JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Katedra informatiky

Služby ve Windows Communication Foundation

Bakalářská práce

Petr Ryska

Vedoucí práce: Ing. Václav Novák, Csc.

2010

Prohlášení

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

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

V Českých Budějovicích dne 20. 4. 2010

.....

Petr Ryska

Anotace

Hlavním záměrem této práce je seznámit zájemce s novou perspektivní technologií Windows Communication Foundation (WCF) od společnosti Microsoft. Vše srozumitelnou, přijatelnou a ucelenou formou na příkladech. V několika lekcích se student seznámí s tvorbou a nastavením aplikací klient – server vytvořených pomocí WCF. Ve výukovém materiálu také zohledním mé poznatky při studiu WCF.

Abstract

The main aim of this work is to introduce an candidates with a new perspective technology Windows Communication Foundation (WCF) from Microsoft Corporation. All in understandable and acceptable form of comprehensive examples. In several lessons, student will learn about creating and setting the application client - server created with WCF technology. In the teaching material, take into account my knowledge in the study of WCF.

Poděkování

Tímto bych rád poděkoval vedoucímu této bakalářské práce, Ing. Václavu Novákovi, Csc., za odborné vedení, připomínky, praktické rady a celkově za veškerý čas, který mi věnoval. Mé poděkování patří i všem respondentům, kteří se ochotně podíleli na testování materiálů pro tuto práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam příkladů

Seznam grafů

1 Úvod

V dnešním moderním a uspěchaném světě se již nelze obejít bez komunikace. V oblasti informačních technologií by se dalo říci, že komunikace je dokonce stěžejním aspektem. Jak již z názvu vyplývá Windows Communication Foundation (dále jen WCF) se zabývá právě komunikací a proto jsem si tuto technologii zvolil jako téma bakalářské práce. Pro tvorbu výukových lekcí jsem se rozhodl, protože mne technologie WCF velice zaujala a chtěl jsem proniknout do jejích tajů, ale bohužel české zdroje mi nebyly příliš nakloněny a strávil jsem spoustu vzácného času pročítáním různých anglických textů.

WCF je robustní technologie využívající moderních přístupů a metod v programování distribuovaných aplikací. WCF představuje jednotný programovací model pro psaní aplikací typu klient-server. Zároveň nabízí obrovské možnosti ve výběru různých komunikačních protokolů pro komunikaci mezi aplikacemi. WCF nabízí jakési sjednocení, standardizování a univerzálnost psaní distribuovaných aplikací předem předurčených pro přenos informací. Propojení WCF s .NET je na opravdu vysoké úrovni, tyto technologie jsou velice úzce spjaty a zároveň se doplňují. WCF je spolu s dalšími zajímavými technologiemi k dispozici v .NET Frameworku ve verzi 3.0 a vyšší.

Cílem mojí práce tedy bude poskytnout zájemci o tuto technologii teoretický základ pro pochopení její vlastní funkce a na příkladech ukázat možné použití této teorie v praxi. Již od počátku jsem se smířil se skutečností, že nemohu obsáhnout veškerá úskalí tvorby aplikací pomocí této technologie, je to z důvodu obsáhlosti WCF a jednoduše nepostačují kapacity této práce na kompletní výuku. Výuka bude probíhat formou výukových lekcí. Pro bezproblémový průchod lekcemi se předpokládá znalost některého z objektově orientovaných jazyků (nejlépe C#).

Lekce doplňuji o vzorové řešené i neřešené příklady a videoukázky pro ještě názornější pochopení právě probírané problematiky. Právě vysvětlování problematiky na ukázkovém příkladě je základem metodiky. Příklad je navíc v lekcích doplněný o videoukázku se zvukovou stopou, jak problematiku krok za

krokem zvládnout. Lekce dělím na dvě části, přičemž první se věnuje úplným začátečníkům a druhá lehce pokročilým zájemcům o technologii WCF. Ve výuce postupuji od nejsnazšího po složitější. První část lekcí na sebe přímo navazuje a po jejich dokončení zájemce získává ucelenou distribuovanou aplikaci složenou z hostované služby (server) a klientské aplikace. Druhá část lekcí předpokládá znalosti veškeré problematiky obsažené v části první, jednotlivé lekce v ní na sebe nenavazují a mohou se absolvovat v různém pořadí.

Vytvořený výukový materiál nabídnu k vyzkoušení menší skupině lidí, která mi formou dotazníku navrátí zpětnou vazbu o jeho kvalitě. Výsledky z testu výukového materiálů samozřejmě zveřejním v této práci.

2 Windows Communication Foundation

2.1 Specifikace WCF

Microsoft měl snahu o sjednocení svých doposud vydaných technologií pro tvorbu komunikačních řešení, jako jsou například Web Services Enhancements (WSE), .NET Remoting atd. Podobně jako se mu dříve podařilo sjednotit různé programovací jazyky v platformě .NET, pod kterou WCF také spadá. Doposud bylo nutné si ujasnit, jaké máme požadavky na funkčnost vytvářeného systému a podle těchto kritérií si zvolit předem komunikační technologii, protože každá ze zmiňovaných technologií měla své určité výhody a nevýhody. WCF není (podle [6], str. 1455) závislá na platformě stejně jako webové služby ASP.NET a nabízí podobné funkce a vlastnosti jako technologie .NET Remoting, služby Enterprise Services či Message Queuing.

Pokud byla například potřeba použít více technologií, tak řešení takové situace nebylo zrovna snadné. Jednou z možností jak tuto potřebu řešit bylo programovat jednotlivé části distribuovaného systému různými technologiemi, což bylo velmi nepraktické. Problémem byla i snaha o změnu doposud používané technologie, kdy většinou následovalo přepracování celé aplikace. Navíc každá technologie vyžadovala od vývojáře odlišné dovednosti pro tvorbu aplikací.

WCF umožňuje psát aplikace pro komunikaci, zpracování či výměnu dat jak pro Windows, tak pro jakékoliv jiné platformy, které podporují nějaký z otevřených standardů pro výměnu dat, např. SOAP. Při vývoji technologie WCF byl kladen důraz na to, aby byla velice obsáhlá a používala propracované a moderní servisně orientované programování. To je totiž přístup, na kterém celé WCF staví. Dalším důležitým prvkem bylo, aby tato technologie obsahovala širokou škálu možností nastavení a celkově nebylo příliš složité pochopit problematiku tvorby distribuovaných aplikací pomocí WCF.

2.1.1 Co přináší WCF

WCF (kapitola dle [6], str. 1456) kombinuje funkčnost webových služeb ASP.NET, technologie .NET Remoting, výměny zpráv (Message Queuing) a služby Enterprise Services. Tato jediná technologie tedy přináší hned několik výhod různých technologií:

- **Hostování komponent a služeb:** Stejně jako je možné využívat vlastní hostitele v rozhraní .NET Remoting a WSE, je při komunikaci mezi dvěma rovnocennými počítači (peer-to-peer) možné ve WCF hostovat služby v ASP.NET, ve službě Windows, procesu COM+, nebo v nějaké aplikaci pro Windows.
- **Deklarativní chování:** Atributy není nutné odvozovat od základní třídy (jako u .NET Remoting a služeb Enterprise Services), ale lze jimi definovat služby. Podobně fungují webové služby vytvořené v technologii ASP.NET.
- **Komunikační kanály:** Technologie .NET Remoting je při změnách komunikačních kanálů velice tvárná, přičemž WCF nabízí výbornou alternativu, protože je stejně flexibilní. WCF nabízí pro komunikaci pomocí protokolů HTTP a TCP více kanálů současně i s kanálem IPC. Dokonce lze pro transportní protokoly vytvořit své vlastní kanály.
- **Bezpečnostní infrastruktura:** Pokud je potřeba implementovat na platformě nezávislé webové služby, je k tomu potřeba využít standardizované bezpečnostní prostředí. Tyto navržené standardy jsou obsaženy ve WSE verze 3.0, na kterou WCF přímo navazuje.
- **Možnosti rozšíření:** .NET Remoting má v oblasti možností rozšiřování vcelku bohaté kořeny. Tato technologie umožňuje si vytvořit vlastní kanály, nebo formátovací nástroje, či proxy. Dále také dovoluje vložit další funkce do zpráv klienta či serveru. WCF disponuje podobnými možnostmi rozšíření, které se zde provádí pomocí hlaviček SOAP.

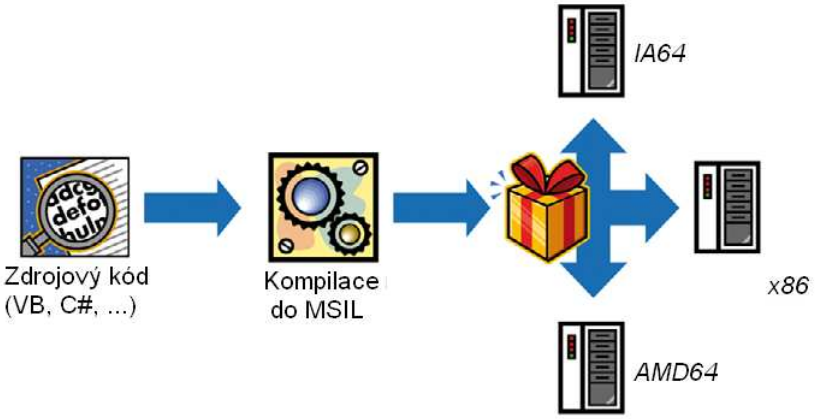
- **Podpora dřívějších technologií:** Technologie WCF je kompatibilní s dříve vydanými a používanými technologiemi, což z ní dělá velice mocný nástroj. Místo kompletního přepisování již existujícího distribuovaného řešení do prostředí WCF je možné WCF integrovat přímo do tohoto řešení vyvinutém v odlišné technologii. WCF poskytuje například kanál, který umí prostřednictvím protokolu DCOM komunikovat se spravovanými komponentami. V technologii WCF lze integrovat i webové služby vyvinuté v ASP.NET.

2.1.2 WCF v .NET Frameworku

Vývojová platforma Microsoft .NET Framework je (čerpáno z [16], str. 8) integrovaná součást v podporovaných operačních systémech Microsoft Windows pro tvorbu, ladění a běh aplikací nové generace a XML webových služeb. Zajišťuje vysoce produktivní, na prověřených a patentovaných standardech založené vývojové prostředí, v němž může vedle sebe společně existovat a spolupracovat více programovacích jazyků pro snadnější podporu při vývoji různorodých počítačových aplikací. .NET Framework také umožňuje vývojářům využívat zkušenosti již nabyté při dřívějším vývoji aplikací a tím, že sjednocuje vlastnosti různých jazyků a platforem výrazně usnadňuje proces programové integrace do již existujících softwarových řešení, modulů či komponent. Programátorům jsou zcela jistě také nápomocny nepřehlédnutelné a užitečné vestavěné inovace pro vývoj, či udržování internetových aplikací, nebo pro vytváření a úpravy distribuovaných řešení. Celá vývojářská platforma .NET Framework staví na dvou základních pilířích, kterými jsou společné prostředí pro běh vytvořených řešení CLR (Common Language Runtime) a jednotné hierarchické uspořádání knihoven s třídami. Tyto knihovny zapouzdřují třídy například pro vývoj výkonných webových aplikací, inteligentních klientských aplikací pro Windows a databázových aplikací.

Platforma .NET Framework umožňuje vývojářům vyvíjet své aplikace v mnoha programovacích jazycích, protože tato platforma jich podporuje celou řadu. Ať se vývojář rozhodne pro kterýkoliv z podporovaných jazyků, finálním

výstupem z .NET kompilátoru bude vždy univerzální jazyková mezivrstva, jejíchž kód je ve formátu jazyka MSIL (Microsoft Intermediate Language). Právě díky této jednotné vrstvě, mohou dvě aplikace napsané v rozdílných dostupných jazycích z široké palety .NET bez potíží spolupracovat. Pro spravování kódu jazyka MSIL je zde již zmiňované společné běhové prostředí CLR zajišťující překlad do nativního kódu srozumitelného instrukční sadě určitého procesoru počítače. Tento proces demonstruje obrázek 1. Po uvedení tohoto prostředí mají vývojáři naprosto volnou ruku při volbě, který z programovacích jazyků pod platformou .NET Framework si vybrat. Jednoduše řečeno, programátoři si teď mohou vybrat „ten svůj“ jazyk a to plně s ohledem na své preference či charakter řešených úloh.

