České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Rozhraní nástroje pro tvorbu rozvrhu

Praha, 2009 **Autor: Ondřej Kunc**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Děkuji své rodině za podporu vynakládanou pa celou dobu mého studia.
Především však, děkuji vedoucímu práce panu Ing. Liboru Waszniowskimu, Ph.D. za odborné vedení diplomové práce, vstřícnost při zodpovídání dotazů.

Abstrakt

Úkolem práce je na pilotním projektu ověřit použitelnost technologie Silverlight pro tvorbu grafického uživatelského rozhraní(GUI) optimalizačniho nástroje. Jako pilotní apliakace bylo zvoleno GUI nástroje pro tvorbu rozvrhu fakulty, které v sobě kombinuje požadavky na komfortni ovladani blízke desktopové aplikaci a zároveň potřebu sdílení dat, kterou naopak nabizí webová aplikace.

Úvodní kapitoly popisují základní pojmy .NETu využívané v Silverlight technologii, stejně jako s instalaci a používání Silverlightu. Dále je popisován vyvinutý Silverlight program, určený pro tvorbu rozvrhu školy. V závěru jsou zhodnoceny osobní zkušenosti a postřehy nabyté při tvorbě programu.

Abstract

The aim of this thesis is to get familiar with new Microsoft technology called Silverlight and create an optimalization tool with it. As a starting application was selected GUI for declaring school's timetable, which combines requires for user friendly interface similar to desktop application and at the same time need of sharing data, which offers web application.

Opening chapters are describing basic .NET conceptions being used in Silverlight technology as well as installation and using Silverlight. Further is described program developed in Silverlight. Personal experience found out during thesis construction are estimated at the end.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Ondřej Kunc**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Rozhraní nástroje pro tvorbu rozvrhu**

Pokyny pro vypracování:

1. Seznamte se s technologií .Net a SilverLight a principy tvorby internetových rozhraní.
2. Navrhněte nástroj pro tvorbu rozvrhu.
3. Implementujte internetové rozhraní tohoto nástroje pomocí zmíněných technologií. Rozhraní musí umožňovat intuitivní interakci uživatele s nástrojem.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Libor Waszniowski, Ph.D.

Platnost zadání: do konce zimního semestru 2009/2010

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 25. 9. 2008

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Web je nejpopulárnější prostředí pro sdílený software. Jsou však věci, které webové aplikace dělat nedokáží, nebo jenom omezeně. Ani přes snahu vybavit své, například ASP.NET webové stránky nejnovějšími věcmi JavaScriptu, nebudeme schopni dosáhnout funkcionality desktopových aplikací. I přesto, že JavaScript poskytuje možnost odpovídat na některé klientské události, budování složitých rozhraní, aspoň tak uživatelsky vstřícných jako stadnartní aplikace desktopové, je nemožné. Hranice webu jsou neustále pokořovány, a tak už nyní není problém si na stránkách zahrát, ne zrovna triviální, plně grafickou hru. Řeknete si to asi není obyčejné HTML, CSS a JavaScript. Říká se jim RIA, internetové aplikace fungující za pomoci plug-inu v prohlížeči.

Tato práce si dává za cíl seznámení s novou technologií Microsoftu na tvorbu RIA aplikací (kap.2), nazývanou Silverlight(kap.3). Silverlight umožňuje vytvářet aplikace přístupné na internetu s funkčností srovnatelnou s aplikacemi desktopovými. Jelikož je Silverlight poměrně nová technologie bude nejdříve třeba objevit její možnosti a limity, stejně jako poznat jak se s ní pracuje. Cílem práce je vytvořit uživatelsky příjemnou aplikaci(kap.5.3) na tvorbu rozvrhu školy, běžící na Silverlightu.

Kapitola 2

RIA

Koncept bohatých internetových aplikací existoval mnoho let pod různými jmény(**Remote Spring**-Microsoft 1998, **X Internet**-Forrester Research 2000), ale až v roce 2002 byl uveden pojem RIA. Ten zavedla společnost Macromedia(nyní Adobe).

Pojem RIA (Rich Internet Application) se nedá jednoznačně specifikovat. Definice *Rich Internet Application* obsahuje tři slova *Internet*, *Application* jejich význam je jednoznačný. Se slůvkem *Rich* už to tak jednoznačné není. Právě ten dělá z RIA nejednoznačný pojem. Obecně jsou RIAs webové aplikace, které se snaží překlenout rozdíly mezi klasickou webovou aplikací a aplikací desktopovou . RIA aplikace se snaží v rámci webového prohlížece napodobovat desktopové aplikace svým vzhledem i chováním a poskytnout tak vyšší uživatelský komfort. Například všem známé GMail nebo OutlookWebAcces jsou RIA. Samozřejmě že ne jenom tyto dvě, jmenovat bychom mohli tisíce dalších webových, v dnešní době hlavně AJAXových(kap.2.1.1.1), aplikací. Pak by bylo možné popsat slůvko *Rich* jako tu funkcionalitu, kterou nám nabízí tyto technologie navíc oproti těm nynějším. Pojem RIA se stává do jisté míry marketingovým tahákem, proto se hodně výrobců snaží dát tuto nálepku produktům, které by dříve za internetové považoval jen málokdo. K těm patří podle Adobe RIA aplikace, dle mého názoru desktopová Adobe Media Player nebo eBay Desktop atd... Tetno pohled na věc je nový ale zcela zřejmý, důraz je kladen na vnímaní uživatele, ale už ne tak na technologii, tedy jestli běží na internetu nebo na našem lokálním počítači. To je možné díky tomu, že není třeba žádné instalace, aplikace se při zobrazení stránky(s RIA aplikací) načte v nejnovější verzi. Při používání RIA je také menší riziko infikace počítače virem.

Tradiční internetové aplikace fungující jako client-server. Všechny se musí vypořádávat s HTTP protokolem a jeho strukturou dotaz-odpověď. Pak i při sebenší změně na straně

klienta je vyvolána žádost, odeslána na server. Po obsloužení žádosti serverem je poslána zpět odpověď. Ta ale obsahuje celou předešlou stránku plus tu jednu malou změnu, kterou uživatel provedl. Tato nevýhoda je jedna z důvodů vzniku RIA.

2.1 Prostředky RIA

Jak jsme zmínili v předchozí kapitole RIA aplikace jsou v dnešní době hlavně AJAXové. Avšak neplatí, že RIA je AJAX. Na trhu je mnoho dalších technologií pro tvorbu bohatých internetových aplikací, které by se daly rozdělit do dvou skupin:

- bez plug-inu do prohlížeče
- s plug-inem do prohlížeče.

Plug-in do prohlížeče znamená instalaci zasuvného modulu do vašeho prohlížeče(Software), ten pak pracuje jako doplňkový modul jiné aplikace a rozšiřuje tak její funkčnost.

2.1.1 Technologie bez potřeby plug-inu

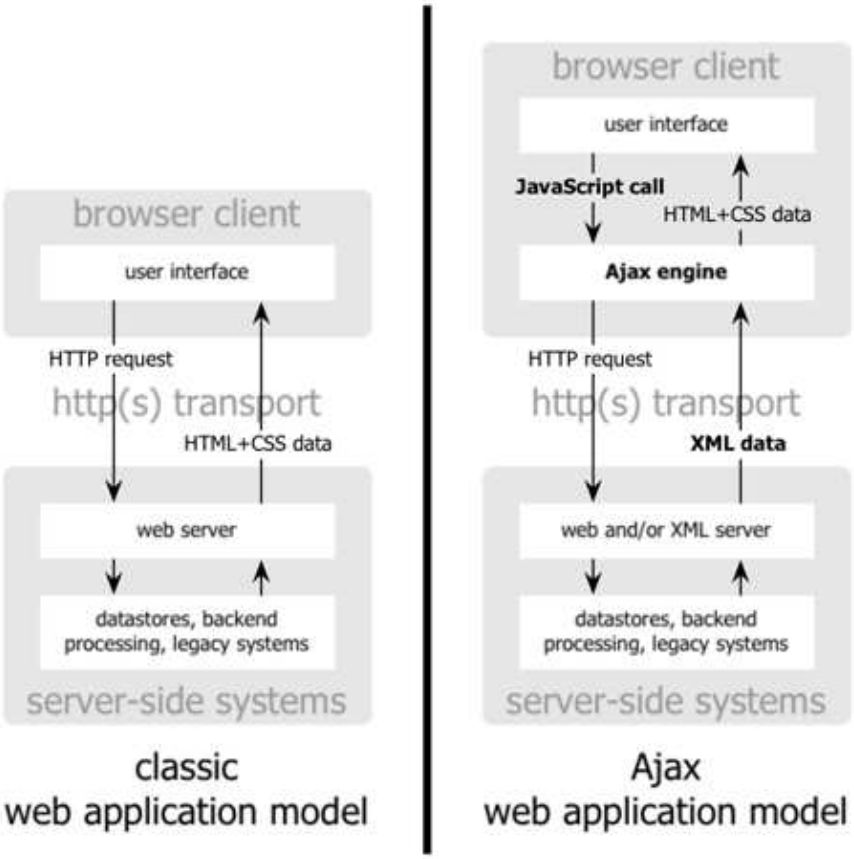
2.1.1.1 AJAX(Asynchronous JavaScript and XML)

Ajax není samostatnou technologií, nýbrž obecným konceptem. AJAX předtavuje cestu, která maximaílně využívá dnešní technologie. Zastřešuje následující z nich:

- HTML, CSS a JavaScript (JScript)
- XMLHttpRequest (Mozilla, MSIE)
- Document Object Model (DOM).

To převratné na AJAXu oproti "normálnímu"webovému konceptu je způsob využití XMLHttpRequest k volání serveru. Oproti klasickému webovému modelu, kde každá změna stavu u klienta vyžaduje obnovení celé stránky, je to u Ajaxu jinak.
AJAX, jak již z názvu vyplývá, pracuje asynchronně. To znamená, že u webu vytvořeného pomocí AJAXu, se neobnovoje celá stránka, ale jen ta část, která je k tomu určena. Požadavek na změnu stavu se vyřizuje přes XMLHttpRequest (asynchronní volání serveru), server obslouží požadavek a pošle odpověď zpět ta je opět zpracována v XMLHttpRequest a nyní nastane změna patřičné části. Klient a server si předávají informace v XML,

AJAX u klienta je reprezentován JavaScriptem. Popsaný rozdíl oproti klasickému konceptu dotaz-odpověď je vidět na obr.2.1 , kde Ajax engine je XMLHttpRequest.



Obrázek 2.1: Porovnání AJAXu a klasického internetového modelu (převzato z [3])

Podrobnější informace o AJAXu najdete v [8].

2.1.2 Technologie s potřebou plug-inu

2.1.2.1 Silverlight

Technologie Silverlight je odpověď Microsoftu na stále větší popularitu, rozšíření RIA aplikací. Je to mladý projekt, tuto technologii představil Microsoft v prosinci roku 2006 ve verzi 1.0, která však byla oficiálně dokončená, se všemi vývojovými nástroji, až v březnu roku 2007. Avšak z pohledu vývoje aplikací je zajímavá až verze Silverlight 2.x, všechny informace zde uvedené se týkají verze 2.x. Detailní popis technologie je uveden v kapitole

3.

Uživatelské rozhraní je definováno ve značkovacím jazyku (XAML), o výkonnou část se stará objektově orientovaný programovací jazyk(některý z .NET jazyků) a k běhu je vyžadován plugin v prohlížeci. Aby měl Microsoft šanci uspět v konkurenci Flash Playeru s více než čtyrmi miliardami instalací a tržním podílem vetším než 90 procent, musel nabídnout nějakou klíčovou výhodu. Něco co konkurence nemá. Tím je .NET. Jeho použití v Silvelrightu je možné díky stejnému CLR(Common Language Runtime) v obou běhových prostředích jak v Silverlightu tak .NETu. Výhody poskytované .NETem jsou například možnost programování v některém z jeho jakzyků ($C\sharp$, VB, nebo také RUBY, Phython, ..) používání dobře známých tříd a hlavně snadná integrace Silverlightu do již vytvořených aplikací. .NET je vyspělá platforma a tím poskytuje Silverlightu velkou výhodu(věci jako LINQ, kompoziční model WPF, ...).

Silverlight je, narozdíl od .NETu, multiplarformní. Stačí prohlíčeč (nyní podporované jsou Explorer, Firefox, Safari pro Linux od Novelu Moonlight), naistalovaný plug-in(velikost instalace je necelých $5MB$) a samozřejmě připojení. Multiplatformnost je velmi důležitým aspektem dnešní doby, proto se Silvelright stává zajímavým nástrojem pro tvorbu RIAs(kap.6.4).

2.1.2.2 Adobe Flex

Technologie Adobe Flex je v dnešní době dominantní na trhu. Je to také proto, že aplikace vytvořené ve Flexu běží ve Flash Playeru s bezkonkurenčním tržním podílem. Také pro vývojáře je přívětivý:

- definice uživatelských rozhraní pomocí MXML(je podobné HTML)
- stylování aplikace lze zařídit pomocí podmnožiny CSS
- ActionScript 3 zase dobře kombinuje rysy JavaScriptu a Javy. Byly zmenšeny bariéry pro vývojáře přicházející z několika ruzných směrů.

Hlavní výhody Flexu jsou v jeho mnohaletém používání a vývoji. To dává předpoklad, že začít budovat aplikaci ve Flexu je sázka na jistotu. Flex je zdarma. Je to open source, jeho nové prostředí Adobe AIR(Adobe Integrated Runtime) může aplikace spouštět nejen v prohlížeči, ale také na desktopu. O Flex se stará velká skupina lidí, existuje řada komponent a tutoriálů, prostě má toho za sebou již mnoho.

To všechno jsou důvody proč je dnes Flex(Flash) zdaleka nejpoužívanější RIA technologie(vedle AJAXu).

2.1.2.3 Java, JavaFX

Java byla jednou z prvních technologii schopnou implementace RIA aplikací. Používala k tomu Java applety. Jde o jednu z prvních, avšak skoro nepoužívanou technologii, protože Java applety si vybudovaly, bůhví proč, tak špatnou reputaci, že jejich používání se považovalo spíš za chybu než přínos.

JavaFX je v současnosti v plenkách, opět bude používat jako běhové prostředí Javu. Veškeré informace o JavaFX jsou jen malé střípky a na jejich slepení dohromady si budeme muset počkat.

2.1.2.4 a ti druzí...

Jak už víme význam RIA není přesně vymezen, proto bychom mezi technologické zástupce mohli řadit Google Gears, XUL, a další. Některé jsou hodně podobné zmiňovaným mainstreamovým technologiím, jiné, jako například Google Gears, nabízejí úplně něco nového, jiného(podpora offline práce). Je asi dobré vědět, že něco takového existuje, ale mezi RIA technologie je řadit s čistým srdcem nemůžeme.

Tyto technologie mají určitě své pro a proti. Mezi pozitiva bychom mohli zařadit jednotnost plug-inu(pro všechny prohlížeče stejný, odpadá ladění pro různé prohlížeče),nejsou zde technologická omezení, jak Flex tak Silverlight mají funkční WYSIWYG a další vývojařské "lahůdky"(pro různé vývojářské nástroje různé) jako je kontrola syntaxe, debugging, atd... Tedy i ve vývoji RIA aplikací je zřejmá snaha přibližování se vývoji desktopových aplikací.

Hlavní nevýhody jsou dvě. Velké. Kvůli nim například Google používá stále AJAX. Je to potřeba plug-inu. Žádná z RIA platforem není tak rozšířená jako webové prohlížeče. A jako druhá je podstata fungovaní RIA aplikací v prohlížeči. Ačkoli se zdá že jde o klasickou webovou stránku některé věci zde nefungují. Nefungují klávesové zkratky prohlížeče, tlačítko "zpět"a další věci. Většina je řešitelná, ale velmi náročná.

2.2 Silverlight vs. Adobe Flash

Silverlight je zelenáč na trhu webové interaktivity. Na začátku své existence (verze 1.0) by klání těchto dvou ryvalů bylo jednoznačné. Bez větších okolků by vyhrál Adobe Flash. Od uvolnění verze Silverlight 2, která přinesla značný pokrok (více v [4]) je srovnání možné a už ne tak jednoznačné. Srovnávat můžeme z několika pohledů(oblastí použití).

Vývoj aplikací

Flex je na trhu již několik let, pro vývoj RIA aplikací se značně osvědčil. Poskytuje background mnoha hotových aplikací a příkladů ve všech možných oblastech použití. Silverlight, však svého soupeře zdatně stíhá, jednoznačným plusem je možnost zápisu aplikační logiky v jazycích .NETu, má lepší podporu spolupráce s prohlížečem než Flex. Zkrátka Silverlight, díky komfortním možnostem ladění a vývoje aplikace bude získávat stále více příznivců a síly soupeřů se budou postupně vyrovnávat.

Jednoduché animace

Nevýhodou Flashe, při tvorbě jednoduchých aplikací, je potřeba Adobe Flashe. Silverlight používá pro grafiku (její reprezentaci) textový formát XAML. Do tohoto formátu existuje množství konvertotů z grafických a animačních programů. Proto lze vytvářet animace bez nutnosti nákupu dalšího softwaru. Silverlight je v této oblasti lepší.

Streamování videa

V této oblasti, i přes zkušenosti soupeře, vyhrává Silverlight. Možnost přehrávání videí v HD kvalitě, na celé obrazovce jej kapatultovalo před Flash. Microsoft poskytuje na svém serveru každému uživateli 10GB místa zdarma pro ukládání videí. Dalším argumentem pro Silverlight je podpora WMW formátu, naproti od flashového FLV jde o standart. V oblasti streamovaného videa se dá předpokládat, že hvězda Silverlightu bude stoupat a naopak Flash upadat.

Kdo se stane vítězem klání dvou hlavních technologií pro tvorbu RIA aplikací? Jsou to běžní uživatelé. Další hráč na tomto poli zvyšuje konkurenci a ta zde dlouhou dobu chyběla. Silverlight bude určitě využíván k přehrávání videa zatímco v komplexnějších aplikacích si bude muset svůj prostor ještě vydobít.

Kapitola 3

Silverlight

"Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web.", tak takhle definuje Microdoft Silverlight na svých stránkách.

