

# **Implementace podpory rozvrhování výuky v prostředí Microsoft Excel**

Implementation support education scheduling in Microsoft Excel

Radek Los

---

Bakalářská práce  
2011



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Radek LOS**  
Osobní číslo: **A08067**  
Studijní program: **B 3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**

Téma práce: **Implementace podpory rozvrhování výuky v prostředí Microsoft Excel.**

Zásady pro vypracování:

1. Popište problematiku vytváření doplňků pro Microsoft Excel.
2. Analyzujte možnosti implementace v prostředí Microsoft Excel.
3. Navrhněte vlastní implementaci.
4. Vytvořte uživatelské rozhraní.
5. Demonstrujte ukázkovou aplikaci.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Bustamante, Leroux Michele. Learning WCF. First edition. United State of America : O'Reilly, 2007. str. 582.**
2. **Albahari, Joseph a Albahari, Ben. LINQ Pocket Reference. First edition. Canada : O'Reilly, 2008. str. 164.**
3. **Henney, Kevlin. 97 Things Every Programmer Should Know. First edition. United States of America : O'Reilly, 2010. str. 229.**
4. **Beaulieu, Alan. Learning SQL. Second Edition. United States of America : O'Reilly, 2009. str. 320.**
5. **Griffith, Ian, Adams, Matthew a Liberty, Jesse. Programming C 4.0: Building Windows, Web, and RIA Applications for the .NET. Sixth Edition. United State of America : O'Reilly, 2010. str. 830.**

Vedoucí bakalářské práce:

**Ing. Erik Král**

Ústav bezpečnostního inženýrství

Datum zadání bakalářské práce:

**25. února 2011**

Termín odevzdání bakalářské práce:

**7. června 2011**

Ve Zlíně dne 25. února 2011



prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

Cílem práce bylo vytvoření doplňku pro aplikaci Microsoft Excel pro podporu rozvrhování výuky. Doplňek slouží jako klient webové služby, která vrací data nutná pro zobrazení karty vyučujícího a karty oddělení. Doplňek je určen pro sekretářky, tajemníky a vedoucí ústavu.

Klíčová slova: WCF, webová služba, C#, Microsoft Excel add-in, WinForm, Rozvrhování výuky, LINQ, Entity Framework, SQL

## **ABSTRACT**

Aim of the thesis is to create Microsoft Excel add-in for implementation support lecture scheduling. The add-in is web service client which returns required data for displaying teacher and department view. The add-in is intended to be used by secretaries and heads of department.

Keywords: WCF, Web service, C#, Microsoft Excel add-in, WinForm, Lecture scheduling, LINQ, Entity Framework, SQL

Tímto bych chtěl poděkovat vedoucímu práce Ing. et Ing. Eriku Královi za odborné vedení a konzultace bakalářské práce. Dále bych chtěl poděkovat Petru Čápkovi za vytvoření databáze.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 WEBOVÁ SLUŽBA</b> .....	<b>11</b>
1.1 PRINCIP SLUŽBY .....	11
1.2 SIMPLE OBJECT ACCESS PROTOCOL (SOAP) .....	12
<b>2 WINDOWS COMMUNICATION FOUNDATION (WCF)</b> .....	<b>13</b>
2.1.1 Chyby a jejich zpracování.....	16
2.1.2 Ukázková aplikace .....	17
2.1.3 Vytvoření služby .....	18
2.2 PROVOZOVÁNÍ WEBOVÉ SLUŽBY .....	20
2.3 KLIENT .....	20
2.3.1 Přidání služby do projektu .....	20
2.3.2 Programová část .....	23
2.3.2.1 Synchronní volání .....	23
2.3.2.2 Asynchronní volání.....	23
<b>3 EXCEL ADD-IN</b> .....	<b>24</b>
3.1 VYTVOŘENÍ RIBBONU .....	24
3.2 PROGRAMOVÁNÍ V MICROSOFT EXCELU .....	25
3.2.1 Ukázková aplikace .....	25
<b>4 DATABÁZE A DOTAZOVACÍ JAZYKY</b> .....	<b>27</b>
4.1 LINQ .....	27
4.1.1 IEnumerable<T> .....	27
4.1.2 LINQ to Object .....	28
4.1.3 LINQ to XML .....	29
4.1.4 LINQ to Entity Framework.....	29
4.1.4.1 Přidání Entity do projektu.....	30
<b>5 UŽIVATELSKÉ ROZHRANÍ</b> .....	<b>34</b>
5.1 VÝMĚNA DAT MEZI FORMULÁŘI .....	34
5.1.1 Zpracování dat modálních formulářů.....	34
5.1.2 Zpracování dat nemedálních formulářů .....	35
<b>II PRAKTICKÁ ČÁST</b> .....	<b>36</b>
<b>6 CÍL APLIKACE</b> .....	<b>37</b>
6.1 KARTA VYUČUJÍCÍHO .....	37
6.2 KARTA ÚSTAVU.....	38
<b>7 VYTVOŘENÍ SLUŽBY</b> .....	<b>39</b>
7.1 ZÁKLADNÍ METODY.....	39
7.2 VYUČUJÍCÍ.....	40
<b>8 EXCEL ADD-IN</b> .....	<b>43</b>

---

8.1	DIALOG VYUČJÍCÍ .....	43
8.2	VYTVOŘENÍ OVLÁDACÍCH PRVKŮ.....	45
8.3	AUTORIZACE UŽIVATELE.....	47
<b>9</b>	<b>VYTVOŘENÁ APLIKACE .....</b>	<b>49</b>
	<b>ZÁVĚR .....</b>	<b>52</b>
	<b>CONCLUSION .....</b>	<b>53</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>54</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>55</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>56</b>
	<b>SEZNAM TABULEK.....</b>	<b>57</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>58</b>

## ÚVOD

Cílem práce je implementace podpory rozvrhování výuky v prostředí Microsoft Excel. V práci je popsáno vytvoření doplňku pro Microsoft Excel 2007, který umí generovat pohledové sestavy. Práce navazuje na aplikaci vytvořenou v platformě jazyka Silverlight [12]. Doplněk je schopen vytvářet karty vyučujícího a zobrazení rozvrhovaných předmětů, bez přiřazených místností. V těchto kartách se získávají informace o jednotlivých ústavech a jejich předmětech. Aplikace je určena pro učitele, sekretářky, tajemníky a vedoucí ústavu pro vygenerování karet bez možnosti editace.

Práce je rozdělena na část teoretickou a praktickou. V první části se zabývám popisem tvorby WCF služeb [1] v programovacím jazyce C#. Při vytváření služby se věnuji její konfiguraci, možnosti fungování na různých protokolech, volání na klientovy služby a odchytávání výjimek a synchronní a asynchronní zpracování služby na straně klienta. Také se věnuji dotazovacím jazykům SQL [9] a jeho nahrazení za LINQ [5] pomocí Entity Frameworku [2]. V teoretické části se také zabývám vytvářením doplňku pro Microsoft Excel 2007 konkrétně vytvoření Ribbonu (záložka v pásu karet) [2] a uživatelským rozhraním WinForm.

V praktické části se zabývám vytvořením vlastního doplňku pro Microsoft Excel zobrazujícím karty vyučujícího a ústavu. Popisuji zde praktické vytvoření webové služby přístupující k databázové vrstvě pomocí Entity Framework s využitím LINQ. Jako klientskou aplikaci používám Microsoft Excel, který pomocí doplňku a jeho ovládacích prvků zobrazuje jednotlivé karty do sešitu aplikace.

## **I. TEORETICKÁ ČÁST**

## 1 WEBOVÁ SLUŽBA

Webová služba je softwarový systém, umožňující komunikaci dvou zařízení na síti, založený na nezávislých standardech XML a HTML. Je popsán ve formátu WSDL [2]. Ostatní zařízení komunikují způsobem popsáním v popisu služby pomocí SOAP [2] protokolu.

Služby komunikují pomocí otevřeného standartu XML, tím se komunikace značně usnadňuje.

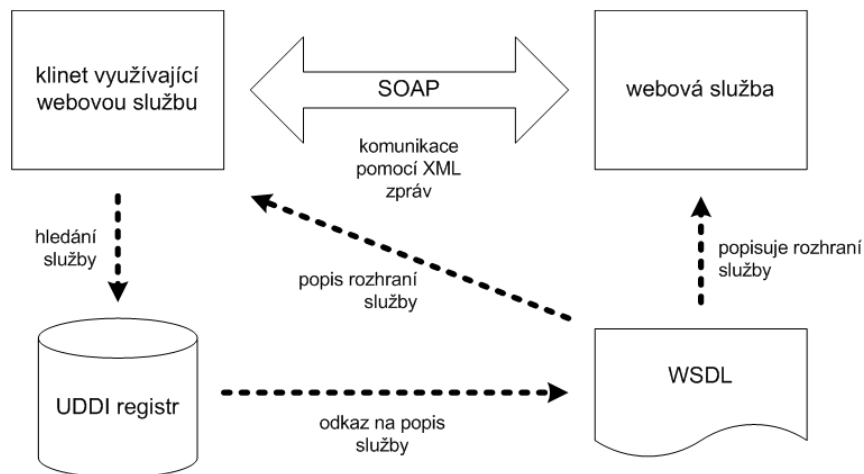
Tato technologie umožňuje sjednotit libovolné aplikace z různých platforem a ovládat je přes internetovou aplikaci (www stránku), desktopovou aplikaci, mobilní aplikaci, atd. Mohou být napsány v libovolném jazyce (dovolujícím vytvořit službu) a volat v jakémkoliv jiném jazyce (dovolujícím webové služby, nebo SOAP). Služba tedy dovoluje platformě nezávislý zdroj dat mezi serverem a klientem.

Další výhoda služeb je komunikace přes port 80 (HTTP), popřípadě 443 (HTTPS), které nejsou většinou blokovány firewallem. HTTPS podporuje bezpečné šifrování tzv. tunel, tudíž není možný odposlech přenesených dat (útok – man in the middle). [1] [2] [3]

### 1.1 Princip služby

Každá webová služba se skládá ze tří částí:

- Poskytovatel služeb – zařízení poskytující službu a její běh.
- Registr služby (UDDI) – uloženy informace o webových službách a jejich poskytovatelích.
- Klient – aplikace využívající službu



Obrázek 1: Princip služby [10]

Klient se odkáže na UDDI registr, kterým se vyhledá WSDL popis – v programovém kódu se z WSDL načtou definice metod – parametry a návratové hodnoty. Po nalezení popisu klient komunikuje pouze se službou prostřednictvím SOAP a XML zpráv. [2] [10]

## 1.2 Simple Object Access Protocol (SOAP)

SOAP je přenos dat přes HTTP/HTTPS ve formě XML zpráv.

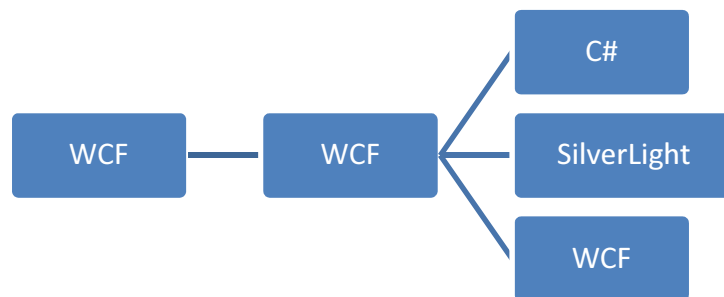
- Data se přenáší jako řetězec (platformní nezávislost)
- XML dokáže přenášet i složitější struktury dat (celé objekty, kolekce objektů)
- Možnost definovat jmenné prostory pro omezení kolize definice objektů

SOAP byl původně vytvořenou společností Microsoft v roce 1998, ale začaly se objevovat podobné protokoly od jiných výrobců. Konsorcium W3C začlenilo SOAP do svého programu v roce 2000 vytvořením verze 1.2. SOAP měl původně přenášet zprávy pouze přes protokol HTTP, postupně byla přidána podpora SMTP. [1] [2]

## 2 WINDOWS COMMUNICATION FOUNDATION (WCF)

WCF služba je část platformy .NET, která poskytuje jednotný programovací model pro tvorbu vzdáleně volaných aplikací.

Služby mohou být lokální (v rámci jednoho počítače), nebo vzdálené (volané přes síť). Aplikace, které využívají služby, se nazývají klienti a mohou být psány v jakémkoliv jiném jazyce platformy .NET. Služby mohou volat jiné služby. Ke službám lze přistupovat z různých klientů, například je možné vytvořit Silverlight aplikaci a připojit ji vzdáleně k WCF a ke stejné službě může přistupovat desktop aplikace pro Windows vytvořená v platformě .NET.