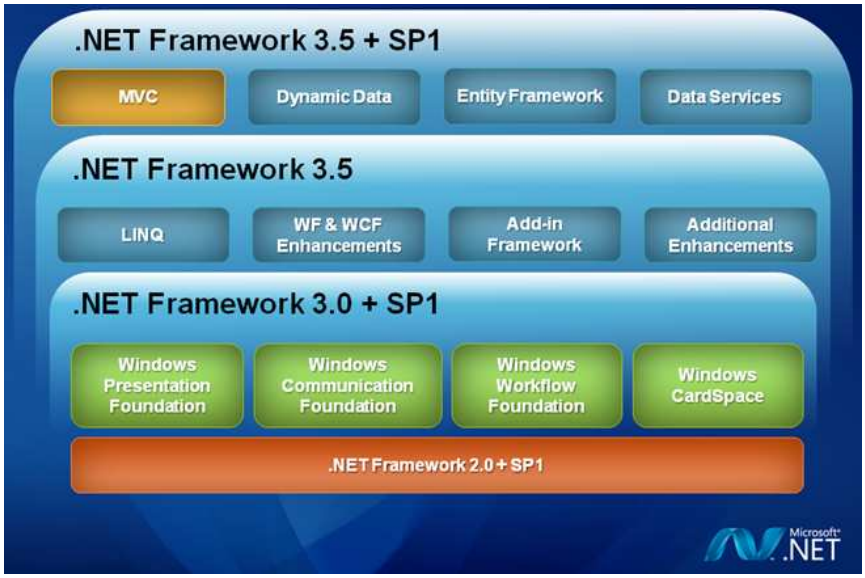


Obrázek 1: Překlad zdrojového kódu do MSIL a instrukční sady procesoru (upraveno z [12], str. 4)

Vývoj .NET Framework je zdařile popsán v [12], str. 4. S příchodem nové platformy .NET Framework 3.0 přivedla firma Microsoft na svět celou řadu nových užitečných technologií. Její uvedení se datuje na listopad 2006. Nové revoluční technologie byly tyto čtyři:

- **Windows Presentation Foundation (WPF)** je technologie grafického subsystému pro tvorbu prezentační vrstvy s bohatou podporou grafiky a multimédií. Někdy se o ní mluví jako o nástupci klasické technologie Windows Forms.

- **Windows Communication Foundation (WCF)** je základním technologickým pilířem pro tvorbu distribuovaných aplikací orientovaných na služby využívající filozofie a principů Servisně-Orientované Architektury (SOA).
- **Windows Workflow Foundation (WF)** slouží k efektivnímu modelování a řízení procesových toků.
- **Windows CardSpace** je subsystém pro správu digitálních identit.



Obrázek 2: Novinky v platformě .NET Framework (převzato z [17])

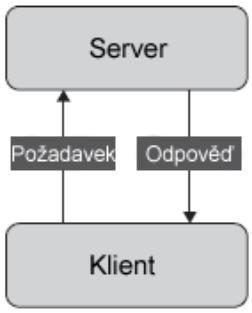
O .NET Frameworku se mluví jako o nástupci zastaralého Win32 API (Application Programming Interface). Pro běh WCF postačuje verze 3.0, avšak doporučuji instalovat nejnovější stabilní verzi a tou je 3.5 SP1, existuje i verze 4.0 ovšem prozatím v testovací verzi beta. Vyšší verze 3.5 SP1 obsahuje i různá rozšíření, která se mohou také hodit, jako je například možnost vývoje aplikací pro mobilní zařízení, či využívání nových standardů JSON, WS*-Standards apod.

Různé verze operačních systémů Windows obsahují automaticky instalovanou součást v podobě různé verze .NET Frameworku. Doporučuji tedy nejprve zkontrolovat výchozí verzi obsaženou v operačním systému a poté podniknout kroky k její aktualizaci či reinstalaci.

2.1.3 Servisně-Orientovaná Architektura

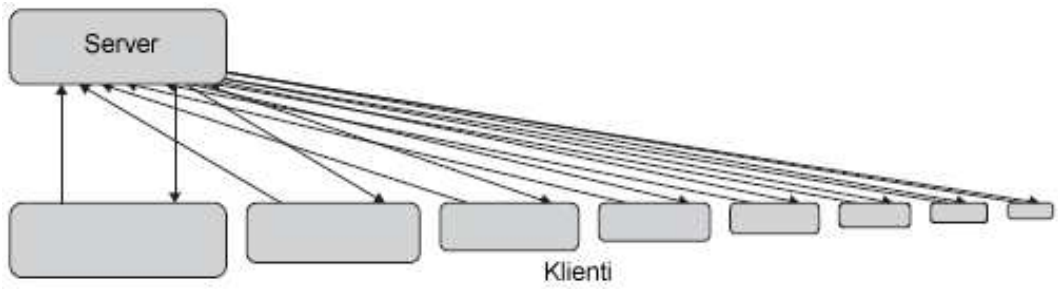
(kapitola podle knihy [6], str. 1626) V dnešní době se tradičně pracuje v síťových aplikacích (včetně internetu) využívajících architekturu klient-server. Demonstrující tuto skutečnost jsou pak webové služby či webové stránky. Pokud si totiž prohlížíte nějakou určitou stránku, posílá se z vaší stanice přes síť internet tzv. žádost na webový server, který ji zpracuje, chcete-li vyhodnotí a zpět Vám odešle odpověď v podobě nějakých požadovaných informací či určitého požadovaného souboru.

Podobně tomu je i u desktopových aplikací, které pracují s nějakou lokální či rozsáhlou sítí. Také se obvykle připojují k serveru ať již v podobě nějaké databáze či serveru s dalšími službami. Tento způsob komunikace znázorňuje obrázek 3.



Obrázek 3: Komunikace architetury klient-server (upraveno z [6], str. 1626)

Na architektuře typu klient-server není v zásadě nic špatného a ve většině případů vyhovuje všem běžným nárokům. Ovšem u této architektury je problém se škálovatelností. Škálovatelnost architektury typu klient-server znázorňuje obrázek 4.



Obrázek 4: Škálovatelnost architektury klient-server (upraveno z [6], str. 1626)

S každým nově připojeným a komunikujícím klientem požadujícím odpověď se zvyšuje zátěž serveru, který musí se všemi klienty komunikovat a obsluhovat je. Pokud se vrátíme k příkladu s webovými stránkami, tak právě toto je problém

občasného kolabování stránek. Pokud se v určitém okamžiku připojí k serveru velké množství uživatelů (klientů), tak se zvýší nároky na režii serveru a ten je tzv. přetížen. Má to za následek, že server jednoduše přestane odpovídat.

Existuje samozřejmě velké množství možností a technik, pomocí kterých je možné podobnou situaci alespoň zmírnit. Je možné škálovat zvýšením výkonu a zdrojů pro server, nebo lze přidat další servery a zátěž tak rovnoměrně rozložit na více serverů. Škálování pomocí zvyšování výkonu je pochopitelně omezeno a má určité hranice v podobě dostupné technologie a ceny tohoto dostupného hardwaru. Škálování pomocí rozložení zátěže mezi více serverů je sice potenciálně pružnější, ale vyžaduje infrastrukturu, která dokáže zajistit, aby mohli klienti komunikovat s jednotlivými servery a zároveň aby se dokázal uchovat jejich stav bez ohledu na server, se kterým právě komunikují. Pro tuto skutečnost existuje mnoho řešení, například webové či serverové farmy.

2.1.4 Interoperabilita a používání formátu XML, SOAP, JSON a WSDL

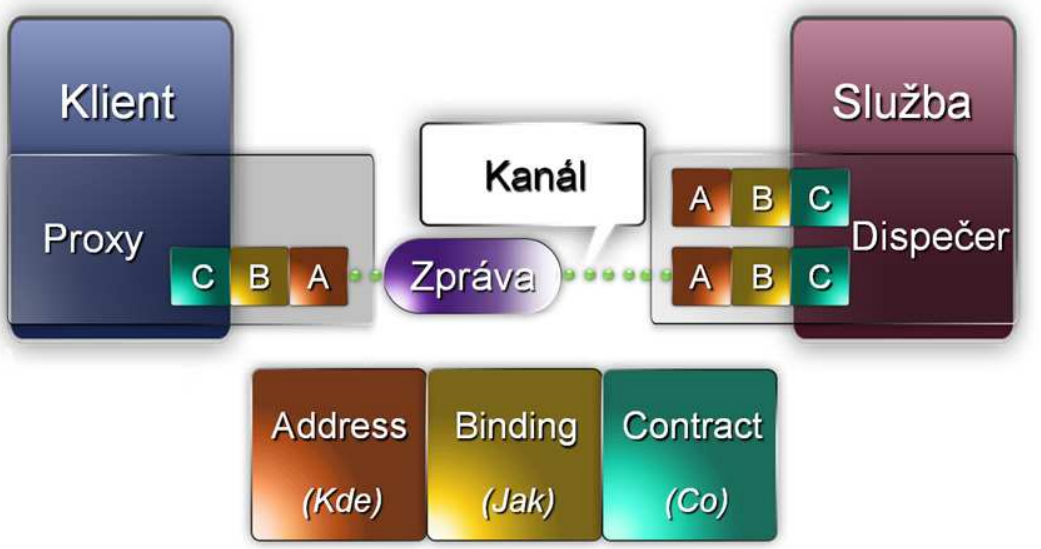
Základním klíčovým faktem pro zpřístupnění služeb napříč různými platformami (Web, apod.) je dohodnutí jednoduchého a jednotného formátu pro popis dat. Tímto vhodným formátem je XML. (dle [15], str. 118 a [6], str. 1457) Služby potřebují formát XML konkrétně k popisu základních věcí jako je přenosový formát, samotný popis služeb apod.

- **Simple Object Access Protocol (SOAP):** Na nejnižší úrovni komunikace potřebují různorodé systémy provádět výměnu dat prostřednictvím nějakého společného jazyka. SOAP v podstatě tvoří základní vrstvu komunikace a poskytuje základní prostředí pro tvorbu složitější komunikace. WCF tento protokol přímo podporuje. Komunikující aplikace musí mít sadu konkrétních pravidel určujících, jak budou vypadat reprezentace různých datových typů (celá čísla, řetězce, pole atd.) a jak budou reprezentovány patřičné operace s těmito daty. SOAP a implementace XML představují jednu společnou množinu těchto pravidel.

- **JavaScript Object Notation (JSON):** Pokud je potřeba přistupovat k WCF službám prostřednictvím jazyka JavaScript, je možné využívat pro odesílání zpráv namísto protokolu SOAP jiný formát pro výměnu dat, kterým je formát JSON. Od verze .NET Framework 3.5 je obsažen serializátor datových kontraktů, díky němuž lze vytvářet objekty zapsané formátem JSON. Hlavní výhodou JSON je, že není tak náročný na výkon jako SOAP, protože nepoužívá XML, ale je optimalizován pro klienty používající JavaScript. Nachází své uplatnění například na ajaxových klientech. Co se týče spolehlivosti zabezpečení a transakcí, tak JSON sice není v tomto ohledu tak propracovaný jako SOAP, avšak tyto funkce nejsou většinou na klientech využívajících jazyk JavaScript potřeba.
- **Web Service Description Language (WSDL):** Pokud tedy mají aplikace obecná pravidla pro reprezentaci datových typů a operací, potřebují najít způsob popisu těchto pravidel, které jsou schopny přijmout. WSDL v podstatě popisuje, co nabízí webová služba za funkce a způsob, jak se jí na to zeptat (definice zpráv služby). Zapisuje se v XML formátu. Dalo by se říci, že popisuje SOAP komunikaci. WSDL je tedy soubor pravidel pro XML popis, který mohou vývojáři a vývojové nástroje používat k popisu webové služby. Popis služby je reprezentován pomocí Metadat, s jejichž pomocí je možné v klientské aplikaci vytvořit komunikační proxy třídu. Ve WCF existuje pro poskytování informací o WSDL speciální endpoint nazývaný jako MEX (Metadata Exchange) endpoint.

2.2 Základy WCF

Při výměně dat (komunikaci) se ve WCF zpravidla jedná o vzájemné zasílání zpráv mezi klientem a službou. Tato komunikace probíhá buď mezi procesy nebo různými systémy, po místní síti či Internetu. Pro zajištění provádění těchto operací nezávisle na platformě a při nejvyšší možné rychlosti vyžaduje WCF (podle [6], str. 1456] dodržení určitých zásad. Vzdálená služba musí nabízet popsaný koncový bod (endpoint) obsahující kontrakt, datové vazby a adresu. Aby měl ke službě klient přístup a mohl s ní komunikovat, je potřeba mu také vytvořit kompatibilní koncový bod.



Obrázek 5: Znázornění komunikace ve WCF (upraveno z [18])

Na obrázku 5 jsou znázorněny komponenty podílející se na komunikaci ve WCF. Klient volá metodu obsaženou v proxy. Proxy pak nabízí metody definované ve službě, ale samotné volání metody převádí na zprávu a tu pak odesílá prostřednictvím kanálu. Kanál se skládá ze dvou částí, a sice klientské a serverové, které spolu komunikují prostřednictvím nějakého síťového protokolu. Zpráva z kanálu pak dále putuje do části zvané dispečer, ten zprávu překládá do tvaru volání metody, které služba provede.

WCF podporuje hned několik různých komunikačních protokolů. Pro umožnění komunikace mezi platformami podporuje webové služby. Ovšem pro komunikaci mezi aplikacemi pod .NET je možné využívat i některé rychlejší a méně zatěžující komunikační protokoly.