Co to je Silverlight jsme již nastínili v kapitole 2.1.2.1, zde se budeme zabývat jeho funkčností. Jeden z důvodů vzniku Silverlightu, vedle toho marketingového, byla potřeba nabídnout vývojářům ASP.NETu alternativu k Flashi. Aplikace s Flashem vyvýjené v ASP vyžadovaly zcela separovaný designerský nástroj, jiný programovací jazyk(ActionScript) i použití programovacího jazyka Flex. Také neexistuje žádný způsob jak generovat Flash aplikace používající kód .NETu na serveru, to znamená, že integrace obsahu Flashe a ASP.NET je obtížná. Silverlight je krokem pro zvýšení a zlepšení možností vývoje bohatě vybavených internetových aplikací. Silverlight má v mnoha ohledech obdobné funkce jako Flash. Má však navíc jednu podstatnou věc a to mateřský jazyk. V Silverlightu je jím $C\sharp$. Zároveň používá celou řadu pojmů z .NETu. Proto mohou vývojáři psát jak Silverlight aplikaci tak kód pro server ve stejném jazyce podporovaném .NETem($C\sharp$, VB).I používání některých abstrakcí jako jsou proudy, collections, LINQ, atd... je zde podporováno.

Na internetových stránkách Microsoftu je možné najít informaci, že plug-in Silverlightu je dnes naistalovován na více než jedné čtvrtině počítačů na světě a stále více webů jej začíná využívat. Zatím největším projektem provedeným pomocí této technologie byly videopřenosy NBC z Olympijských v Pekingu.

3.1 Historie

Silverlight je odpovědí Microsoftu na již zaběhnutý, a z 90-ti% rozšířený na všech počítačích, Adobe Flash. Ten se svým Flexem poskytuje podobné funkce jako Silverlight. Microsoft si myslí, že pokud bude mít Silvelight dostatečně zajímavý obsah uživatelé si jej do svých prohlížečů nainstalují. Od počátku, za který se dá považovat první uvolnění Silverlight verze 1.0 5.9.2007, kde bylo možné programovat pouze pomocí JavaScriptu, byla hlavním tahákem pro vývojáře vidina možnosti psaní Silvelright aplikací v .NET programovacích jazycích (vyšší verze by měly obsahovat .NET Framework 3.0). Tato možnost přišla s verzí 1.1, která se díky svému velkému přínosu a radikálním změnám vůči verzi 1.0 přejmenovala na Silvelright 2.0. Zde je již zmiňovaná možnost opřít vytvářenou aplikaci o některý z .NET jazyků, jsou implentovány základní prvky(Button, TextBlock, ...). Verze Silvelright 2.0 přišla na světlo světa v březnu roku 2008, ale byla ve fázi Beta. To znamenalo jediné. Ne všechno fungovalo tak jak by si člověk představoval. Porovnání obou verzí je k vidění na `http://silverlight.net/GetStarted/overview.aspx`. V dnešní době je k mání finální verze Silverlight 2, uvolněná na podzim roku 2008.

Prostředkem pro design Silverlight aplikací mají být produkty řady Microsoft Expression. Počáteční verze programů obsažených v tomto balíku získal Microsoft akvizí jiné firmy. Na stránkách Expression najdete detaily k jednotlivým produktům. Dnes jsou k dispozici Beta verze. Tým vývojářů však neustále pracuje na dalších uvolněních.

3.2 Instalace

Instalací Silverlightu(SL) rozumíme instalaci zásuvného modulu do prohlížeče. Pokud uživatel navštíví stránku se Sivlerlight obsahem a nemá potřebný plug-in, na místě kde má být SL část, se objeví banner pro jeho instalaci(viz. obr.3.1).

Obrázek 3.1: Banner nepřímé instalace Silverlightu

Po kliknutí na banner se otevře instalační stránka

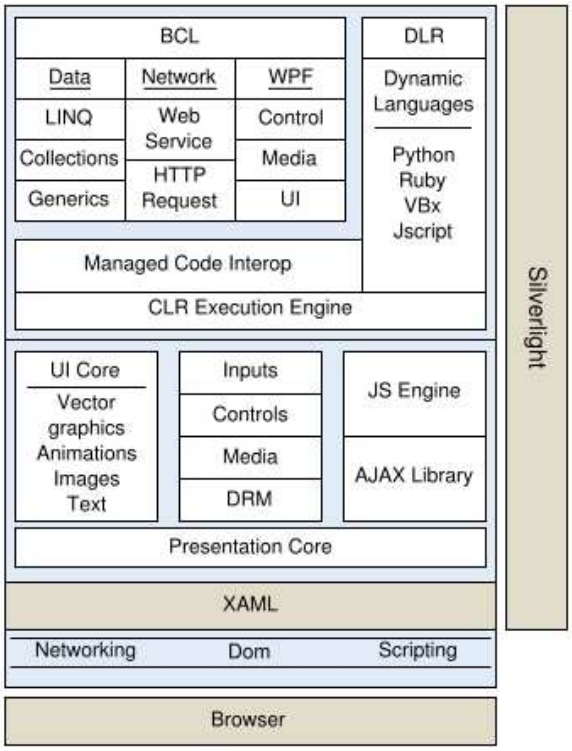
`http://www.microsoft.com/silverlight/install.aspx`. Zde je ověřen používaný prohlížeč a systém. Pokud vše vyhovuje, dalším krokem je potvrzení tlačítka Install(začínáme se stahováním souboru o velikosti okolo 4.5MB a příponou .exe). Otevřením staženého souboru se zobrazí okno s průběhem instalace. Po dokončení je třeba ještě restartovat všechny prohlížeče a SL je připraven k použití. Pokud je instalace provedena například v Internet Exploreru, tato instalace je automaticky používána i dalšími prohlížeči.

Možná jste si všimli názvu obrázku obr.3.1. Je v něm nepřímá instalace Silverlightu. Při budování SL aplikace můžeme použít vlastnost `inplaceInstallPrompt` a nastavit ji na false. Tuto vlastnost najdeme v souboru `názevProjektu.htm`, v tagu `<script>`(funkce createSilverlight). Tím docílíme nepřímé instalace odpovídající popisu nahoře(po kliknutí na obr.3.1 se dostaneme na silverlight stránku, odtud stáhneme instalační soubor ten pak naistalujeme). Opačným nastavením vlastnosti docílíme přímé instalace. Přímá instalace neobsahuje krok ověřování na SL stránce, tedy po kliknutí na banner(ten nevypadá stejně jako na obr.3.1, je doplněn o licenční podmínky) rovnou stáhneme instalační soubor.

Odinstalovat Silverlight je možné v položce, Panelu nástrojů, Přidat nebo odebrat programy.

3.3 Architektura

Architektura Silverlightu se odvíjí od jeho členství ve skupině .NET Frameworku 3.0.



Obrázek 3.2: Silverlight platforma(převzato z [7])

Platforma obsahuje dvě hlavní části, plus instalační a update komponentu(viz.tab.3.1).

Komponenta	Popis
Core presentation framework	Obsahuje komponenty a služby orientované na interakci uživatele, stejně jako XAML(kap.3.4.1) pro deklaraci layoutu. tab.3.2
.NET Framework for Silverlight	Podmnožina .NET Frameworku, která obsahuje komponenty a knihovny, zahrnující CLR, garbage collection, base class libraries,...
Installer and updater	Instalace a aktualizace, které zjednodušují proces instalování aplikace, stejně jako následné updaty.

Tabulka 3.1: Popis hlavních částí Silverlight platformy

Platforma je kombinací již existujících technologií, nástrojů a služeb. Tato skutečnost ulehčuje práci vývojářům, stejně jako jim poskytuje dobrý základ ověřených technologií.

Feature	Popis
Vstup	Obsluhuje vstupy z hardwarových zařízení jako je klávesnice, myš, a další vstupní zařízení.
User Interface vykreslování	Vykresluje vektorovou a bitmapovou grafiku, animace, text. O to se stará UI thread
Media	Uvádí playback a management různých typů audio a video souborů jako *.WMP a $*.mp3$ soubory.
Control	Podporuje rozšiřitelné contorly, ty se dají přizpůsobit stylováním a pomocí šablon.
Layout	Umožňuje dynamické umisťování UI elementů.
Data binding	Umožňuje spojení UI elementů s daty.
XAML	Poskytuje parser pro XAML.

Tabulka 3.2: Core presentation features Silverlight platformy

.NET Framework pro Silverlight je pilířem k robustnímu, objektově orientovanému vývoji internetových aplikací, pro které nebývala tato podpora obvyklá.

Feature	Popis
Data	Podporuje LINQ to SQL a LINQ to XML díky nimž je proces integrace dat z různých zdrojů velmi usnadněn. Pro obsluhu dat jsou také podporovány serializace a použití XML.
Base class library	Soubor .NET knihoven, které poskytují základní programátorské funkce(regulární výrazy, vstupy, výstupy, ...).
WCF	Poskytuje prostředky pro zjednodušení přístupu k vzdáleným datům a službám. To zahrnuje browser object, HTTP požadavek a odpověď, podpora pro JSON, POX a SOAP services.
CLR	poskytuje paměťový management, garbage colection, vyhodnocování vyjímek.
WPF	Poskytuje bohatý soubor kontrol, zahrnující Button, Calendar, CheckBox, DataGrid, DatePicker, HyperlinkButton, ListBox, RadioButton a ScrollViewer..
DLR	Podporuje dynamické programovací jazyky(IronPython, JavaScript) pro tvorbu Silverlight aplikací.

Tabulka 3.3: .NET Framework pro Silverlight

Silverlight nabízí některé další nástroje pro tvorbu RIA. Některé jsou popsány v tabulce níže.

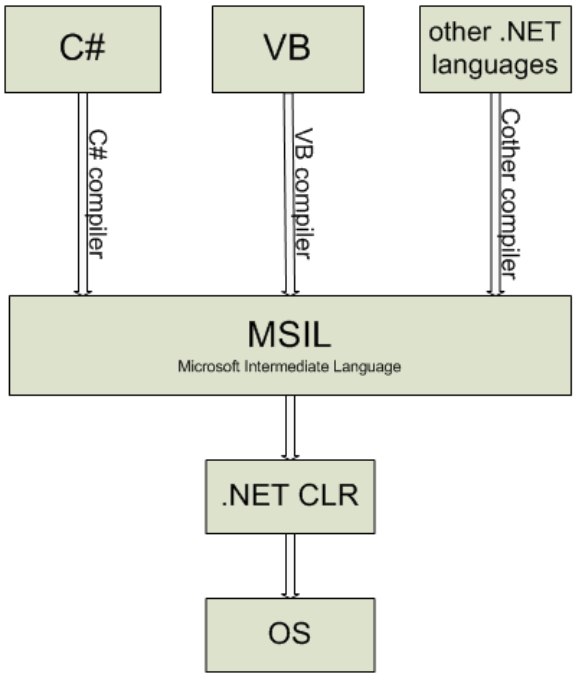
Feature	Popis
Isolated storage	Bezpečný přístup ze Silverlight aplikace na soubor uložen na místním disku počítače. Omezená velikost prostoru(1MB).
Souborový management	Bezpečný File Open dialog k lehčímu načítání souborů.
Serializace	Podpora serializace CLR typů do XML(i JSON).
Packing	Aplikační třída a nástroje na vybudování .xap balíčku. .xap balíček obsahuje aplikaci a entry point pro běh Silverlight plug-inu.
XML knihovny	Třídy XmlReader a XmlWriter zjednodušují práci s XML daty z Web servisů. XLinq díky němu je možné dotazovat se z programovacích jazyků .NETu přímo na XML data.

Tabulka 3.4: Přidané Silverlight Programming Features

V další části této kapitoly budou podrobněji popsány některé části Silverlight platformy. Budou zde zmíněny hlavně ty se kterými budeme dále pracovat nebo jsou nezbytné pro chod a tvorbu Silvelright aplikace.

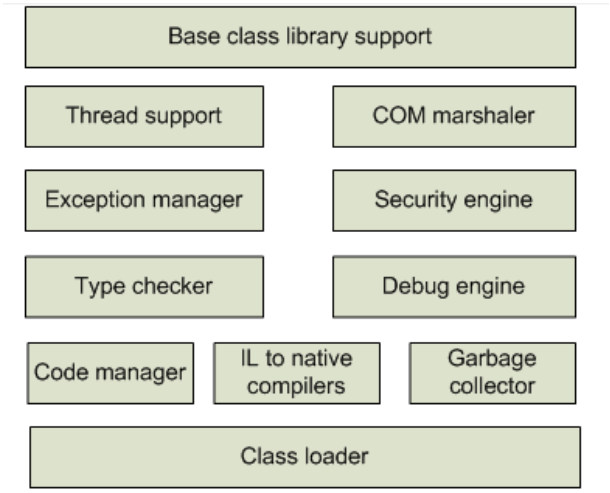
3.3.1 CLR(Common Language Runtime)

Univerzální běhové prostředí pro aplikace v .NET. Při návrhu aplikace začínáme výběrem programovacího jazyka. V prostředí .NET nám právě CLR poskytuje celou řadu jazyků, nezáleží mu na tom v jakém jazyce je program napsán, dokonce se může skládat z jednotlivých částí, které jsou napsány v různách jazycích. Funguje to následovně. Zdrojové kódy libovolného programovacího jazyka .NET jsou přeloženy do intermediálního jazyka (MSIL – Microsoft Intermediate Language) a až teprve při spuštění programu se zkompiluje program z MSIL do kódu pro daný procesor a to je práce pro CLR. Jak celý cyklus vypadá je vidět na obr.3.3.



Obrázek 3.3: Překlad .NET kódu

Není to však jediná funkce CLR. Stará se o typovou bezpečnost, ověřování kódu, management paměti, sběr odpadků atd... Vše co pokrývá je přehledně vidět na obr.3.4.



Obrázek 3.4: Oblasti činnosti CLR

Intermediální jazyk MSIL byl zaveden kvůli přenositelnosti na různé platformy pod-

porující OS Windows. V některých literaturách se pojem CLR připodobňuje pojmu virtuální stroj, který používá Java.
Více o tématice najdeme v literatuře [2].

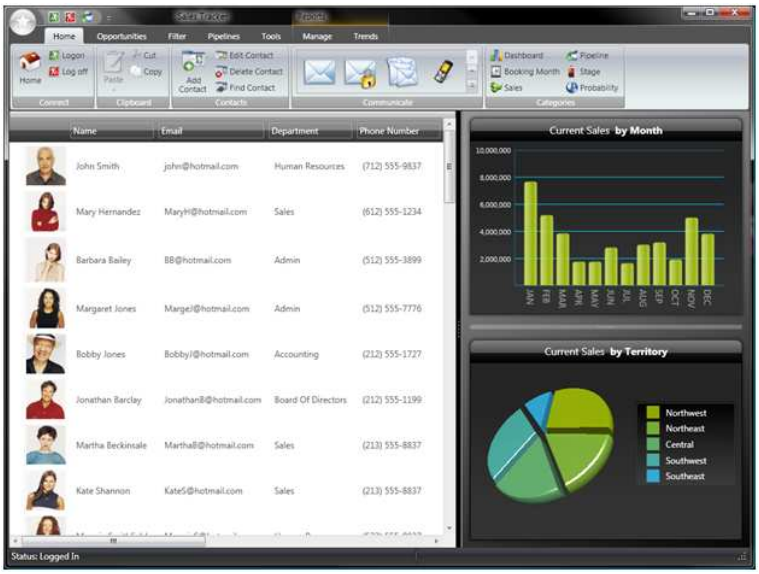
3.4 WPF(Windows Presentation Foundation)

Grafický systém .NET Frameworku 3.0 a vyšší. Jde o celý nový grafický Framework pro psaní Windows aplikací. Předchůdcem WPF jsou WinForms, mohlo by se zdát, že WPF nahrazuje WinForms, ale to pravda není. WPF poskytuje další způsob deklarace, tvorby Windows aplikací. Prostředkem je jazyk XAML(kap.3.4.1). Pořád zde mluvíme o Windows aplikacích, technologie WPF se ale používá i u aplikací internetových(Silverlight).

Hlavní důvod vzniku WPF byla grafika. Vzpomeňme na některé windows aplikace, většina z nich je, nebo do nedávna bývala, formulářového typu. Pokud máte fantazii a popustíte jí uzdu, jste díky WPF schopni využívat mnoho různých možností k tvorbě grafiky:

- vektorová grafika
- animace(časová, použití frames, 3D animace, atd...)
- multimédia(audio, video)
- DataBinding(provázání dat)
- efekty(stíny, záře, atd...)

... a pak je možné, že WinForm aplikace bude vypadat třeba tak jak na následujícím obrázku(obr.3.5)



Obrázek 3.5: Ukázka WPF desktopové aplikace

Můžete si říct, "No jo, ale takovéhle aplikace budou hodně náročné, procesor bude mít co dělat."Zatímco aplikace zpracovává procesor, grafická karta je méně, téměř vůbec, vytížena. WPF běží na vrstvě DirectX, proto jsou WPF aplikace(jejich renderování) zpracovávány grafickou kartou zatímco procesor může dělat jiné věci.

WPF je k dispozici v .NET Frameworku 3.0 a vyšším. Pro korektní běh WPF aplikací je nutné mít naistalován .NET Framework 3.0 a vyšší na vašem počítači. V OS Vista je již součástí základní instalace.

Detaily, týkající se WPF, lze najít v [12].

3.4.1 XAML(Extensible Application Markup Language)

Jde o značkovácí jazyk založený na XML k tvorbě uživatleského rozhraní. Tento jazyk je součástí WPF(kap.3.4). XAML vznikl, protože tvorba UI za pomoci značkovacích jazyků(markup languages) je o mnoho elegantnější a přehlednější než v jazycích objektově orientovaných(VB, C♯). Mezi značkovací jazyky patří i známý HTML, ten definuje UI u webových stránek. Další velkou výhodou XAML je jistě oddělení grafické a programové části. Při tvorbě WPF aplikace jsou vytvořeny dva soubory. Jeden s příponou ∗.xaml, zde je definován vzhled aplikace. Druhý má příponu ∗.cs(přípona podle použitého programovacího jazyka), v něm je implementována logika aplikace.