Obrázek 1: Ukázka možnosti volání služby

Služby tedy nabízejí abstraktní objektový model pro práci s daty. Práce v kódu se službami je stejná jako s třídami. Zavoláme potřebnou metodu a vrácená data zpracujeme podle potřeby. Nemusíme vědět, jestli jsou data uložena na stejném serveru jako služba, nebo jestli se načítají z SQL, XML nebo jakých jiných zdrojů. Pouze nám stačí vědět jakou službu volat a jakými metodami dostaneme potřebná data. [1] [2] [3]

Visual Studio 2010 nabízí možnost vytvořit [3] [6]:

- WCF aplikaci – hostována na IIS serveru
- WCF knihovna – dynamická knihovna (*dll* soubor), který se může nahrát do aplikace a pracovat s ní

### Adresy

Každá služba je asociována unikátní adresou. Adresa ukazuje, kde se služba nachází a jaký používá protokol pro komunikaci. WCF podporují tyto protokoly:

- HTTP
- TCP
- IPC

- MSMQ
- Peer network

Adresy mají vždycky následující formát [1]:

`[protokol]://[server nebo doména][:port]/[URI]`

## Contract

Každá služba vystavuje tzv. contract. Contract je způsob služby jak definovat a popsat, co služba dělá a jaké ke své činnosti používá metody. [1]

## ServiceContract

Popisuje, jaké operace klient může provádět na službě. Označují se atributem, který se přidá před název třídy, nebo rozhraní (`ServiceContract`). Jednotlivé operace (metody) se označují atributem (`OperationContract`). Můžeme také definovat jmenný prostor v `ServiceContract` atributu, kterým můžeme omezit kolizi názvů. [1]

```
[ServiceContract (Name="IServicePortal")]
public interface IService
{
    [OperationContract (Name="NejakaOperace")]
    string getData();
}
```

## DataContracts

Definuje, jaká data prochází službou. Výchozí datové typy jako `int`, `string` ... jsou definovány implicitně, uživatelské datové typy a třídy se musí definovat explicitně atributem `DataContract` před názvem třídy (rozhráním), strukturou, výčtovým typem. Data se poté označují `DataMember`. [1]

```
[DataContract]
public class ScheduleSubject
{
    [DataMember]
    public string name;
}
```

## Fault contracts

Definuje, jaké chyby mohou nastat ve službě a jak je zachytit a odeslat na klienta. [1]

## Message contracts

Dovoluje službě komunikovat se zprávami. [1]

## Binding

Definují, jakým způsobem *endpoint* komunikuje se světem. *Binding* definuje protokoly a kódování (textové nebo binární). Mohou obsahovat elementy pro specifikaci, jakým způsobem se zprávy mohou šifrovat. [1] [2]

- Basic binding (*BasicTcpBinding*) – pro starší verze klientů, aby mohli komunikovat s novými službami
- TCP binding (*NetTcpBinding*) – používá se pro cross-machine na intranetu. Podporuje celou řadu funkcí, včetně spolehlivosti, protokolů a bezpečnosti a je optimalizován pro WCF-WCF komunikaci
- IPC binding (*NetNamedPipeBinding*) – pro přenos na jednom PC. Nejvíce bezpečná služba, protože nemůže být volána z jiného počítače.
- Web service (WS) binding (*WSHttpBinding*) – používají *HTTP* nebo *HTTPS* protokol a jsou navrženy pro komunikaci přes Internet.
- Federated WS binding (*WSFederationHttpBinding*) – WS binding, nabízející podporu pro federalizované zabezpečení.
- Duple WS binding (*WSDualHttpBinding*) – WS binding, podporující dvousměrnou komunikaci ze služby ke klientovi.
- MSMQ binding (*NetMsmqBinding*) – nabízí podporu pro odpojené dávkové volání.
- MSMQ integration binding (*MsmqIntegrationBinding*) – obousměrný převod WCF na MSNMQ zprávy. [1]

## Formátování a kódování

Každý ze standardů binding používá jiný protokol a jiné kódování (Tabulka 1).

Tabulka 1: Přehled kódování a protokolů služby [1] [2] [6]

<b>Binding</b>	<b>Protokol</b>	<b>Kódování</b>
<i>BasicHttpBinding</i>	<i>HTTPS/HTTP</i>	Text
<i>WSHttpBinding</i>	<i>HTTPS/HTTP</i>	Text
<i>WSDualHttpBinding</i>	<i>HTTP</i>	Text
<i>WSFederationHttpBinding</i>	<i>HTTPS/HTTP</i>	Text
<i>NetTcpBinding</i>	<i>TCP</i>	Binární
<i>NetNamedPipeBinding</i>	<i>IPC</i>	Binární
<i>NetMsmqBinding</i>	<i>MSMQ</i>	Binární
<i>NetPeerTcpBinding</i>	<i>P2P</i>	Binární
<i>MsmqIntegrationBinding</i>	<i>MSMQ</i>	Binární

### 2.1.1 Chyby a jejich zpracování

Ve službě mohou nastat kdykoliv neočekávané chyby. Otázkou je, jak tyto chyby odchytit a vrátit je klientovi. V normálních aplikacích se chyby odchyťávají pomocí bloků try-catch-finally). V místě kódů, kde může nastat chyba, se zabalí do **try**, který v případě chyby zavolá automaticky **catch**. **Finally** se volá vždy po skončení buď **try** (kód prošel bez chyb), nebo **catch** (nastala nějaká chyba).

WCF funguje jinak, protože musí zajistit funkčnost v každé klientské aplikaci, aby mohl zachytit vrácenou chybu. Proto služby musí převést normální .NET výjimky jako SOAP zprávu, které bude klient rozumět a zpracuje ji jako chybu.

Zatímco služba nespadne během vracení chyby, klient při nezachycené výjimce může ztratit připojení na proxy (nebo kanál) na službu. Přesný vliv výjimky na klienta zaleží na stupni režimu. [1] [2]

Na straně klienta mohou nastat celkem 3 druhy chyb, když se pokouší vyvolat službu. První typ je chyba komunikace, jako je dostupnost sítě, špatná adresa, hostitelský proces neběží, ... Komunikační výjimka volá **CommunicationException**.

Druhý typ chyb klienta se vztahuje na stav proxy a kanály, jako je pokus o navázání spojení na již uzavřené proxy, což je `ObjectDisposedException`, nebo také nesoulad v contractu a binding zabezpečení.

Třetím a posledním typem chyb jsou chyby na straně služby, tedy služba sama vrátí `Exception`. V zájmu zapouzdření a oddělení, se výjimky umísťují na stranu služby, ta je poté vrací jako `FaultException`.

`FaultException` je přesně definovaná standard, který je nezávislý na jakékoliv technologii – CLR, Java nebo C++ – SOAP faults. [1]

### 2.1.2 Ukázková aplikace

Ukázková aplikace bude demonstrovat vracení výjimky v případě dělení nulou. V definici `OperationContract` se vloží parametr signalizující vracení `FaultContract` přesně definovaného typu.

```
[OperationContract]
[FaultContract(typeof(DivideByZeroException))]
double Divide(double number1, double number2);
```

Pokud se `number2` rovná nule, vrátí se `FaultException`:

```
public double Divide(double number1, double number2)
{
    if (number2 == 0) {
        DivideByZeroException exception = new DivideByZeroException();
        throw new FaultException<DivideByZeroException>(exception, "number2 is zero");
    }

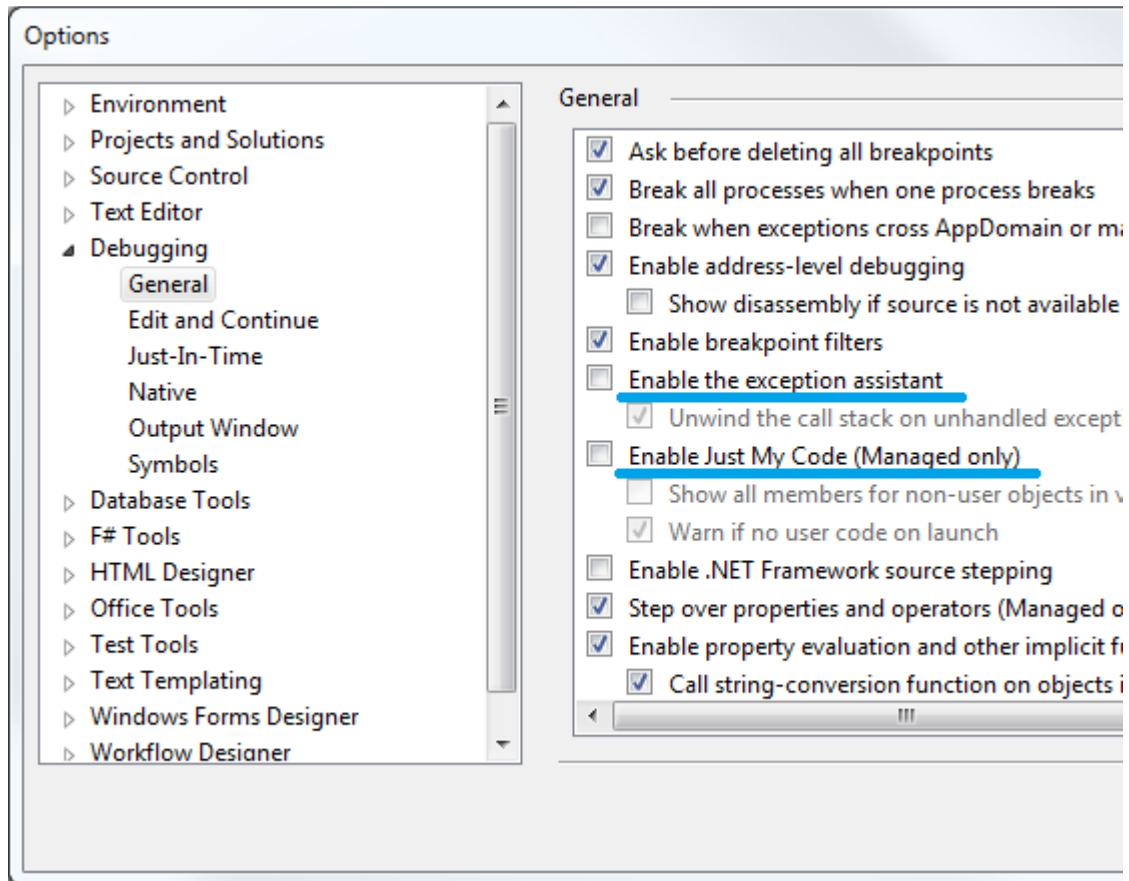
    return number1 / number2;
}
```

Na straně klienta se výjimky ošetří standardním způsobem try-catch-finally

```
string result = "";

try {
    Service.ServiceClient s = new Service.ServiceClient();
    result = s.Divide(1, 0).ToString();
} catch (DivideByZeroException ex) {
    result = ex.Message;
} catch (CommunicationException ex) {
    result = "Chyba připojení";
} catch (ObjectDisposedException ex) {
    result = "Chyba připojení (proxy)";
} finally {
    MessageBox.Show(result);
}
```

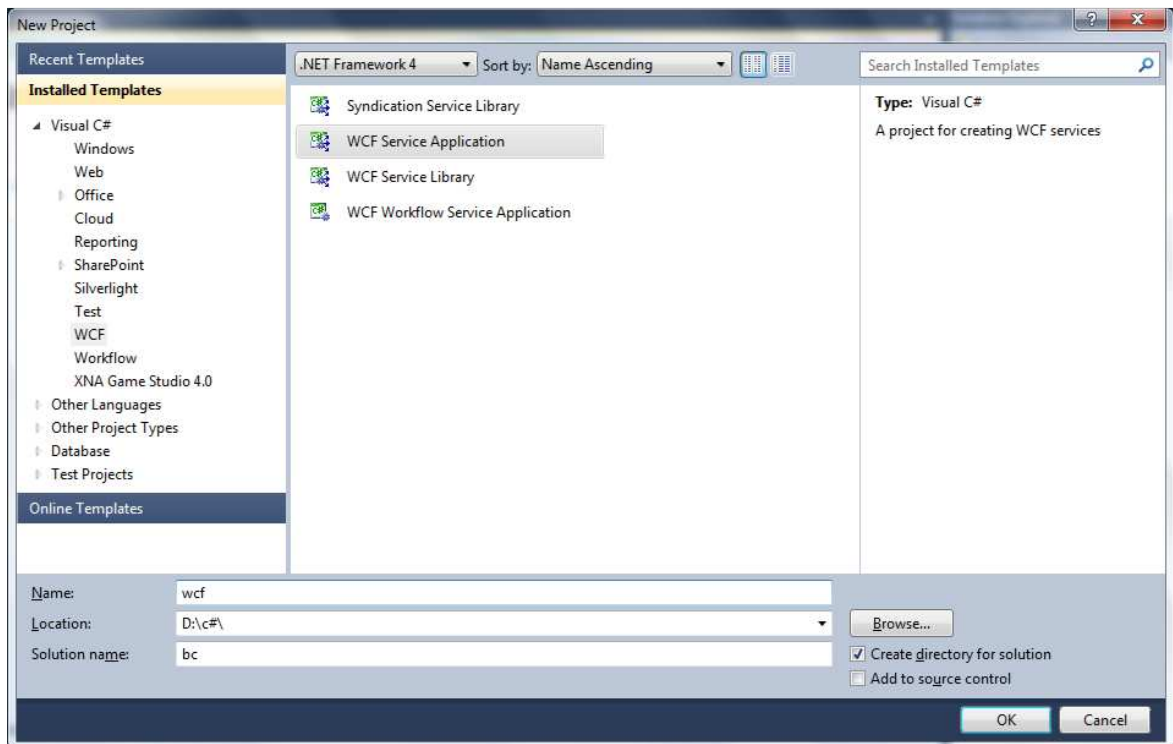
Během testování funkčnosti kódu mi Visual Studio 2010 nevracelo výjimku přes `throw` a odchytil ji sám jako: `System.ServiceModel.FaultException`1 was unhandled by user code`. V tomto případě se musí změnit nastavení v `Tools → Options` a odškrtnout dvě možnosti, viz Obrázek 2.