2.2.1 Endpoint

Endpoint je základem pro komunikaci ve WCF, který by se měl vrýt každému vývojáři do paměti a jeho definice by se dala vyjádřit třemi výrazy Adress, Binding a Contract (tzv. abeceda). Za základní kámen je považován hlavně z důvodu jeho nutnosti u každého komunikujícího prvku ve WCF, tedy jak na straně klienta, tak na straně služby. WCF na této koncepci komunikace staví a přísně vyžaduje její dodržování. Kontrakt definuje operace nabízené službou, vazba poskytuje informace o protokolu a použitém kódování a adresa představuje umístění služby. Jako většina nastavení, tak i endpoint lze ve WCF nastavit dvěma způsoby, buď v konfiguračním souboru s příponou config, nebo přímo v kódu aplikace.

```
private ServiceHost host;
private void StartService()
{
    Uri address = new Uri("http://localhost:8080/ShopService");
    host = new ServiceHost(typeof(ShopService));
    BasicHttpBinding binding = new BasicHttpBinding();
    host.AddServiceEndpoint(typeof(IShopService), binding,
address);
    host.Open();
}
```

Příklad 1: Konfigurace endpointu přímo v kódu aplikace

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="Shop.ShopService">
        <endpoint address="http://localhost:8080/ShopService/"
                  binding="basicHttpBinding"
                  contract="Shop.IShopService" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Příklad 2: Konfigurace endpointu v konfiguračním souboru

2.2.1.1 Address

(čerpáno z [8], str. 111) Každý endpoint musí mít vlastní adresu určující přesné místo, na kterém se služba nachází, naslouchá a tudíž, kde je možné se s ní spojit. Tato adresa musí být vždy jedinečná, pak funguje jako jednoznačné URI (Uniform Resource Identifier) endpointu, což by se dalo volně přeložit jako jednotný identifikátor zdroje.

Struktura stavby adresy je vcelku jednoduchá a v podstatě je nám tento způsob tvorby adresy známý například z webových stránek. Skládá se z definice transportního protokolu, jména počítače či serveru, portu a umístění, kde konkrétně na serveru služba běží. Všechny části kromě portu jsou povinné, navíc port je závislý na použitém protokolu. Pokud nenastavíme port u protokolu vyžadujícího jeho nastavení, tak dojde k inicializování výchozí hodnotou.

- **TCP:** Velice často používaný způsob komunikace. U protokolu typu TCP (Transmition Control Protocol) je vyžadován port, avšak pokud není uveden, použije se výchozí port 808. Pokud se tedy vývojář rozhodne pro některý z NetTcp bindingů, tvar jeho adresy bude následující:

  net.tcp://{hostname}:[port]/{umístění služby}
  net.tcp://localhost/MojeSluzba
  net.tcp://localhost:82/MojeSluzba

- **IPC:** Pro komunikaci protokolu tohoto typu se nepožaduje port. IPC (Inter-process Communication) využívá net.pipe a je určená pro komunikaci mezi procesy na lokálním systému. Nelze tedy s její pomocí komunikovat mezi vzdálenými systémy. Adresa je v následujícím tvaru:

  net.pipe://localhost/{umístění služby}
  net.pipe://localhost/MojePipeSluzba

- **HTTP a HTTPS:** Tento typ adresy je asi nejpoužívanější. Zde je vyžadováno uvedení portu pro komunikaci, ovšem pokud jej neuvedeme, použije se výchozí port 80 pro HTTP a port 443 pro HTTPS. HTTPS je zabezpečená verze HTTP. Tvar jejich adres je následující:

  http://{hostname}[:port]/{umístění služby}
  http://localhost:8094/MojeSluzba
  https://{hostname}[:port]/{umístění služby}
  https://localhost:8094/MojeSluzba

- **MSMQ:** Tento způsob komunikace využívá metodu front zpráv. MSMQ (Microsoft Message Queue) nedovoluje vývojáři změnu nastavení portu pro komunikaci, protože používá pouze jeden jediný port 1801 a ten je neměnný. Adresa tohoto způsobu komunikace využívá net.msmq a obsahuje veřejné či soukromé typy front. Veřejné mohou být přístupné vzdáleně (zvenčí), soukromé pak pouze lokálně. U privátních front se uvádí v adrese private a pokud se jedná o veřejné fronty, tak se neuvádí nic. Tvar adresy pak může vypadat takto:

  net.msmq://{hostname}/[private/]{název fronty}
  net.msmq://localhost/private/MojePrivatniMsMqSluzba
  net.msmq://localhost/MojeVerejnaMsMqSluzba

  Pro MsmqIntegrationBinding má adresa následující formát:

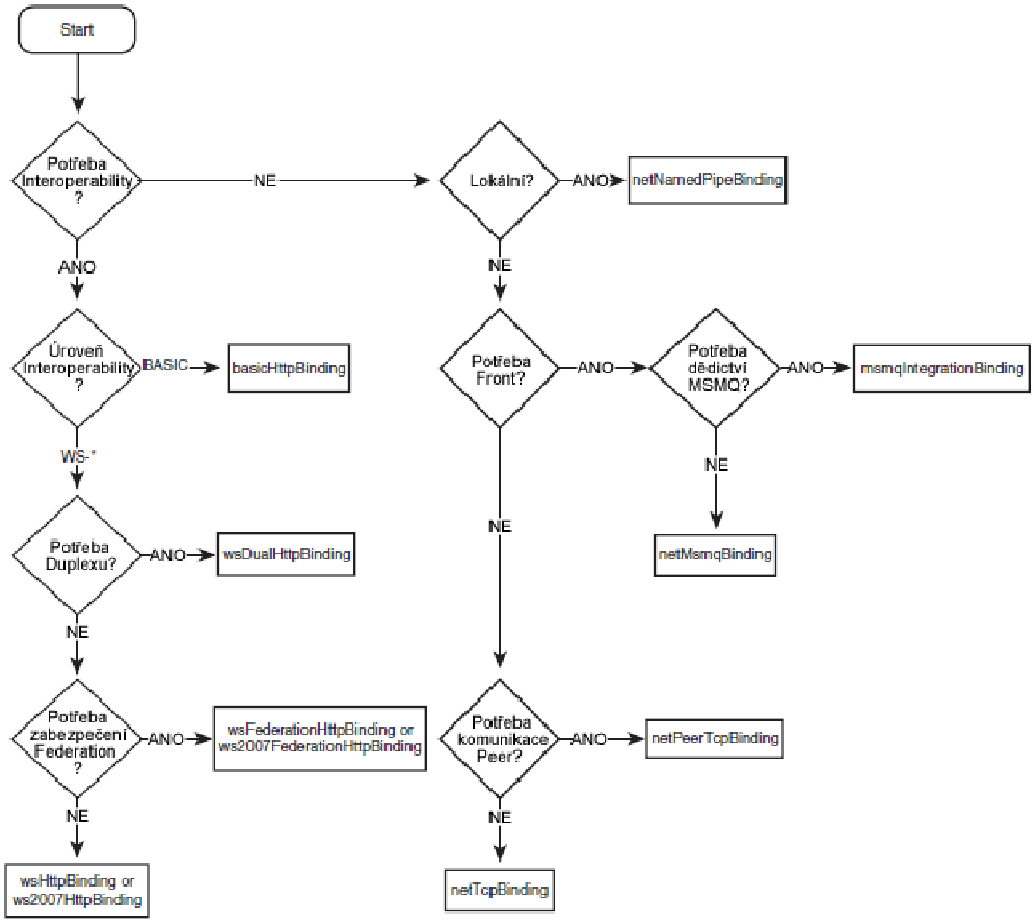
  msmq.formatname:{název MSMQ formátu}
  msmq.formatname:DIRECT=OS:.\\private$\\Orders

- **Peer:** Adresu tohoto typu využívá Peer-to-Peer Networking tedy komunikace způsobem rovný s rovným. Peer vyžaduje nastavení portu, avšak pokud není uveden, použije se výchozí port 0. Adresa je pak v tomto formátu:

  net.p2p://{název mesh}[:port]/{umístění služby}
  net.p2p://MojeMesh/MojeSluzba
  net.p2p://MojeMesh:4242/MojeSluzba

2.2.1.2 Binding

Binding, nebo také vazba je definice způsobu komunikace, zabezpečení přenosu a šifrování dat. Problematika jejího správného výběru je perfektně vysvětlena včetně ukázkových grafů v [8], str. 152. WCF disponuje celou řadou již předdefinovaných bindingů, které v naprosté většině případů bohatě postačují nárokům vývojáře. Popřípadě je možné lehce modifikovat nastavení některého z již předdefinovaných bindingů, nebo si vytvořit svůj vlastní. Výběr bindingu je velice důležitá a zásadní volba, která ovlivňuje chod celé aplikace. Při volbě bindingu je potřeba si upřednostnit, jaké preference od něj očekáváme, zda podporu relací, transakce, či duplexní komunikaci. Ve výběru může být velice nápomocný a přínosný následující diagram.



Obrázek 6: Diagram výběru binding (přeloženo z [8], str. 118)

Typy binding implementované přímo v .NET Framework (přeloženo a převzato z [6], str. 1478):

- **BasicHttpBinding:** Tato vazba je vazbou pro velice širokou interoperabilitu, použitelná i pro první generaci webových služeb. Využívá transportních protokolů HTTP a HTTPS. Jediné její zabezpečení je to, které má v sobě transportní protokol.
- **WSHttpBinding:** Vazba, která je určena i pro další generaci webových služeb, nebo platforem implementujících rozšíření SOAP pro zabezpečení, spolehlivost a transakce. Využívá transportních protokolů HTTP a HTTPS. Pro zabezpečení jsou implementovány specifikace standardu WS-Security. Transakce mají zajištěnou vynikající podporu díky specifikacím implementovaných standardů WS-Coordination, WS-AtomicTransaction a WS-BusinessActivity. Spolehlivé posílání zpráv je zajištěno implementací standardu WS-ReliableMessaging. Další implementovaný standard WS-Profile, zajišťuje posílání příloh pomocí kódování protokolu MTOM (Message Transmisson Optimization Protocol).
- **WS2007HttpBinding:** Je odvozena od třídy základní vazby WSHttpBinding. Podporuje specifikace zabezpečení, spolehlivosti a transakcí definované organizací OASIS (Organization for the Advancement of Structured Information Standards). Třída této vazby je novinkou v .NET Framework 3.0 SP1.
- **WSHttpContextBinding:** Také odvozena od třídy základní vazby WSHttpBinding a i bez používání cookies přidává podporu kontextu. Vazba proto obsahuje element ContextBindingElement, aby bylo možné vyměňovat informace o kontextu.
- **WebHttpBinding:** Tato vazba se využívá pro služby zveřejněné prostřednictvím požadavků HTTP, nikoliv prostřednictvím požadavků SOAP. Její využití je vhodné pro skriptování klientů, například pro technologii ASP.NET AJAX (Asynchronous JavaScript and XML).

- **WSFederationHttpBinding:** Je zabezpečená a interoperabilní vazba, která disponuje mechanismy pro ověření a autorizaci, či podporou sdílení identit mezi více systémy.
- **WSDualHttpBinding:** Tato vazba má vlastnosti jako vazba WSHttpBinding, ovšem navíc podporuje duplexní zprávy.
- **NetTcpBinding:** Vazba využívající binární kódování vhodné pro komunikaci mezi .NET aplikacemi. Vyznačuje se vyšší rychlostí a pro komunikaci používá protokol TCP/IP.
- **NetTcpContextBinding:** Tato vazba je odvozena od základní třídy vazby NetTcpBinding. Podobně jako je tomu u WSHttpContextBinding i tato vazba přidává kvůli výměně kontextu se záhlavím SOAP element ContextBindingElement.
- **NetPeerTcpBinding:** Vazba odvozená od základní třídy NetTcpBinding poskytující spojení aplikacím komunikujícím pomocí peer-to-peer (P2P).
- **NetNamedPipeBinding:** Tato vazba je optimalizována pro komunikaci mezi různými procesy, avšak pouze v jednom systému.
- **NetMsmqBinding:** Tato vazba obohacuje komunikaci WCF o možnost pracovat s frontou zpráv. Zprávy jsou zde seřazovány do již zmíněné fronty.
- **MsmqIntegrationBinding:** Je vazbou pro existující aplikace, které pracují s frontami zpráv. Tato vazba však vyžaduje, aby se aplikace WCF nacházely jak na straně klienta, tak i na straně serveru.
- **CustomBinding:** Vazba sloužící pro kompletní přizpůsobení dle požadavků vývojáře. Dovoluje vybrat a nastavit transportní protokol i bezpečnostní požadavky.

V závislosti na použité vazbě jsou nabízeny určité specifické funkce pro využití v daném prostředí. Vazby začínající prefixem WS nejsou závislé na platformě, jsou lépe zabezpečené než pouze HTTP vazby a odpovídají

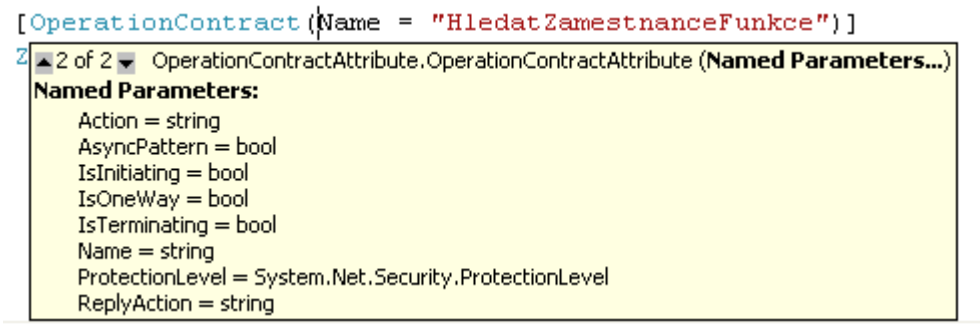
specifikacím webových služeb. Vazby jejichž prefix je Net používají binární kódování pro dosažení rychlejší komunikace mezi aplikace typu .NET. Následující tabulka 1 znázorňuje vlastnosti obsažené v předdefinovaných vazbách.