Následující kód představuje stránku v Silverlight aplikaci(jménem SilverlightApplication9)

deklarovanou XAMLem. Zobrazí se bílá plocha o rozměrech 400px na 300px, uprostřed které je nápis *"Hello amigos"*, dole na stránce je tlačítko s textem *newControl*.

```
<UserControl x:Class="SilverlightApplication1.SilverlightControl2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Name="txtb1" Text="Hello amigos" HorizontalAlignment="Center"
                   VerticalAlignment="Center" FontSize="20"/>
        <Button x:Name="button1" Content="newControl" Click="button1_Click"
                VerticalAlignment="Bottom" Width="90" Margin="20" Background="Red"/>
    </Grid>
</UserControl>
```

Obrázek 3.6: Ukázka XAML kódu

Pro experimenty s XAMLem v reálném čase lze využít XAMLPad, jednoduchý XAML editor.

3.5 WCF(Windows Communication Foundation)

WCF je ucelený Framework na tvorbu komunikace mezi dvěma částmi přeložitelného kódu. WCF se objevilo stejně jako WPF s příchodem .NET Frameworku 3.0(prosinec 2006). Říkáte si na co další komunikační technologie, vždyť v současné době máme k dispozici věci jako DDL, DCOM, NET Framework remoting, Enterprise Services, Web Services, WSE,... Příchod WCF je ale zlomový pro tvorbu distrubuovaných aplikací. Sjednocuje funkci všech doposud používaných Microsoft technologií pro tvorbu těchto aplikací. To vše je možné díky komunikaci založené na webových službách. Jeden z důvodů vzniku WCF je lepší implementace SOA(Service Oriented Architecture). SOA má rozdílné části kódu/služby běžící na zařízeních kolem světa, klienti mohou používat tyto kódy/služby. Tedy zásadou SOA je izolovanost, to je lépe pochopitelné na příkladu. Řekněme, že máme webovou službu, která přijímá PSČ a vrací předpověď počasí pro danou oblast. Vše o co se stará uživatel je správné zadání PSČ(což je vstup). Mezi serverem(host) a klientem je izolace. Nekomunikují mezi sebou přímo, ale pomocí služby. To jak komunikují si "domluvili"v contractu(říká co služba dělá) služby(kap.3.5.1).

WCF používá SOAP zprávy pro komunikaci mezi dvěma procesy, to vytváří schopnost WCF aplikací spolupracovat s jakýmkoli dalším procesem komunikujícím přes SOAP(viz.

kap.3.5.1). Jakákoli aplikace, používající WCF, má WSDL interface, proto je schopna pracovat s jakýmkoli WebServicem bez ohledu na to, na jaké platformě služba funguje. Před vznikem WCF závisel výběr technologie na požadavcích vytvářené aplikace a tak se někdy stávalo, že jsme museli technologie kombinovat. Například část aplikace používala pro komunikaci .NET remoting(důraz na výkonnost aplikace) a pro komunikaci s jinými než .NET aplikacemi zase WSE. V projektu založeném na WCF je vše založeno na službách a pomocí služeb komunikujeme .

Detaily o WCF lze najít například v [11].

3.5.1 Web Services(Webové služby)

Webové služby jsou konsorciem $W3C$([3]) defionovány jako "software system designed to support interoperable machine-to-machine interaction over a network". Jinými slovy jde o prostředek podporující komunikaci mezi aplikacemi napsanými v různých programovacích jazycích, běžících na různých platformách nebo počítačích. Jak jsme zmínili v kap.3.5 je webová služba prostředkem pro komunikaci ve WCF. Poskytovatel služby nabízí data, klient najde adresu služby v registru webových služeb, načte si její popis a může ji volně využívat. Webová služba je tedy prostředek pro tvorbu distribuovaných systému. Systémů jejichž části jsou na různých místech(počítačích) propojených sítí.

WebServices je technologie pro vzdálené volání procedur pomocí přenosu SOAP zpráv(psané v XML) protokolem HTTP. S WebServices jsou spojeny tři základní pojmy:

- SOAP zprávy(vzdálené volání procedur)
- WSDL jazyk (popis poskytovaných služeb)
- UDDI a WSIL mechanismy(nalezení služeb v "seznamu"kde získám jejich WSDL popis).

SOAP(Simple Object Access Protocol)

Soap je protokol pro komunikaci pomocí zpráv. Zprávy umožňují volat vzdálené funkce protokolem HTTP. Ten při volání služby přenese zprávu vytvořenou v XML. Tato zpráva popisuje volanou funkci a její parametry. Jestliže máme webovou službu s takovýmto obsahem.

```
....
[WebMethod]
 public int Add(int a, int b)
 {
   return a+b;
 }
 ...
```

SOAP zpráva posílaná jako dotaz na tuto službu může být:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
     <Add xmlns="http://tempuri.org/" />
        <a>5</a>
        <b>8</b>
     </Add>
  </soap:Body>
</soap:Envelope>
```

a služba poté odpoví zprávou:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/" />
      <AddResult>13</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

Takto vypadá jednoduchý případ komunikace webové služby pomocí SOAP zpráv. Vnitřku zprávy nemusíme rozumět, je generován a není možné jej upravovat.

WSDL(Web Services Description Language)

Abychom mohli protokolem SOAP volat, musíme vědět co volat, ale i jak to volat. Jazyk WSDL popisuje rozhraní služeb. To zahrnuje jména dostupných operací, typy parametrů a návratových hodnot. Také to na jakém místě(port, URL, stroj) a jak(HTTP, HTTPS, ...) je služba k dispozici. Je to jeden ze standardů používaných pro popis služeb a rozhraní pomocí XML.

UDDI(Universal Description, Discover and Integration)

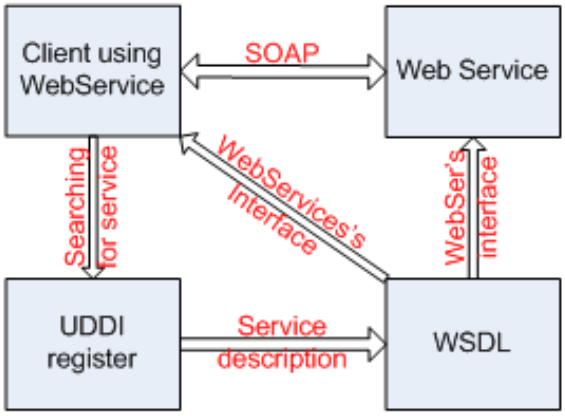
Je pohodlné mít možnost hledat služby podporující XML snadno na Interntu. Proto byl zaveden UDDI registr, který umožňuje vyhledávání a publikování informací o WebServicess. Publikují se pouze služby, které jsou pro širokou veřejnost, ne už tak ty, které

používá jen určitá vývojová skupina(tým).

Webová služba jako taková je tvořena dvěma hlavními částmi. Je to contract(obsah webové služby), který je ve WCF obvykle implementován jako interface s atributem `[ServiceContract]`. Druhou částí je implementace metod obsažených v interfacu(metadata). Mezi další nezbytné části patří

- Adress tedy místo kde službu najdeme
- Binding tedy jaký protokol používáme pro komunikaci se službou.

Spolu s contractem tvoří tyto dvě části EndPoint. Ten udává jak ServiceHost předkládá službu. Potom klient může používat metody obsažené v contractu. A proto ten kdo používá webovou službu, na základě daných metadat a EndPointu, může vytvořit Proxy. Proxy, to je prázdná schránka, typ, který předkládá všechny detaily služby. Konečně klient a host spolu komunikují skrze různé kanály(channels). Kanál je popisován jako to skrze co prochází zpráva při komunikaci klient-host a naopak. Okolo kanálů mohou být deklarovány různé Behaviors spojené se službou. Behavior umožňuje modifikaci zpráv procházejících kanálem. Příklad deklarace těchto částí je vidět na obr.4.7. Jak vypadá komunikace mezi hostem a klientem pomocí webové služby, ukazuje následující schéma (obr.3.7).



Obrázek 3.7: Schéma komunikace WebService

V prostředí Microsoft Visual Studia je k dispozici celá řada druhů webových služeb. Výběr záleží na tom, kde bude služba nasazena, v jakém typu aplikace. Příchod WCF

přinesl již hotové šablony webových služeb pro různé technologie .NETu. S tím i Silverlight-enabled WCF WebService, specielně pro Silverlight. Tato speciální služba pro Silverlight je vyjímečná svou stavbou, nemá oddělený Interface, ten je v jednom souboru spolu s definicemi obsahu služby(metadaty). Také použitým typem bindingu viz.kap.4.2.2. Samozřejmě není stryktně dáno, že pro Silverlight lze používat pouze tuto službu. Volba je na vývojáři, tato šablona je pouze doporuřena.

Shrnout WCF bychom mohli do následující věty. Máme hosta a klienta, ti "souhlasí"s contractem. Host(server) předkládá EndPointy, ty jsou kombinací Adresy, Bindingu a Contractu. Klient používá proxy, která je provázána s jednotlivými endpointy-provázána s jednotlivými contracty.

3.6 LINQ(Language Integrated Query)

Jde o označení pro skupinu rozšíření .NET Frameworku. Toto rozšíření je označováno za revoluční v přístupu k datům, a to jakýmkoli. LINQ bylo uvedeno spolu s .NET Frameworkem 3.5.
Data jsou v dnešní době součástí téměř každé aplikace. .NET umožňoval již od počátku práci s relačními(pravoúhlé uspořádání dat-řádky, sloupce) nebo hierchackými(je zde nějaká struktura uložených dat, např. strom) daty několika způsoby zastřešenými obsaženými v ADO.NET([2]). Narozdíl od těchto způsobů přináší LINQ podporu dotazování přímo v jazyce .NET(C♯, VB). Tento přímý přístup přináší několik výhod, kde asi největší je dotazování pomocí klíčových slov(query words) podobných SQL syntaxi. Nesporná výhoda je kontrola kódu při kompilaci programu, většina chyb je odhalena již v této fázi. LINQ je možné použít na téměř jakákoli data. Podstata LINQ je podobná ADO.NET, kde vytvoříme provider pro určitý typ databáze. LINQ není omezeno pouze na relační databáze, díky novince v .NET, Expression Trees(data jsou ukládána do stromové struktury). Ty umožňují práci s kódem jako s daty, proto může být LINQ dotaz přeložen providerem pro určitý datový zdroj a nezáleží na typu dat. LINQ má několik skupin(implementací):

- LINQ to Object-stadartní kolekce z paměti
- LINQ to SQL-Microsoft SQL Server typy

- LINQ to XML-práce s XML daty
- LINQ to DataSet-práce s ADO.NET datasety

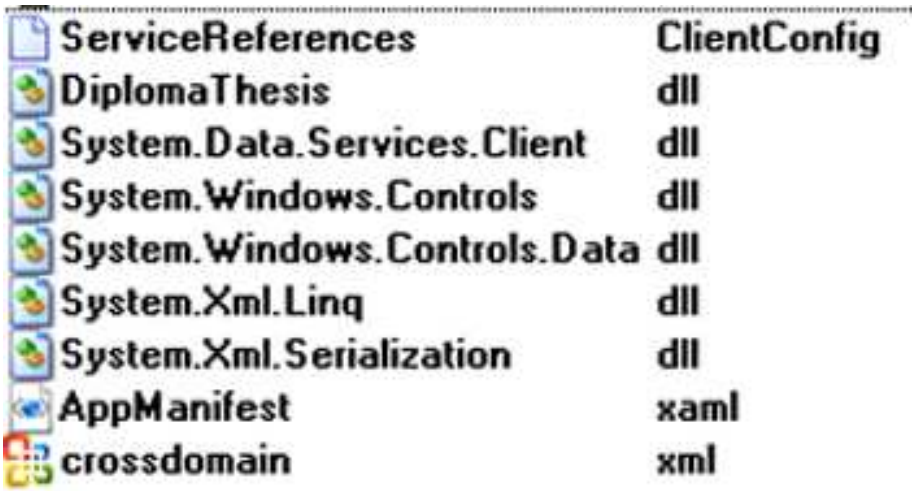
Asi nejvíce používanou implementací je LINQ to SQL, ta umožňuje dotazování nad relačními databázemi. LINQ to SQL umožňuje i modifikaci dat v databázi, tím dostáváme velmi silný nástroj pro práci s nimi. V následující ukázce provedeme LINQ dotaz nad dvěma kolekcemi. Je zde vidět způsob dotazování a klíčová slova podobná těm používaným v SQL jazyce.

```
using System.Limq;
...
        System.Collections.ObjectModel.ObservableCollection<Teacher> listOfTeachers;
        System.Collections.ObjectModel.ObservableCollection<Locations> listOfLocations;
...
var teachers = (from teacher in listOfTeachers
                join location in listOfLocations on
                teacher.idLocation equals location.idLocation
                select new { location , teacher  }).ToList();
```

Syntaxe se opravdu velmi podobá SQL dotazu. Liší pouze umístěním klauzule from a select. Zatímco v SQL je select na začátku dotazu u LINQ je na konci. Dotaz začíná klíčovým slovem from. Díky této obměně nám je intellisense ve VS 2008 schopen napovídat, protože je hned známo, jakého typu je datový zdroj. Více informací o LINQ lze získat například na oficiálním webu microsoftu [2] nebo [10].

3.7 Struktura Silverlightu

Spolu se Silverlightem přichází i nová přípona ∗.XAP[zap]. Jde o komprimovaný soubor obsahující všechny soubory potřebné ke správnému běhu Silverlight aplikace. Ke kompresi ∗.xap se používá zip komprese. Soubor se vytvoří při kompilaci projektu, je uložen v adresáři Client-Bin(obr.4.9) a jmenuje se stejně jako projekt(aplikace), ke kterému patří. Přepíšeme-li příponu xap na zip, je možné nahlédnout do obsahu ∗.xap souboru(viz. obr.3.8). Obsahuje soubor application manifest (AppManifest.xaml), DLL knihovny nezbytné pro bezproblémový běh a bezpečnostní soubory(4.2.3). První dll knihovnou je zkompiloná verze aplikace se stejným názvem.



Obrázek 3.8: Obsah souboru XAP odpovídající aplikaci s názvem DiplomaThesis

Soubor AppManifest.xaml obsahuje popis aplikace pro spuštění v doplňku Silverlight. Soubor je aktualizován při kompilaci aplikace, na základě referenční assembly je připojen soubor AppManifest. Přikládané soubory se dají spravovat, pro assembly, kterou nechceme zahrnout do ∗.xap, nastavíme vlastnost "Copy Local"na *false*. Pakliže nezahrneme assembly do ∗.xap, její stažení bude vyžádáno při jejím výskytu v kódu. Toto je užitečné z hlediska velikosti stahovaného souboru, obsahuje pouze nezbytné assemblies a ty, co jsou použity v kódu, jsou staženy až když jsou potřeba. Mechanismus je velkmi užitečný používá-li určitou assembly pouze část aplikace.
Příklad souboru AppManifest.xaml je níže.