Obrázek 2: Změna nastavení pro správné vracení výjimek [2]

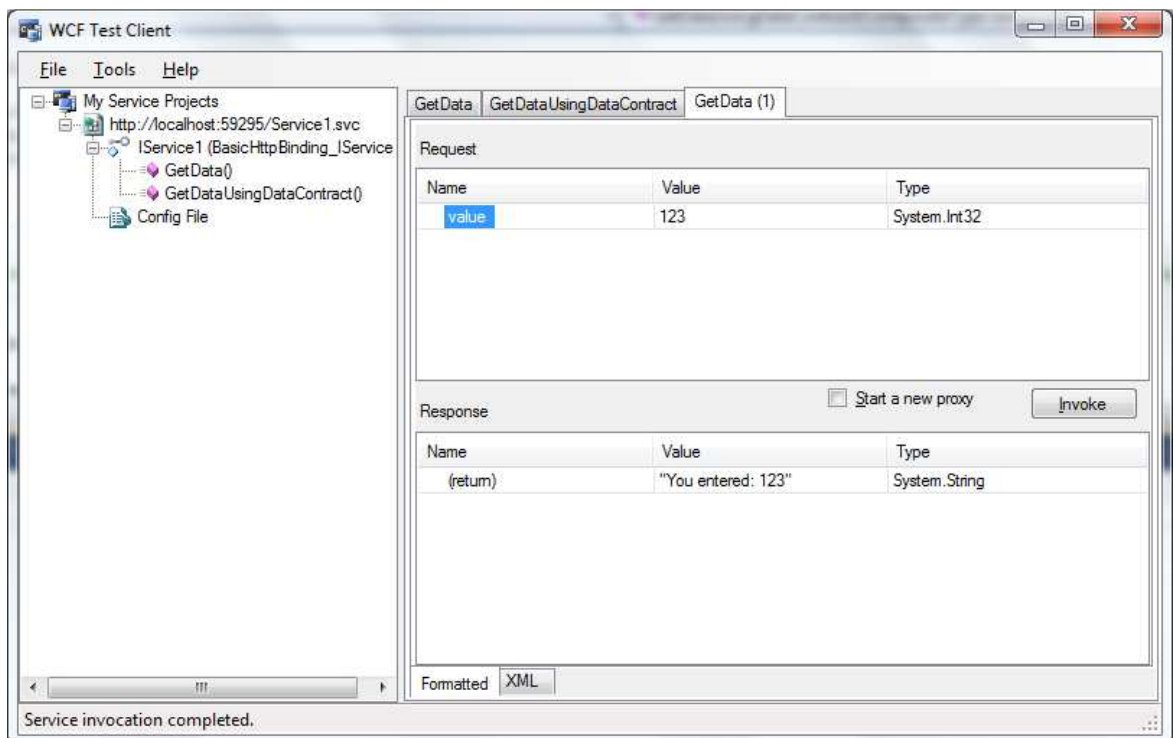
### 2.1.3 Vytvoření služby

WCF služby se vytváří především v programu Microsoft Visual Studio. Dále bude popsán postup vytvoření služby ve vývojovém prostředí Visual Studio 2010. Jako první krok se musí vytvořit nový projekt (`File → New → Project`). V zobrazeném dialogovém okně se vybere šablona `WCF → WCF Service Application` a zvolí název projektu a služby. [2]



Obrázek 3: Vytvoření nové služby

Po spuštění služby se otevře dialogové okno WCF Test Client, ve kterém můžeme volat metody služby. Pokud testovací klient vypadá stejně jako na obrázku „Obrázek 4“, služba se vytvořila správně.



Obrázek 4: Testovací klient

Dialogové okno WCF Test Client má na levé straně seznam dostupných metod, v pravé části se pracuje s daty. Pravá strana je rozdělena na horní polovinu – parametry metody a na spodní polovinu – návratovou hodnotu. Tlačítkem *Invoke* se dá metoda vyvolat, na spodním řádku můžeme přepnout zobrazení vrácených dat – *Formatted* – stromová struktura, nebo *XML*.

V testovacím klientu je vidět rozhraní *IService*, které nabízí dvě metody. Pro jakoukoliv aplikaci je lepší toto rozhraní vymazat a vytvořit si vlastní rozhraní, pojmenované podle účelu pro snadnější orientaci. Soubory *IService1.cs* a *Service1.svc* je vhodné vymazat. [2]

## 2.2 Provozování webové služby

WCF mají výhodu při jejich provozování a dostupnosti služby, zatímco webové služby mohou být hostovány pouze na webovém serveru, u WCF máme na výběr z několika možností. [3] [6]

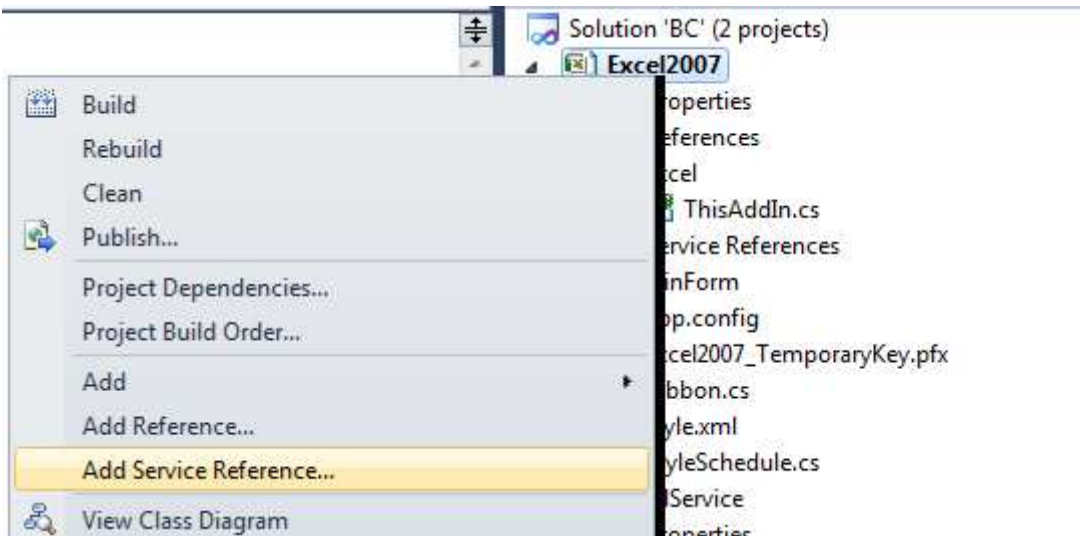
- **Internet Information Services (IIS):** Obdoba provozování webových služeb nebo webových aplikací. Při požadavku klienta se vytvoří proces, který je řízen IIS serverem. Nevýhoda IIS je pouze podpora HTTP protokolu.
- **Windows Service:** Proces systému Windows, který vytváří prostředí pro běh služby, která má možnost být řízena jako všechny ostatní služby v systému. Výhodou je možnost použití všech dostupných protokolů.
- **Self hosting:** Za životní cyklus hostující služby je zodpovědný zdrojový kód implementovaný programátorem.

## 2.3 Klient

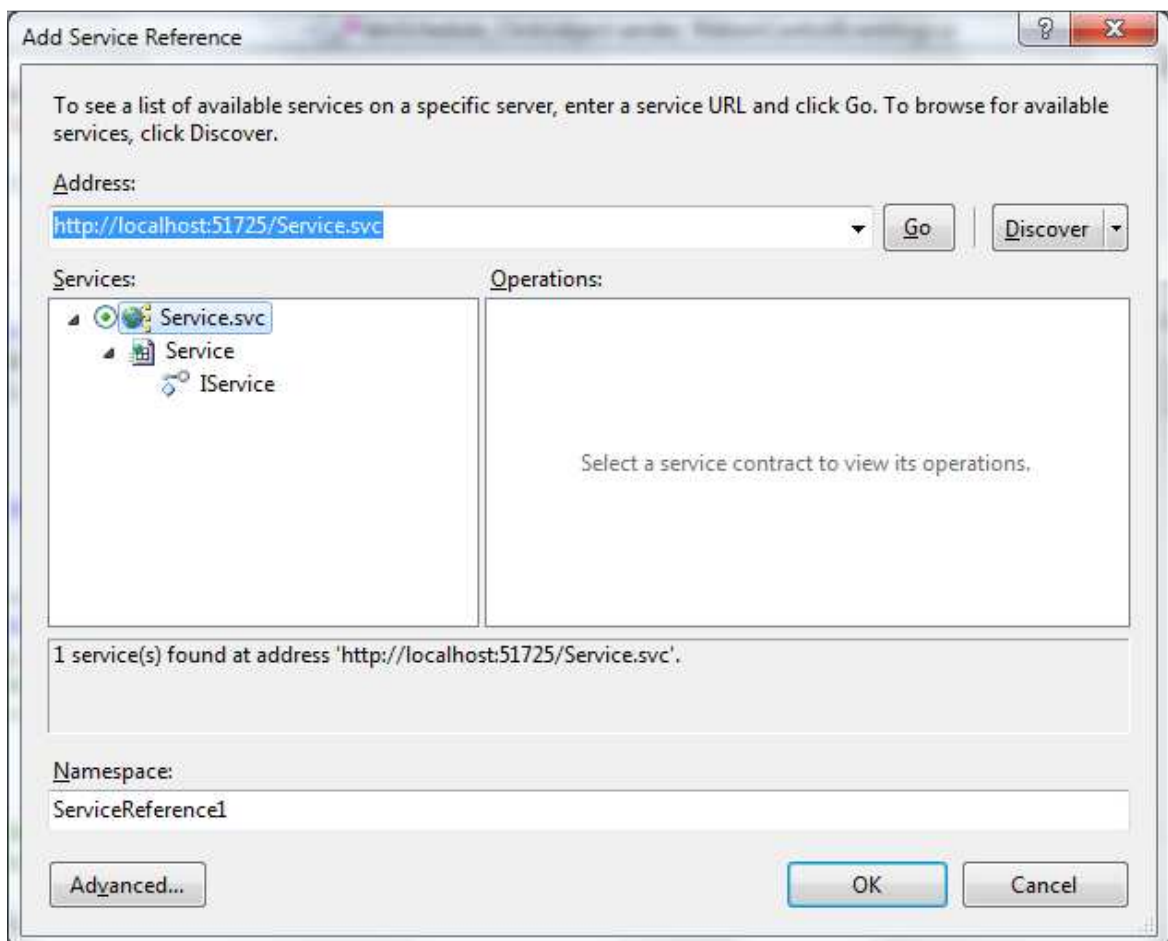
Klient je libovolná aplikace, která využívá službu a volá její metody (*OperationContract*). Se službami se pracuje stejně jako s třídami, pouze se musí přidat reference do projektu. [3]

### 2.3.1 Přidání služby do projektu

V nabídce projektu se otevře dialogové okno *Add Service Reference*.



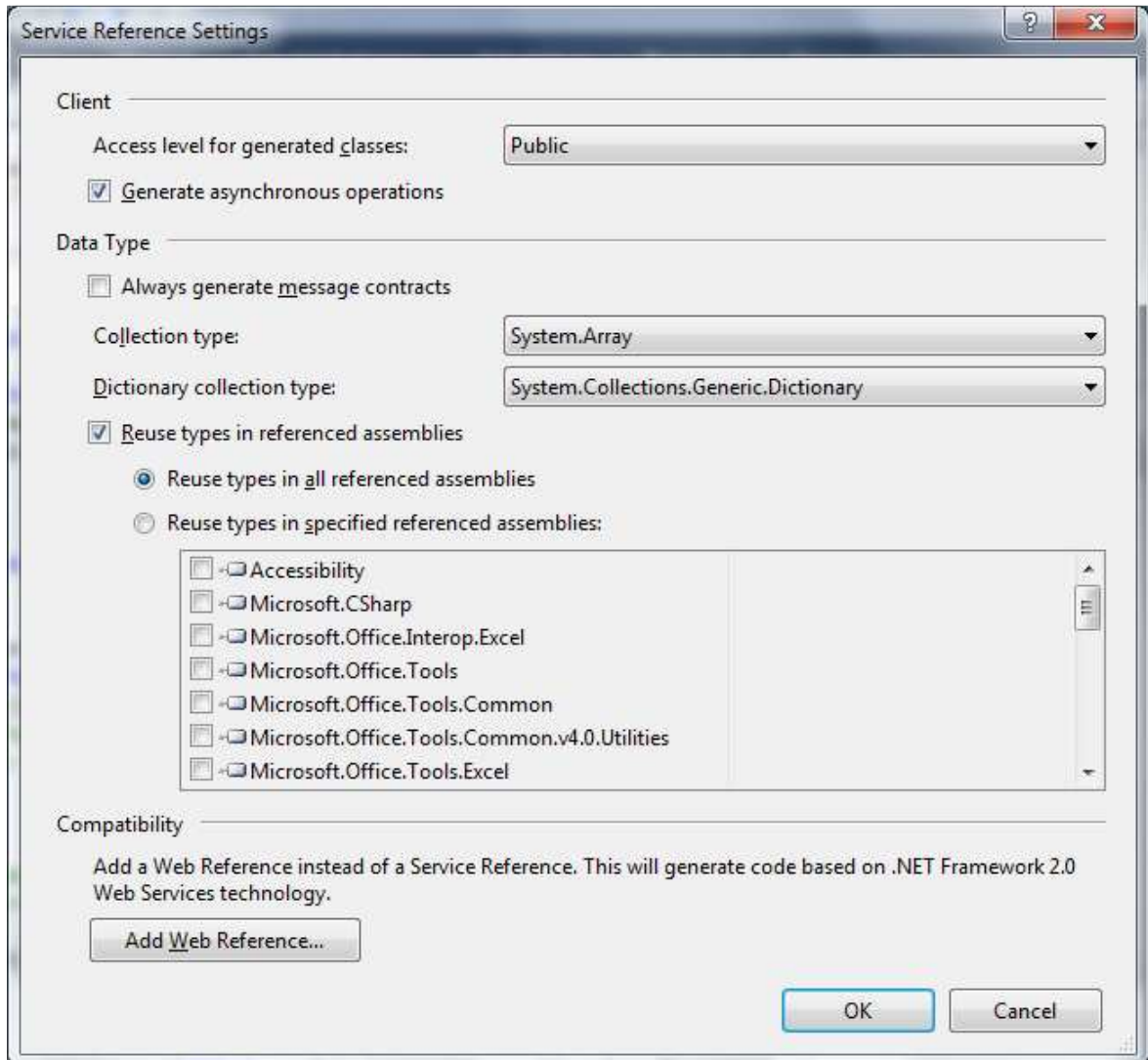
Obrázek 5: Přidání službu do projektu



Obrázek 6: Okno - Add Service Reference

V nově otevřeném okně služby automaticky najdeme služby v projektu stiskem tlačítka *Discover*. V tree-view *Services* se poté objeví služby v projektu (strom nabízí seznam operací).