Název Binding	Transport-Level Security	Message-Level Security	WS-* Interoperabilita	WS-*Transakce	Odolné a spolehlivé zprávy	Spolehlivé Sessions	Výkon	Komunikace		
								Request/Reply	One-way	Duplex
basicHttpBinding	✔	✔	✔	-	-	-	Dobrý	✔	✔	-
wsHttpBinding	✔	✔	✔	✔	-	RS*	Dobrý	✔	✔	-
wsDualHttpBinding	✔	✔	✔	✔	-	RS*	Dobrý	✔	✔	✔
netTcpBinding	✔	✔	-	✔	-	RS*	Lepší	✔	✔	✔
netNamedPipeBinding	✔	-	-	✔	-	-	Nejlepší	✔	✔	✔
netMsmqBinding	✔	✔	-	-	✔	-	Lepší	-	✔	-
netPeerTcpBinding	✔	-	-	-	-	-	Dobrý	-	✔	✔
msmqIntegrationBinding	✔	-	-	-	✔	-	Lepší	-	✔	-
wsFederationHttpBinding	✔	✔	✔	-	-	RS*	Dobrý	✔	✔	-
ws2007HttpBinding	✔	✔	✔	✔	-	RS*	Dobrý	✔	✔	-
ws2007FederationHttpBinding	✔	✔	✔	-	-	RS*	Dobrý	✔	✔	-

Tabulka 1: Vlastnosti předdefinovaných binding (přeložena z [8], str. 117)

2.2.1.3 Contract

Jako poslední část ze tří stěžejních pilířů endpointu uvádím kontrakt. Kontrakt by se dal jednoduše chápat jako definice dat a metod, které služba nabízí. Daty jsou myšleny vytvořené složitější datové typy. Dále může kontrakt definovat operace nabízené službou, tedy operace, které může volat klient. Na základě kontraktu si klient vytváří proxy třídu, pro komunikaci se službou. Není závislý na použité adrese či bindingu. WCF disponuje nejrůznějšími typy kontraktů definované pomocí atributů, které mohou obsahovat různá další nastavení pomocí parametrů. Parametry se píší za název kontraktu do klasické závorky, jak znázorňuje následující obrázek 7.



Obrázek 7: Ukázka parametrů u OperationContract atributu

Datový kontrakt (DataContract)

Pokud je potřeba přenášet (přijímat i odesílat) jiné datové typy než základní (například int, string, bool atd.), musí se definovat DataContract, který popisuje strukturu těchto dat. Dalo by se říci, že definuje námi vytvořené datové struktury ať v podobě třídy, struktury či výčtového typu. Atribut DataContract se píše před hlavičku definující třídu, která bude využita pro přenos. Atribut DataMember se píše před jednotlivé členy, které struktura obsahuje.

```
[DataContract]
public class Osoba
{
    private int RodneCislo;
    private string Name;

 [DataMember]
    public int RodneCisloValue
    {
        get { return Id; }
        set { Id = value; }
    }

 [DataMember]
    public string NameValue
    {
        get { return Name; }
        set { Name = value; }
    }
}
```

Příklad 3: Definice datového kontraktu

Kontrakt služby (ServiceContract)

Tento kontrakt se využívá k popisu služby. Je to ve skutečnosti rozhraní, které implementuje naše třída obsahující definici služby. Toto rozhraní využívá klient, pro výčet metod, které daná služba nabízí. Na základě tohoto kontraktu se vytváří proxy třída na straně klienta. Jak tomu u rozhraní bývá, obsahuje pouze hlavičky metod. Atributem ServiceContract se buď v rozhraní, nebo třídě označuje definice kontraktu služby. Atributy OperationContract se označují hlavičky metod nabízených takovou službou.

```
[ServiceContract]
public interface IShopService
{
    [OperationContract]
    void AddProductToCart(int id, string name, int count);

    [OperationContract]
    List<Product> GetShoppingCart();
}
```

Příklad 4: Definice kontraktu služby

Kontrakt zprávy (MessageContract)

Kontrakt zprávy se využívá v případech, kdy je potřeba mít nad zprávou SOAP plnou kontrolu, neboť modifikuje přímo strukturu SOAP messages. Sami si pak určujeme, co půjde do hlavičky a co do těla zprávy, navíc lze různým částem zprávy nastavovat různé stupně zabezpečení. Definování pořadí elementů v těle zprávy lze určením parametru Position, jako je tomu v příkladě 5. Ve většině případů je postačující tyto nastavení s klidem přenechat WCF. Tento kontrakt se definuje opět přidáním určitých atributů, konkrétně MessageContract před hlavičku třídy a dále pak atributy MessageHeader označující hlavičku zprávy a atributy MessageBody označující tělo zprávy.

```
[MessageContract]
public class Product
{
    private int Id;
    private string Name;
    private string Sign;

    [MessageHeader]
    public int IdValue
    {
        get { return Id; }
        set { Id = value; }
    }

    [MessageBody(Position = 0)]
    public string NameValue
    {
        get { return Name; }
        set { Name = value; }
    }

    [MessageBody(Position = 1)]
    public string SignValue
    {
        get { return Sign; }
        set { Sign = value; }
    }
}
```

Příklad 5: Definice kontraktu zprávy

Chybový kontrakt (FaultContract)

Definuje, jaké informace o vzniklé chybě mohou být odeslány službou na stranu klienta. Chybový kontrakt se přiřazuje k hlavičce operace v definici kontraktu služby, přičemž je možnost jich přiřadit hned několik k jedné operaci. Nejdříve je potřeba vytvořit definici struktury dat, které se zobrazí v chybovém hlášení na straně klienta. Tato struktura dat s hlášením o vzniklé chybě se pak předává jako parametr s jejím typem do definice chybového kontraktu. Vytváří se stejným způsobem jako definice datového kontraktu, tedy pomocí atributů DataContract a DataMember.

```
[DataContract]
public class ProductFault
{
    private string Problem;
    private string Message;
    private string Solution;

    [DataMember]
    public string ProblemValue
    {
        get { return Problem; }
        set { Problem = value; }
    }

    [DataMember]
    public string MessageValue
    {
      get { return Message; }
      set { Message = value; }
    }

    [DataMember]
    public string SolutionValue
    {
      get { return Solution; }
      set { Solution = value; }
    }
}
```

Příklad 6: Struktura dat odesílaných jako chybové hlášení

Poté je potřeba určit, ve kterých operacích se může tento typ chyby vyskytnout. Následující příklad 7 demonstruje přiřazení chybových kontraktů k jednotlivým operacím v definici kontraktu služby. Jak již bylo řečeno, je možné přiřadit i více chybových kontraktů pro jednu operaci.

```
[ServiceContract]
public interface IShopService
{
    [OperationContract]
    void AddProductToCart(int id, string name, int count);

    [OperationContract]
    [FaultContract(typeof(ProductFault))]
    [FaultContract(typeof(...))]
    List<Product> GetShoppingCart();
}
```

Příklad 7: Definice FaultContract

3 Výukové lekce Windows Communication Foundation

3.1 Zmapování situace a východiska

Před zadáním tématu pro tuto bakalářskou práci jsem se bohužel s technologií WCF nikdy nesetkal. Ačkoliv je to velice perspektivní, široce konfigurovatelný a univerzální nástroj k tvorbě distribuovaných aplikací využívající moderní přístupy. Ovšem při mém vlastním studiu této technologie jsem narazil na mnoho problémů a úskalí. Asi největší překážkou mi bylo pochopení problematiky z literatury v anglickém jazyce. Dalším problémem v mém ucelování znalostí bylo prodírání se velkým množstvím článků na internetu ve většině případů napsaných také v anglickém jazyce.

Cílem této práce je zmapovat možnosti tvorby distribuovaných aplikací pomocí technologie WCF a zkompletovat informace o určitých oblastech této technologie. Na základě těchto shromážděných informací následně vytvořit multimediální kurz pro výuku této perspektivní technologie. Chtěl bych předem upozornit, že se nebude jednat o kompletní výuku všech aspektů týkajících se WCF, ale pouze o vybrané kapitoly.

Práce obsahuje několik výukových lekcí, pomocí kterých se zájemce postupně seznámí s naprostými základy WCF spolu s vysvětlením pojmů týkajících se této technologie až po ukázky mírně pokročilých pravidel a možností tvorby distribuovaných aplikací. Celý výukový materiál bude formou webcastu. V každé výukové lekci zohledňuji své vlastní poznatky a zkušenosti při mém studiu WCF. Každá výuková lekce se skládá z krátké textové části popisující teoretický základ, doplněný o různé zdrojové kódy a obrázky demonstrující postup danou lekcí. Dále každá lekce kromě úvodní obsahuje jeden, či více vzorových příkladů spolu s videoukázkou obsahující zvukovou stopu s mluveným komentářem, jak právě probíranou problematiku zvládnout krok za krokem.

Chtěl bych podotknout, že se bude jednat o první ucelenou publikaci v českém jazyce. V anglickém jazyce vyšla celá řada publikací od nejrůznějších nakladatelství, ale v češtině doposud žádná. Právě toto byl impuls pro mne, jakožto zájemce o technologii WCF, k tvorbě těchto materiálů. Pravdou je, že se na internetu objevují nějaké seriály o WCF, ale zpravidla neobsahují ucelené a komplexní informace, nebo jen povrchně komentují zdrojový kód. Právě díky tomuto faktu se při jejich studiu výsledné znalosti formují mnohem složitěji a delší dobu.

Mnou vytvořený výukový materiál jsem poskytl malé skupině lidí s různými stupni znalostí v oblasti objektově orientovaného programování, aby si jednotlivými lekcemi sami prošli a vyzkoušeli si vyřešit přiložené vzorové příklady. Každý navíc ještě obdržel mnou vytvořený dotazník s otázkami, na které odpověděl. Poskytnutá zpětná vazba mi pomohla ke zjištění kvality těchto výukových materiálů, a zda se mi podařilo do lekcí promítnout dostatek kvalitních a srozumitelně publikovaných informací o problematice týkající se technologie WCF. Zpracované výsledky z těchto dotazníků vyplněných respondenty jsou samozřejmě publikovány v této práci.

3.2 Metodika

3.2.1 Způsob výuky

Všeobecně dlouhé texty nejsou pro čtenáře příliš příjemné na čtení, proto jsem výuku rozložil do jednotlivých lekcí. Tyto lekce postupně ukazují řešení jednotlivých částí problematiky tvorby aplikací v technologii WCF. Navíc výukový materiál psaný jako jediná dlouhá lekce by působil chaotickým dojmem. Zejména při hledání pouze určitých informací by se v něm čtenář ztrácel.

V každé lekci se zájemci dostane do ruky výukový text popisující krok za krokem řešení dané problematiky. Tento text je doplněn obrázky demonstrující jednotlivé úkony, pro lepší představivost studenta. Dále pro lepší přehlednost je

text doplněn barevně zvýrazněnými útržky částí zdrojových kódů. Pro větší přehlednost je toto barevné zvýraznění provedeno naprosto shodně jako ve vývojovém nástroji Visual Studio 2008.

Každá výuková lekce kromě úvodní obsahuje i možnost stažení neřešených či kompletně řešených zdrojových kódů a dále je doplněna o výukovou videoukázku, která vždy demonstruje zvládnutí nějaké vybrané tématiky z dané lekce. Tyto videa tedy nemusí vždy obsahovat kompletní řešení všech aspektů lekce, ale vždy jsem zvolil problematiku, která se mi jevila jako zajímavá či složitější na pochopení. Každá videoukázka je doplněna o zvukovou stopu s komentářem k dané problematice.

Na konci některých lekcí je ponechán prostor i pro samostatné řešení nějaké problematiky. Řešení těchto úkolů není nutné ke zvládnutí a pochopení celkového rámce výuky, ale spíše pro rozšíření znalostí nebo poukázání na další možnosti, proto se jimi podrobně nezabývám a ponechávám zájemci prostor pro jejich samostatné řešení.

V řešení vzorových příkladů v lekcích nepřikládám příliš velkou váhu na grafické úpravy a vzhled výsledných aplikací, protože si myslím, že podobné úpravy nesouvisí s WCF a jsou závislé spíše na potřebách a cítění tvůrce aplikace. Můj zájem je směrován především na kvalitu a funkčnost zdrojového kódu, což nechává ještě větší prostor pro tvůrčí činnost zájemce studujícího tyto lekce.

3.2.2 Segmentace výuky

Řazení kapitol má také svůj patřičný význam. Podle mého názoru není chaotické a ledabylé psaní jednotlivých lekcí (chcete-li kapitol) příliš správné už jen proto, že se může klidně stát, že nějakým nedopatřením se dostane kapitola s pokročilejším tématem před nějakou ze základních kapitol. Navíc si myslím, že můj systém psaní lekcí pomáhá zvýšit efektivnost výuky a přehlednost zaměření jednotlivých kapitol. Právě proto jsem členění nenechal náhodě a výukové lekce rozdělil na dvě části.

Navíc jsem v jednotlivých lekcích postupoval od řešení snazšího učiva po složitější. Sice částečně počítám s tím, že pokročilý zájemce bude ve většině případů přeskakovat mezi různými lekcemi a vybírat si jen určité pasáže, ale tyto výukové lekce jsem jako komplet tvořil primárně pro začínající zájemce. Těm tento způsob koncepce výuky bude jistě příjemnější.