```
<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 EntryPointAssembly="DiplomaThesis" EntryPointType="DiplomaThesis.App" RuntimeVersion="2.0.31005.0">
  <Deployment.Parts>
    <AssemblyPart x:Name="DiplomaThesis" Source="DiplomaThesis.dll" />
    <AssemblyPart x:Name="System.Data.Services.Client" Source="System.Data.Services.Client.dll" />
    <AssemblyPart x:Name="System.Windows.Controls.Data" Source="System.Windows.Controls.Data.dll" />
    <AssemblyPart x:Name="System.Windows.Controls" Source="System.Windows.Controls.dll" />
    <AssemblyPart x:Name="System.Xml.Linq" Source="System.Xml.Linq.dll" />
    <AssemblyPart x:Name="System.Xml.Serialization" Source="System.Xml.Serialization.dll" />
  </Deployment.Parts>
</Deployment>
```

Obsahuje tag `<Deployment>`, který definuje aplikaci a obsahuje další vnořené tagy. Atribut EntryPointAssembly definuje které assembly, z níže definovaných(`<AssemblyPart>` nód), je hlavní dll knihovnou aplikace. EntryPointType atribute specifikuje třídu z dll

knihovny(assembly), definované v EntryPointAssembly atributu, která je hlavní třídou pro start aplikace. `<Deployment>` nód má atribut definující verzi Silverlightu ve které je aplikace napsána. Jak je vidět, nódy `<AssemblyPart>` definují assembly(dll knihovny) použité v ∗.xap souboru a každé knihovně přidělují jméno.
Tento ∗.xap soubor je stažen prohlížečem, přeložen zásuvným modulem a jeho obsah(Silverlight aplikace) zobrazen na stránce.

Kapitola 4

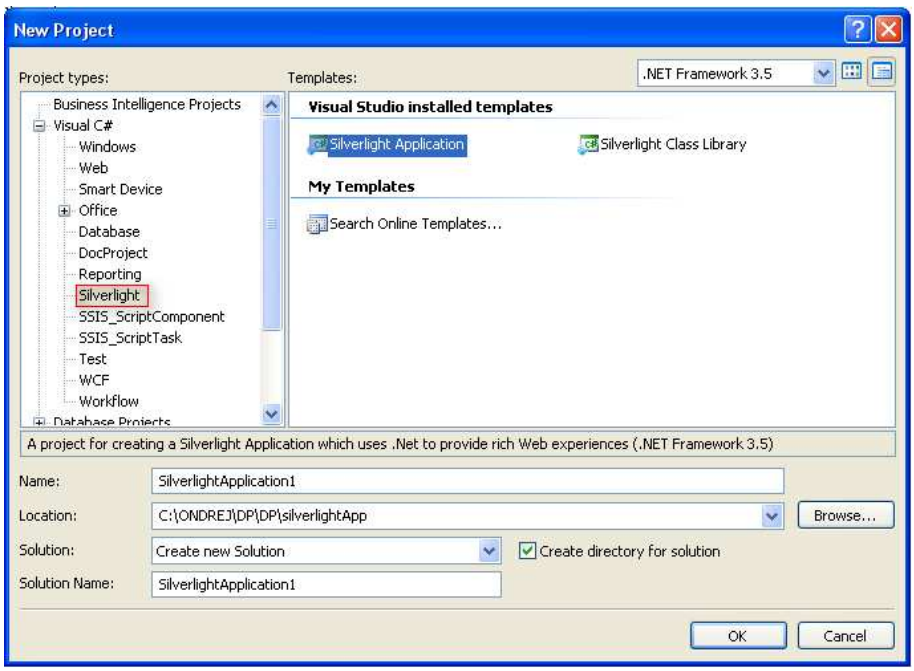
Vytvoření RIA aplikace

RIA aplikaci můžeme vytvořit pomocí různých technologií. V této práci byl na tyto účely použit Silverlight. Proto následující popis bude zaměřen na Silverlight aplikaci.

4.1 Tvorba Silverlight aplikace

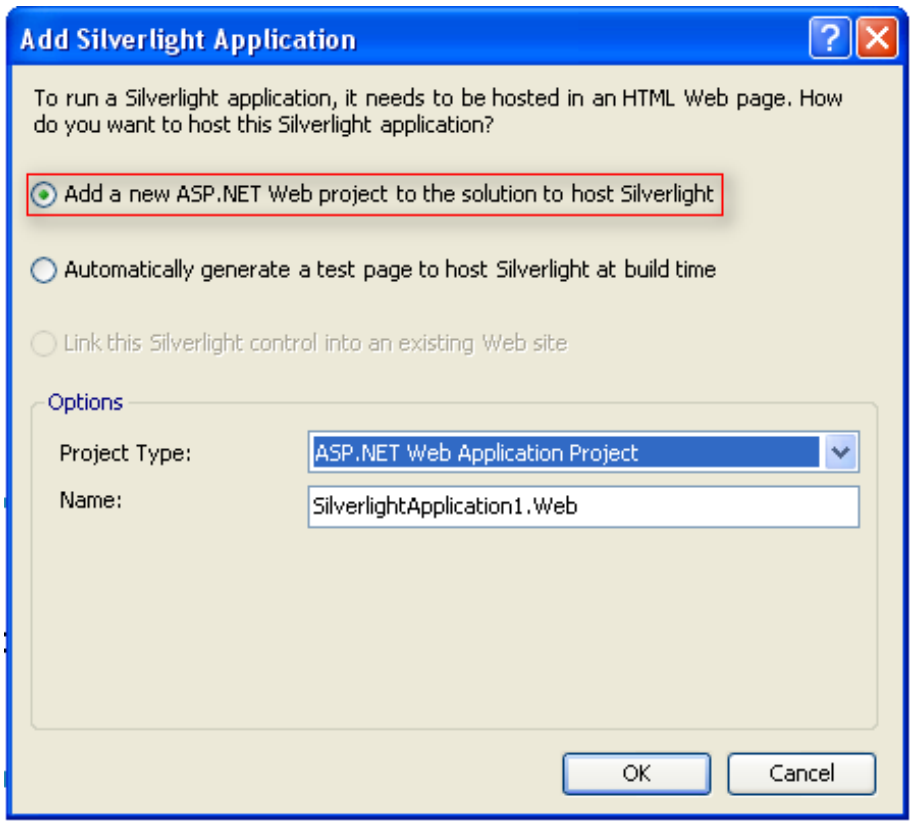
Rozhodneme-li se naprogramovat Silvelight aplikaci musíme mít k dispozici pár nezbytných věcí. Jako první je prostředí, kde bychom vlastní kód implementovali. Takovým vývojovým prostředím je například Microsoft Visual studio(VS). Pro Silverlight ta nejnovější verze VS 2008. Tím dostáváme komplexní nástroj pro tvorbu všech druhů aplikací dostupných v .NET. VS je nástroj určený hlavně pro vývojáře, značně ulehčuje práci věcmi jako jsou intelisence, code snippes, atd... Protože Silverlight aplikace jsou graficky velmi vyspělé a tvorba grafiky(animace, vlastní šablony prvků, ...) je důležitá, Microsoft vyvýjí paralelně nástroje Expression. Jejich výčet a více informací o nich je na webu. Díky této koncepci je tedy striktně oddělena práce grafiků a vývojářů. Práce se stává efektivnější. VS a Expression mezi sebou kooperují, změní-li se něco v jednom prostředí, ve druhém se při otevření projektu s provedenou změnou automaticky provede refresh a změna se zapíše.

Instalací Visual Studia 2008 ještě nejsme schopni vyvíjet Silverlight aplikace. Musíme doinstalovat updaty pro Silverlight. Ty jsou popsány v literatuře [6].
Po doinstalaci veškerého potřebného softwaru vybereme z nabídky projektů VS ten s názvem Silverlight(vlevo) a šablonu Silverlight Application, jak je vidět na obr.4.1.



Obrázek 4.1: Výběr šablony pro tvorbu Silverlight aplikace

Po vyplnění všech potřebných údajů a potvrzení se dostaneme na výběr druhu Silverlight aplikace. Na obr.4.2 je vidět dialogové okno, kde je na výběr mezi SL aplikací, která je hostována na ASP.NET stránce (my používáme tento typ obr.4.9) nebo tou, kde je Silvelight hostován na stránce automaticky generované. Tato aplikace pak neobsahuje ASP.NET část. Je sice možné vytvořit web, který je založený jen na Silverlightu, ale není pravděpodobné, že by se někdo touto cestou vydal. Skutečnost, že Silverlight je vlastně ještě v plenkách, také to, že nepodporuje poděděné klienty (nepodporuje uživatele Windows ME, Windows 2000 a Windows 98), obojí dohromady to znamená, že rozsah možných implementací není srovnatelný s obyčejným HTML. I přesto mnohé weby, které používají Silverlight, jej nepoužívají již jen k odlišení a vyzdvižení se od ostatních online konkurentů, ale také kvůli možnosti vytvořit téměř jakoukoli funkčnost. Web se stává prostorem téměř bez mantinelů.



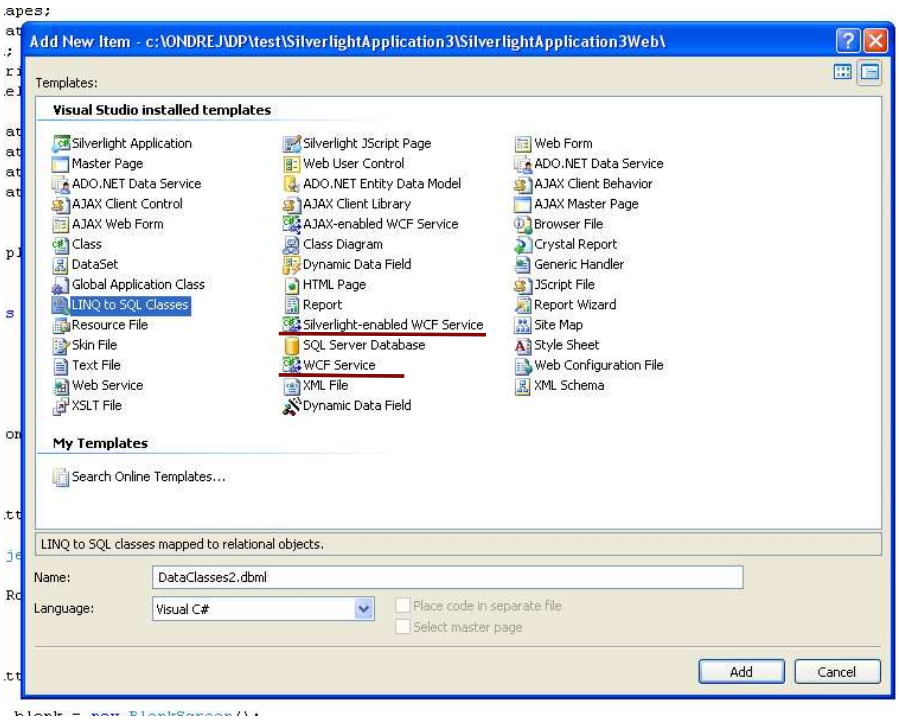
Obrázek 4.2: Dialogové okno pro výběr druhu Silverlight aplikace

Po potvrzení se vytvoří Silverlight projekt, který obsahuje základní složky a vypadá podle toho, jaký druh jsme zvolili.

4.2 Nastavení komunikace s databází

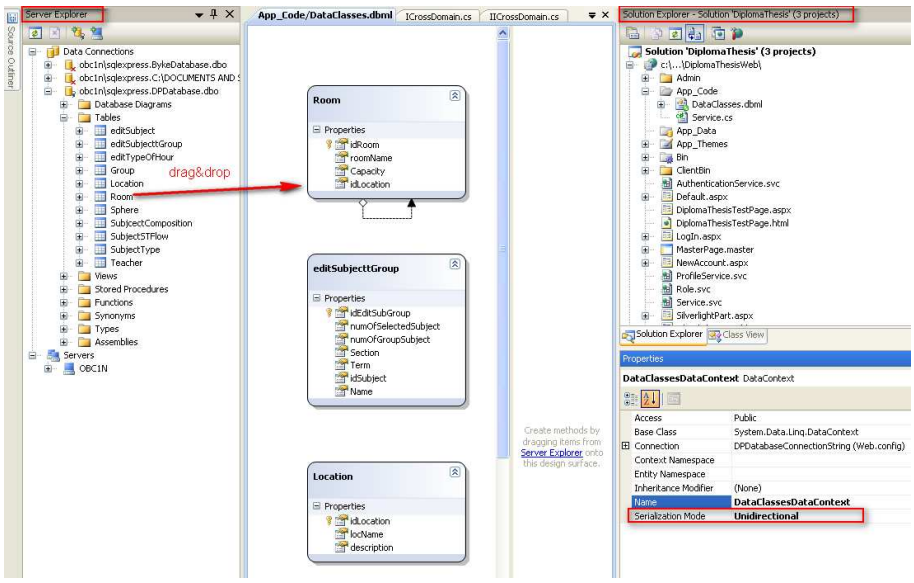
V Silverlightu je možné ke komunikaci s databází použít webové služby. Protože pracujeme v Silverlightu můžeme použít speciální template, obsažen ve Visual Studiu, jménem *Silverlight-WCF enabled WebService*, který je popsán v 3.5.1. Spolu s webovými službami je příhodné používat technologii LINQ(kap.3.6), usnadňující práci s daty. Proces připojení aplikace k databázi je popsán v následujícím textu.

Máme databázi vytvořenou v Microsoft SQL Server jménem DPDatabase. Tuto databázi připojíme k naší aplikaci tak abychom mohli s daty pracovat pomocí LINQ. K tomu použijeme šablonu VS jménem LinqToSql class, kterou přidáme do projektu.



Obrázek 4.3: Tvorba třídy Linq to Sql

Po připojení LinqToSql třídy k projektu(jméno.dmbl) se soubor přidá do záložky AppCode ve webové části projektu(obr.4.9). Metodou drag and drop přidáváme tabulky databází zobrazených v Server Exploreru do DataClass, jak je vidět na obr.4.4.



Obrázek 4.4: Přidávání prvků do Linq třídy

Přidáním tabulek se vytvoří spojení s databází. Spojení s databází je deklarováno connectionStringem, což je, už podle názvu, řetězec spojení. Tvar connection stringu se liší od typu Databázového serveru. V SQL Server 2008 má ConnectionString DPDatabese databáze následující tvar a je obsažen v souboru *Web.config*(obr.4.2.1).

```
<connectionStrings>
...
<add name="DPDatabaseConnectionString" connectionString="Data Source=OBC1N\SQLEXPRESS;
          Initial Catalog=DPDatabase;Integrated Security=True"
          providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Nesmíme zapomenout nastavit vlastnost třídy Serialization mode z None na Unidirectional, protože pro práci třídy se službou je třeba aby byla serializovatelná. Vytvořili jsme poskytovatele dat. Ten bude v budoucnu poskytovat data službě (kap.4.2.2). Tedy

pro komunikaci s databází vytvoříme webovou službu, která bude k datům přistupovat přostřednictvím třídy LinqToSql. Vše, webovou službu, LinqToSql třídu umístíme na stejný server. Nastavíme contract službě a skrz ni potom můžeme používat data zahrnutá v LinqToSql třídě.

4.2.1 Web.config

Tak jako mají například Windows uložena svá nastavení v registrech, má svá nastavení uložena i webová .NET aplikace. Soubor *Web.config* je konfigurační soubor ASP.NET aplikací. Protože jsme zvolili Silverlight projekt hostovaný na ASP.NET stránce(obr.4.2), jmenuje se konfigurační soubor aplikace Web.config. Ostatní .NET aplikace mají konfigurační souboru pojmenovány podle jména aplikace jménoAplikace.config. Soubor $*$.config je psán v XML, rozdělen na několik částí(appSettings, connectionStrings, ...). Konfigurační soubor je přiložen k aplikaci a můžeme jej měnit(nastavování connectionStringů, ...). Za zmínku ještě stojí, že každá ASP.NET aplikace dědí základní *Web.config* soubor od web.config souboru stroje (ten je uložen $root \setminus \setminus Microsoft.Net \setminus Framework \setminus v*.*.*.*\setminus CONFIG$). Základní stavba, souboru *web.config* je vidět v následující ukázce.

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
    <appSettings/>
    <connectionStrings/>
    <!--nastavujeme napríklad autentizaci a autorizaci ASP.NET aplikace-->
    <system.web>

    <!--nastavujeme chování naší aplikace pri kompilaci
    debug na true, znamená to, že se naše aplikace bude
    kompilovat v režimu ladení a budou se nám vypisovat
     všechna chybová hlášení-->

        <compilation debug="false"/>
        <authentication mode="Windows"/>
        <!--
            customErrors umožňují nastavení reakcí na neočekávané události
        -->
        <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm"/>
            <error statusCode="404" redirect="FileNotFound.htm"/>
        </customErrors>

    </system.web>
</configuration>
```

V této fázi máme připraveny data. Nyní můžeme vytvořit službu na spojení s databází.

4.2.2 Připojení webové služby

Jak už zde několikrát bylo řečeno ve Visual Studiu je pro Silverlight speciální webová služba. Nejedná se o žádnou novinku ve smyslu nové služby, nýbrž jenom o šablonu upravenou na míru Silverlightu. Ve verzi Silverlightu Beta 1 jsme jako WCF službu byli nuceni používat WCF Service, jak je vidět na obr.4.3 tato služba(resp. šablona) zůstala v nabídce VS. Pro to, aby služba v naší aplikaci pracovala tak jak si představujeme, je potřeba několik malých úprav. Tyto úpravy se týkají bindingu(kap.3.5.1). Současná verze Silverlightu podporuje výhradně jeden typ bindingu a to basicHttpBinding. Proto je třeba změnit položku binding, u WCF služby, v souboru *web.config* z wsHttpBinding na basicHttpBinding. Potom bychom neměli zapomenout ani na vzájemnou kompatibilitu ASP.NET a prvků použitých ve webové službě. Za tímto účelem se připojují některé dslší atributy do Web.config. Abychom nemuseli tyto věci dopisovat a neustále je hlídat, byla vytvořena právě Silverlight-WCF enabled Web service. Jde tedy jenom o další ulehčení práce programátorů. Samozřejmě lze použít jakoukoli jinou službu, jen je třeba dbát na již zmíněné, nezbytné, úpravy.

Služba(Silverlight-WCF enabled Web Service) se skládá ze dvou souborů. Jeden *názevSlužby.svc* a druhý *názevSlužby.cs*(pro kód psaný v C♯). Soubor s příponou svc popisuje službu. Specifikuje jazyk, ve kterém budou psány metody(podle jazyka se odvíjí přípona souboru s kódem služby ∗.cs, ∗.vb), cestu k souboru s těmito metodami(metadata), název. Jde obvykle o jeden řádek, příklad obsahu takového souboru je vidět v následujícím textu. V tomto případě je *názevSlužby* ekvivalentní s MainService.

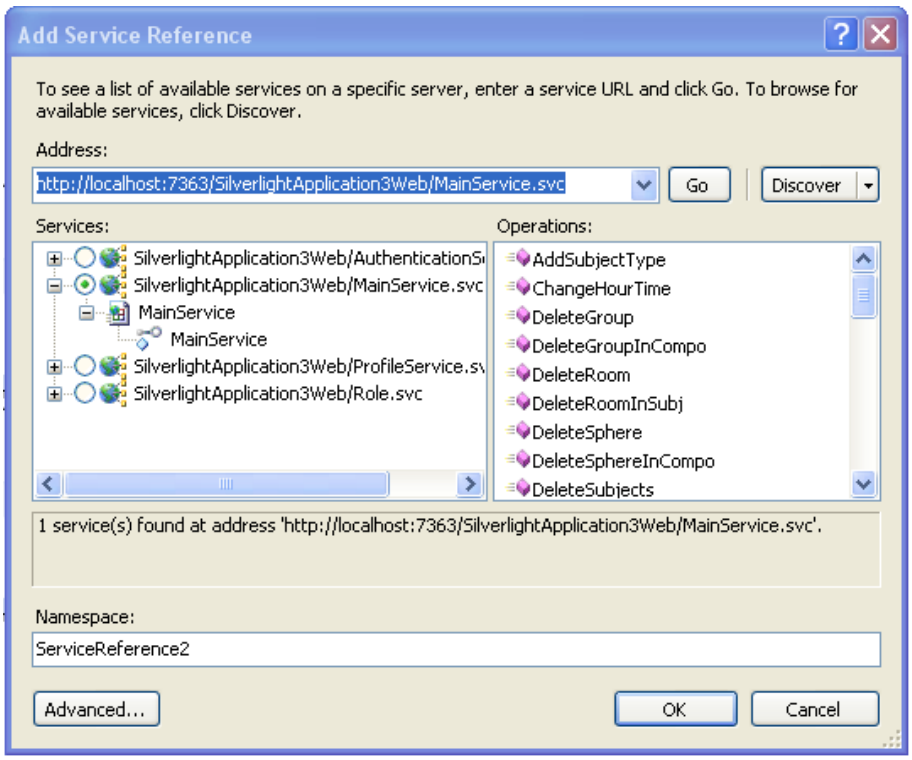
```
<%@ ServiceHost Language="C#" Debug="true" Service="MainService" CodeBehind="~/App_Code/MainService.cs" %>
```

Soubor s příponou cs obsahuje metody služby, tedy obsah, funkčnost metody. Z obsahu ∗.svc je dále zřejmé, že soubor ∗.cs je v aplikaci umístěn ve složce AppCode(obr.4.9). Na obr.4.5 je obsah souboru MainService.cs. Metody jsou zde uvozeny atributem [OperationContract]. Atribut [OperationContract] se vkládá kvůli lepší kontrole generace WSDL. Nejčastěji zůstává atribut tak, jak jej je vidět na obr.4.5, ale například pro obousměrné zpávy(duplex messages) je třeba nějakým způsobem deklarovat některé specifikující(doplňující) vlastnosti. Co je možné v OperationContract definovat najdete v [2]. Jinak je práce v kódu v pozadí WCF služby velmi podobná praci v klasické třídě, tedy, říkáme službě co má dělat. Služba MainService obstarává komunikaci s datbází DPDatabase. Na obr.4.5 je uvedena pouze její malá část(jedna metoda). Tato metoda vrací tabulku Teacher z DPDatabase obsaženou v DataClass(viz.obr.4.4) provideru.

