus in zemePusobnosti)
    {
        seznamZemi.SelectedItems.Add(zemePus.InnerText);
    }
}
```

Uživatel může ve formuláři upravit zobrazené informace, doplnit zatím prázdné údaje nebo nějaké smazat. V průběhu editace se žádné změny neukládají a vše je nutné potvrdit tlačítkem Ulož změny.

Úprava dat se provádí velice jednoduše. Je to pouze opačný postup zobrazení dat ve formuláři. Tedy textová nebo jiná hodnota formulářového prvku se přiřadí uzlu v XML DOM. Ukázka v příkladu 4-7 zobrazuje ověření, zda se údaje liší a v případě, že ano, změnu hodnoty.

Příklad 4-7 Přiřazení hodnoty z formulářového prvku do uzlu v XML DOM

```
if (uUrad.SelectSingleNode("telefon").InnerText != telefon.Text)
{
  uUrad.SelectSingleNode("telefon").InnerText = telefon.Text;
}
```

4.2.6 Vložení nových dat

Pro vložení nových dat do XML DOM je potřeba nejprve vytvořit nový uzel a ten připojit do stromu XML dokumentu. Pro vytvoření nového elementu se používá třída

`XmlNode`, pro vytvoření nového atributu třída `XmlAttribute` z jmenného prostoru `System.Xml`.

Nový element se tvoří metodou `CreateElement(String)` třídy `XmlDocument`. Tato metoda vytvoří nový prázdný prvek pojmenovaný podle zadaného řetězce. Hodnota se do uzlu přidává pomocí metody `InnerText` třídy `XmlNode`. Do nadřazeného uzlu se nový element přidává metodou `AppendChild(XmlNode)` třídy `XmlNode`.

Velice podobně se vytváří i nový atribut. Pomocí metody `CreateAttribute(String)` třídy `XmlDocument` se vytvoří nový prázdný atribut pojmenovaný podle zadaného řetězce. Hodnota se mu však přidává tím, že se změní hodnota vlastnosti `Value`. Atributy jsou u třídy `XmlNode` vedeny jako vlastnost s názvem `Attributes` a ve formě kolekce (`XmlAttributeCollection`). Nový atribut se tedy přidává do této kolekce pomocí metody `SetNamedItem(XmlAttribute)`.

Všechny nově připojované uzly k nadřízenému uzlu se vždy řadí na konec seznamu podřízených uzlů.

V aplikaci vytvářené v rámci této práce se do XML dokumentu mohou přidávat pouze nové úřady, tedy uzel `item` a všechny jeho podřízené elementy a atributy (viz příklad 4-1). V ukázce kódu v příkladu 4-8 je zobrazeno, jak se přebírají hodnoty z formuláře a ukládají do XML DOM.

Příklad 4-8 Vytvoření nových uzlů a jejich připojení k nadřízenému uzlu

```
XmlNode nItem = uradyXML.CreateElement("item");
XmlNode nPusobnost = uradyXML.CreateElement("pusobnost");

foreach (string selZeme in seznamZemi.SelectedItems)
{
  XmlNode n = uradyXML.CreateElement("zeme");
  XmlAttribute a = uradyXML.CreateAttribute("iso");
  n.InnerText = selZeme;
  a.Value = rootRejstrik.SelectSingleNode("asset[text()='" +
selZeme + "']").Attributes["isocode"].Value;
  n.Attributes.SetNamedItem(a);
  nPusobnost.AppendChild(n);
}
nItem.AppendChild(nPusobnost);
// připojování dalších uzlů (ekonom, vedouci)
XmlNode nn;
nn = uradyXML.CreateElement("telefon");
nn.InnerText = telefon.Text;
nItem.AppendChild(nn);
// připojování dalších uzlů (fax, email, atd.)
rootUrady.AppendChild(nItem);
```

Nejprve je vytvořen nový uzel s názvem `item` metodou `CreateElement(String)` a přiřazen do nové instance třídy `XmlNode` s názvem `nItem`. Postupně se do elementu `nItem` připojí všechny nově vytvořené uzly metodou AppendChild(XmlNode) v pořadí, které odpovídá XML schématu, a nakonec se element `nItem` připojí do XML dokumentu.

4.2.7 Uložení XML dokumentu

K uložení XML dokumentu z XML DOM do souboru je určena metoda `Save(String)` třídy `XmlDocument`, kde řetězec `String` značí název souboru. V aplikaci je k uložení využito dialogové okno `SaveFileDialog` z jmenného prostoru `Microsoft.Win32`.

Využití tohoto dialogového okna je užitečné pro možnosti jeho nastavení. Lze nastavit výchozí adresář, kam se soubor bude ukládat, výchozí název souboru, kontrolu přepsání stejnojmenného souboru a další (viz příklad 4-9).

Příklad 4-9 Uložení XML dokumentu do souboru pomocí dialogového okna