První část je určena pro naprosté začátečníky, kteří neví o WCF vůbec nic. Tato část obsahuje řešení nejzákladnějších problémů a úskalí tvorby distribuovaných aplikací ve WCF. Lekce v této části jsou spolu velice úzce spjaty, a proto jsou psané tak, aby na sebe navazovali, protože je to nezbytné. Například nelze řešit problematiku hostování služby, pokud zatím nevíme jak vůbec nějakou službu vytvořit, nebo není možné řešit klientskou aplikaci, pokud nevíme jak hostovat službu, ke které by bylo možné se připojit a komunikovat s ní. První část obsahuje například kapitoly jako vytvoření a hostování služby, vytvoření klientské aplikace, ale i úvodní lekci s teoretickými základy technologie WCF. V těchto kapitolách vždy demonstruji řešení na nově založeném projektu a zájemce si zkouší vytvořit aplikaci od začátku krok za krokem, jako kdyby vytvářel svou vlastní distribuovanou aplikaci. Tento způsob tvorby příkladů na nově založeném projektu jsem zvolil po zkušenostech z tutoriálů a videolekcí pro výuku jiné technologie. Často se mi totiž stávalo, že si tutoři ve videolekci prostě odněkud okopírovali část zdrojového kódu, ten pak ani neuveřejnili, nebo pokud ho uveřejnili, tak s nějakými chybami atd. Aplikace vytvořené podle jejich návodu pak nefungovali a opravení chyb mi zabíralo spoustu času.

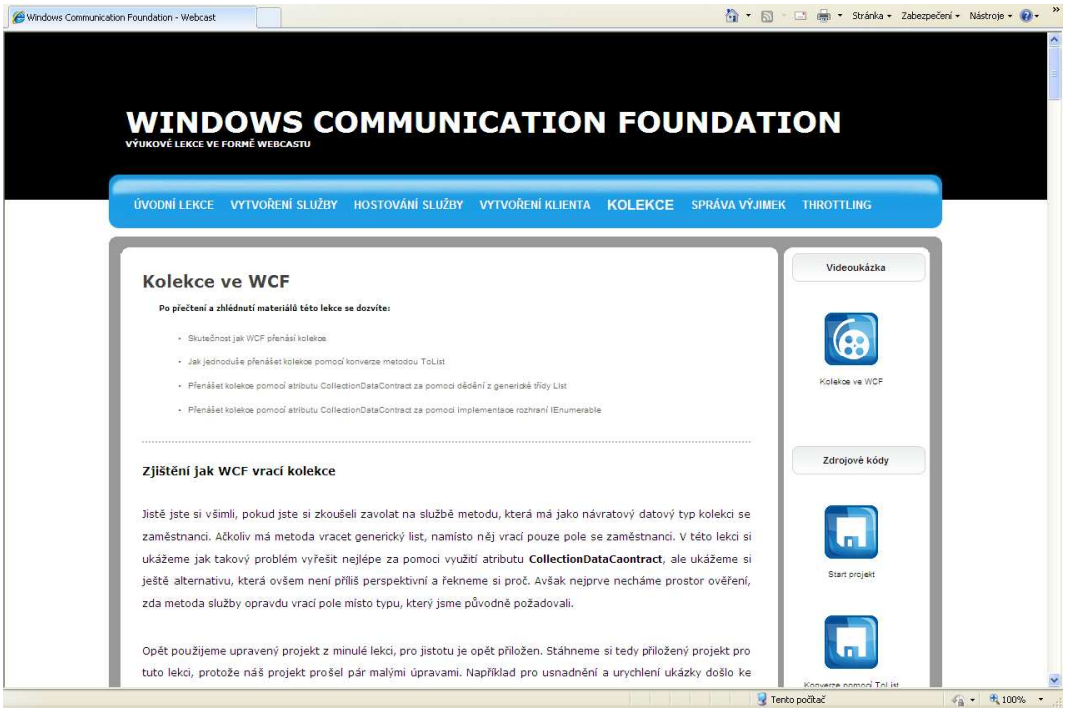
Ve druhé části si zájemce osvojí i pokročilejší témata určená mírně zkušenějším programátorům, kteří již přišli do styku s technologií WCF a rozumí jejím základům. Nyní se již v každé lekci nezačíná na nově založeném projektu, protože v těchto již pokročilejších kapitolách by to byla spíše ztráta času. Druhá část přímo nenavazuje na předchozí, i přestože se v ní stále řeší nějaká fiktivní firma. Ponechal jsem schválně tento projekt vytvořený již v předchozích lekcích, protože ho už zájemci dobře znají a nemusím je zatěžovat představováním nového projektu v každé lekci. Projekt jsem již v počátcích navrhoval tak, aby vyhovoval

i nárokům problematiky probírané v pokročilejších lekcích. Myslím si, že příliš nezáleží na tom, zda vytvořím nový projekt s podobnými vlastnostmi a budu na něm demonstrovat řešení problému, nebo použiji již známý a vytvořený projekt, který svými parametry naprosto dostatečně vyhovuje. Spíš se vždy snažím v každé lekci představit pouze provedené změny, pokud byly vůbec nějaké změny provedeny, což mi přijde mnohem přínosnější, než jak jsem již zmínil zbytečně zatěžující představování nových projektů. V těchto kapitolách se zájemce seznámí například s problematikou přenosu kolekcí ve WCF, nebo škrcením služeb atd. Druhá část lekcí je však samozřejmě koncipována tak, aby ji byl schopen zvládnout i naprostý začátečník, který si prostudoval a pochopil první část výukových lekcí.

3.2.3 Rozhraní mezi studentem a výukovými materiály

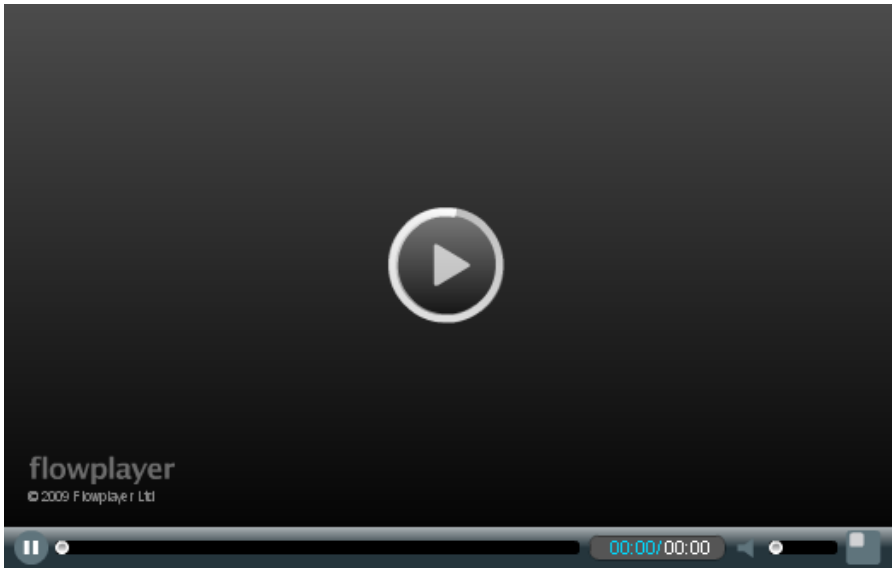
Pro realizaci tohoto výukového materiálu jsem zvolil webové rozhraní. Je to zejména proto, že každý počítač dnes obsahuje nějaký webový prohlížeč. Dnes si nedokážu si představit počítač bez prohlížeče webových stránek. Toto rozhraní jsem tedy zvolil právě kvůli kompatibilnosti a přenositelnosti na jiný počítač, kdy může chybět nejrůznější software pro čtení elektronického obsahu apod.

Web je naprogramovaný v jazyku XHTML 1.0 Transitional je kompletně validní a striktně dodržuje tento vytyčený standard pro psaní webových stránek. Tudíž neobsahuje žádné nepovolené elementy, všechny elementy jsou psané malými písmeny a jsou dodržena i další specifika pro tento jazyk.



Obrázek 8: Webové rozhraní s výukovými lekcemi

Pro přehrávání videí jsem použil formát FLV, což je zkratka pro Flash Video. Toto je také velice kompatibilní formát, kdy pro přehrávání videa postačuje pouze do počítače zdarma nainstalovat Flash Player od firmy Adobe. Nejnovější verze instalačního souboru tohoto nutného prvku je samozřejmě také přiložena u webových stránek s výukovými materiály a přímo z těchto stránek vede odkaz na jeho instalaci. Ovšem i po instalaci se nemusí video správně přehrávat, proto webové stránky obsahují i stránku obsahující nápovědu s nastavením, pokud nelze video přehrát. I přes tento malý problém s nastavením jsem zvolil právě tento formát, který využívají i komerční servery typu *youtube.com* apod. Přímo ve stránkách pak používám free verzi výborného přehrávače Flowplayer.



Obrázek 9: Vzhled přehrávače Flowplayer

Veškeré útržky zdrojových kódů či ukázky konfiguračních souborů jsou barevně sladěny se standardem barevného značení ve Visual Studiu. Značně to přidává na srozumitelnosti a přehlednosti ukázek. Čtenář tak hned rozezná co je proměnná, třída či řetězec apod.

Jednotlivé zdrojové kódy ke stažení jsou vytvořeny ve Visual Studiu 2008 a jsou komprimovány do archivu s koncovkou zip, který lze snadno extrahovat i ve Windows XP bez instalace jakéhokoliv komerčního archivátoru.

3.3 Úvodní lekce

Úvodní lekce má za úkol připravit zájemce na studium technologie WCF a na hladký průchod jednotlivými lekcemi. Úvod je rozdělen celkem do čtyř částí. Každá část v sobě nese užitečné informace, bez jejichž vstřebání není možné absolvovat výukový kurz. Jsou to informace týkající se základních vlastností WCF, předpokladů a nároků na studenta těchto lekcí, koncepce jednotlivých lekcí, co vše je potřeba pro tvorbu distribuovaných aplikací pod platformou .NET a kde tyto různé nástroje získat. Na konci úvodní lekce se student dozví krátké, ale výstižné definice objasňující základní pojmy spjaté s WCF.

První část úvodu obsahuje základní charakteristiku WCF. Snažím se čtenáře v pár kratších odstavcích seznámit s filozofií této moderní technologie, aby zjistil a pochopil, jaká problematika bude náplní výuky. Čtenář se ve zkratce dozví kdy, a proč vznikla, a hlavně co přináší technologie WCF. Poukazuji také na zásadní vlastnost WCF a tou je multiplatformnost, která zajišťuje, že klientská část aplikace může být naprogramována v platformě odlišné od .NET, stačí aby zvolená platforma podporovala nějaký z otevřených standardů výměny dat.

Ve druhé části seznamuji čtenáře s požadovanými nároky na jeho znalosti. Upozorňuji, že pro bezproblémový průchod výukou předpokládám znalosti objektově orientovaného programování, nejlépe pak znalost jazyka C#, ve kterém jsou jednotlivé příklady pro výuku napsány. Dále popisuji koncepci výuky zahrnující, co čtenáře v jednotlivých lekcích čeká a upozornění na způsob rozdělení lekcí.

Pro čtenáře je nezbytně nutné, aby věděl co vlastně potřebuje k tvorbě a běhu aplikací pod .NET. Proto další část lekce seznamuje čtenáře s platformou .NET Framework a její nutností, nebo spíše obsahuje ve zkratce stručný popis této platformy. Vypsal jsem i systémové požadavky na .NET Framework ve verzi 3.5 Service Pack 1, aby si zájemce mohl ověřit, zda má dostatečnou hardwarovou a softwarovou výbavu svého počítače. Dále nabádám zájemce k instalaci stejné verze Frameworku jako používám já, tím zajišťuji správnou funkci přiložených příkladů. Poukazuji i na novější verzi .NET Framework 4 beta a zároveň upozorňuji, že je spíše ve fázi vývoje, proto ho nedoporučuji. Já používám stabilní verzi .NET Framework 3.5 Service Pack 1. Připisuji návod jak si .NET Framework stáhnout a přikládám i přesný odkaz na stránky firmy Microsoft, který zájemce přesměruje přímo na stránku pro stažení Frameworku.

Dalším užitečným, dalo by se říci i nutným nástrojem pro vývoj aplikací pod platformou .NET je Visual Studio. Věnuji proto další část úvodu tomuto vývojovému nástroji. Poukazuji i na možnost tvorby aplikací pomocí jakéhokoliv textového editoru a nějakého z .NET kompilátorů (tzv. překladačů). Zároveň od tohoto způsobu tvorby odrazuji, protože je zbytečně náročný. Visual Studio

obsahuje různé podpůrné nástroje pro tvorbu a jednoduše dělá spoustu práce za nás. Vývoj pomocí textového editoru a kompilátoru v textu dokonce přirovnávám k extrému. Ve zkratce popisuji vlastnosti nástroje Visual Studio a poukazuji na jeho různé mutace od Standard Edition, přes Proffesional Edition až po Team Suite. Představuji čtenáři i nekomerční verzi Express Edition, která však neobsahuje veškeré součásti, proto varuji, že jsem výukové lekce tvořil v komerční verzi Proffesional. Nabádám čtenáře k instalaci shodné verze Visual Studia s verzí, kterou používám já, tou je verze 2008 Service Pack 1. Zmiňuji i existenci verze 2010, ale nedoporučuji její používání, protože se jedná o novinku a nemusí být ještě odladěná. Čtenář se opět seznámí i s minimálními nároky na systém pro běh Visual Studia 2008 Express Edition, aby si mohl ověřit, zda má požadovanou výbavu. Nakonec přikládám i přesný odkaz na stránku, kde je možné si zdarma stáhnout Visual Studio 2008 v již zmiňované mutaci Express Edition.