```
MainService.svc | ServiceReferences.ClientConfig | App_Code/DataClasses.dbml | Page.xaml.cs | Page.xaml | App_Code/MainService.cs | Role.svc | Web.config
MainService                                                                                                                                db

using System;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Activation;
using System.Collections.Generic;


[ServiceContract(Namespace = "")]
[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
public class MainService

{
    DataClassesDataContext db = new DataClassesDataContext();
    [OperationContract]
    public List<Teacher> GetTeachers()
    {

        var teacher = from u in db.Teachers
                      select u;
        return teacher.ToList();

        //db.Teachers;
    }
```

Obrázek 4.5: Soubor s kódem služby

Vytvoříme-li službu zbývá ji připojit k aplikaci. Stiskem pravého tlačítka, nad položkou References, je vyvoláno kontextové menu v jehož nabídce vybereme Add Service Refences. Otevře se dialogové okno obr.4.6.



Obrázek 4.6: Dialogové okno připojení reference na službu

Z nabídky služeb vybereme tu, kterou chceme připojit, pojmenujeme připojovanou referenci. Dialogové okno na obr.4.6 je rozděleno na dvě části, Services a Operations. Operations jsou metody použité ve službě, vybrané v části Services.

Po přidání služby k projektu(přes referenci) je vygenerován soubor ServiceReferences.ClientConfig. Tento soubor obsahuje některé konfigurační informace. Ve verzi Silverlightu Beta 1 byl vytvořen, ale už ne použit. V Silverlightu 2 je soubor vygenerován, připojen k XAP(obr.3.8) a používán jako podmnožina WCF konfigurace. Obsah je v XML jazyce. Příklad obsahu je na obr.4.7. Je zde vidět popis služby, tedy druh bindingu a endpoint. Druh bindingu specifikuje protokol používaný určitým endpointem. EndPoint je to jak je služba předkládána, aby klient mohl vyvolat operace definované v contractu.

```
ServiceReferences.ClientConfig | App_Code/DataClasses.dbml | Page.xaml.cs | Page.xaml | App_Code/MainService.cs | Role.svc | Web.config | SilverlightApp...3TestPage.aspx
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_AuthenticationService" maxBufferSize="2147483647"
          maxReceivedMessageSize="2147483647">
          <security mode="None" />
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:7363/SilverlightApplication3Web/AuthenticationService.svc"
        binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_AuthenticationService"
        contract="ServiceReference2.AuthenticationService" name="BasicHttpBinding_AuthenticationService" />
    </client>
  </system.serviceModel>
</configuration>
```

Obrázek 4.7: Obsah ServiceReferences.ClientConfig souboru

Pokud uděláme nějakou změnu ve službě a pak obnovíme referenci na tuto službu, změny se automaticky promítnou i do souboru ServiceReferences.ClientConfig. Potíže které se vyskytli během implementace jsou popsány v kapitole 6. Pokud žádné potíže nenastaly můžeme službu začít používat v aplikaci(kap.4.4).

4.2.3 Cross-Domain access policy

Výše popsaný postup připojení služby, bude fungovat pokud je služba umístěna na stejném serveru jako aplikace(která službu používá). To však není ve většině případů pravda, proto je třeba doplnit aplikaci o soubor, který umožní pracovat se službou umístěnou na jiné doméně(cross-domain acces). Silverlight podporuje dva různé způsoby umožnění přístupu cross-domain.

- Vytvořit soubor *clientaccesspolicy.xml* a přidat jej na "páteř"domény, kde je hostována webová služba. Ten nastaví službu tak, aby bylo možné používat cross-domain přístup.
- Vytvořit soubor *crossdomain.xml* a přidat jej na "páteř"domény, kde je hostována webová služba. Formát tohoto souboru je podporován i Adobe Flashem.

Jsou to dvě zdánlivě stejná řešení, a proto vyvstává otázka, kterou z těchto dvou možností použít. Ačkoli *crossdomain.xml* používá i Adobe Flash, je zde omezení pro Silverlight. Aby byla služba přístupná musí být *crossdomain.xml* nastaven tak, že webová služba je zpřístupněna z jakékoli domény. Není-li soubor nastaven takto, Silverlight službu nerozezná.

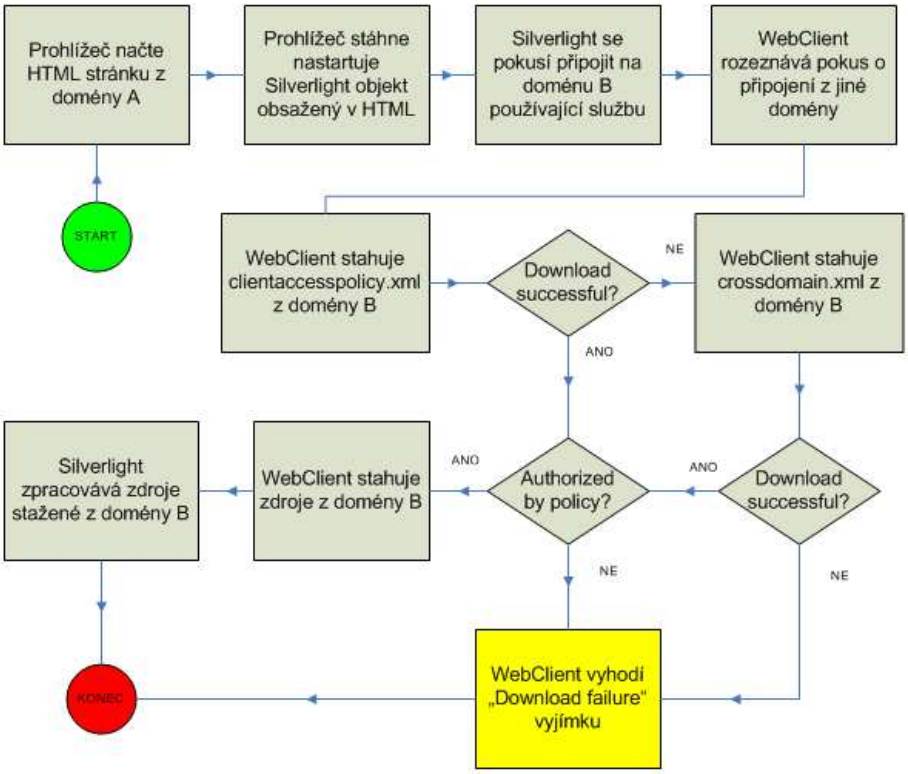
```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-http-request-headers-from domain="*" headers="*"/>
</cross-domain-policy>
```

Druhá možnost je opoznání flexibilnější. Soubor *clientaccesspolicy.xml* umožňuje specifikaci domén, které mají přístup ke službě. V ukázce obsahu *clientaccesspolicy.xml* je umožněn přístup pouze na volání z adresy *http://myApp.com*.

```
<?xml version="1.0" encoding="utf-8"?>
        <access-policy>
          <cross-domain-access>
            <policy>
              <allow-from http-request-headers="*">
                <domain uri="http://myApp.com"/>
              </allow-from>
              <grant-to>
                <resource path="/" include-subpaths="true"/>
              </grant-to>
            </policy>
          </cross-domain-access>
        </access-policy>
```

Když použijete oba soubory(umístíte je na "kořen"domény), klientská Silverlight aplikace zkontroluje nejprve *clientaccesspolicy.xml*, nastaví omezení na určité domény. Nezáleží na obsahu *crossdomain.xml*(ten bude použit klienskými aplikacemi FLASHe). Nejlepší odpovědí na otázku který způsob použít, je asi oba dva.

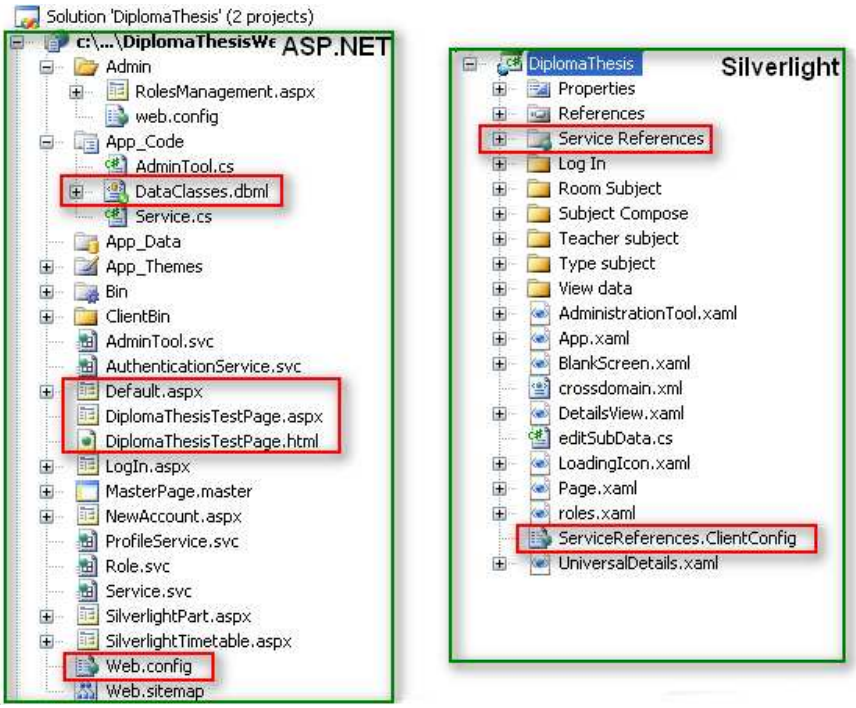
Co je třeba mít neustále na paměti je ukládání těchto dvou souborů(*clientaccesspolicy.xml*, *crossdomain.xml*). Musí být uloženy na kořen domény, ve které je služba hostována. Například pokud je služba na http://localhost:7363/SilverlightApplication3Web/ potom přístupové soubory musí být umístěny na adrese
http://localhost:7363/SilverlightApplication3Web/crossdomain.xml.



Obrázek 4.8: Průběh získávání přístupových informací (převzato z [13])

V obrázku 4.8 se vyskytuje pojem WebClient. WebClient je třída .NETu, obsahující metody pro posílání a přijímání dat ze zdroje identifikovaného určitým URI.

4.3 Přidání Silverlight aplikace na ASP.NET stránku



Obrázek 4.9: Solution explorer Silverlight aplikace

Na obrázku 4.9 je vidět příklad obsahu Silverlight aplikace(Solution explorer ve VS2008). Jde o jedno řešení(solution) obsahující dva projekty. Jak jsme zmiňovali v kapitole 4.1 SL projekty se obvykle vytváří spojené s ASP.NET stránkami, které SL aplikaci zapouzdřují. Na obrázku jsou vidět dvě části. ASP.NET a Silverlight část. ASP.NET stránka má tři potencionální vstupní kanály pro Silverlight. Jsou to

- Default.aspx
- DiplomaThesisTestPage.aspx
- DiplomaThesisTestPage.html.

Pro připojení SL do ASP.NET stránky jsou nezbytné dva tagy. První je `<asp:Silverlight>`. Druhým je AJAX `<asp:ScriptManager>`(), tak jak je vidět níže

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs" Inherits="Default2" %>
<%@ Register Assembly="System.Web.Silverlight" Namespace="System.Web.UI.SilverlightControls"
    TagPrefix="asp" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
</head>
<body>
    <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <div>
        <asp:Silverlight ID="Silverlight1" runat="server" Height="480px"
         MinimumVersion="2.0.31005.0"
            Source="~/ClientBin/DiplomaThesis.xap" Width="640px">
        </asp:Silverlight>
    </div>
    </form>
</body>
</html>
```

Text víše ukazuje způsob připojení SL aplikace do webové stránky.

4.4 Volání služby v kódu

Kvůli nejisté době odezvy serveru jsou všechny služby v Silverlightu 2 volány asynchronně. Získávání dat je pak operace o dvou krocích. Prvním je nastavení event handleru, který bude zavolán když je služba dokončena. Druhou operací je zavolání asynchronní metody služby.

```
private void testService()
{
    //vytvoření reference služby na referenci
    ServiceReference1.ServiceClient client = new ServiceReference1.ServiceClient();
    //event handler
    client.GetTeachersCompleted+=new EventHandler<GetTeachersCompletedEventArgs>(client_GetTeachersCompleted);
    //asynchronní volání služby
    client.GetTeachersAsync();
}

void client_GetTeachersCompleted(object sender, GetTeachersCompletedEventArgs e)
{
    //uložení tabulky teacher do proměnné teacherIn typu ObservableCollection<Teacher>
    ObservableCollection<Teacher> teacherIn = e.Result;
    teacher = teacherIn;
}
```

Obrázek 4.10: Volání služby v kódu

Asynchonní volání služeb v Silverlightu je velmi kontroverzní, občas způsobující memalé problémy(programátor neví kdy přijde odpověď). Proto je možné, že to bude jedna z

věcí které se v budoucnu změní. Ať je asynchronní přístup jakkoli nepopulární, je nejrozšířenější a je podporován širokým spektrem prohlížečů a operačních systémů. Také pomáhá při kontrole dostupnosti vlákna obsluhujícího uživatelské rozhraní(UI).UI vlákna se starají o překreslování obsahu oken tab.3.2). Abychom přidali ještě nějaká pozitiva, jakmile je použita služba s asynchronním voláním, kód pro její volání je čistý, lehce čitelný. Event handler má event args, které zahrnují i návratovou hodnotu(pokud metoda služby něco vrací) *e.Result* viz.obr.4.10.

4.5 Práce s daty

Data, získaná z databáze pomocí webové služby, jsme uložili podle obr.4.10 do proměnné teacherIn. Dejme tomu, že nyní budeme chtít vybrat všechny učitele, kteří mají křestní jméno začínající na I. Protože jsme použili LINQ class, můžeme se dotazovat jazykem LINQ. Podle kap.3.6 můžeme napsat dotaz:

```
var teacherI = from i in teacherIn
               select i.FirstName.StartsWith("I");
```

Jednoduchým zápisem jsme dostali seznam učitelů, jejichž křestní jména začínají na I. Stejným způsobem provádíme dotazy nad veškerými tabulkami obsaženými v DataClass.dbml souboru. Typ *var*, používaný k implicitní definici proměnných, je v tomto případě použit kvůli anonymitě výsledku získaného z dotazu. V příkladu výše je výsledkem jeden sloupec(FirstName) stringů(pole stringů) začínajících na I. Tedy zde není úplně nutný typ *var*, protože typ, získaný v dotazu, je známý. Může se však změnit málo a rázem je uplně jiný návrátový typ.

```
var teacherI = from i in teacherIn
               select new{i.firstName, i.lastName};
```

Nyní je proměnná teacherI anonymního typu a typ *var* je zde nezbytný. K implicitní deklaraci není *var* moc využíván, protože takovéto použití vnáší do kódu značný zmatek. Více o tématice práce s daty v Silverlightu naleznete například zde [2],[10].

Kapitola 5

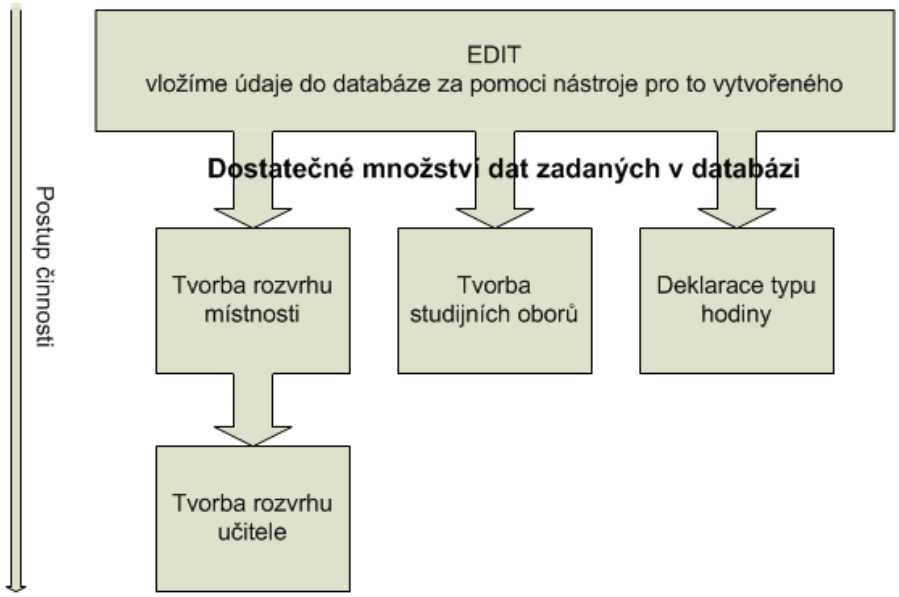
Popis programu pro tvorbu rozvrhu

Jedním z úkolů práce je vytvořit SL aplikaci pro tvorbu rozvrhu školy. Na této aplikaci byla vyzkoušena většina vlastností a možností Silverlightu. Výsledkem je aplikace popsaná v kap.5.3.

Jak je známo tvorba školního rozvrhu není jednoduchá věc.Využít Silverlightu k řešení takového problému je zajímavá alternativa. Byl zvolen protože jde o směr kterým se ubírají moderní aplikace. Tedy jsou přístupné na internetu, k jejich správné funkci je potřeba jednoduchý plug-in. Jsou podporovány velkým množstvím prohlížečů, ale hlavně poskytují funkčnost a možnosti podobající se desktopovým aplikacím.