Pokud bychom chtěli volat metody služby asynchronně, musíme vytvořit metody, které to dovolují. Visual Studio 2010 nám dovoluje tyto metody vytvořit automaticky, po stisku tlačítka *Advanced* se otevře nové dialogové okno. [3]



Obrázek 7: Okno Service Reference Setting – Advanced

V sekci klient se zaškrtně políčko *Generate asynchronous operatoion*. Poté se všechna dialogová okna potvrdí stiskem *Ok*. Po úspěšném přidání služby je možné ji vidět ve složce *Service Rereferences*. [3]

## 2.3.2 Programová část

Služba si vytvořila vlastní jmenný prostor (`Excel2007.Service`). Ve jmenném prostoru se nachází třída s názvem služby a postfixem `Client`, tato třída obsahuje všechny operace služby.

```
ServiceClient service = new ServiceClient();
```

### 2.3.2.1 Synchronní volání

Pokud voláme metody služby synchronně, aplikace čeká na provedení operací. Následující kód vrátí rozvrh skupiny `3I3X`. [3]

```
schedule = service.getSchedule("3I3X");  
//zpracovani vysledku...
```

### 2.3.2.2 Asynchronní volání

Při asynchronním volání požadavek na službu běží paralelně s aplikací. Musí se nejdříve nadefinovat událost, která se provede po dokončení služby a poté službu zavolat asynchronní metodou.

```
service.getScheduleCompleted += new  
EventHandler<getScheduleCompletedEventArgs>(service_getScheduleCompleted);  
service.getScheduleAsync("3I3X");
```

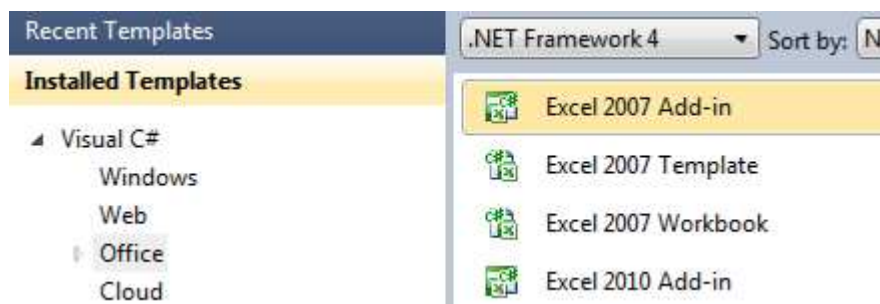
Data služby se zpracují v metodě `getScheduleCompleted`, musíme si ji definovat:

```
private void service_getScheduleCompleted(object sender,  
getScheduleCompletedEventArgs e) { //zpracovani vysledku... }
```

Vrácené data metody se nacházejí v *e.result*.

### 3 EXCEL ADD-IN

Pomocí Visual Studia můžeme vytvářet různé doplňky do programu Microsoft Excel. Pro vytvoření nového doplňku se vytvoří, nebo přidá nový projekt. (*File* → *Add* → *New Project*, *File* → *New Project*). V šablonách vybereme *Office* a poté *Excel 2007 Add – in*

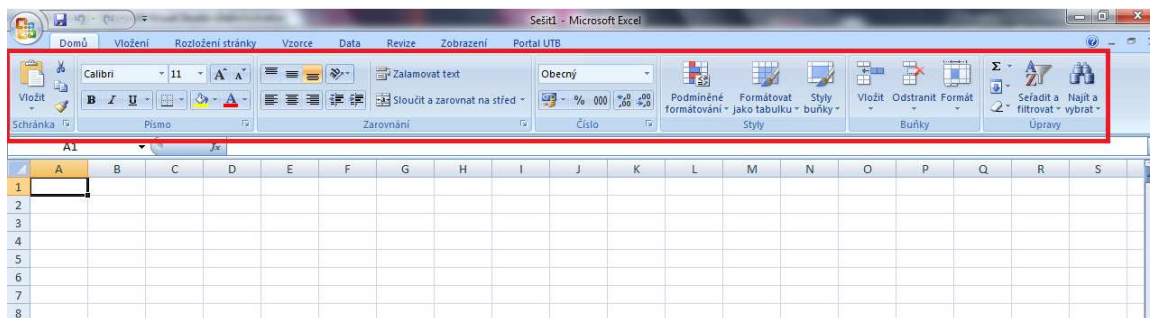


Obrázek 8: Dialogové okno pro vytvoření doplňku

Po spuštění projektu se doplněk automaticky nainstaluje do aplikace Microsoft Excel. Výchozí šablona pouze doplněk inicializuje, ale neprovádí žádnou akci. Třída *ThisAddIn* obsahuje metody, které se provádí při spuštění aplikace a druhá při jejích ukončování. [2]

#### 3.1 Vytvoření Ribbonu

Ovládací prvky jsou od verze MS Office 2007 umístěny do záložek tzv. ribbonů. Obrázek 9 znázorňuje ribbon v červeném obdélníku.



Obrázek 9: Ribbon v Microsoft Excel

Ribbon se do doplňku přidá jako nový prvek. Visual Studio nabízí dva typy:

- Ribbon (Visual Designer) – WYSIWYG editor.
- Ribbon (XML) – XML soubor. Vytváření objektů (tlačítek, galerií, ...) podobně jako v XAML.

Hlavní výhodou *Ribbon (Visual Designer)* je skutečnost, že nemusíme znát syntaxi XML objektů a hned vidíme, jak výsledný ribbon bude v Excelu vypadat. Po přidání do projektu se automaticky spouští v Excelu po startu projektu.

Jednotlivé kontrolory se přidávají v záložce *Toolbox*.

Tabulka 2: Ribbon controls [2]

Box	Pro sjednocení ovládacích prvků
Button	Tlačítko
groupButton	Skupina tlačítek
Checkbox	Zaškrtávací políčko
Combobox	Rozbalovací nabídka
DropDown	Rozbalovací nabídka
Editbox	Psaný text
Gallery	Rozšířená rozbalovací nabídka – velké obrázky
Group	Skupina tlačítek (se spodním popisem)
Label	Text
Menu	Rozbalovací nabídka
Seperator	Oddělovač
SplitButton	Rozbalovací nabídka
Tab	Nová záložka (Ribbon)
toggleButton	Vratné tlačítko

## 3.2 Programování v Microsoft Excelu

Microsoft Excel pro práci s ním obsahuje různé třídy, nacházející se ve jmenném prostoru `Microsoft.Office.Interop.Excel`. Kompletní seznam lze najít na adrese: [msdn.microsoft.com](http://msdn.microsoft.com). Používaný jmenný prostor se může zmínit v direktivě `using` na začátku zdrojového kódu [2]:

```
using Excel = Microsoft.Office.Interop.Excel;
```

Používané objekty při programování doplňku:

- Worksheet – sešit
- Range – vybraná oblast v sešitu

### 3.2.1 Ukázková aplikace

Ukázková aplikace po načtení doplňku ribbon vypíše v buňce sešitu řetězec „test“. Musí se vybrat sešit, do kterého se vloží hodnota. Na sešity se odkazuje pomocí pole

`Globals.ThisAddIn.Application.Worksheets`, indexovány od 1, poté se vybere buňka nebo oblast metodou `get_Range` v našem případě A1. [2]

```
public partial class Ribbon
{
    private Excel.Worksheet ActiveWorkBook;
    private Excel.Range A1;

    private void Ribbon_Load(object sender, RibbonUIEventArgs e) {
        ActiveWorkBook = Globals.ThisAddIn.Application.Worksheets[1];
        A1 = ActiveWorkBook.get_Range("A1");

        A1.Value = "test";
    }
}
```

## 4 DATABÁZE A DOTAZOVACÍ JAZYKY

Databáze je uspořádaná struktura dat, rozdělená do tabulek. Tabulky pak obsahují řádky a sloupce. Sloupce určují typ dat (jméno, příjmení, datum narození) a řádky jednotlivá data. Každý řádek by měl obsahovat jedinečnou hodnotu pro jeho přesné určení, jak pro editaci, tak pro spojení s jinými tabulkami tzv. relace. [2] [4] [9]

- SQL (Structured Query Language) – standardní dotazovací jazyk pro práci s daty v relačních databázích.
- LINQ (Language Integrated Query) – rozšíření pro .NET Framework pro dotazování. Výhodou je dotazovat se na jakýkoliv typ dat.
  - LINQ to Objects – standardní kolekce nacházející se v paměti
  - LINQ to SQL – pro práci s SQL
  - LINQ to XML – pro práci s XML
  - LINQ to DataSet<sup>1</sup> – pro práci s ADO .NET datasety

### 4.1 LINQ

#### 4.1.1 IEnumerable<T>

Data v LINQ se prochází pomocí příkazu `foreach`, který závisí na rozhraní `IEnumerable`, podobně jako direktiva `using` na rozhraní `IDisposable`. Rozhraní `IEnumerable` podporuje šablony a může se implementovat jakýkoliv typ, který se pak bude procházet v cyklu [4] [5]:

```
IEnumerable<Employee> q = from e in employee select e;
foreach(Employee e in q) {
    //zpracování výsledku - pr.: e.name
}
```

---

<sup>1</sup> Zastaralé, musí se definovat DataAdapter pro každý objekt.

### 4.1.2 LINQ to Object

LINQ se nachází ve jmenném prostoru `System.Linq`. Dotaz začíná slovem `from`, kde definujeme data, která se budou načítat. Posledním příkazem LINQ je `select`, který určuje datovou strukturu výstupních dat. LINQ vrací data v `IEnumerable<T>`, což je implementace generického rozhraní. [4] [5]

```
IEnumerable<Employee> linq = from alias in zdrojDat ... select vystupDat;
```

Výpis zaměstnanců starších 60 let:

```
List<Employee> employee = new List<Employee>();  
var employeeLinq = from e in employee  
                  where e.age > 60  
                  select e;
```

Seřazený seznam podle jména a poté příjmení:

```
var employeeLinq = from e in employee  
                  where e.age > 60  
                  orderby e.firstName, e.secondName  
                  select e;
```

Seznam jednotlivých oddělení a počet zaměstnanců v nich, příkaz si vytvoří vlastní strukturu dat:

```
var employeeLinq = from e in employee  
                  group e by e.deparment into d  
                  orderby d.Key  
                  select new {Department = d.Key, Count = d.Count()};
```

Výpis dotazu:

```
foreach(IEnumerable<Employee> em in employeeLinq) s{  
    //...  
}
```

Dotaz LINQ se pouze nadefinuje, je spuštěn až při práci s ním. Nucené vyhodnocení LINQ se může provést následnými metodami. Každá metoda dotaz převede na určitý datový typ. [4] [5]

- `ToArray()`
- `ToDictionary()`
- `ToList()`
- `ToLookup()`
- `First()` – vrátí pouze první záznam

### 4.1.3 LINQ to XML

Od svých počátků je LINQ navrženo tak, aby manipulovalo s XML daty stejně snadno, jako manipuluje s relačními daty. LINQ to XML představuje nové API pro XML, což v XPath a XQuery je ještě mnohem jednodušší.

Například předpokládejme, že zdroj dat pro aplikaci „firma“, kde jsou uloženy informace o zaměstnancích a oddělení, jsou uloženy v souboru *company.xml*. [4] [5]

```
XElement company = XElement.Load(@"company.xml");
XElement employee = company.Element("employee");
XElement department = company.Element("department");

var employees = from e in employee.Elements() select e;
```

Pro výpis všech zaměstnanců, kteří se jmenují „Karel“:

```
var karlove = from e in employee.Elements() where e.Element("Name").Value == "Karel"
select e;
```

Jak můžeme vidět, dotazování XML dokumentů s LINQ je úplně stejné jako s LINQ to Object. Rozdíl je v tom, že strukturu XML dokumentu je třeba vzít v úvahu, např. v tomto případě dokument je hierarchicky navržen.

Pomocí XML se může transformovat log soubor IIS serveru do XML:

```
#Software: Microsoft Internet Information Services 5.1
#Version: 1.0
#Date: 2006-06-23 12:37:18
#Fields: time c-ip cs-method cs-uri-stem sc-status
12:37:18 127.0.0.1 GET /cobrabca 404
12:37:25 127.0.0.1 GET /cobranca 40
```

LINQ pro rozdělení IIS logu do XML:

```
var logIIS = new XElement("LogIIS",
    from line in File.ReadAllLines("file.log")
    where !line.StartsWith("#")
    let items = line.Split(' ')
    select new XElement("Entry",
        new XElement("Time", items[0]),
        new XElement("IP", items[1]),
        new XElement("Url", items[3]),
        new XElement("Status", items[4])));
```

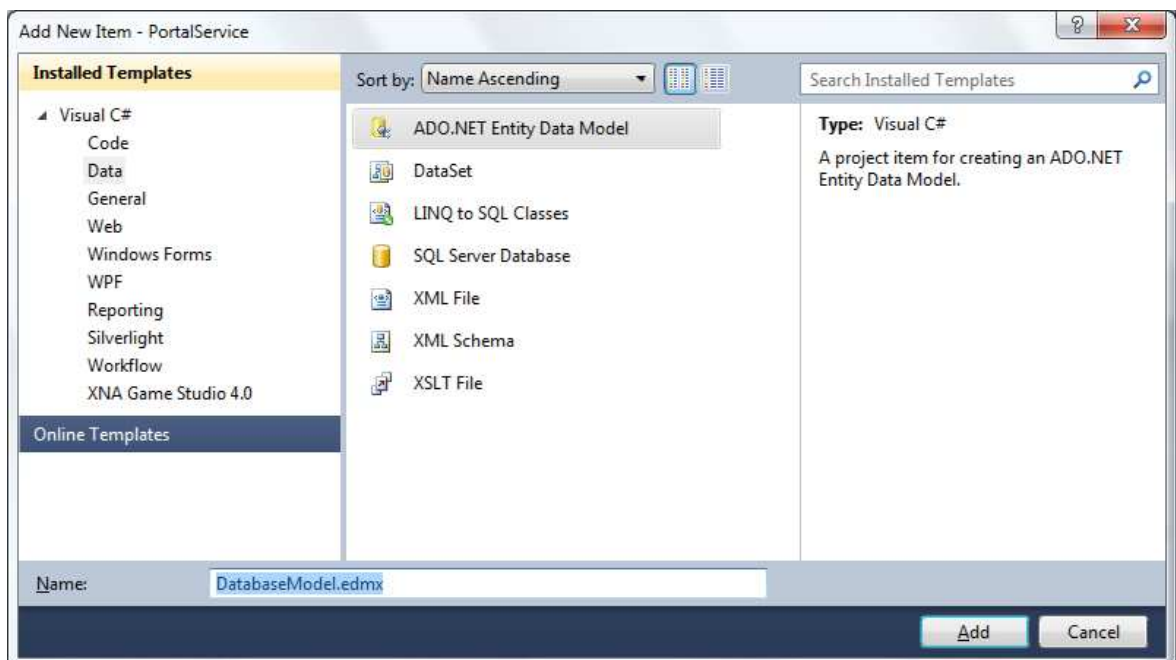
### 4.1.4 LINQ to Entity Framework

Entity Framework (dále Entity) je nástroj pro objektově relační mapování. LINQ to SQL podporuje pouze připojení na SQL na rozdíl od Entity. Někdy je výhodné používat univerzální framework pro databáze z důvodu přechodu na jiný databázový jazyk.