Úvod zakončuji objasněním základních pojmů týkajících se WCF. Zájemce se dozví význam nejrůznějších termínů, například vysvětluji, co je servisně-orientovaná architektura, služba, endpoint, ukazuji různé typy kontraktů, proxy třída a další. Vše je popsáno spíše ve zkratce, velmi zřetelně a jasně, nezabíhám zbytečně do detailů, které nejsou potřeba pro další výuku.

3.4 Lekce implementace služby

Tato lekce se věnuje hlavnímu pilíři WCF a sice výuce tvorby služby. Zájemce si vyzkouší, jak vytvořit jednoduchou službu představující nějakou fiktivní firmu. Tuto službu budu využívat po celou dobu výuky, zejména protože jsem ji již v počátcích navrhoval tak, aby vyhovovala i nárokům pozdějších lekcí. Zároveň se snažím o dobrou srozumitelnost tím, že směruji zájemce na tvorbu aplikace od samého počátku, aby nedocházelo k vytváření a vrývání do paměti zájemce nějakých zlozvyků v oblasti programování.

Ukazuji zájemci jaké má možnosti při tvorbě WCF služby. Vysvětluji jaký je mezi nimi rozdíl a popisuji jejich možnosti použití. Dále upozorňuji, že možnost WCF Service Library využívaná v těchto lekcích chybí ve Visual Studiu 2008 Express Edition.

3.4.1 Tvorba služby

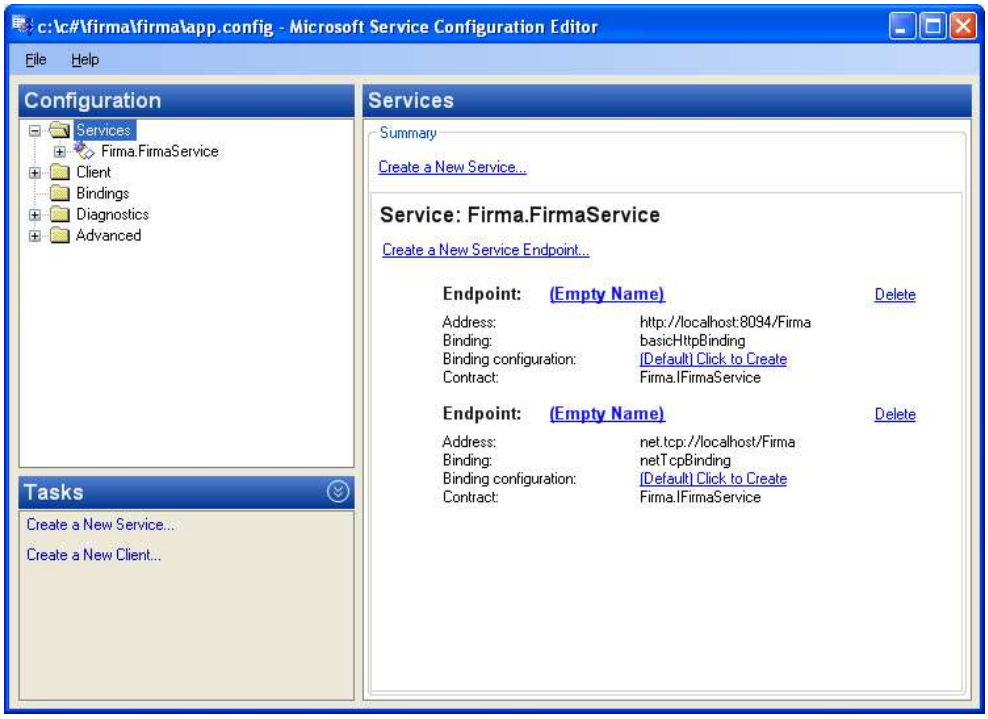
Lekce postupně přechází do samotné tvorby služby. Vzhledem k předešlému popisu a z něj vyplývající univerzálnosti použití WCF Service Library volím právě tuto možnost, která je ideální pro využití i v dalších lekcích. Seznamuji čtenáře s rozhraním plnícím funkci Service kontraktu a doporučuji toto rozhraní pojmenovat přesně na míru své aplikace. Stejnou změnu přejmenování doporučuji i u třídy obsahující definice metod, které bude později nabízet vytvořená služba. Zájemce se seznámí i s využitím atributů Data Contract a Data Member, které definují objekty a struktury přenášené pomocí služby. Dále demonstruji jak správně nadefinovat již zmiňované rozhraní se Service kontraktem. Upozorňuji na možnosti nastavení dalších užitečných parametrů u nejrůznějších atributů. Názorně ukazuji a vysvětluji nastavení parametru Name u atributu Operation Contract. Následuje dokončení definice funkce služby.

```
[OperationContract(Name = "HledatZamestnanceId")]
Zamestnanec Hledat(int id);

[OperationContract(Name = "HledatZamestnanceFunkce")]
Zamestnanec Hledat(string funkce);
```

Příklad 8: Ukázka atributu OperationContract s parametrem Name

3.4.2 Nastavení služby

Poslední část se věnuje správnému nakonfigurování služby. Čtenář se v textu dozví, kde takové nastavení provádět a zároveň jaký nástroj je k tomu určený. Vyzkouší si konfiguraci od úplného začátku, kdy konfigurační soubor neobsahuje žádná nastavení a je potřeba nastavit všechna nastavení. V této lekci ukazuji možnost nastavení právě pomocí vestavěného nástroje ve Visual Studiu, který se jmenuje WCF Service Configuration Editor. Lekce předvede, jak s tímto práci usnadňujícím nástrojem pracovat.



Obrázek 10: Ukázka WCF Service Configuration Editoru

Zároveň upozorňuji na možnost konfigurace ručně, která je také často využívaná pro svou rychlost menších úprav. Ruční konfiguraci si zájemce vyzkouší v pozdější lekci, kdy si nadefinuje celý konfigurační soubor psaním jednotlivých elementů ručně. Ve většině případů se však kombinují oba způsoby, jak využití WCF Service Configuration Editoru, tak ruční konfigurace. Konfigurace se provádí buď v App.config, nebo ve Web.config souboru, podle toho kde službu hostujeme. Pro tuto lekci obsahuje konfigurace služby tvorbu endpointů, nastavení behaviors (chování) služby a přidání bázové adresy. Toto jsou nutná nastavení pro testovací hostování služby.

Při tvorbě endpointu se ve většině informativních zdrojů dodržuje tzv. abeceda, což je definice Adresy, pak Bindingu a nakonec Contractu. Ovšem pokud si pozorně prohlédneme konfiguraci přes WCF Service Configuration Editor, tak ta nás vede v duchu obráceného definování tedy nejdříve, kontrakt, poté binding a nakonec adresa. Tento postup se mi z návrhového hlediska jevil příjemnější, protože vždy si musíme nejprve rozmyslet, co budeme pomocí služby nabízet, poté jak bude služba komunikovat a nakonec kde bude komunikovat. Abych vše

upřesnil, je lepší postupovat tak, aby měl vývojář volné ruce a postupoval od nesvazujících nastavení po zavazující. I když je pravdou, že před samotnou tvorbou aplikace by měl být hotový kompletní návrh definující i způsoby komunikace apod. Proto neodsuzuji způsoby mnohých informativních zdrojů, ale respektuji je.

Na konci lekce si zájemce vyzkouší i přidání druhého endpointu, aby mohla služba komunikovat přes více komunikačních protokolů, konkrétně přes http a net.tcp. Celou funkci projektu si nakonec zájemce otestuje. Lekce samozřejmě obsahuje ke stažení zdrojový kód vytvořené služby pro oba způsoby komunikace.

3.5 Lekce hostování služby

Tato lekce obsahuje seznámení s různými typy hostování, konkrétně s IIS hostingem a self-hostingem. Vybral jsem pouze tyto dva způsoby, protože je lekce již tak dosti obsáhlá a nedostalo by se pak na jiná témata. Absolvováním této lekce vstřebá zájemce obě zmiňované techniky hostování, přičemž navazuje na lekci předchozí, ze které využijeme již vytvořenou službu Firma a budeme ji hostovat. První z nich jsem kvůli menším modifikačním odlišnostem v nové verzi vyzkoušel jak na IIS 5.1 ve Windows XP, tak i na verzi 7.5 obsažené ve Windows 7. Samozřejmě jsem věnoval pár odstavců i upozornění na nutnost instalace IIS, správnému namapování a registrace komponent pomocí utility ServiceModelReg.

3.5.1 Verzování IIS

Nejprve se čtenář dozví upozornění na různé názvy IIS, které může z počátku působit poněkud zmatek a chaos. Při mém vlastním studiu jsem také nejprve nevěděl, proč se IIS někdy nazývá Internet Information Server a někdy Internet Information Service. Musel jsem proto procházet značné množství článků a diskusí, než jsem zjistil, že Internet Information Server je označení pro staré verze. Dále varuji zájemce, že verze IIS je plně závislá na používaném operačním systému (OS), dělám to proto, abych ho vyvaroval zbytečným pokusům o instalaci odlišných verzí do svého OS, jelikož to prostě není možné.

Verze IIS	Dostupnost	Operační systém
1.0	Součástí Windows NT 3.51 SP3 (nebo jako samostatné stažení).	Windows NT Server 3.51
2.0	Součástí systému Windows NT Server 4.0.	Windows NT Server 4.0
3.0	Součástí systému Windows NT Server 4.0 Service Pack 3 (Internet Information Server 2.0 je automaticky inovován na Internet Information Server verze 3.0 během instalace SP3).	Windows NT Server 4.0
4.0	Samostatné stažení z www.microsoft.com nebo z disku CD-ROM Windows NT Option Pack.	Windows NT Server 4.0 SP3 a Microsoft Internet Explorer 4.01
5.0	Vestavěná součást Windows 2000.	Windows 2000
5.1	Vestavěná součást systému Windows XP Professional a Windows XP Media Center Edition.	Windows XP Professional a Windows XP Media Center Edition
6.0	Vestavěná součást systému Windows Server 2003.	Windows Server 2003
7.0	Vestavěná součást systému Windows Vista a Windows Server 2008.	Windows Vista a Windows Server 2008
7.5	Vestavěná součást systému Windows 7 a Windows Server 2008 R2	Windows 7 a Windows Server 2008 R2

Tabulka 2: Závislost verzí IIS na různých operačních systémech Windows (upraveno z [19])

Úsek lekce s IIS hostintgem je rozdělen do dvou částí, v první se zájemce dozví jak IIS nainstalovat, nastavit a vyzkoušet správnou funkčnost. Druhá část se zabývá samotným hostingem služby z dřívější lekce na IIS.

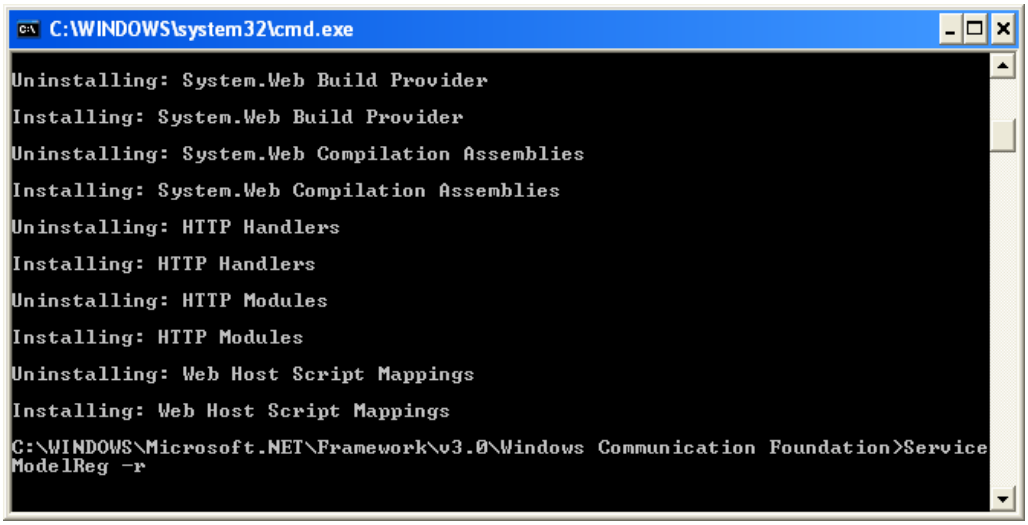
3.5.2 Instalace, nastavení a spuštění IIS

Pro úspěšné hostování na IIS je v prvé řadě nejdůležitější provést jeho instalaci. Upozorňuji na to, protože instalace není provedena automaticky při instalaci systému, ale je třeba tuto jeho součást přidat ručně. Instalaci jsem pro zájemce zmapoval na více systémech, protože se v různých verzích OS Windows liší. Přikládám tedy způsob instalace pod OS Windows XP, ale i pod Windows Vista či Windows 7.