5.1 Stručný popis

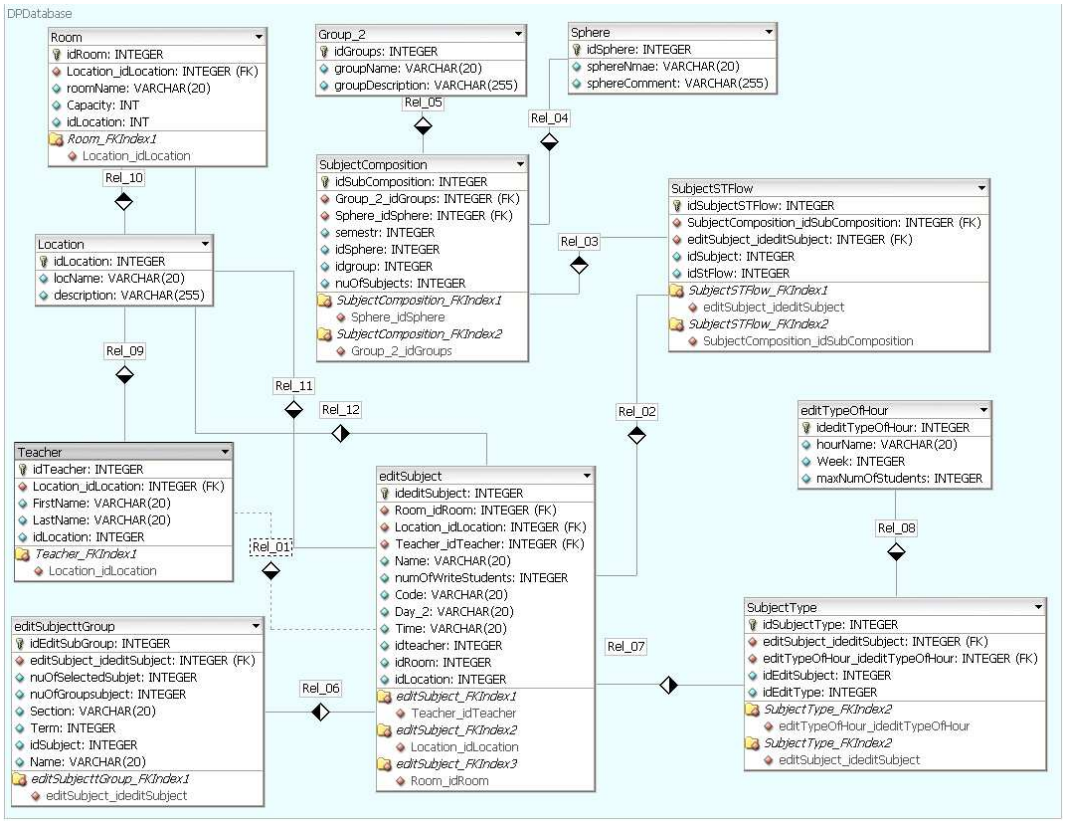
Co to znamená vlastně tvorba rozvrhu? Je to přidělování vlastností(učí kdy, učí se kde) jednotlivým entitám(učitel, předmět), zároveň hlídání kolizí podle pravidel vycházejících z koncepce rozvrhu školy(učitel nemůže učit zároveň dva předměty, atd...). Tedy pilířem je databáze, nad kterou postavime aplikaci. Co by měl takový systém umožňovat? Určitě vkládání jednotlivých entit(učitelů, předmětů), jejich mazání, určit má být co učeno, ... Tyto možnosti obstarávají jednotlivé Silverlight User Controls(je to jedna Silvelright "stránka") kap.5.2.2. Hlavní funkce aplikace jsou znárorněny v blokovém schématu.



Obrázek 5.1: Blokové schéma aplikace

5.1.1 Databáze

Dobrý návrh strukturované relační databáze je složitý. Při návrhu se provádí datová analýza, která se skládá z několika fází. V jejich průběhu se postupně vytváří stále konkrétnější schémata, která znázorňují požadovaná data. V první fázi se prostřednictvím konzultací s uživateli a zadavateli systému formulují a shromažďují přesné požadavky na to, co vše má být v databázi uloženo. Z takto získaných informací je vytvořen konceptuální model. Konceptuální datový model popisuje data v databázi na abstraktní úrovni nezávisle na fyzickém uložení. Výsledkem je konceptuální model znázorněný jako schéma nebo diagram, který co nejvýstižněji zachycuje pohled člověka na danou část reálného světa. Nejznámějším konceptuálním datovým modelem je ER(entity relationship) model. ER model k návrhu databáze byl použit i v tomto případě(obr.5.2).



Obrázek 5.2: ER model databáze

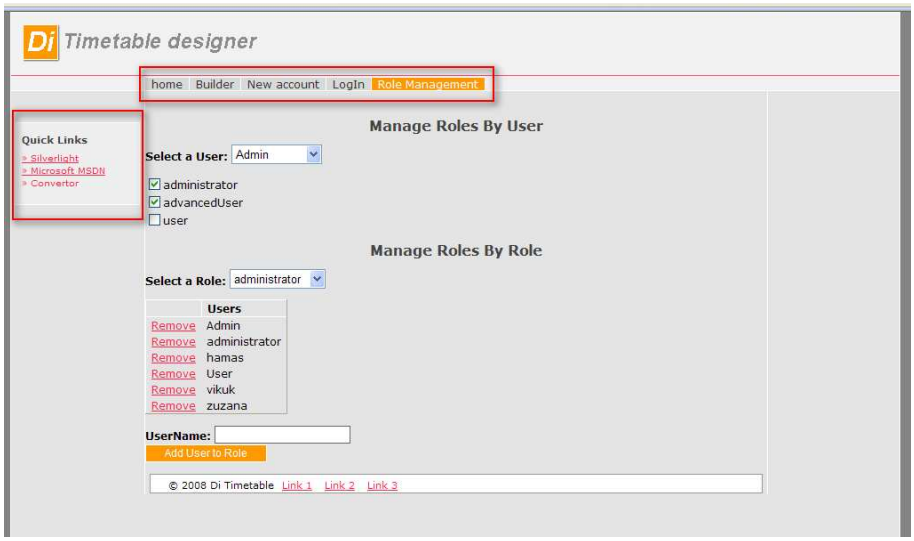
MS Sql databáze navržená podle modelu byla vytvořena v Microsoft SQL Server 2008 Express. Stručný popis jednotlivých entit(tabulek) je uveden níže. Vytvořená databáze se k Silverlight projektu připojí postupem popsaným v kap.4.2. Pár slov k funkci jednotlivých tabulek

- Room-místnosti ve kterých se může učit
- Group-skupina předmětů, jako je například povinné, povinně volitelné, atd...
- editTypeOfHour-typ hodiny, cvičení, přednáška, ...
- Location-umístění, atribut pro rozeznávání, např. Katedra Řízení
- editSubject-předměty
- Teacher-učitel
- Sphere-databáze oborů, což je například Řídící technika

- SubjectComposition-říká, kde všude se předmět vyučuje, například v oboru Řídící technika v 2. semestru je jeden z předmětů typu S Matematika pro Kybernetiku a měření.

5.2 Tvorba GUI

Grafické uživatelské rozhraní, je neméně důležitou částí a jeho tvorba by neměla být podceňována. Návrh GUI je samotným vědním oborem a má svá pravidla a zákonitosti (jednoduchost, přehlednost, nevtíravé, ...). Dobré uživatelské rozhraní je základ úspěšné aplikace, protože lidé nebudou používat něco co se používá špatně, nepohodlně nebo nestandartně.
GUI Silverlight aplikace je integrováno do ASP.NET stránky (obr.5.3). Není dobré, ani výhodné, vytvářet webové stránky založené pouze na Silverlightu, protože SL nepodporuje věci jako jsou tlačítko zpět v prohlížeči(stejně jako AJAX, Flash), neodchytává události myši tak jak jsme zvyklí(kap.6.4), ... Proto použití ASP.NET integrace.

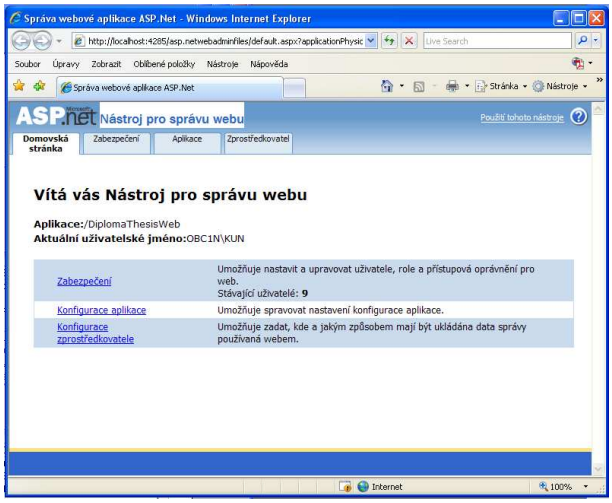


Obrázek 5.3: Vzhled GUI

Protože návrh rozvrhu, ve smyslu jeho tvorby, je distribuovaný problém(učitel nemůže vytvářet rozvrhy místností,...), je v aplikaci použit systém uživatelských rolí.

5.2.1 Administrace rolí

Používání uživatelských rolí v Silverlight aplikacích(pokud je hostována v ASP.NET stránce) znamená nastavení *ASP.NET Nástroje pro správu webu* obr.5.4. Ten najdeme ve Visual Studiu nabídce `Website > ASP.NET Configuration`.



Obrázek 5.4: ASP.NET nástroj pro správu webu

Zde v položce Zapezpečení nastavíme umožnění rolí. Vytvoříme uživatele a zařadíme je do rolí. Pro zpřístupnění ASP.NET Authentication systému v Silverlightu, připojíme WCF službu([2]). ASP.NET Authentication je proces získávání autorizačních informací uživatelů (přístup ano, ne). Tato služba bude pouze ukazovat na tu defaultní, tedy tu poskytovanou pro ASP.NET. Proto nemusí mít žádnou třídu s kódem v pozadí. Vytvoříme ji prostým přidáním textového souboru. Nazveme jej AuthenticationService.svc. obsahem bude jeden řádek

```
<%@ ServiceHost Language="C#" Service="System.Web.ApplicationServices.AuthenticationService" %>,
```

ten prováže službu s implementací používanou v ASP.NET. Jak jsme zmiňovali není zde žádný odkaz na soubor v pozadí. K nastavení služby patří ještě doplnění souboru Web.config o některé tagy.

```
<System.web>
...
<!-- umožnění forms authentication nastavením z Windows na Forms -->
<authentication mode="Forms" />
...
</System.web>
<!-- přidání system.web.extensions umožnění autentifikaci
     být přístupná přes webService  -->
<system.web.extensions>
    <scripting>
      <webServices>
        <authenticationService enabled="true" requireSSL="false"/>
      </webServices>
    </scripting>
  </system.web.extensions>

<system.serviceModel>
    <services>
      <!-- this enables the WCF AuthenticationService endpoint -->
      <service name="System.Web.ApplicationServices.AuthenticationService"
               behaviorConfiguration="AuthenticationServiceTypeBehaviors">
        <endpoint contract="System.Web.ApplicationServices.AuthenticationService"
                  binding="basicHttpBinding" bindingConfiguration="userHttp"
                  bindingNamespace="http://asp.net/ApplicationServices/v200"/>
      </service>

    </services>
    <bindings>
      <basicHttpBinding>
        <binding name="userHttp">
          <!-- this is for demo only. Https/Transport security is recommended -->
          <security mode="None"/>
        </binding>
      </basicHttpBinding>
    </bindings>
    <behaviors>
      <serviceBehaviors>
        <behavior name="AuthenticationServiceTypeBehaviors">
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <!-- this is needed since this service is only supported with HTTP protocol -->
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>
  </system.serviceModel>
```

Připojení reference na službu je shodné s postupem popsaným v kap.4.2.2. Pokud nastane chyba při připojování je třeba znovu překontrolovat všechny části nastavení. Připojíme-li službu k naší aplikaci je možné používat metody AuthenticationServicu. Jaké to jsou se dozvíte například v [2].

Pro práci s detaily uživatele(správa účtu) je třeba přidat službu s názvem ProfileService.

Přidáme opět textový soubor, přejmenujeme jej na ProfileService.svc s obsahem

```
<%@ ServiceHost Language="C#" Service="System.Web.ApplicationServices.ProfileService" %>.
```

Stejně jako v předchozím případě, je ruční dopsání některých tagů do Web.config nezbytné.

```
...
<service name="System.Web.ApplicationServices.ProfileService"
               behaviorConfiguration="ProfileServiceTypeBehaviors">
        <endpoint contract="System.Web.ApplicationServices.ProfileService"
                  binding="basicHttpBinding" bindingConfiguration="userHttp"
                  bindingNamespace="http://asp.net/ApplicationServices/v200"/>
</service>
....
<behavior name="ProfileServiceTypeBehaviors">
          <serviceMetadata httpGetEnabled="true"/>
</behavior>
```

Umístíme tagy na stejné místo jako u předchozí služby. Poslední službou, potřebnou k plnohodnotné práci s rolemi, je RoleService, neboli správa rolí. Defaultně není uživatel zařazený do žádné role. Přidáním uživatele do role, nebo více rolí dostáváme lepší možnost administrovat aplikaci. Podle role, do které uživatel patří, se k němu aplikace "chová". Jak tedy připojit službu pro správu rolí? Postup je analogický. Tagy, pro tuto službu, přidávané do Web.configu mají takovouto podobu.

```
....
<service behaviorConfiguration="AppServicesBehavior" name="System.Web.ApplicationServices.RoleService">
      <endpoint binding="basicHttpBinding" bindingConfiguration="userHttp"
        bindingNamespace="http://asp.net/ApplicationServices/v200"
         contract="System.Web.ApplicationServices.RoleService" />
    </service>
....
<behavior name="AppServicesBehavior">
        <serviceMetadata httpGetEnabled="true" />
</behavior>
...
```

Obsah služby RoleService.svc je stejný jako u předešlých dvou. Liší se pouze atributem `Service=System.Web.ApplicationServices.RoleService`. Jsou-li všechny tři služby připojeny, dostáváme možnost obsluhovat uživatele, jejich role přes obsahy těchto služeb(jejich Contract).

Standartní možností, ve většině aplikací, u přihlašovacích formulářů, je možnost pamatování si uživatelského jména, popřípadě hesla. V Silverlightu můžeme tuto funkčnost zavést díky IsolatedStorage. IsolatedStorage by se dal přirovnat ke Cookies známým z HTML. Toto izolované úložiště je jediné místo kam může Silverlight aplikace ukládat

vytvořené soubory. Soubory jsou oddělené podle uživatele a webových stránek ve kterých byly uloženy. Velikost úložiště je omezena na $1MB$. Jak takové úložiště používat je vidět na obr.5.5.

```
//VYTVOŘENÍ
using (IsolatedStorageFile store = IsolatedStorageFile.GetUserStoreForApplication())
{   //vytvoření streamu pro vytvoření souboru
    using (IsolatedStorageFileStream strm = new IsolatedStorageFileStream("MyOwnFile.txt", FileMode.Create, store))
    {
        using (StreamWriter sw = new StreamWriter(strm))
        {   //zapis dat do souboru
            sw.Write("You can write some text over here");
        }
    }
}
//ČTENÍ
using (IsolatedStorageFile store = IsolatedStorageFile.GetUserStoreForApplication())
{   //otevření proudu pro čtení
    using (IsolatedStorageFileStream fs = new IsolatedStorageFileStream("MyOwnFile.txt", FileMode.Open, store))
    {
        using (StreamReader sr = new StreamReader(fs))
        {
            String txt = sr.ReadLine();
        }
    }
}
```

Obrázek 5.5: Použití IsolatedStorage

Přihlašovací stránka, která je také tou uvodní v aplikaci, vypadá takto obr.5.6. Pokud jsou přihlašovací údaje správné, a uživatel je zařazen do některé z rolí, naskytne se mu možnost vybrat si z nabídky akcí, které by chtěl dělat. Nabídka je na obr.5.7.

5.2.2 Navigace v Silverlightu

Navigací rozumíme přepínání mezi jednotlivými částmi aplikace. Tyto části aplikace jsou reprezentovány jednotlivými Silverlight User Controly. User Control je šablona z nabídky Visual Studia, která pomocí .NETu umožňuje tvorbu Silverlight aplikací. Pak výběr jakékoli položky z nabídky znamená zavolat UserControl, která se zobrazí na místě té stávající. V Silverlightu neexistuje něco jako je odkaz ve smyslu přesměrování na jinou HTML(aplikační) stránku, je nutné používat mechanismus přidávání a odtraňování jednotlivých stránek(UserControl). Jak již bylo řečeno, layout stránky je deklarován v jazyce XAML. Potom když vytvoříme nový Silverlight UserControl, například s názvem newControl s následujícím obsahem.

```
<UserControl x:Class="SilverlightApplication1.newControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Name="txtb1" Text="newControl" HorizontalAlignment="Center"
                   VerticalAlignment="Center" FontSize="20"/>
    </Grid>
</UserControl>
```

Máme v projektu dvě UserControly(newControl a tu z obr.3.6 nazvanou SilverlightControl2). Nyní je možné zobrazit newControl na místě SilverlightControl2(přepnout jejich zobrazení). Tlačítko `button1` v SilverlightControl2 obsluhuje událost Click. Po kliknutí na něj je vyvolána metoda `button1_Click`. Tu obslouží metoda obsažená v kódu v pozadí(pro C♯ soubor SilverlightControl2.cs). Přepnutí mezi dvěma UserControly je popsáno v textu níže.

```
private void button1_Click(object sender, RoutedEventArgs e)
        {
            //vytvoreni instance tridy newControl
            newControl newcontrol = new newControl();
            //pridani newcontrol na misto SilverlightControl2
            this.LayoutRoot.Children.Add(newcontrol);
        }
```

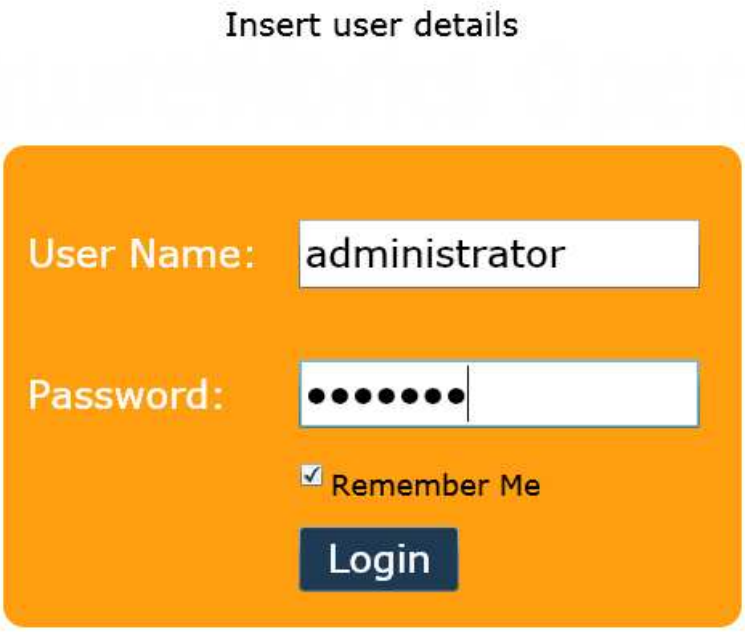
Po kliknutí na tlačítko se tedy zobrazí obrazovka s nápisem *newControl* uprostřed. Zpět na původní zobrazení(to z 3.4.1) se dostaneme podobně. Místo metody Add použijeme Remove s opačnými parametry.