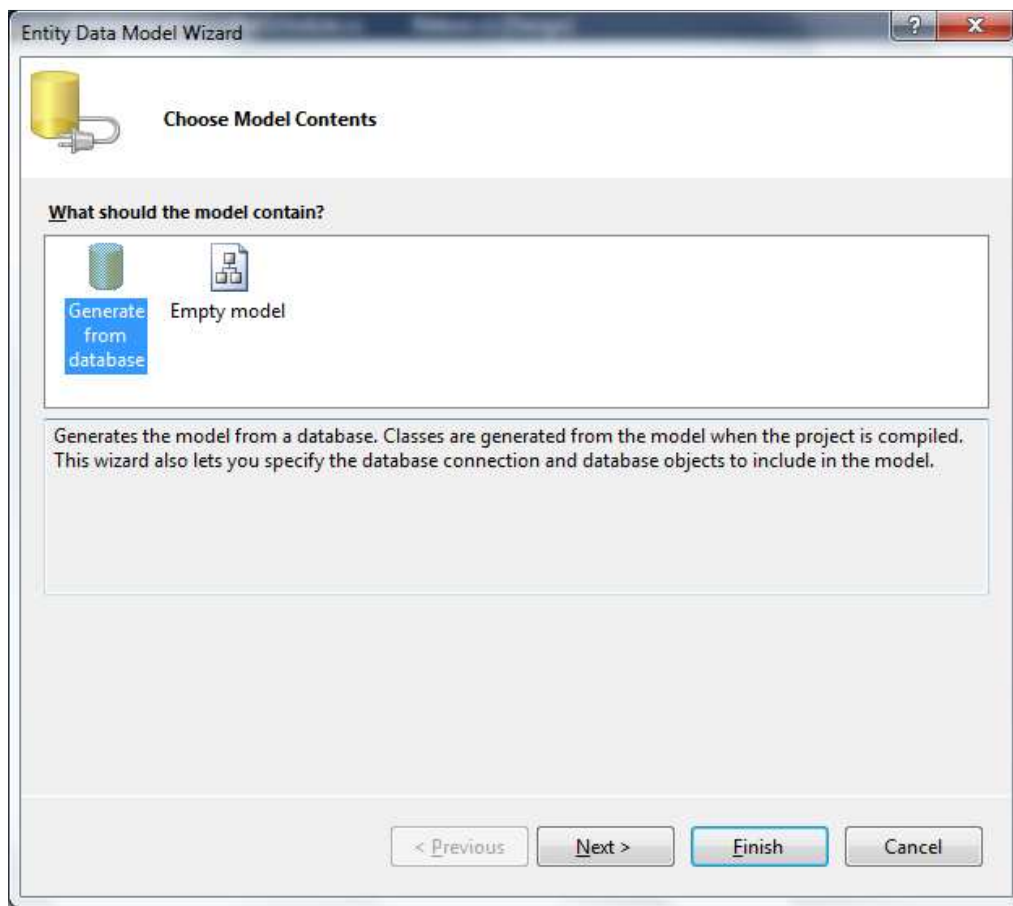
Nevýhoda LINQ to SQL je, že Microsoft už tuto technologii nepodporuje a může se používat pouze pro mapování dat 1:1. [2] [4] [5]

#### 4.1.4.1 Přidání Entity do projektu

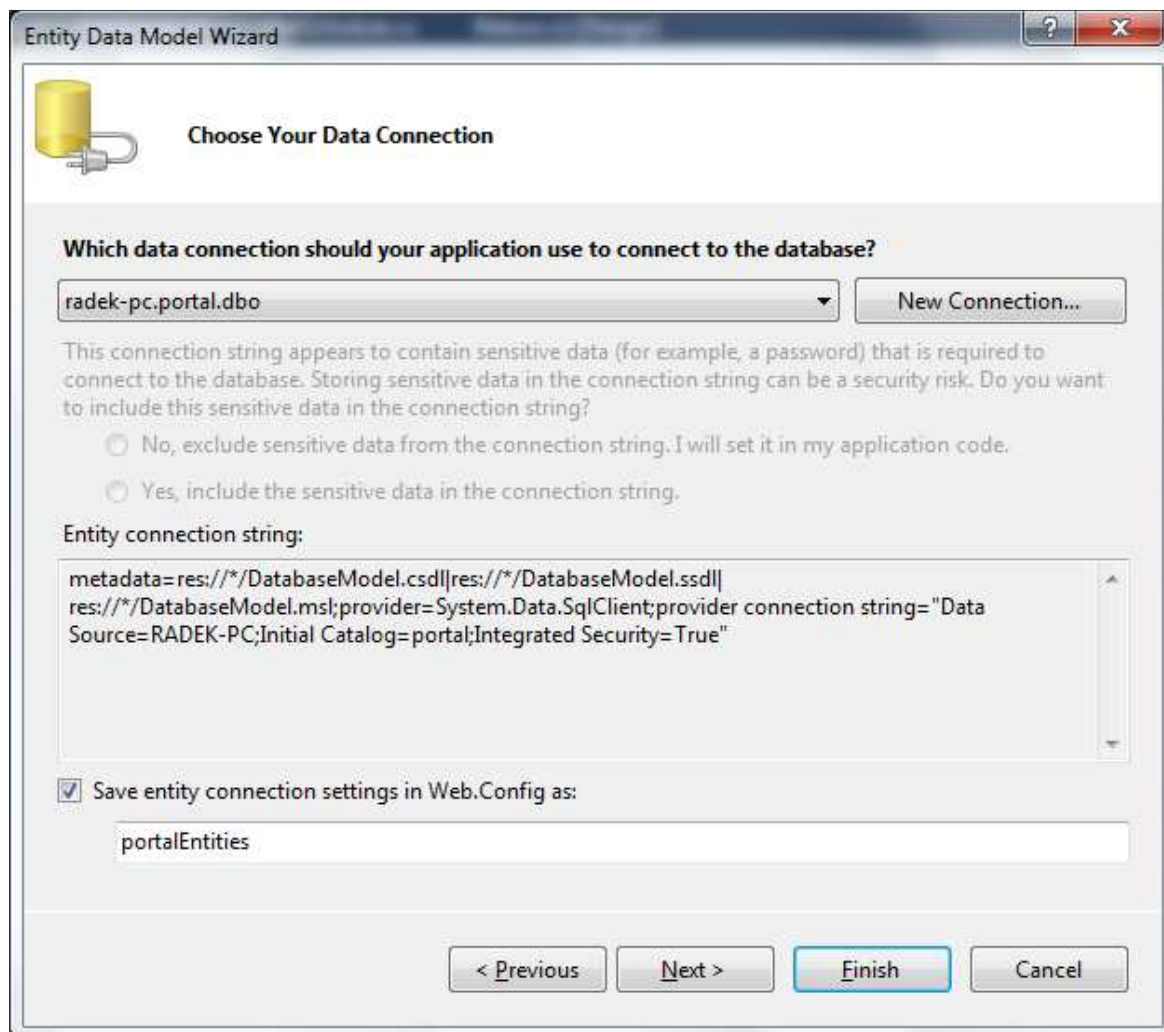
Entity se do projektu přidá podobně jako nová třída (*Add* → *New Item*). Otevře se dialogové okno (Obrázek 10). Potvrzením vytvoření nové položky se automaticky otevře nové dialogové okno na výběr modelu (Obrázek 11), pokračováním na další dialogové okno se vybere připojení na databázi (Obrázek 12), poslední krokem je výběr tabulek (Obrázek 13), které se exportují ze serveru a budou jim vytvořeny objektové modely pro použití v kódu. [2] [4] [5]



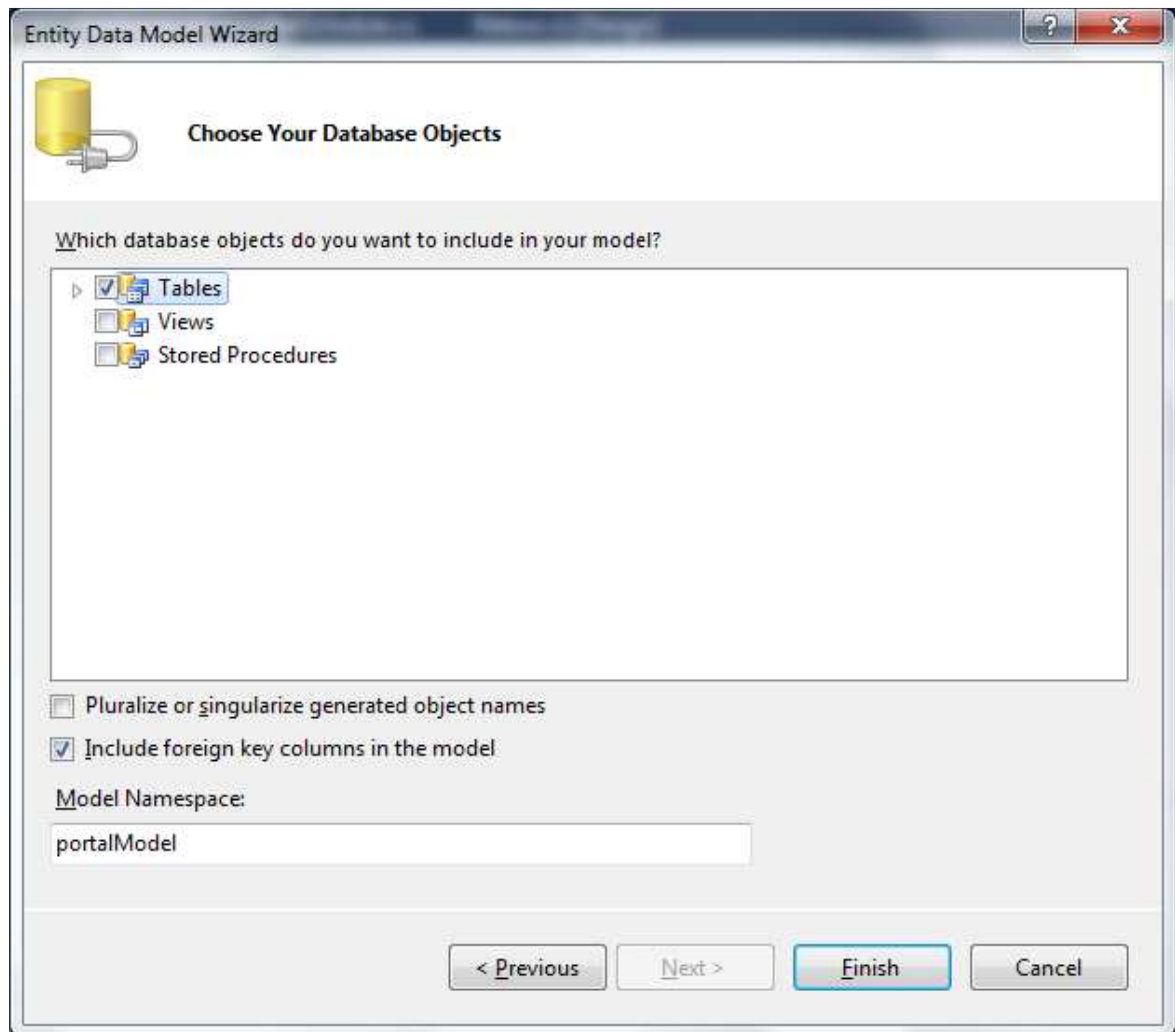
Obrázek 10: Entity: Přidání nového souboru



Obrázek 11: Entity: vybraní typu modelu



Obrázek 12: Entity: Připojení na databázi



Obrázek 13: Entity: Výběr tabulek, pohledů a procedur pro export

Musí se inicializovat třída `PortalEntities`, která obsahuje jednotlivé odkazy na tabulky v členských proměnných třídy. Vypis seznamu skupin:

```
PortalEntities portalEntities = new PortalEntities();  
IEnumerable<string> zkratky = from g in portalEntities.skupina orderby g.zkratka  
select g.zkratka;
```

## 5 UŽIVATELSKÉ ROZHRAŇÍ

WinForms je název pro grafické API součástí .NET Framework od verze 1.0 poskytující nativní přístup k Microsoft Windows rozhraní vytvářené v kódu aplikace, nebo pomocí Designéra Windows.