Nezapomínám upozornit na nutnost registrace komponent pomocí utility ServiceModelReg či problematiku správného namapování. Někdy nebývá problematika namapování zmíněna. Je to možná proto, že toto nastavení přímo nesouvisí s WCF a počítá se předem se znalostí těchto formalit. Ovšem i já jsem se při studiu WCF potýkal s problémy správného nastavení a instalace IIS, proto se v lekci této podle mého názoru důležité problematice věnuji. V lekci přikládám odkaz právě na stránku s vysvětlením správného namapování a seznamuji zájemce i s utilitou ServiceModelReg, která se defaultně nachází v tomto umístění:

\%SystemDirectory%\Microsoft.NET\Framework\v3.0\Windows Communication Foundation

Pro utilitu ServiceModelReg představuji dva užitečné parametry prvním je –r pro spuštění přeregistrování a druhým je –y pro vypnutí nutnosti potvrzovat změnu. Dále přikládám přesný odkaz na MSDN stránky firmy Microsoft, kde se zájemce dozví vyčerpávající informace o utilitě ServiceModelReg a jejích nejrůznějších parametrech.



Obrázek 11: Registrace komponent s pomocí utility ServiceModelReg

Ještě před samotným hostováním služby s firmou ukazuji, jak vyzkoušet funkci IIS na defaultní stránce IISHelp, kterou obsahuje ihned po instalaci. Zájemce tak sám vidí, že IIS na jeho lokálním počítači opravdu běží.

3.5.3 Hosting služby na IIS

Zájemci jsou nyní v lekci představeny základní změny, které je potřeba provést v projektu Firma vytvořeném původně jako WCF Service Library, aby jej bylo možné hostovat i na IIS. Změn je hned několik, avšak všechny je v lekci popisuji a demonstruji. První z nich je fakt, že IIS používá místo konfiguračního souboru App.config soubor Web.config. Ukazuji, že postačí pouze přejmenovat již stávající soubor a pozměnit jeho obsah. Změna obsahu spočívá ve vymazání bázové adresy a adres u jednotlivých endpointů, protože umístění služby se převezme z umístění virtuálního adresáře na IIS, tudíž není potřeba je uvádět.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="FirmaServiceBehavior">
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service behaviorConfiguration="FirmaServiceBehavior"
name="Firma.FirmaService">
        <endpoint address="" binding="basicHttpBinding"
            bindingConfiguration="" contract="Firma.IFirmaService"
/>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Příklad 9: Konfigurační soubor Web.config pro IIS

Soubor Web.config je potřeba ještě připojit k našemu projektu, aby byl součástí buildu aplikace. Provádí se to ve vlastnostech projektu v záložce Build změnou umístění výstupu s projektem ze stávajícího bin\Debug pouze na bin\, čímž se změní umístění přeloženého projektu. Poslední změnou je přidání souboru s příponou svc, přes který se pak přistupuje ke službě hostované na IIS. Tímto se zájemce seznámil s potřebnými změnami WCF Service Library, aby jí bylo možné hostovat na IIS.

3.5.4 Problémy s podporou komunikace v IIS

V lekci upozorňuji, že provedením těchto změn je služba připravena pro hostování na IIS ve verzi 7.0 obsažené ve Windows Vista nebo Windows Server 2008, popřípadě na vyšší verzi IIS obsažené v novější verzi operačního systému Windows (například Windows 7). Je to z důvodu používání net.tcp komunikace, protože tyto verze IIS podporují i tzv. non-http komunikaci. Pokud čtenář používá některou ze zmíněných verzí IIS, tak je vše v pořádku a služba mu v pořádku poběží. Pouze poukazuji na potřebná nastavení. Prvním je nutnost povolit již zmiňovanou non-http komunikaci a ukazuji jak ji povolit. Druhým je v nabídce nastavení virtuálního adresáře vytvořeného na IIS zapnout komunikační protokoly http a net.tcp.

Ovšem pokud je využívána starší verzi (například IIS 5.1 obsažená ve Windows XP), tak upozorňuji na nutnost odebrat endpointy využívající jiný způsob komunikace než http. Http je totiž jediný způsob komunikace, který tyto starší verze IIS podporují.

3.5.5 Hosting služby pomocí self-hostingu

Při vývoji distribuovaných aplikací je ve většině případů výhodné psát službu přímo v projektu s hostující aplikací. Psaní služby tímto způsobem je výhodou převážně proto, že není potřeba vytvářet dvě assembly, tudíž je služba perfektně spjata s hostující aplikací, vše (služba i host aplikace) se upravuje na jednom místě a celkově to přidává na přehlednosti. S touto možností se zájemce seznámí až v pozdější lekci, nyní jsem zvolil druhý způsob, kterým je připojit odděleně vyvinutou službu v podobě knihovny dll. Kde je hlavní výhoda v tom, že je služba perfektně přenositelná do více projektů s různým hostováním. Navíc pak není potřeba duplikovat kód a provedené změny na jednom místě se projeví ve všech projektech, kde je knihovna se službou využívána.

Pro tuto možnost hostování opět využívám službu Firma z minulé lekce, která je tvořena pomocí knihovny a je pro jistotu přiložena ke stažení. Pomocí self-hostingu je možné hostovat v nejrůznějších aplikacích. Já jsem zvolil Windows Forms Application, protože bývá v drtivé většině nejčastějším řešením aplikací pro prostředí Windows.

Jako v předchozích lekcích i zde ukazuji, jak vytvořit hostující aplikaci od založení nového projektu. Na začátku této části lekce o hostování shrnuji v kostce, že k hostování služby pomocí self-hostingu je především zapotřebí použít namespace System.ServiceModel a z jeho obsahu třídu ServiceHost. Dále představuji budoucí aplikační řešení pomocí okna se dvěma tlačítky a popiskem s výpisem stavu služby. Následuje detailní popis jak toto řešení realizovat, jak připojit knihovnu se službou k projektu s hostující aplikací, nebo jak vytvořit objekt typu ServiceHost a spustit hostování pomocí metody Open.

Poslední část lekce se věnuje konfiguračnímu souboru, který je v hostující aplikaci nezbytný. Jak jsem předesílal v dřívější lekci, že zájemce seznámím i s možností nevyužít vestavěný WCF Service Configuration Editor, ale celý konfigurační soubor si napsat ručně, tak nyní přišel ten správný čas. Ukazuji kde najít možnost vytvořit takový soubor a jak ho smysluplně a funkčně naplnit, tak aby hostující aplikace správně fungovala.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="FirmaServiceBehavior">
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service behaviorConfiguration="FirmaServiceBehavior"
name="Firma.FirmaService">
        <endpoint address="FirmaService"
                  binding="basicHttpBinding"
                  contract="Firma.IFirmaService" />
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8094/Firma/" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Příklad 10: Konfigurační soubor App.config pro self-hosting

3.6 Lekce vytvoření klienta

Po zvládnutí problematiky obsažené v této lekci zájemce získá poslední dílek k ucelení distribuované aplikace, která se bude nyní skládat z hostované služby a z klienta, který bude se službou komunikovat. Distribuovaná aplikace bývá většinou realizována nějakou službou hostovanou na serveru a klientskými aplikacemi využívanými uživateli na vzdálených stanicích. Tito klienti se pak vzdáleně připojují k serveru a prostřednictvím něho se službou komunikují a vyměňují si data. Dokončením a vstřebáním této lekce se završí první část výukového materiálu určená pro naprosté začátečníky. Veškeré doposud získané znalosti v těchto lekcích budu nadále považovat za zvládnuté a budu předpokládat jejich znalost, tudíž se k nim již nebudu vracet.

Z důvodu komplexnosti ukazuji zájemci celkem tři způsoby, jak vytvořit klientskou aplikaci komunikující se službou. V některých internetových tutoriálech je zmíněn i pouze jeden způsob (např. [13]), což nepovažuji za správné a ukazuji tvorbu klienta pomocí Service Reference, nebo utility svcutil či pomocí ChannelFactory. Všechny klienty realizuji opět pomocí Windows Forms Application.

Nejprve ukazuji kroky, které jsou společné pro všechny způsoby tvorby klientů. V těchto krocích se zájemce dozví jak si připravit základ aplikace, který využije ve všech možnostech tvorby klientské aplikace. Ukazuji vytvoření nového projektu s formulářovou aplikací nazvaného Klient a dále jak si připravit vzhled aplikace spočívající v přidání potřebných referencí a umístění určitých komponent do okna formuláře pro pozdější obsluhu.

3.6.1 Klient pomocí Service Reference

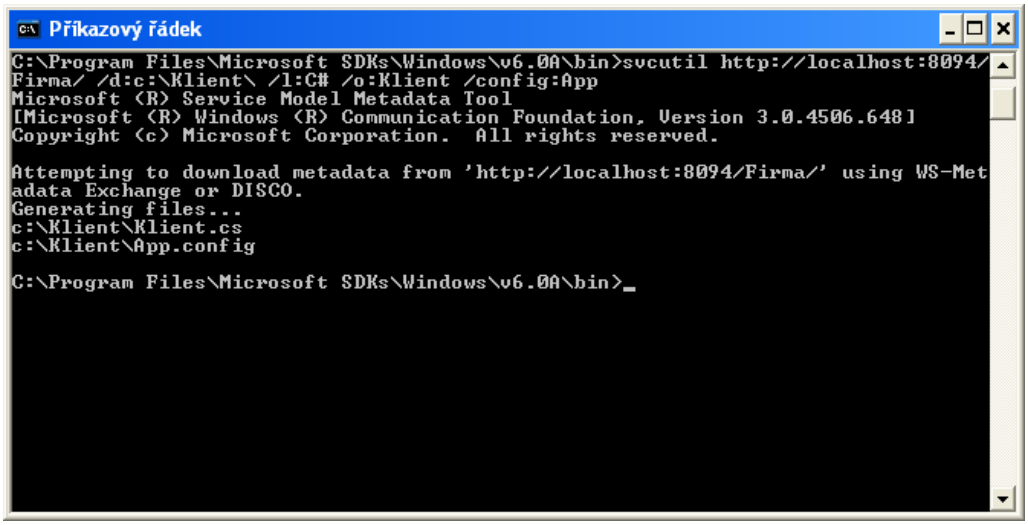
Jako první způsob demonstruji postup návrhu klienta (podle [6], str. 1482) pomocí reference (odkazu) na službu. Toto je nejběžnější způsob, avšak ne jediný, jak získat informace o endpointu z metadat. Snadno lze tuto referenci přidat pomocí Visual Studia, které od studentů těchto lekcí očekávám. Ukazuji jak snadno přidat referenci na službu pomocí volby Add Service Reference. Ovšem nezapomínám upozornit na nutnost službu někde hostovat, tudíž publikovat její metadata, aby bylo možné stáhnout potřebné údaje pro nově vzniklou referenci. Dále upozorňuji na nutnost provedení aktualizace reference při provedení jakýchkoliv změn ve službě. Ve zdrojovém kódu obslužných tlačítek je pak vidět, jak se vytváří objekt této reference a jak se pomocí něho volají různé operace služby. Klienta tvořeného tímto způsobem využívám i v pozdějších lekcích.

3.6.2 Klient pomocí svcutil

Ukazuji i způsob jak získat klientskou aplikaci nikoliv pomocí Visual Studia, ale pomocí utility svcutil, která se spouští nejlépe z příkazové řádky s nejrůznějšími parametry. Defaultně se nachází v následujícím umístění:

C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin

Tato utilita dokáže vygenerovat proxy třídu a konfigurační soubor, které jsou potřebné ke správné funkci klienta. Dokáže to, jak z metadat vypublikovaných službou, tak i z knihovny obsahující implementaci služby. V lekci ukazuji samozřejmě oba možné způsoby získání těchto souborů. Vysvětluji použití nejzákladnějších parametrů této utility a přikládám i přesný odkaz na stránku obsahující její vyčerpávající popis včetně všech jejích parametrů.



Obrázek 12: Generování potřebných souborů z metadat služby pomocí utility svcutil

Zájemce se samozřejmě dozví, jak vygenerované soubory připojit do projektu s klientem. Následuje ukázka, jak vytvořit objekt proxy třídy a způsob jakým s její pomocí volat operace na službě.

3.6.3 Klient pomocí ChannelFactory

Posledním předváděným a méně častým způsobem vytváření klientské aplikace, je tvorba pomocí generické třídy ChannelFactory. Ke komunikaci využívá také třech základních prvků endpointu a je vhodná zejména pokud máme k dispozici rozhraní s kontraktem. Tento kontrakt v lekci získávám nakopírováním knihovny se službou a jejím připojením k projektu.

V lekci je předvedeno jak vytvořit objekt představující adresu a objekt obsahující binding. Tyto objekty jsou spolu s kontraktem předány nově vytvořené instanci třídy ChannelFactory. V příkladě je znázorněno, i jak s ChannelFactory pracovat a jak s její pomocí volat operace na službě.

```
public void VolaniOperaceSluzby()
{
    BasicHttpBinding binding = new BasicHttpBinding();
    EndpointAddress adresa = new
EndpointAddress("http://localhost/AdresaSluzby");
    ChannelFactory<IFirmaService> channel = new
ChannelFactory<IFirmaService>(binding, adresa);

    IFirmaService proxy = channel.CreateChannel();
    vystupniLabel.Text = proxy.OperaceSluzby();
}
```

Příklad 11: Vzorová implementace ChannelFactory

Nakonec v lekci nechávám prostor pro samostatné řešení obsluhy ostatních operací nabízených službou a navrhuji samostatné vyzkoušení vytvoření klientské aplikace pomocí WPF či Console Application. Ovšem tyto kroky nejsou potřeba pro zvládnutí základů WCF.