V této fázi máme obecný přehled o tom jak se tvoří takové grafické rozhraní v Silverlightu. Hlavním stavebním kamenem je Silverlight UserControl(logický prvek zapouzdřující další logické prvky). Tyto prvky provazujeme do sebe, tím vytváříme aplikaci.

5.3 Vlastní aplikace

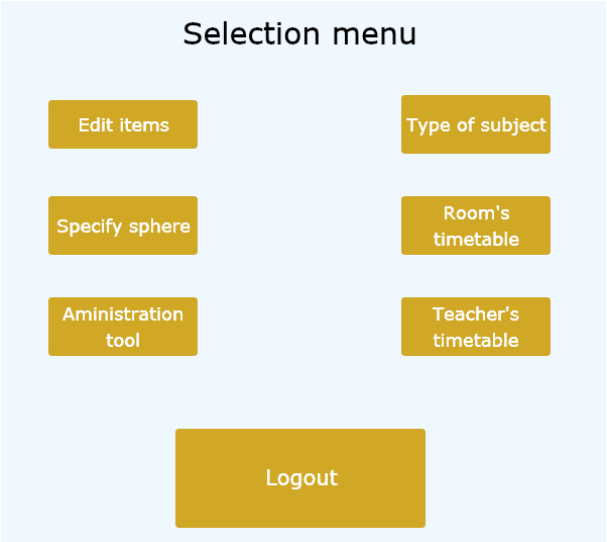
5.3.1 Editace dat

Popis vlastní aplikace jsme tu již nastínili. Je zapouzdřena v ASP.NET stránce (obr.5.3), používá role pro přístup. Tedy pokud se úspěšně zalogujeme viz.obr.5.6, jsme zařazeni v některé z rolí, zobrazí se nám výběr(menu) položek.



Obrázek 5.6: Uvodní stránka aplikace pro tvorbu rozvrhu

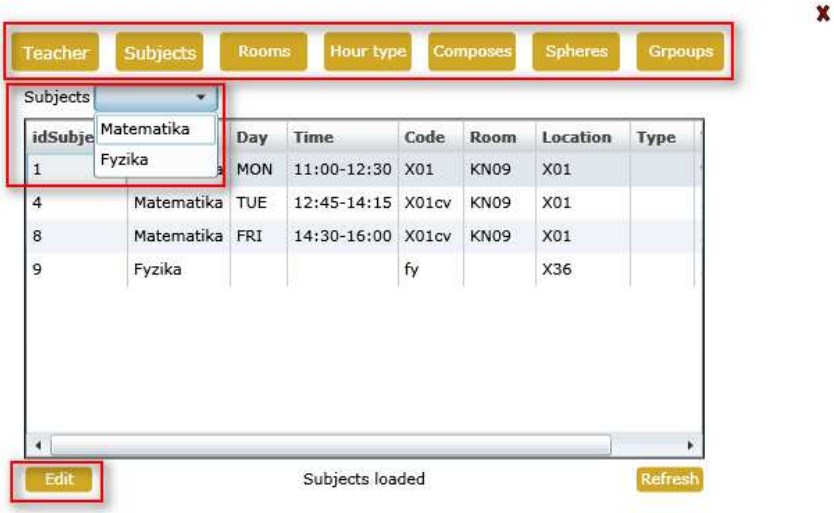
Asi tou nejočekávanější možností aplikace je editace jednotlivých tabulek, ta je pod políčkem menu Edit items(obr.5.7).



Obrázek 5.7: Hlavní menu aplikace

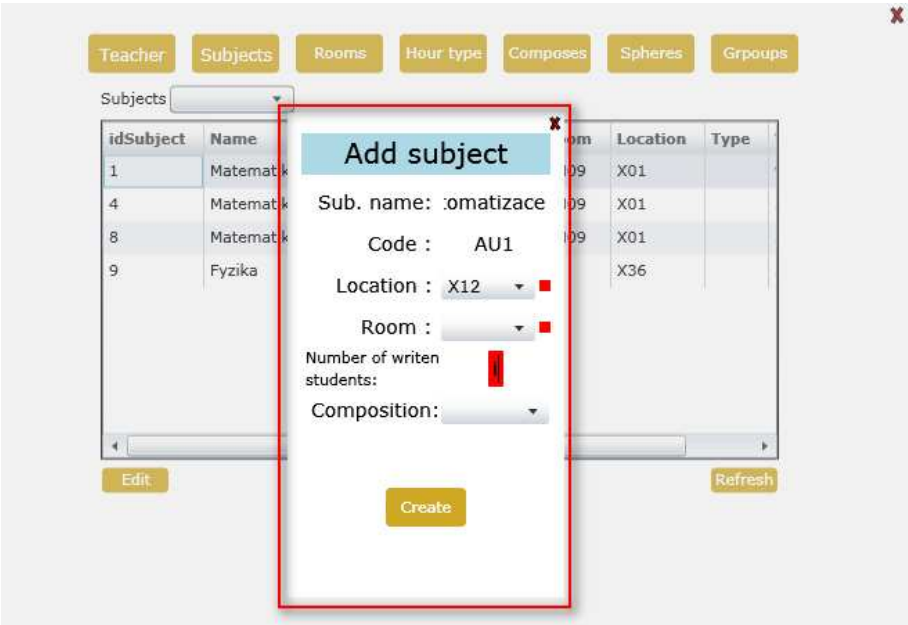
Při potřebě vkládání nebo umazání stávajích údajů(nedají se editovat, kvůli následné

změně mnoha údajů, třeba i počtu předmětů) zvolíme tabulku kterou chceme editovat. Tabulky vybíráme pomocí tlačítek v horní části dialogového okna(viz.obr.5.8).



Obrázek 5.8: Funkce odkazu View

Obsah tabuek je možné filtrovat a řadit vzestupně a sestupně jednotlivé sloupce. Pro přidání nového záznamu zvolíme Edit, vlevo dole.

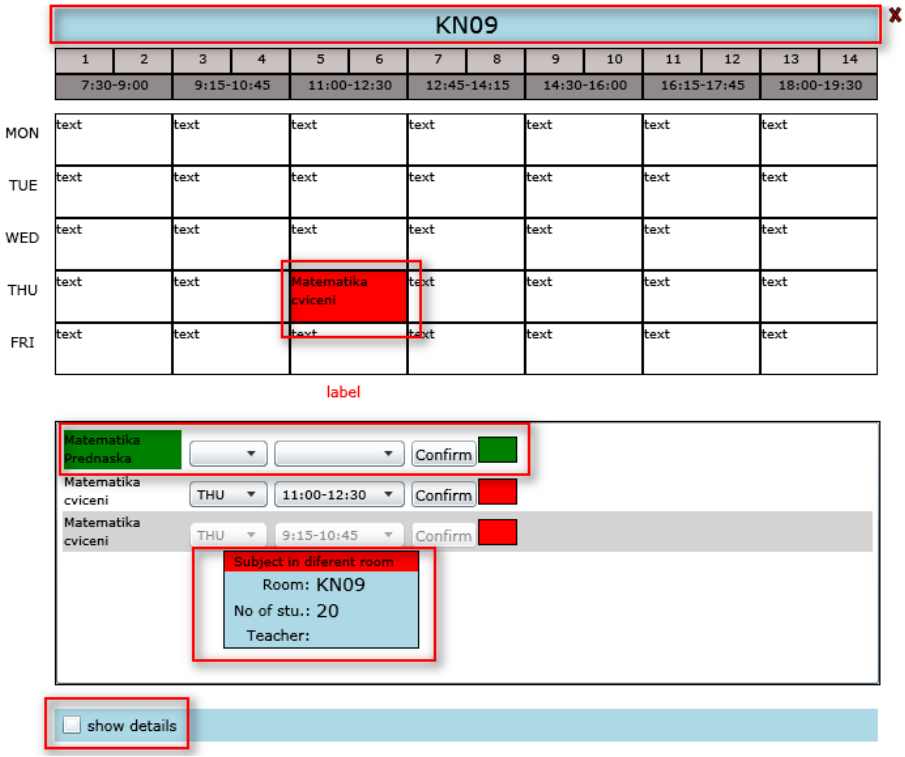


Obrázek 5.9: Přidávání nových záznamů

Tím se otevře dialogové okno pro vkládání položek do tabulky. Vkládání je bezpečné. Aplikace nenechá uživatele vložit neplatný údaj(5.9). Pro vymazání záznamu jej stačí vybrat v tabulce a stisknout klávesu Delete. Po obou akcích(smazání, přidání záznamu) se tabulka a filtry obnoví podle aktuální situace. Pro ruční obnovu dat v zobrazovaných tabulkách slouží tlačítko Refresh.

5.3.2 Rozvrh místnosti

Další operací je přiřazení předmětu do místnosti. Tedy tvorba rozvrhu místnostu. Tuto volbu umožňuje dialogové okno na obr.5.10.



Obrázek 5.10: Přiřazení předmětů místnostem

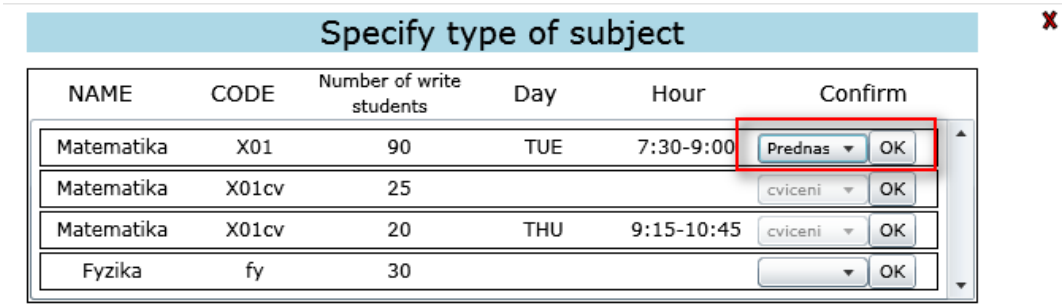
Dialogové okno vyvoláme kliknutím na příslušnou položku menu, následně se na obrazovce objeví seznam místností, jednu z nich vybereme. Podle kapacity místnosti se nám zobrazí jen ty předměty, které mají počet studentů menší a patří do stejné lokace jako vybraná místnost. Pak dostaneme například takový scénář jako je na obr.5.10. Ve vrchní části je název místnosti(KN09). V prostřední části rozvrh této místnosti s červeně zvýrazněnými, již zapsanými předměty. Ve spodní části je seznam dostupných předmětů.

Spodní část zobrazuje předměty ve třech různých stavech. První(zelený) je předmět ještě nezařazený do žádné místnosti. Předmět s šerveným označením a možností editovat čas a den v Combo boxech je umístěn v aktuální místnosti(na obr.5.10 vprostřed). Posledním je ten předmět, který svými vlastnosti vyhovuje této místnosti, ale již byl dříve umístěn do jiné. Zde není umožněna editace a barevný příznak je taktéž červený. Při najetí kurzorem myši nad předmět z jiné místnosti se zobrazí jeho detaily(viz.obr.5.10). Pokud chceme přepnout na editaci místnosti, ve které je předmět umístěn, stačí stlačit tlačítko myši kdekoli nad ním.

K zapisování předmětů do místnosti máme opět dvě možnosti. První je výběr dne a času konání předmětu pomocí comboboxů a potrzení tlačítkem confirm. Druhou alternativou je metoda drag and drop. "Chytneme"předmět(jeho zelený název) a přetáhneme jej nad kolonku rozvrhu. I na změnu doby výuky předmětu lze aplikovat stejné dva způsoby.

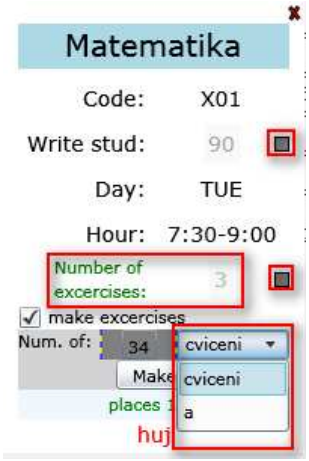
5.3.3 Typ předmětu

Pro specifikaci typu předmětu, tedy jde-li o cvičení, přednášku nebo něco jiného, je připraven nástroj na obr.5.11.



Obrázek 5.11: Určení druhu předmětu

Zde vybereme typ předmětu z nabídky a potvrdíme tlačítkem. Pokud jsme vybrali jako typ přednášku, potvrdíme volbu tlačítkem, zobrazí se okno pro editaci cvičení stejné jako na obr.5.12.

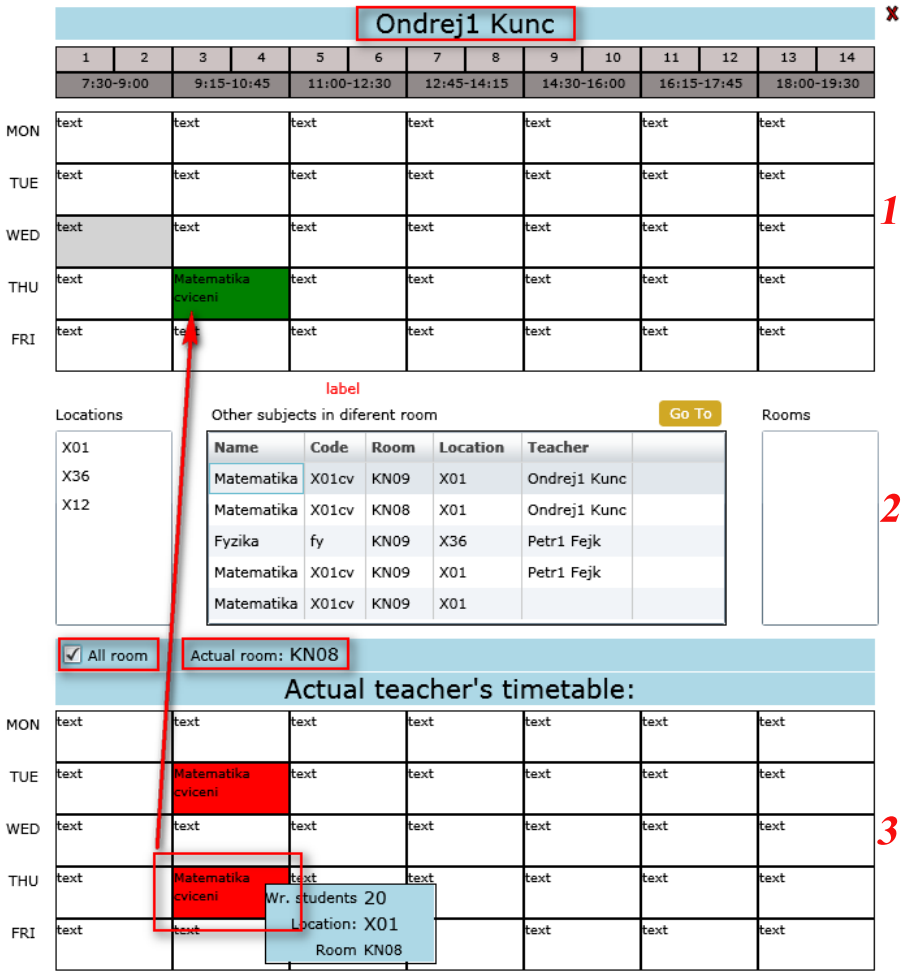


Obrázek 5.12: Vytvoření cvičení

Pokud zaškrtneme možnost Make Excercise, zadáme počet studentů v jednotlivých hodinách a vybereme typ hodiny, dialogové okno nám umožní vytvořit cvičení nebo jakýkoli jiný typ předmětů(laboratoře, počítačová cvičení) propojený s danou přednáškou. Při potvrzení dialogového okna se vytvoří právě tolik cvičení(nebo jiných typů předmětu), kolik je třeba. Počet generovaných předmětů se odvíjí od počtu studentů navštěvujících přednášku a zadáním počtu studentů na cvičení.

5.3.4 Rozvrh učitele

Tvorba rozvrhu učitele, to je další dialogové okno a zároveň možnost editace rozvrhu na obr.5.13.



Obrázek 5.13: Tvorba rozvrhu učitele

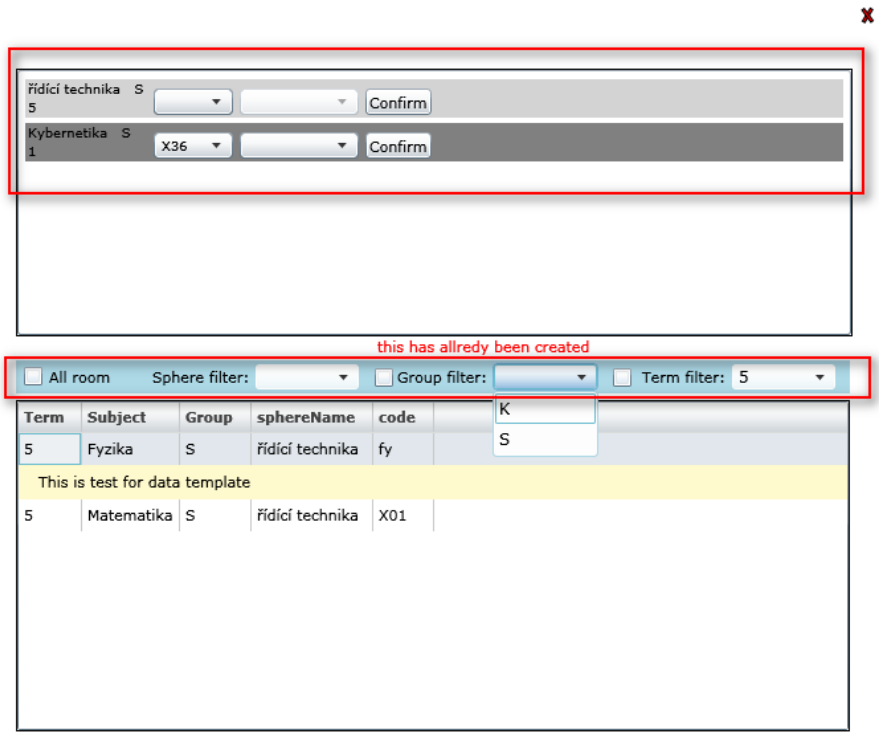
Opět, stejně jako u kap.5.3.2 je okno vyvoláno výběrem učitele ze seznamu. První zobrazení respekntuje zařazení učitele do skupiny(například X01) a nabídne na výběr pouze předměty, zařazenéve stejné skupině. Pracovní plocha je rozdělena na tři části(na obr.5.13 popsány čísly v pravo). První část(číslo 1 vpravo) ukazuje rozvrh vybrané místnosti. Druhá, prostřední část pak obsahuje výběrová data. Poslední, třetí část zobrazuje rozvrh učitele.