Od .NET 3.0 se objevilo nové uživatelské prostředí WPF, píše se v jazyku XAML. Technologie je vestavěná do Windows Vista, Windows 2008 a novějších. Účelem WPF je sjednotit uživatelské rozhraní, animace, vektorovou a rastrovou grafiku, 2D a 3D grafiku, vázání dat a audio a video. WPF se dá považovat za náhradu starých WinForms. [2] [4] [7]

### 5.1 Výměna dat mezi formuláři

Formulář by měl aktualizovat data rodičovské aplikace po stisku tlačítka *OK*. Je možné změnit vlastnosti formulářů z privátních na veřejné, a tím k nim přistupovat z hlavní aplikace. Pokud ale dojde ke změně prvků, musí se upravit všechny metody, ve kterých jsou použity. [2] [4] [7]

#### 5.1.1 Zpracování dat modálních<sup>2</sup> formulářů

Abychom mohli zpracovat data z formuláře, musí metoda `ShowDialog` vrátit řízení, to znamená, že dialogové okno se musí zavřít, čehož docílíme vyvoláním funkce `Form.Close`. Naštěstí `ShowDialog` standardně vrací `DialogResult.Cancel`. Formulář nám ale může vracet i hodnoty z výčtu [2] [4] [7]:

```
public enum DialogResult {  
    None = 0,  
    OK = 1,  
    Cancel = 2,  
    Abort = 3,  
    Retry = 4,  
    Ignore = 5,  
    Yes = 6,  
    No = 7,  
}
```

Hodnoty `Abort`, `Ignore`, `No` a `Retry` většinou vrací `MessageBox`, ale můžeme je používat ve svých formulářích jakkoliv. Jestli formulář vrátil `OK`, můžeme ověřit následujícím kódem:

---

<sup>2</sup> Nevrací řízení aplikaci, ale čeká na zavření formuláře.

```
WinForm.Teacher t = new WinForm.Teacher();
t.ShowDialog();
if (t.DialogResult == DialogResult.OK) {
    //...
}
```

Standardně každý formulář vrací hodnotu `None`, proto by se měla vlastnost `Form.DialogResult` před zavřením ve formuláři nastavit. U modálních formulářů při nastavení hodnoty `Form.DialogResult` se automaticky vyvolá metoda `Close`. [2] [4] [7]

```
private void okButton_Click(object sender, EventArgs e) {
    this.DialogResult = DialogResult.OK;
}
```

U tlačítek můžeme nastavit vlastnost – *výchozí tlačítko* – tedy to, které se stiskne s klávesou `Enter` (OK) nebo `Esc` (Storno). Obě vlastnosti mohou být nastaveny v designérovi nebo v kódu. Výhodou nastavení výchozích tlačítek je automatické nastavení návratové hodnoty formuláře. [2] [4] [7]

```
this.AcceptButton = this.btn_confirm;
this.CancelButton = this.btn_storno;
```

### 5.1.2 Zpracování dat nemodálních formulářů

Modální formuláře se otevírají metodou `Show()`, která vrací okamžitě řízení hlavní aplikaci. Při nastavení `Form.DialogResult` se formulář automaticky nezavře, jako je tomu u modálních formulářů. K zpracování dat v tomto případě slouží události, které si definujeme jako veřejné ve formuláři:

```
public event EventHandler accept;

private void confirm(object sender, EventArgs e) {
    if(this.accept != null) this.accept(this, EventArgs.Empty);
    this.Close();
}
```

Událost se definuje jako delegát na volanou metodu, ve kterém se na formulář odkáže pomocí parametru `sender`.

```
private void btn_teacher_Click(object sender, RibbonControlEventArgs e) {
    WinForm.Teacher t = new WinForm.Teacher();
    t.accept += new EventHandler(getTeacherInfo);
    t.ShowDialog();
}

private void getTeacherInfo(object sender, EventArgs e) {
    //...
}
```

## **II. PRAKTICKÁ ČÁST**

## 6 CÍL APLIKACE

### 6.1 Karta vyučujícího

Aplikace, pomocí služby běžící na IIS serveru (během vývoje je hostována na stejném počítači), vrací úvazek vybraného vyučujícího se seznamem vyučovaných předmětů do aplikace Microsoft Excel. Zobrazená data není možné editovat, tzn. změna dat v tabulce není uložena do SQL databáze.

Aplikace bude vytvářet „karty vyučujícího“ zobrazující jméno, úvazky (A, B, C) a vyučované předměty v předem vybraném období.

<b>Vyučující:</b>	...					<b>Procent úvazku:</b>	...		
<b>Úvazek (ZH):</b>	...					<b>Úvazek ZH:</b>	...		
<b>A) Přímá výuka</b>						<b>Celkem:</b>	<b>756</b>		
<b>Předmět</b>	<b>Týdnů</b>	<b>PŘ</b>	<b>CV</b>	<b>SE</b>	<b>PŘ</b>	<b>CV</b>	<b>SE</b>	<b>A</b>	
...	...	...	...	...	..	..	...	...	...
...	...	...	...	...	..	..	..	...	...
<b>B) Zkoušení a tutoring</b>						<b>Celkem:</b>	<b>..</b>		
<b>Předmět</b>								<b>B</b>	
...	...	...	...	...	..	..	...	...	...
...	...	...	...	...	..	..	..	...	...

Obrázek 14: Návrh grafického rozhraní – karty vyučujícího

Úvazek a C je zadán v databázi pro výpočet A a B každého předmětu platí následující rovnice:

$$A_p = K \cdot h \cdot t \cdot q$$

Rovnice 1: Výpočet úvazku A jednoho předmětu

Kde:

- $K$  – konstanta typu předmětu
- $h$  – počet hodin (60 minut) v rámci jednoho vyučování
- $t$  – počet výukových týdnů
- $q$  – počet skupin

Výpočet úvazku je nezávislý na oboru, kde učitel učí, výpočet pouze rozlišuje konstanta  $K$  určená jinak pro každý typ výuky (seminář, přednáška a cvičení). Celkové A se vypočítá jako suma úvazků všech vyučovaných předmětů nezávisle na oboru:

$$A = \sum_{i=1}^{p_n} A_i$$

Rovnice 2: Výpočet celkového úvazku  $A$

Výpočet úvazku  $B$  závisí na počtu studentů a typu ukončení předmětů:

$$B_p = K \cdot n_p$$

Rovnice 3: Výpočet úvazku  $B$  jednoho předmětu

Kde:

- $K$  – konstanta typu ukončení (zkouška, zápočet a neklasifikovaný zápočet)
- $n$  – počet studentů

$$B = \sum_{i=1}^{p_n} B_i$$

Rovnice 4: Výpočet celkového úvazku  $B$

## 6.2 Karta ústavu

Ze stejného zdroje (WCF a SQL) se do klientské aplikace Microsoft Excel zobrazí karta ústavu, tzn. seznam předmětů, počet skupin a studentů rozdělených podle oboru, a k nim přiřazení vyučující s počtem vyučujících skupin a přednášející.

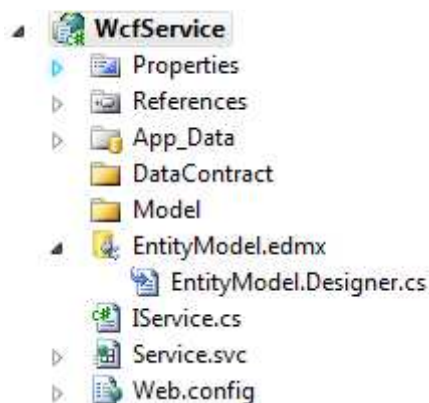
A4AUT	Cvičení	Skupin	Vyučující
	<i>Obor...</i>	...	...
			...
	...	...	...
	Seminář	Skupin	Vyučující
	...	...	...
	Zkoušení	Skupin	Vyučující
	...	...	...
	...	...	...

Obrázek 15: Návrh grafického rozhraní – karty ústavu

## 7 VYTVOŘENÍ SLUŽBY

Praktickou část jsem začal vytvořením služby, která bude zpracovávat data z SQL souboru a vracet je klientovi (Microsoft Excel). V nově vytvořené službě jsem si přejmenoval šablonové soubory a třídy služby a upravil si strukturu adresářů přidáním nové složky `DataContracts`, obsahující data služby, adresář `Model`, kde budou třídy přístupující k datům uloženy v databázi. Pokud změníme název třídy `ServiceContract`, měli bychom zkontrolovat konfiguraci služby, jestli je při spuštění volána správná služba.

Do projektu jsem si přidal Entity model, ten vytvořil třídy pro přístup k SQL pomocí LINQ to Entity.



Obrázek 16: Struktura adresáře služby

### 7.1 Základní metody

Klientská aplikace bude zobrazovat data v závislosti na odděleních a období (školním roce). Vytvořil jsem si třídu ve složce Model:

```
static class Department
{
    public static Dictionary<int, string> getList()
    {
        DatabaseEntities de = new DatabaseEntities();
        return (from p in de.ustav
                orderby p.nazev
                select new
                {
                    id = p.id,
                    name = p.nazev
                }).ToDictionary(p => p.id, p => p.name);
    }
}
```

Druhá třída na vracení období:

```
static class Period
{
    public static Dictionary<int, string> getList()
    {
        DatabaseEntities de = new DatabaseEntities();
        return (from p in de.obdobi
                orderby p.rok
                select new
                {
                    id = p.id,
                    name = p.rok
                }).ToDictionary(p => p.id, p => p.name.ToString());
    }
}
```

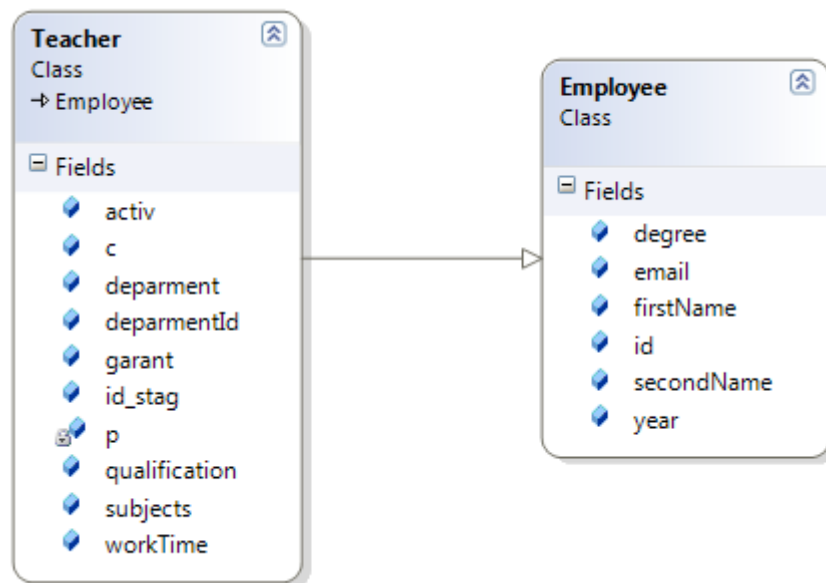
Ve službě stačí upravit `ServiceContract` a zavolat jednotlivé statické metody a hned je vrátit `OperationContract` metodami. Příklad použití těchto metod viz kapitola 8.1 Dialog Vyučující.

```
public Dictionary<int, string> getDepartmentList()
{
    return Model.Department.getList();
}

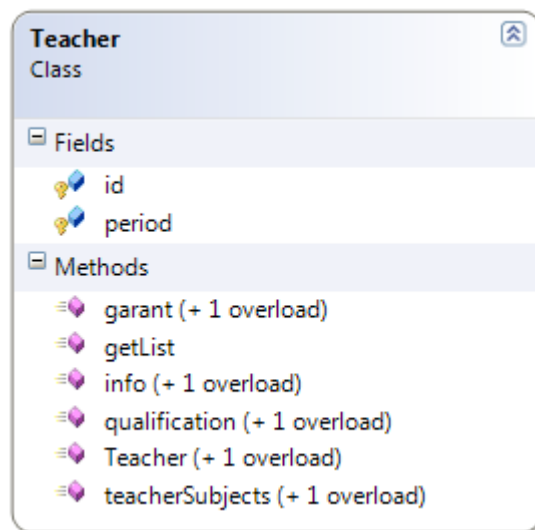
public Dictionary<int, string> getPeriodList()
{
    return Model.Period.getList();
}
```

## 7.2 Vyučující

Vytvořil jsem si třídu `DataContract` reprezentující zaměstnance a učitele. Třída `Teacher` je potomkem `Employee`. Každá členská proměnná obou tříd je `DataMember`, aby byla přístupná ze strany klienta. Třídy v `DataContract` obsahují pouze definice členských proměnných, veškerá logika je ve třídách `Model`.

Obrázek 17: Diagram třídy `DataContracts.Teacher`

Ve složce *Model* jsem si vytvořil třídy pro práci s `DataContract`. Třída obsahuje přetížené metody pro práci s členskými proměnnými i s libovolnými parametry.