```
SaveFileDialog dlgUlozit = new SaveFileDialog();
dlgUlozit.DefaultExt = "xml";
dlgUlozit.AddExtension = true;
dlgUlozit.FileName = "obchodniZastupci";
dlgUlozit.InitialDirectory = @"C:\Users\"+
System.Security.Principal.WindowsIdentity.GetCurrent().Name+"\\Docu
ments\\";
dlgUlozit.OverwritePrompt = true;
dlgUlozit.Title = "Obchodní zástupci MZV";
dlgUlozit.ValidateNames = true;

if (dlgUlozit.ShowDialog().Value)
{
    uradyXML.Save(dlgUlozit.FileName);
}
```

5 Závěr

Cílem této bakalářské práce bylo analyzovat možnosti zpracování XML dokumentu v prostředí .NET Frameworku. V úvodní části práce jsem popsal, co je to XML, jak tento jazyk vznikl a přiblížil jeho syntaxi a použití. V následující části jsem stručně popsal prostředí .NET Framework a jeho vývoj a přiblížil možnosti zpracování XML dokumentů pomocí nástrojů XML reader, XML writer, XML DOM a LINQ to XML.

V praktické části bakalářské práce jsem nejprve analyzoval požadavky na aplikaci, popsal její funkčnost a použité technologie. V poslední části práce jsem konkrétně popsal, jak vytvořit uživatelské prostředí pomocí aplikace Visual Studio 2010 a technologie XAML a jak naprogramovat funkčnost aplikace pomocí jazyka C# s využitím knihoven .NET Framework.

Vysledná aplikace a zdrojové soubory s daty potřebnými pro běh aplikace je uložena na přiloženém CD.

6 Seznam použitých zdrojů

6.1 Tištěné zdroje

1. HAROLD, Elliotte Rusty, MEANS, W. Scott. *XML v kostce – pohotová referenční příručka.* Vydání první. Praha: Computer Press, 2002. 439 s. ISBN 80-7226-712-4.
2. ESPOSITO, Dino. *XML – efektivní programování pro .NET.* Vydání první. Grada Publishing, 2004. 596 s. ISBN 80-247-0775-6.
3. MLÝNKOVÁ, Irena, NEČASKÝ, Martin, POKORNÝ, Jaroslav, RICHTA, Karel, TOMAN, Kamil, TOMAN, Vojtěch. *XML technologie – Principy a aplikace v praxi.* Vydání první. Praha: Grada Publishing, 2008. 282 s. ISBN 978-80-247-2725-7.
4. SHARP, John. *Microsoft Visual C# 2010 – Krok za krokem.* Vydání první. Brno: Computer Press, 2010. 696 s. ISBN 978-80-251-3147-3.
5. TREY, Nash. *C# 2010 – Rychlý průvodce novinkami a nejlepšími postupy.* Vydání první. Brno: Computer Press, 2010. 624 s. ISBN 978-80-251-3034-6.

6.2 Internetové zdroje

6. VISUAL STUDIO. *.NET Framework 4.* [online]. 13. 10. 2012 [cit. 2013-02-13]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2%28v=vs.100%29.aspx>.
7. MSDN. *.NET Framework 4.* [online]. [cit. 2013-02-13]. Dostupné z WWW: <http://msdn.microsoft.com/cs-cz/library/w0x726c2%28v=vs.100%29.aspx>.
8. CAMBIUCCI, Waldemir. *.NET Framework 4.0 architektura aplikace.* [online]. 4. 7. 2011 [cit. 2013-02-13]. Dostupné z WWW: <http://blogs.msdn.com/b/wcamb/archive/2011/07/04/net-framework-4-0-e-arquiteturas-de-aplica-231-245-es.aspx>.
9. VISUAL STUDIO. XML Document Object Model (DOM). [online]. 2. 8. 2012 [cit. 2013-02-25]. Dostupné z WWW: < http://msdn.microsoft.com/en-us/library/vstudio/hf9hbf87%28v=vs.100%29.aspx>.

10. MSDN. *XmlReader.Create Method (XmlReader, XmlReaderSettings)*. [online]. [cit. 2013-02-28]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/x1h1125x(v=vs.110).aspx>.

7 Přílohy

7.1 XML schéma

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="item">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="pusobnost">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="zeme">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="iso" type="xs:string"
use="required" />
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="ekonom">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="vedouci" type="xs:string" />
                    <xs:element minOccurs="0" maxOccurs="unbounded"
name="jmeno" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="telefon" type="xs:string" />
              <xs:element name="fax" type="xs:string" />
              <xs:element name="email" type="xs:string" />
              <xs:element name="www" type="xs:string" />
              <xs:element name="urad">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="nazev" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
```

```
              <xs:element name="poznamka" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="id" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="zeme" type="xs:string" use="required"
/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

7.2 XML dokument (ukázka)