Začali bychom popisem prostřední(2) části. Ta zobrazuje tři tabulky. První(Locations) je seznam lokací, druhá je seznam předmětů a informací o nich. Třetí je seznam místností z dané lokace. Dále zde vidíme CheckBox All Room. Jeho zaškrtnutím zahrneme do tabulek všechny předměty ve všech lokacích(viz.obr.5.13). Nic nám již nebrání ve výběru jakýchkoli předmětů.

Přiřazení předmětu učiteli je možné v první části stránky. Vybereme místnost, zobrazíme její rozvrh(předměty nepřidělené učiteli jsou červené). Klikneme-li na předmět, který chceme aby učil vybraný učitel, předmět se mu hned zapíše. Předměty obsazené jinými kantory se nezobrazují. Chceme-li přejít na tvorbu rozvrhu jiného učitele, vybereme jeden z jeho předmětů ze seznamu v prostřední části a potvrdíme tlačítkem Go To.

5.3.5 Tvorba oborů

Předměty snad na každé škole spadají do různých oborů a skupin. V návrhovém systému kromě toho, že jsou ve skupinách (X01, X36, ...) a skupinách předmětů(povinně volitelné-S, oborové-L, ...), je další členění a to podle oboru v kterém se učí. Tedy Matematika ze skupiny X01 bude vyučována v oboru Řídící technika v 5.semestru skupině předmětů S. Rozřazování je umožněno dialogovým oknem obr.5.14.



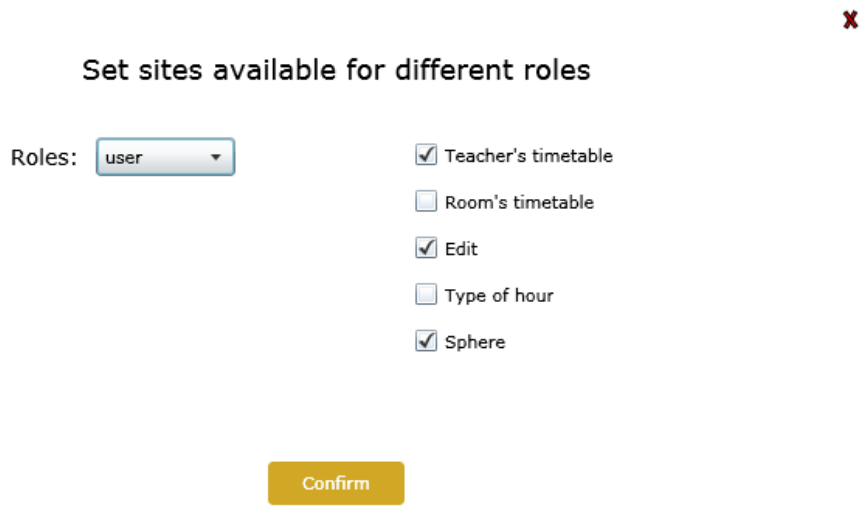
Obrázek 5.14: Tvorba oborů

Funkčnost je opět podobná předchozím dialogovým oknům a předměty se do oborů

zařazují pomocí ComboBoxů. Opět zde platí, že aplikace nás nenechá udělat žádný nedovolený krok(duplicitní údaje, předmětů je v oboru již moc, ...). Je možné filtrovat data a to buď jednotlivě, nebo pomocí CheckBoxů filtry "svázat".

5.3.6 Nastavení přístupu

K nastavení přístupu k jednotlivým nástrojům byl vytvořen administrační formulář. Tento formulář je přístupný pouze administrátorům. Těm umožňuje nastavovat přístupy ke stránkám pomocí rolí.



Obrázek 5.15: Nástroj pro administraci přístupu k funkcím aplikace

Kapitola 6

Postřehy a problémy zaznamenané při tvorbě programu

Při tvorbě programu jsem narážel na mnoho chyb způsobených chybami v Silverlightu. To se ustálilo až s příchodem konečné verze Silverlight 2. Chyby tohoto charakteru zde neuvádím. Jsou zde uvedené pouze ty hlavní problémy, nezpůsobené nevyladěním technologie.

6.1 Potíže při přístupu k webové službě

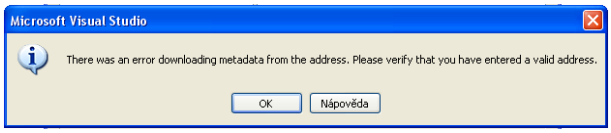
Pakliže se Silverlight snaží získat webovou službu na jiném serveru než se nachází aplikace, potýkáme se s cross-domain problémem. Může zde dojít k mnoha různým potížím. Tou nejčastější je ta na obr.6.1, tedy server nenalezen.

Obrázek 6.1: Chyba při přísrupu k webové službě

Tato chyba je nejčastěji vyvolána nepřítomností nebo chybou v souborech pro přístup ke službě(*clientaccesspolicy.xml*, *crossdomain.xml*). Soubory musí být správně nastaveny a uloženy na správném místě, tedy na kořeni domény kde je služba(viz.4.2.3)

6.2 Připojování reference na službu

Při připojování reference na službu dochází často k chybám. Tou nejčastější je asi ta odpovídající obr.6.2.

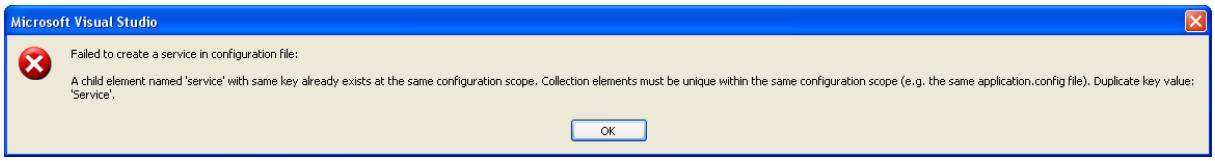


Obrázek 6.2: Chyba při připojování reference na webovou službu

To znamená, že služba není dostupná, VS 2008 nemůže stáhnout metadata potřebná k jejímu vybudování. Může jí způsobit mnoho různých věcí. Co na většinou pomůže je restart Visual Studia. Pak je dobré zkontrolovat obsah souboru web.config, příslušejícího k dané aplikaci. V něm musí být správně definované tagy `<behavior>` a `<service>`. Část souboru Web.config, deklarující připojovanou službu, může vypadat takto.

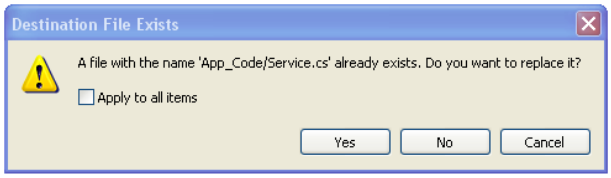
```
...
<system.serviceModel>
  <behaviors>
   <serviceBehaviors>
     <behavior name="AppServicesBehavior">
     <serviceMetadata httpGetEnabled="true"/>
     </behavior>
   </serviceBehaviors>
  </behaviors>
  <services>
    <service behaviorConfiguration="AppServicesBehavior"
    name="System.Web.ApplicationServices.RoleService">
      <endpoint binding="basicHttpBinding" bindingConfiguration="userHttp"
        bindingNamespace="http://asp.net/ApplicationServices/v200"
        contract="System.Web.ApplicationServices.RoleService"/>
    </service>
</services>
...
```

Další možnou chybou je ta na obr.6.3 ta nastane, když se snažíme připojit službu, jejíž název se shoduje s názvem služby již někdy připojené.



Obrázek 6.3: Chyba při připojování služby

Při rušení služby jsme zapomněli vymazat deklaraci služby v souboru Web.config. Stačí jen umazat nódy `<behavior>`, `<service>` z Web.config, příslušející nepoužívané službě.

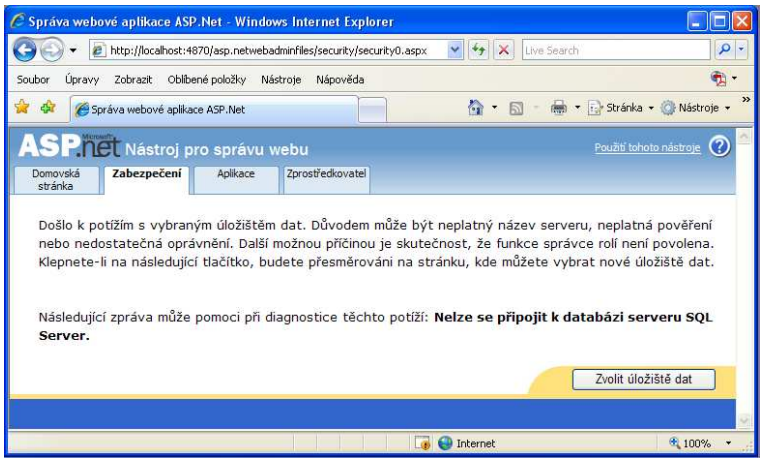


Obrázek 6.4: Chyba při připojování reference na webovou službu

Pokud jsou splněny všechny víše zmíněné podmínky, s připojením služby by neměl být žádný problém.

6.3 Problémy s ASP.NET configurátorem

Chcete-li administrovat vaši ASP.NET stránku ASP.NET konfiguračním nástrojem a při jeho vyvolání vyvstane chyba stejná jako na obr.6.5, znamená to absenci SQL databáze.



Obrázek 6.5: ASP.NET Configuration chyba

V normálním případě se při prvním otevření ASP.NET konfigurátoru k aplikaci do složky `App_Data` připojí databáze s názvem ASPNETDB.MDF. V našem případě ji musíme připojit ručně. To znamená ji buď stáhnout z webu Microsoftu, nebo získat nějakým jiným způsobem a přes její connectionString, který přidáme do souboru Web.config, ji připojit. Pak stačí restartovat ASP.NET konfigurátor a vše funguje jak má.

6.4 Odchytávání událostí myši

Zdá se to být zvláštní, ale je to tak. Silvelright nepodporuje obsluhu událostí myši jako jsou MouseDoubleClick, MouseMiddleButton, MouseRightButton. Událostmi, které můžeme použít při programování SL aplikace, jsou ty pro levé tlačítko myši, a samozřejmě obsluha událostí klásnice. Rada na to jak tento problém obejít není.

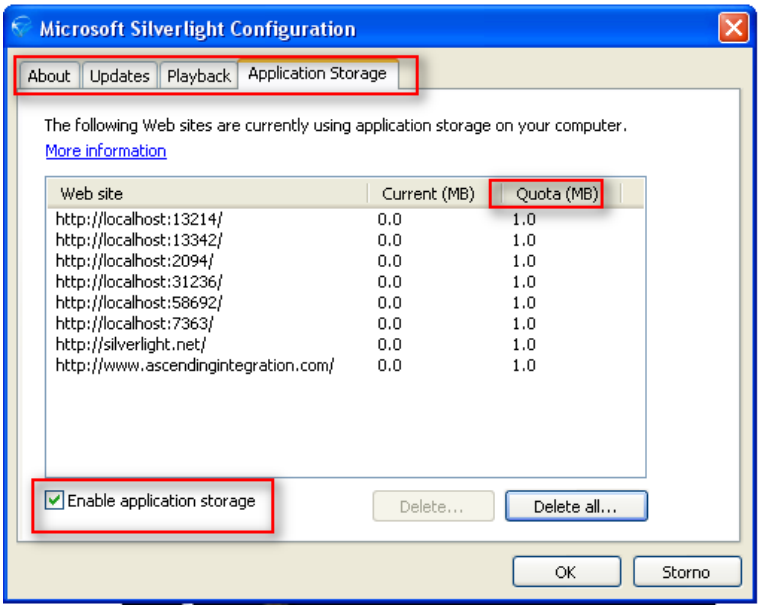
A důvod absence těchto událostí? Je to přenositelnost. Bylo to vždy tak, že to co fungovalo v IE nebo FireFoxu nefungovalo v Mac-Safari, tak nezbývalo než se zaměřit na tyto druhy prohlížečů a kód upravit na míru každému zvlášť. To je silná stránka Silverlightu, opravdová přenositelnost. Tlačítka myši zde hrají velkou roli. Donedávna platilo, že Mac měl tradičně jedno tlačítko myši. To se automaticky pokládá za to levé

na myších používaných ve Windows. Nové Macy mají ovládací zařízení s více než jedním tlačítkem, avšak s minulostí se musí počítat. A jak do toho zapadá double click? To má původ v přenositelnosti Silvelrightu na mobilní zařízení kde je dvouklik problém.

Silverlight je velmi komplexní technologie a na to musíme myslet při implementaci aplikací v něm.

6.5 Silverlight configuration

Jak jsme mluvili o nemožnosti používání pravého tlačítka myši v Silverlightu neřekli jsme co se stane použijeme-li ho. Použitím pravého tlačítka myši u jakékoli SL aplikace vyvoláme nástroj pro správu Silverlightu.



Obrázek 6.6: Nástroj pro správu SL aplikace

Tento nástroj umožňuje nastavení chování Silverlightu na klientském počítači. Je zde možné nastavovat druh aktualizací pro Silverlight, umožnění ukládání dat do Isolated-Storage.

Odpovědi na hodně otázek okolo Silverlightu lze najít například v [4].

Kapitola 7

Závěr

V rámci této diplomové práce byla vyzkoušena nová technologie Microsoftu pro tvorbu RIA aplikací, Silverlight. Úkolem práce bylo se s technologií Silverlight seznámit, popsat její možnosti, klady a zápory. Výsledky byly porovnány s největším konkurentem, Adobe Flash. Silverlight komunikuje s databázemi pomocí webových služeb. V rámci práce byly vyzkoušeny přístupy k různým datovým zdrojům pomocí webových služeb. Všechny tyto technologie a poznatky byly použity při tvorbě pilotního projektu, kterým byla zvolena aplikace pro tvorbu rozvrhu.

Vytvořený program umožňuje tvorbu rozvrhu školy od základního zadávání položek do databáze, až po nastavování přístupu k aplikaci podle rolí uživatelů. Vše bylo integrováno do ASP.NET stránky a vlastní nástroj dostal grafické užatelské rozhraní, založené na rolích uživatelů. Na základě požadavků na aplikaci byla vytvořena relační databáze uchovávající data aplikace. Vytvořené rozhraní umožňuje úplnou tvorbu rozvrhu a je dobrým základem pro vývoj rozvrhového systému.

Práce se Silverlightem, z programátorského hlediska, nebyla vždy na výbornou. Při tvorbě rozhraní jsem narážel neustále na problémy s technologií Silverlight. Je třeba říci, že během tvorby této práce se aktuální, nejnovější, verze Silverlightu několikrát změnila. Až v posledním uvolnění, na podzim roku 2008, byla k dispozici konečná verze(Silverlight 2), tedy verze bez větších chyb.

Literatura

[1] *ASP.NET* 2.0 *a C♯ profesionálně*. Matthew MacDonald, Mario Szpuszta, 2006.

[2] *Microsoft Developer Network MSDN*. dostupný na `http://msdn.microsoft.com/en-us`.

[3] *W3C konsorcium*. dostupné na `www.w3.org/`.

[4] *Oficiální stránky Silverlightu*, dostupné na `http://silverlight.net/`.

[5] *Blog Scotta Guthrie věnovaný Silverlightu*, dostupné na `http://weblogs.asp.net/scottgu/archive/2007/05/07/silverlight.aspx`.

[6] *Návod na nastavení Visual Studia pro implementaci Silverlight aplikací*, dostupné na `http://silverlight.net/GetStarted/`.

[7] *Blog s příspěvky o vývoji webových aplikací*, dostupné na `http://www.scottklarr.com`.

[8] *Integrace AJAXu do ASP.NET*, dostupné na `http://www.asp.net/ajax/`.

[9] Hana Kozelková, *Bakalářská práce:Editor ER diagramů s podporou převodu do relačního modelu*, Praha: UK MAT-FYZ, 2007.

[10] *Blog Scotta Guthrie věnovaný LINQ*, dostupné na `http://weblogs.asp.net/scottgu/archive/2007/05/19/using-linq-to-sql-part-1.asp`.

[11] *Oficiální stránky Microsoft WCF*, dostupné na `http://msdn.microsoft.com/en-us/netframework/aa663324.aspx`.

[12] *Oficiální stránky Microsoft WPF*, dostupné na `http://msdn.microsoft.com/en-us/netframework/aa663326.aspx`.

[13] *Portál s programátorskými tématy*, dostupné na `http://community.dynamics.com/blogs`.

Příloha A

Seznam použitého softwaru

- Microsoft Windows XP©
- Microsoft Visual Studio Team System 2008
- Microsoft Expression Blend 2.5 Beta
- Microsoft SQL Server 2008 Express
- Microsoft Office Visio 2003
- MikTex 2.7
- WinEdt 5
- FastStone Capture
- BSR Screen Capture
- Adobe Photoshop CS

Příloha B

Obsah přiloženého CD

- **Zdrojové kódy**: Zdrojové kódy vytvořené aplikace pro Silverlight.
- **Databáze**: Databáze potřebné pro chod aplikace.
- **Dokumentace** : Tento dokument ve formátu PDF, zdrojové kódy pro LaTeX.
- **Video** : Videozáznam o použití aplikace.