Obrázek 18: Diagram třídy `Model.Teacher`

Metody pro načtení informací o učiteli vypadají následovně:

```
public DataContracts.Teacher info() {
    return this.info(this.id, this.period);
}

public DataContracts.Teacher info(int id, int period)
{
    DatabaseEntities de = new DatabaseEntities();
    DataContracts.Teacher ret = (from te in de.vyucujici
        join ep in de.vyucujici_obdobi on te.id equals ep.id_vyucujici
        join pe in de.obdobi on ep.id_obdobi equals pe.id
        where pe.id == period && te.id == id
        select new DataContracts.Teacher(){
            id = te.id, id_stag = te.id_stag,
            deparment = te.ustav.nazev,
            email = te.email,
            firstName = te.jmeno, secondName = te.prijmeni,
            degree = te.titul,
            activ = te.planovat,
            c = ep.c, workTime = ep.uvazek,
        }).First();

    ret.subjects = this.teacherSubjects(id, period);
    ret.qualification = this.qualification(id, period);
    ret.garant = this.garant(id);

    return ret;
}
```

Každý `OperationContract` služby vrací `FaultContract` z důvodu jakékoliv chyby na straně služby. Na straně klienta se pak odchytí pomocí try-catch.

```
[ServiceContract]
public interface IService
{
    [OperationContract]
    [FaultContract(typeof(Exception))]
    List<DataContracts.Employee> getTeachers(string deparment, int period);

    [OperationContract]
    [FaultContract(typeof(Exception))]
    Dictionary<int, string> getDeparmentList();

    [OperationContract]
    [FaultContract(typeof(Exception))]
    Dictionary<int, string> getPeriodList();

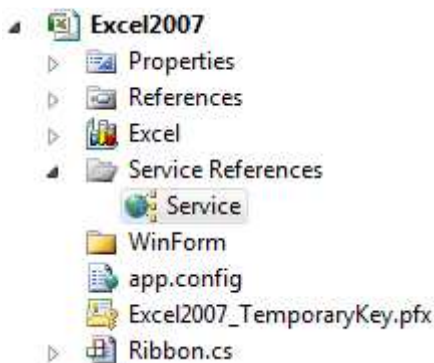
    [OperationContract]
    [FaultContract(typeof(Exception))]
    DataContracts.Teacher teacherInfo(int id, int period);

    [OperationContract]
    [FaultContract(typeof(Exception))]
    List<DataContracts.Subject> getSubject(int deparment, int year);

    [OperationContract]
    [FaultContract(typeof(Exception))]
    List<DataContracts.Teaching> teaching(int deparment, int period);
}
```

## 8 EXCEL ADD-IN

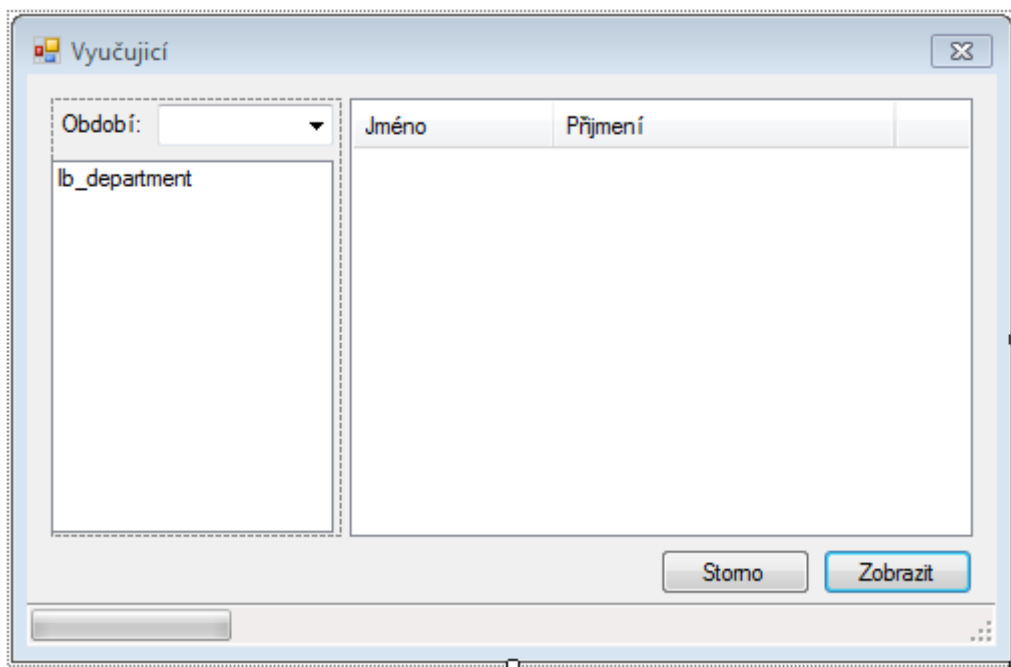
Vytvořil jsem doplněk a přidal Ribbon (Visual Designer) a referenci vytvořené služby. Protože je zapotřebí dialogové okno, pro přehlednost jsem vytvořil složku WinForm.



Obrázek 19: Adresářová struktura Excel Add-in

### 8.1 Dialog Vyučující

Do nově vytvořeného Ribbon, jsem si přidal tlačítko s popiskem Vyučující. Protože budeme vybírat ze seznamu učitelů, který vrátí služba, vytvořil jsem dialogové okno **Teacher** (Windows Form).



Obrázek 20: Dialogové okno Vyučující

V levé části ListBox slouží na zobrazení oddělení a ComboBox s výběrem období. Po kliknutí na ListBox se v pravé části zobrazí seznam vyučujících ve vybraném oddělení. Do

dialogového okna jsem přidal status bar, zobrazující načítání dat ze služby a label pro výstup chyb. V metodě Teacher\_Load se musí inicializovat služba a zavolat asynchronně metoda na vracení seznamu oddělení, do události po dokončení se vyplní ListBox.

```
private void Teacher_Load(object sender, EventArgs e)
{
    this.bg_worker.Style = ProgressBarStyle.Marquee;
    Service.ServiceClient s = new Service.ServiceClient();
    s.getDepartmentCompleted += new
EventHandler<Service.getDepartmentListCompletedEventArgs>(s_getDepartmentCompleted);
    s.getDepartmentAsync();

    s.getPeriodListCompleted += new
EventHandler<Service.getPeriodListCompletedEventArgs>(s_getPeriodListCompleted);
    s.getPeriodListAsync();
}

void s_getPeriodListCompleted(object sender, Service.getPeriodListCompletedEventArgs
e)
{
    this.cmb_period.Items.Clear();
    foreach(KeyValuePair<int, string> p in e.Result){
        this.cmb_period.Items.Add(p.Value);
    }
    this.bg_worker.Style = ProgressBarStyle.Blocks;
}

void s_getDepartmentCompleted(object sender,
Service.getDepartmentListCompletedEventArgs e)
{
    foreach(KeyValuePair<int, string> dep in e.Result){
        this.lb_department.Items.Add(dep.Value);
    }
    this.bg_worker.Style = ProgressBarStyle.Blocks;
}
```

Definoval jsem si událost na kliknutí na lb\_department, která podobným způsobem zobrazí seznam vyučujících, pro lepší čitelnost a možnost řadit položky na kliknutí na záhlaví jsem zvolil ListView (Vlastnost View = Details).

```
private void lb_department_SelectedIndexChanged(object sender, EventArgs e)
{
    if(this.cmb_period.SelectedItem == null){
        this.lbl_status.Text = "Nebylo vybráno období";
    } else {
        this.lbl_status.Text = "";
        this.bg_worker.Style = ProgressBarStyle.Marquee;
        this.lv_teachers.Items.Clear();

        Service.ServiceClient s = new Service.ServiceClient();
        s.getTeachersCompleted += new
EventHandler<Service.getTeachersCompletedEventArgs>(s_getTeachersCompleted);
        s.getTeachersAsync(this.lb_department.SelectedItem.ToString(),
Int32.Parse(this.cmb_period.SelectedItem.ToString()));
    }
}

void s_getTeachersCompleted(object sender, Service.getTeachersCompletedEventArgs e)
```

```

{
    if (e.Result.Count() == 0) {
        this.lbl_status.Text = "Žádný učitel nebyl nalezen";
    } else {
        this.lbl_status.Text = "";
        foreach (Service.Employee i in e.Result) {
            ListViewItem item = new ListViewItem();
            item.Text = i.firstName;
            item.SubItems.Add(i.secondName);
            item.Tag = i.id;
            this.lv_teachers.Items.Add(item);
        }
    }
    this.bg_worker.Style = ProgressBarStyle.Blocks;
}

```

V každé funkci volající službu se zobrazí progress bar, který se schová po dokončení dat. Není možné zjistit stav přenosu, proto se zobrazuje pouze to, že se děje něco na pozadí, nikoli přesná procenta (style = Marquee). V metodách se mění styl progress baru `Blocks` na `Marquee` z důvodu zastavení neexistující vlastnosti pro zastavení ukazatele činnosti. Animace se může zastavit nastavením vlastnosti rychlosti animace na 0. Ukazatel se pouze zastaví, ale nezmizí, proto jsem použil změnu stylu.

Tlačítka jsem si nastavil jako `OK` a `Cancel`. Data z tohoto formuláře zpracovávám modálním způsobem – hlavní aplikace se zastaví a čeká na vrácení dat z formuláře.

```

public int teacherId;
public int year;

//..

protected void confirm(object sender, EventArgs e)
{
    if (this.lv_teachers.SelectedItems.Count == 0 ||
        !Int32.TryParse(this.lv_teachers.SelectedItems[0].Tag.ToString(), out
            this.teacherId)) {
        this.DialogResult = DialogResult.None;
        return;
    }

    this.DialogResult = DialogResult.OK;
    this.Close();
}

```

## 8.2 Vytvoření ovládacích prvků

Ribbon bude obsahovat dvě skupiny – `Vyučující` a `Výuka`. Ve skupině `Vyučující` bude tlačítko zobrazující dialogové okno `Vyučující` (viz kapitola 8.1), v druhé skupině se nachází dva uživatelské ovládací prvky `ComboBox`, které se automaticky vyplní daty ze služby (období a oddělení) a tlačítko pro zobrazení `Výuky`.



Obrázek 21: Vytvořený Ribbon – neautorizovaný uživatel



Obrázek 22: Vytvořený Ribbon – autorizovaný uživatel

Po načtení Ribbonu se provede uzamčení skupin volající službu.

```
private void Ribbon_Load(object sender, RibbonUIEventArgs e)
{
    this.groupEnabled(this.gr_teacher);
    this.groupEnabled(this.gr_teaching);
}

private void groupEnabled(RibbonGroup group, bool status = false){
    foreach (RibbonControl i in group.Items) {
        i.Enabled = status;
    }
}
```

Na tlačítka Ribbonu jsem přidal události, tlačítko *Úvazek* zobrazí dialogové okno, jehož data zpracuje modálním způsobem. Druhé tlačítko načte a zavolá službu a pomocí seskupování a podmínek zobrazí kartu ústavu.

Na formátování hlaviček jsem si vytvořil třídu definující styl v Excelu, který se pouze přiřadí dané buňce:

```
class ThemeLoader
{
    static int numOfInstance = 0;
    public const string header = "header";
    private Excel.Workbook wb;

    public ThemeLoader(Excel.Workbook w) {
        this.wb = w;

        if(ThemeLoader.numOfInstance == 0) {
            Excel.Style style = w.Styles.Add(ThemeLoader.header, Type.Missing);
        }

        this.setHeaderStyle();
    }
}
```

```

        numOfInstance++;
    }

    private void setHeaderStyle(){
        Excel.Style style = wb.Styles[ThemeLoader.header];
        style.Font.Bold = true;
    }
}

```

### 8.3 Autorizace uživatele

Po načtení doplňku ribbon se automaticky spustí metoda `Ribbon_Load`, která volá metodu zamykající jednotlivé ovládací prvky. Metodu na zamčení je možné volat i na odemčení přidáním druhého parametru (`false` – zamkne, `true` – odemkne).

```

private void Ribbon_Load(object sender, RibbonUIEventArgs e)
{
    this.groupEnabled(this.gr_teacher);
    this.groupEnabled(this.gr_teaching);
}

private void groupEnabled(RibbonGroup group, bool status = false){
    foreach (RibbonControl i in group.Items) {
        i.Enabled = status;
    }
}

```

Po stisku přihlašovacího tlačítka se zadané uživatelské jméno a heslo ověřuje s univerzitním systémem Novell (*nw-central.utb.cz:389*), na který se připojuje pomocí LDAP protokolu. Autentizace je definována v metodě `User.login`.

Možnost lokální autentizace je pro hosta, který nemusí mít vytvořen účet na univerzitním systému Novell.

```

class User
{
    const string source = @"App_Data\localUsers.xml";

    public static void login(string userName, string password, bool local = false)
    {
        try {
            if (userName == "" || password == "") throw new Exception();
            if (local == true) {
                User.localAuthenticator(userName, password);
            } else {
                User.remoteAuthenticator(userName, password);
            }
        } catch (Exception ex) {
            throw new Exception("Autorizace se nepodařila");
        }
    }

    protected static void remoteAuthenticator(string userName, string password)
    {
        try {

```

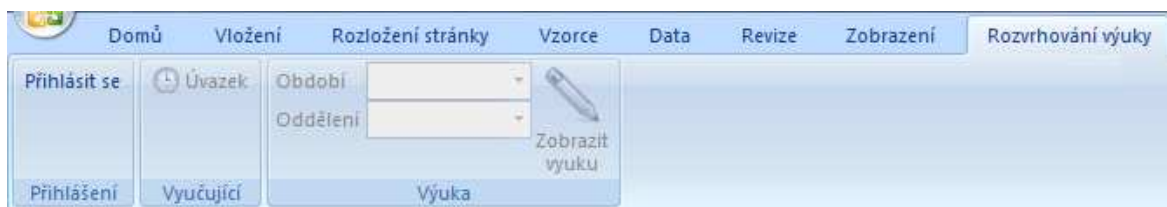
```
LdapConnection con = new LdapConnection("nw-central.utb.cz:389");
con.AuthType = AuthType.Basic;
con.Bind(new System.Net.NetworkCredential("cn=" + userName + ",ou=fai-
st,o=utb", password));
con.Dispose();
} catch (Exception ex) {
    throw new Exception(ex.Message);
}
}

protected static void localAuthenticator(string userName, string password)
{
    try {
        XElement styleXML = XElement.Load(source);
        int authenticatorQuery = (from s in styleXML.Descendants("user")
                                where s.Element("login").Value == userName &&
                                       s.Element("password").Value == password
                                select s).Count();

        if (authenticatorQuery == 0) throw new Exception();
    } catch (Exception ex) {
        throw new Exception(ex.Message);
    }
}
}
```

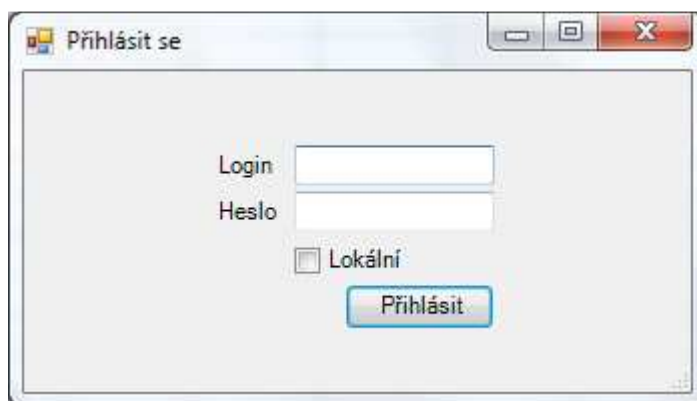
## 9 VYTVOŘENÁ APLIKACE

Doplňek je integrován do aplikace Microsoft Excel a po jeho spuštění se automaticky přidá do pasu karet jako nová karta s popiskem „Rozvrhování výuky“. Po načtení jsou skupiny doplňky zamčeny, jediná aktivní volba je „Přihlásit se“.