```xml
<?xml version="1.0" encoding="utf-8"?>
<data>
  <item id="1" zeme="mexiko">
    <pusobnost>
      <zeme iso="MX">mexiko</zeme>
      <zeme iso="BZ">belize</zeme>
      <zeme iso="GT">guatemala</zeme>
      <zeme iso="HN">honduras</zeme>
      <zeme iso="CR">kostarika</zeme>
      <zeme iso="NI">nikaragua</zeme>
      <zeme iso="PA">panama</zeme>
      <zeme iso="SV">salvador</zeme>
    </pusobnost>
    <ekonom>
      <vedouci>Ing. Milena Petříčková</vedouci>
      <jmeno>Ing. Josef Kostiha</jmeno>
    </ekonom>
    <telefon>+52-55-5531-2544</telefon>
    <fax>+52-55-5531-1837</fax>
    <email>commerce_mexiko@mzv.cz</email>
    <www>http://www.mzv.cz/mexico</www>
    <urad>
      <nazev>Velvyslanectví ČR v Mexiku</nazev>
    </urad>
    <poznamka>
    </poznamka>
  </item>
  <item id="2" zeme="albanie">
    <pusobnost>
      <zeme iso="AL">albanie</zeme>
    </pusobnost>
    <ekonom>
      <vedouci>Ing. Vítězslav Schwarz, Ph.D</vedouci>
    </ekonom>
    <telefon>+355-42-23-4004</telefon>
    <fax>+355-42-23-2159</fax>
    <email>Commerce_Tirana@mzv.cz</email>
    <www>http://www.mzv.cz/tirana</www>
    <urad>
      <nazev>Velvyslanectví ČR v Tiraně</nazev>
    </urad>
    <poznamka>
    </poznamka>
  </item>
  <item id="3" zeme="afghanistan">
    <pusobnost>
      <zeme iso="AF">afghanistan</zeme>
    </pusobnost>
    <ekonom>
```

```
      <vedouci>Mgr. Štěpánka Litecká </vedouci>
    </ekonom>
    <telefon>+93-79-841-7418</telefon>
    <fax>
    </fax>
    <email>kabul@embassy.mzv.cz</email>
    <www>http://www.mzv.cz/kabul</www>
    <urad>
      <nazev>Velvyslanectví ČR v Kábulu</nazev>
    </urad>
    <poznamka>
    </poznamka>
  </item>
  <item id="4" zeme="argentina">
    <pusobnost>
      <zeme iso="PY">paraguay</zeme>
      <zeme iso="UY">uruguay</zeme>
      <zeme iso="AR">argentina</zeme>
    </pusobnost>
    <ekonom>
      <vedouci>Ing. Tibor Chochula </vedouci>
    </ekonom>
    <telefon>+54-11-4807-3107</telefon>
    <fax>+54-11-4800-1088</fax>
    <email>Commerce_BuenosAires@mzv.cz</email>
    <www>http://www.mzv.cz/buenosaires</www>
    <urad>
      <nazev>Velvyslanectví ČR v Buenos Aires</nazev>
    </urad>
    <poznamka>
    </poznamka>
  </item>
  <item id="5" zeme="armenie">
    <pusobnost>
      <zeme iso="AM">armenie</zeme>
    </pusobnost>
    <ekonom>
      <vedouci>PhDr. Petr Mikyska</vedouci>
    </ekonom>
    <telefon>+374/96/695395</telefon>
    <fax>
    </fax>
    <email>yerevan@embassy.mzv.cz</email>
    <www>http://www.mzv.cz/yerevan</www>
    <urad>
      <nazev>Velvyslanectví ČR v Jerevanu</nazev>
    </urad>
    <poznamka>
    </poznamka>
  </item>
</data>
```

7.3 XAML dokument

```
<Window x:Class="ODEV.ObchodniZastupci"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        Height="615" Width="815" Name="obchodniZastupci"
Closing="obchodniZastupci_Closing"
        Title="Obchodní zástupci na úřadech MZV" mc:Ignorable="d">
    <Grid>
        <Label Content="Název úřadu" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,74,0,0" Name="label1"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Země" Height="28" HorizontalAlignment="Left"
HorizontalContentAlignment="Right" Margin="10,108,0,0" Name="label2"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Působnost úřadu" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,142,0,0" Name="label3"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Obchodní zástupce" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,232,0,0" Name="label4"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Další OZ" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,266,0,0" Name="label5"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Telefon" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,300,0,0" Name="label6"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Fax" Height="28" HorizontalAlignment="Left"
HorizontalContentAlignment="Right" Margin="10,334,0,0" Name="label7"
               VerticalAlignment="Top" Width="119" />
        <Label Content="URL adresa" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,395,0,0" Name="label8"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Poznámka" Height="28"
HorizontalAlignment="Left" HorizontalContentAlignment="Right"
Margin="10,424,0,0" Name="label9"
               VerticalAlignment="Top" Width="119" />
        <Label Content="Email" Height="28" HorizontalAlignment="Left"
Margin="12,366,0,0" Name="label10" VerticalAlignment="Top" Width="117"
               HorizontalContentAlignment="Right" />
```

```
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,76,0,0" Name="nazev" VerticalAlignment="Top" Width="318"
IsEnabled="False"
                 TextChanged="nazev_TextChanged" />
        <ComboBox Height="23" HorizontalAlignment="Left"
Margin="135,108,0,0" Name="zemeUradu" VerticalAlignment="Top"
Width="318" IsEnabled="False"
                  SelectionChanged="zemeUradu_SelectionChanged"
IsReadOnly="False" />
        <ListBox Height="86" HorizontalAlignment="Left"
Margin="135,142,0,0" Name="seznamZemi" VerticalAlignment="Top"
Width="318" SelectionMode="Multiple"
                 IsEnabled="False"
SelectionChanged="seznamZemi_SelectionChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,234,0,0" Name="ekonomVedouci" VerticalAlignment="Top"
Width="318" IsEnabled="False"
                 TextChanged="ekonomVedouci_TextChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,268,0,0" Name="jmenaOZ" VerticalAlignment="Top"
Width="318" IsEnabled="False"
                 TextChanged="jmenaOZ_TextChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,302,0,0" Name="telefon" VerticalAlignment="Top"
Width="318" IsEnabled="False"
                 TextChanged="telefon_TextChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,336,0,0" Name="fax" VerticalAlignment="Top" Width="318"
IsEnabled="False"
                 TextChanged="fax_TextChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,368,0,0" Name="email" VerticalAlignment="Top" Width="318"
IsEnabled="False"
                 TextChanged="email_TextChanged" />
        <TextBox Height="23" HorizontalAlignment="Left"
Margin="135,397,0,0" Name="www" VerticalAlignment="Top" Width="318"
IsEnabled="False"
                 TextChanged="www_TextChanged" />
        <TextBox Height="67" HorizontalAlignment="Left"
Margin="135,426,0,0" Name="poznamka" VerticalAlignment="Top"
Width="318" IsEnabled="False"
                 TextChanged="poznamka_TextChanged" />
        <ListBox Height="417" HorizontalAlignment="Left"
Margin="459,76,0,0" Name="seznamUradu" VerticalAlignment="Top"
Width="318"
                 SelectionChanged="seznamUradu_SelectionChanged" />
        <Button Content="Upravit" Height="23"
HorizontalAlignment="Left" Margin="459,508,0,0" Name="upravit"
VerticalAlignment="Top"
                Width="75" Click="upravit_Click" IsEnabled="False" />
```

```
        <Button Content="Smazat" Height="23"
HorizontalAlignment="Left" Margin="621,508,0,0" Name="smazat"
VerticalAlignment="Top" Width="75" IsEnabled="False"
                Click="smazat_Click" />
        <Button Content="Nový" Height="23" HorizontalAlignment="Left"
Margin="540,508,0,0" Name="novy" VerticalAlignment="Top" Width="75"
Click="novy_Click"
                IsEnabled="False" />
        <Button Content="Ulož změny" Height="23"
HorizontalAlignment="Left" Margin="135,508,0,0" Name="ulozZmeny"
VerticalAlignment="Top" Width="75" IsEnabled="False"
                Click="ulozZmeny_Click" />
        <Button Content="Uloz XML" Height="23"
HorizontalAlignment="Left" Margin="702,508,0,0" Name="ulozXML"
VerticalAlignment="Top" Width="75"
                IsEnabled="False" Click="ulozXML_Click" />
        <StatusBar Height="23" HorizontalAlignment="Stretch"
Margin="0" Name="statusBar1" VerticalAlignment="Bottom" Width="Auto"
                    />
        <Button Content="Vyčistit" Height="23"
HorizontalAlignment="Left" Margin="216,508,0,0" Name="vycistit"
VerticalAlignment="Top" Width="75"
                IsEnabled="False" Click="vycistit_Click" />
        <Button Content="Otevřít" Height="23"
HorizontalAlignment="Left" Margin="459,47,0,0" Name="otevrit"
VerticalAlignment="Top" Width="75"
                Click="otevrit_Click" />
        <Button Content="Stáhnout" Height="23"
HorizontalAlignment="Left" Margin="540,47,0,0" Name="stahnout"
VerticalAlignment="Top" Width="75"
                Click="stahnout_Click" />
    </Grid>
</Window>
```