Obrázek 23: Výchozí podoba načteného doplňku

Po stisku tlačítka se zobrazí modální dialogové okno s možností zadání přihlašovacího jména a hesla. Volba „Lokální“ slouží pro autentizaci uživatelů, kteří nemají účet na univerzitním systému Novell. Přihlašovací údaje lokálního uživatele jsou uloženy v XML souboru v projektu aplikace (login: „test“, heslo: „test“).



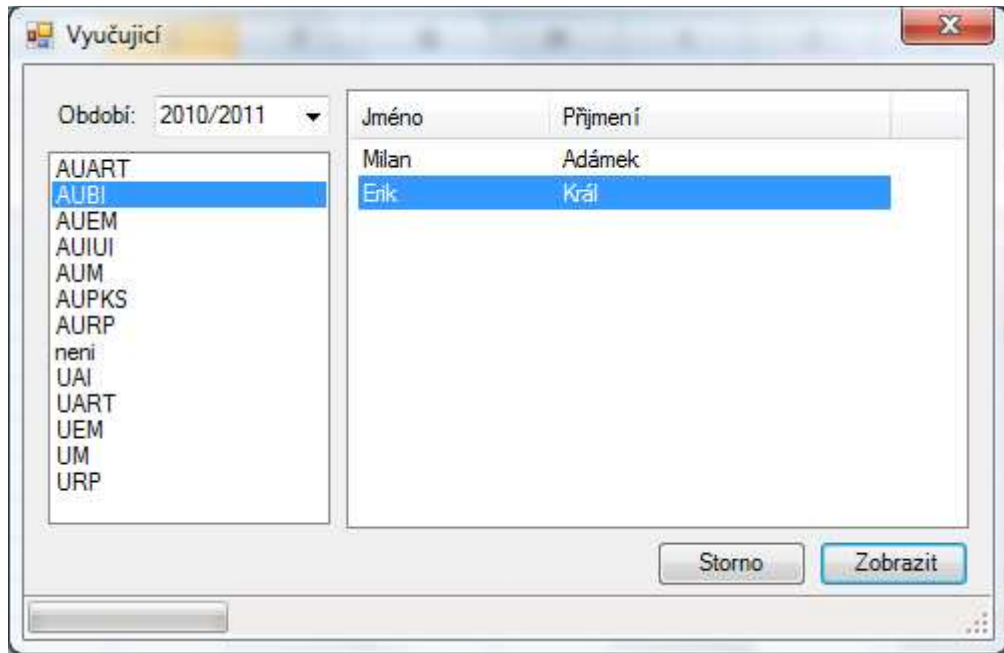
Obrázek 24: Přihlašovací dialogové okno

Po správném zadání přihlašovacích údajů, se první skupina změní na „Uživatelský panel“ s možností odhlásit (Obrázek 25.1).



Obrázek 25: Doplňek po přihlášení uživatele.

Po stisku tlačítka „Úvazek“ (Obrázek 25.2) sloužící pro zobrazení karty vyučujícího, se otevře nové dialogové okno s možností výběru oddělení a učitele (Obrázek 26).



Obrázek 26: Dialogové okno pro nastavení voleb karty vyučujícího

Po potvrzení okna se do Microsoft Excel vyplní karta učitele zobrazující vyučované předměty, zkoušení a vypočet celkového úvazku:

Uživatelský panel		Vyučující		Výuka						
E7		fx		2						
	A	B	C	D	E	F	G	H	I	J
1	<b>Vyučující:</b>	Ing., Král Erik				<b>Procent úvazku:</b>			<b>112</b>	
2	<b>Úvazek (ZH):</b>	900				<b>Úvazek ZH:</b>			<b>1015</b>	
3										
4	<b>A) Přímá výuka</b>								<b>Celkem:</b>	<b>756</b>
5	<b>Předmět</b>	<b>Týdnů</b>	<b>PŘ</b>	<b>CV</b>	<b>SE</b>	<b>PŘ</b>	<b>CV</b>	<b>SE</b>	<b>A</b>	
6	A7HRP	14	2	0	2	0	0	1		56
7	A5PGR	14	1	0	2	0	0	2		112
8	A4OPR	14	0	0	3	0	0	2		168
9	A2PRG	14	0	1	3	0	0	5		420
10										
11	<b>B) Zkoušení a tutoring</b>								<b>Celkem:</b>	<b>60</b>
12	<b>Předmět</b>								<b>B</b>	
13	A7HRP	kbBT								6
14	A5PGR	kbBT								12
15	A4OPR	kbBT								12
16	A2PRG	pbIRT								30
17										

Obrázek 27: Výstup doplňku – karta vyučujícího

Druhá skupina „Výuky“ zobrazuje karty oddělení, nabízí možnost vybrat období a oddělení. Po stisku tlačítka „Zobrazit výuku“ (Obrázek 25.4) se sešit vyplní:

Uživatelský panel		Vyučující		Výuka					
Odhlásit se		Úvazek		Období 2010/2011	Zobrazit výuku				
				Oddělení AUART					
K9									
	A	B	C	D	E	F	G	H	I
1	A4AUT	Cvičení	Skupin	Vyučující					
2		kdII-EN		5	2	Milan Adámek			
3					2	Michal Brázda			
4					1	Josef Čejka			
5		pbBT		1	1	Milan Adámek			
6		Seminář	Skupin	Vyučující					
7		pbBT		1	1	Milan Adámek			
8		Zkoušení	Skupin	Vyučující					
9		kdII-EN		5	2	Milan Adámek			
10					2	Michal Brázda			
11					1	Josef Čejka			
12		pbBT		2	1	Milan Adámek			
13									

Obrázek 28: Výstup doplňku – karta oddělení

## ZÁVĚR

Cílem práce bylo vytvoření doplňku do aplikace Microsoft Excel 2007 pro implementaci podpory rozvrhování výuky. Doplňěk umí vytvářet dvě pohledové sestavy – karta vyučujícího a karta oboru. Tedy zobrazení vyučovaných předmětů ve vybraném období a výpočet procentuálního úvazku vyučujícího z přímé výuky, zkoušení a tutoringu a ostatních pedagogických činností. Druhá karta informuje o předmětech ústavu, kým je předmět vyučován, počet skupin a počet zkoušených studentů v rámci oborů fakulty.

Práce je rozdělena do dvou částí – teoretické a praktické. V teoretické části rozebírám vytvoření WCF služby v programovém prostředí Visual Studio 2010 v programovacím jazyku C#. V druhé části popisuji vytvoření funkčního doplňku, který komunikuje se službou a vypisuje karty zaměstnanců a oddělení v sešitu Microsoft Excel 2007.

Služba zpracovává veškerou logiku aplikace a vytváří datové kontejnery pro zobrazení obou karet na straně klienta. Pro přístup k databázové vrstvě využívá Entity Framework a LINQ. Doplňěk neslouží pro editaci dat, tedy služba nabízí operation contract pouze pro vrácení dat.

Do Microsoft Excel 2007 jsem vytvořil novou záložku v pásu karet tzv. Ribbon, komunikující s vytvořenou službou. Doplňěk zobrazuje požadované karty po stisku ovládacích prvků v sešitu aplikace. Pro zabezpečení informací je pro ovládání doplňku nutné přihlášení pomocí přihlašovacího jména a hesla prostřednictvím univerzitního systému Novell, po správném zadání se ovládací prvky odemknou.

Protože doplňěk je integrován do známé aplikace Microsoft Excel 2007, jeho ovládaní je velice zjednodušeno a uživatel není nucen se učit nový software. Vytvořený doplňěk je určen pro sekretářky, tajemníky a vedoucí ústavu. Editace dat se provádí v samostatné aplikaci [12] využívané pro sestavení výuky pro další akademický rok. Microsoft Excel nepodporuje v sešitech složitější struktury dat, ale pouze vypsání řetězce na určité místo v tabulce, proto není vhodný pro editaci dat.

## CONCLUSION

The aim of the thesis is to create add-in for Microsoft Excel 2007 used for implementation support education scheduling. Add-in can create Teacher view and Department view. Teacher view contains list of taught subjects in selected school year, calculates working hours from teaching, examining and other educational activities. Department view informs about subjects at selected department, who teaches these subjects and how many groups and students are taught.

Thesis has two parts theoretical and practical. In the theoretical part I describe how to create WCF service and add-in for Microsoft Excel in Visual Studio 2010. At the practical part I explain how to create add-in that displays data of view through WCF service.

WCF Service processes all data from database and sends collection of data contract to client through HTTP. Service uses LINQ to Entity to access SQL database. Add-in does not allow editing data because service provides read-only operation contract.

I have created a new tab in ribbon bar for Microsoft Excel, which communicates with service and receives required data which are shown in sheet. Ribbon contains a group for user name and password authorization. After user name and password are filled in correctly other groups get unlocked.

Working with Microsoft Excel add-in is uncomplicated because users know the application and they do not have to learn new software interface. The add-in is ordered to be used by secretaries or heads of department. Data are edited at Silverlight application [12]. This application accesses to the same database source. Microsoft Excel spreadsheet supports to display only string, integer or double so the spreadsheet is not suitable for data editing.

**SEZNAM POUŽITÉ LITERATURY**

- [1] BUSTAMANTE, Leroux Michele. *Learning WCF*. First edition. United State of America : O'Reilly, 2007. str. 582.
- [2] MSDN: *Visual C# Developer Center* [online]. 2011 [cit. 2011-05-01]. Dostupné z WWW: <<http://msdn.microsoft.com/cs-cz/>>.
- [3] W3C: *Latest SOAP versions* [online]. 2007 [cit. 2011-05-09]. Dostupné z WWW: <<http://www.w3.org/TR/soap/>>.
- [4] GRIFFITH, Ian, ADAMS, Matthew a LIBERTY, Jesse. *Programming C# 4.0: Building Windows, Web, and RIA Applications for the .NET*. Sixth Edition. United State of America : O'Reilly, 2010. str. 830.
- [5] ALBAHARI, Joseph a ALBAHARI, Ben. *LINQ Pocket Reference*. First edition. Canada : O'Reilly, 2008. str. 164.
- [6] JANÍČEK, Miloš. *Zabezpečení Windows Communication Foundation*. Zlín, 2010. 91 s. Diplomová práce. Univerzita Tomáše Bati.
- [7] SELLS, Chris. *Windows forms programming in C#*. United States of America : Addison-Wesley, 2004.
- [8] FREEMAN, Adam. *Introducing Visual C# 2010*. United States of America : Apress, 2010.
- [9] BEAULIEU, Alan. *Learning SQL*. Second Edition. United States of America : O'Reilly, 2009. str. 320
- [10] KOSEK, Jiří. *Kosek* [online]. 2008 [cit. 2011-05-01]. Webové služby. Dostupné z WWW: <<http://www.kosek.cz/vyuka/4iz228/prednasky/xml/foil10.html>>.
- [11] HENNEY, Kevlin. *97 Things Every Programmer Should Know*. First edition. United States of America : O'Reilly, 2010. str. 229.
- [12] ČÁPEK, Petr. *Informační systém pro podporu řízení, správu a zjišťování aktuálního stavu rozvrhované výuky*. Zlín, 2011. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

.NET	Prostředí potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak potřebné knihovny.
API	Application program interface
HTML	HyperText Markup Language
HTTP/HTTPS	Hypertext Transfer Protocol / HTTP Secure
LINQ	Language Integrated Query
MS	Microsoft
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
VS 2010	Visual Studio 2010
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
WSDL	Web Service Definition Language
WYSIWYG	What you see is what you get (co vidíš, to dostaneš)
XML	Extensible Markup Language

**SEZNAM OBRÁZKŮ**

Obrázek 1: Princip služby [10] .....	12
Obrázek 2: Změna nastavení pro správné vracení výjimek [2] .....	18
Obrázek 3: Vytvoření nové služby .....	19
Obrázek 4: Testovací klient .....	19
Obrázek 5: Přidání službu do projektu .....	21
Obrázek 6: Okno - Add Service Reference .....	21
Obrázek 7: Okno Service Reference Setting – Advanced .....	22
Obrázek 8: Dialogové okno pro vytvoření doplňku .....	24
Obrázek 9: Ribbon v Microsoft Excel .....	24
Obrázek 10: Entity: Přidání nového souboru.....	30
Obrázek 11: Entity: vybraní typu modelu.....	31
Obrázek 12: Entity: Připojení na databázi .....	32
Obrázek 13: Entity: Výběr tabulek, pohledů a procedur pro export.....	33
Obrázek 14: Návrh grafického rozhraní – karty vyučujícího .....	37
Obrázek 15: Návrh grafického rozhraní – karty ústavu.....	38
Obrázek 16: Struktura adresáře služby .....	39
Obrázek 17: Diagram třídy <code>DataContracts.Teacher</code> .....	41
Obrázek 18: Diagram třídy <code>Model.Teacher</code> .....	41
Obrázek 19: Adresářová struktura Excel Add-in.....	43
Obrázek 20: Dialogové okno Vyučující .....	43
Obrázek 21: Vytvořený Ribbon – neautorizovaný uživatel.....	46
Obrázek 22: Vytvořený Ribbon – autorizovaný uživatel .....	46
Obrázek 23: Výchozí podoba načteného doplňku .....	49
Obrázek 24: Přihlašovací dialogové okno .....	49
Obrázek 25: Doplněk po přihlášení uživatele.....	49
Obrázek 26: Dialogové okno pro nastavení voleb karty vyučujícího.....	50
Obrázek 27: Výstup doplňku – karta vyučujícího .....	50
Obrázek 28: Výstup doplňku – karta oddělení .....	51

**SEZNAM TABULEK**

Tabulka 1: Přehled kódování a protokolů služby [1] [2] [6] .....	16
Tabulka 2: Ribbon controls [2] .....	25

## SEZNAM PŘÍLOH

Zdrojové kódy doplňku a služby uloženy na přiloženém CD.