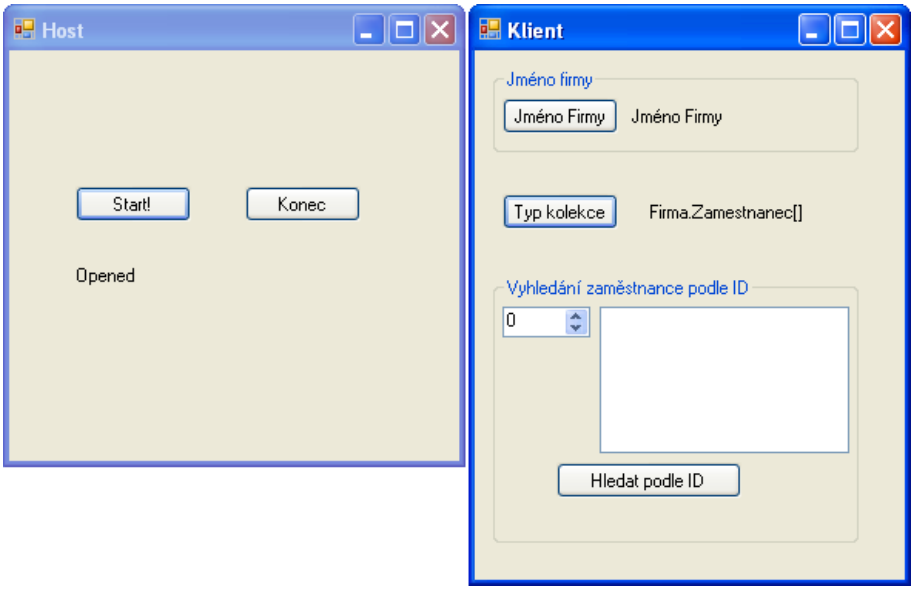
3.7 Lekce kolekce ve WCF

Nyní začíná druhá část výukového materiálu určená pro pokročilejší zájemce o technologii WCF. Zde se sice stále lekce zabývají problematikou fiktivní firmy, ovšem již na sebe přímo nenavazují. Projekt v lekci může projít nejrůznějšími změnami pro lepší pochopení probírané problematiky, avšak zájemce s těmito změnami seznamuji hned na začátku lekce a vždy přikládám upravený projekt ke stažení.

3.7.1 Vracení kolekcí ve WCF

Je zvláštní, že se mnohé publikace (například jako publikace [5]) vůbec správným přenosem kolekcí nezabývají. Já jsem jiného názoru a mám pocit, že tato problematika je velice důležitá, tudíž by se neměla opomínat. První část lekce seznamuje čtenáře se způsobem návratu kolekcí v technologii WCF. Ukazuji, jak jednoduše zjistit skutečný typ navrácené generické kolekce typu List. Zájemce tak

sám zjistí fakt, že tam, kde by se měl správně vracet generický list se ve skutečnosti ve WCF vrací obyčejné pole. Prvním ověřením je zhlédnutí návratového typu metody pro návrat všech zaměstnanců firmy ve třídě implementující službu. Dále dokazuji, že po spuštění hostování služby je vše jinak. V internetovém prohlížeči je v xsd schématu vidět návratový typ této metody jako ArrayOfZamestnanec. Posledním ověřením, které provádím je výpis datového typu do komponenty Label. Opět je vypsaným návratovým typem pouze pole, tento Label má význam i později v této lekci.



Obrázek 13: Návrat kolekce ve WCF

3.7.2 Přenos kolekcí pomocí konverze

První předvedený způsob sice není příliš ideální, avšak i přesto může nalézt své uplatnění především díky jednoduchosti. Není vůbec potřeba měnit službu, ale pouze provést konverzi kolekce na straně klienta. Pokud kompletně celou distribuovanou aplikaci, tedy službu včetně klienta, vyvíjí jeden tým vývojářů či jedna korporace, tak lze konverzi bez problémů využít. Jde totiž o to, aby programátor klientské aplikace perfektně znal strukturu odesílaných dat, tedy v jaké kolekci byla data původně odeslána.

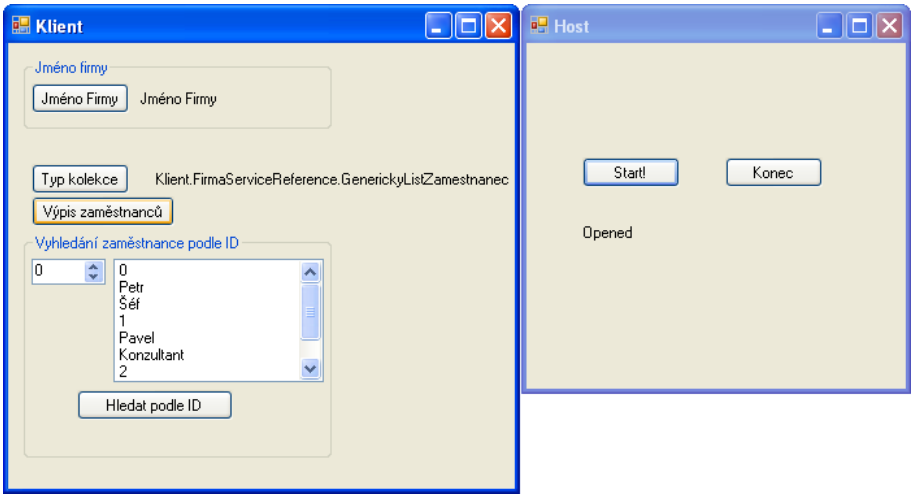
3.7.3 Přenos kolekcí pomocí atributu CollectionDataContract

Druhá předvedená možnost je optimálním způsobem přenosu kolekcí ve WCF. Využívá se k tomu vlastní vytvořená generická kolekce, jejíž implementace se označí právě atributem CollectionDataContract. Třída této vlastní kolekce buď implementuje rozhraní IEnumerable, aby bylo možné používat cyklus foreach, nebo dědí od již existujícího typu kolekce, jako je například List nebo Dictionary. V lekci ukazuji samozřejmě oba tyto možné způsoby. Pro využití atributu CollectionDataContract nesmí zájemce opomenout využití namespace System.Runtime.Serialization, na což také upozorňuji.

```
[CollectionDataContract(Name = "GenerickyList{0}")]
public class MujList<T> : IEnumerable<T>
{
    private List<T> list = new List<T>();

    public void Add(T item)
    { list.Add(item); }

    public T GetItem(int pozice)
    { return list[pozice]; }

    public void ClearList()
    { list.Clear(); }

    public int Count
    { get { return list.Count; } }

    #region IEnumerable<T> Members

    public IEnumerator<T> GetEnumerator()
    { return list.GetEnumerator(); }

    #endregion

    #region IEnumerable Members

    System.Collections.IEnumerator
System.Collections.IEnumerable.GetEnumerator()
    { return list.GetEnumerator(); }

    #endregion
}
```

Příklad 12: Implementace rozhraní IEnumerable pro přenos kolekcí

Lekce provádí zájemce i ostatními změnami v různých částech služby, jako je změna návratového typu v rozhraní s kontraktem služby, pak stejná změna ještě v implementaci služby a nakonec změna datového typu i ve třídě nahrazující databázi. Na samém konci lekce je i ukázka demonstrování použití cyklu foreach.



Obrázek 14: Ukázka správného datového typu kolekce s výpisem jejího obsahu

3.8 Lekce správy výjimek ve WCF

Tato lekce provede zájemce možnostmi správného ošetření nejrůznějších výjimek ve WCF. Hlavním bodem, který je třeba si uvědomit je to, že ve výchozím nastavení nejsou zprávy o chybě vzniklé na službě nijak odesílány na stranu klienta. Pro spoustu zájemců začínajících s technologií WCF je tento fakt poněkud nepochopitelný, jako tomu bylo při mém vlastním studiu této technologie. Avšak v lekci ospravedlňuji toto výchozí nastavení, protože není přípustné, aby uživatel na straně klienta věděl, přesné údaje o chybě vyskytlé na serveru hostujícím službu. Především vidím oprávnění výchozího nastavení ve větší bezpečnosti. Informace o vzniklé chybě je třeba spravovat a protokolovat přímo ve službě, nikoliv na straně klienta a navracet pouze publikovatelné informace o chybě.

Pořadí probírané problematiky v lekci je řazeno postupně podle obtížnosti pochopení a podle návaznosti využití. Nejprve při vyvolání výjimky a následném pokusu ji ošetřit se zájemce dozví o funkci atributu

includeExceptionDetailInFaults, který patřičně popíši. Dále zjistí, jak správně ošetřit tuto výjimku bez použití již zmíněného atributu includeExceptionDetailInFaults. Nakonec si zájemce osvojí způsob jak odesílat ve WCF strukturovaný chcete-li detailnější popis chyby.

Nejprve je v lekci předvedeno, jak uměle nasimulovat způsobení výjimky na straně služby. Jednoduše to lze provést přidáním jedné řádky ve zdrojovém kódu implementace této služby. Vyvolání výjimky spočívá ve vytvoření instance třídy Exception a předání nějakého řetězce s textem popisujícím chybu do parametru jejího konstruktoru.

3.8.1 Atribut IncludeExceptionDetailInFaults

Následuje ukázka, jak tuto výjimku ošetřit na straně klienta. Odchycení výjimky spočívá v přidání bloků try a catch. Ovšem při pokusu zobrazit text popisující chybu se zobrazí hláška vyzývající k nastavení atributu includeExceptionDetailInFaults na true. Ukazuji tedy změnu nastavení tohoto atributu nacházejícího se v konfiguračním souboru aplikace a zároveň upozorňuji na to, že je určen pouze pro ladění aplikace. Nyní se již zobrazuje na straně klienta i obyčejná chyba služby korektně v dialogovém okně s popisem chyby, ale při finální podobě aplikace musí být tento parametr nastaven na false. Tudíž nelze toto řešení považovat za konečné, ale pouze jako možnost si zobrazit informace o vyskytlé chybě při vývoji a ladění distribuované aplikaci.

```
<behaviors>
  <serviceBehaviors>
    <behavior name="FirmaServiceBehavior">
      <serviceMetadata httpGetEnabled="true" />
      <serviceDebug includeExceptionDetailInFaults="true" />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

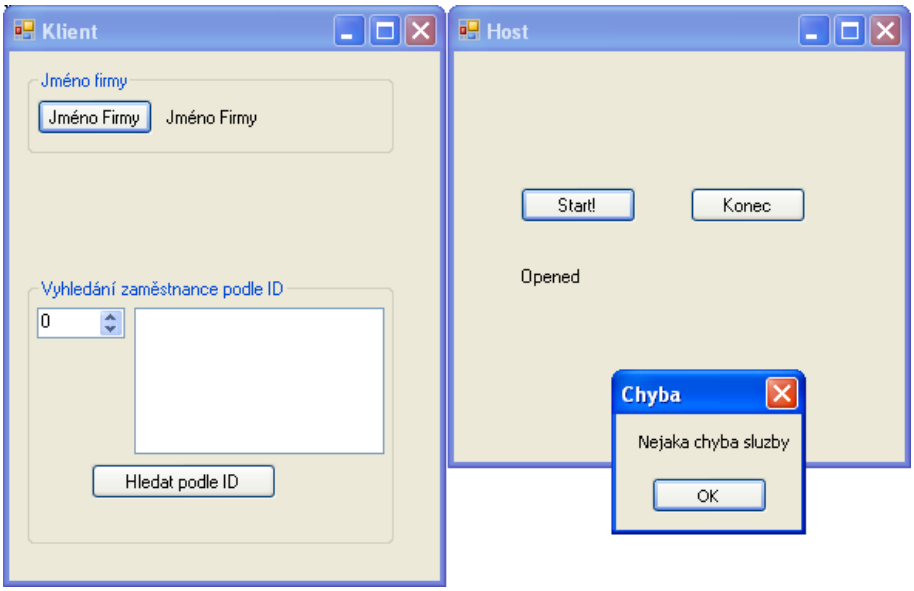
Příklad 13: Definice atributu includeExceptionDetailInFaults v konfiguračním souboru

3.8.2 FaultException chyba

Správný způsob jak ve WCF odesílat informace o chybě vzniklé ve službě na straně serveru je právě přes FaultException. Ukazuji, že v implementaci služby stačí pouze v místě výskytu chyby přidat blok try a catch. Do bloku catch jako odchycení chyby služby umístit řádku vyvolávající výjimku FaultException a popis chyby jí předat buď zprávou vyskytlé chyby, nebo vložit libovolný textový řetězec. Tím je zaručené, že se na straně klienta zobrazí pouze taková zpráva o chybě, kterou si vývojář sám navolí. Zájemce na příkladu zjišťuje, že se opravdu zobrazuje popis chyby, který si sám zvolil a vidí, že má nad obsahem odesílaného hlášení mnohem větší kontrolu.

```
try
{
    throw new Exception("Nejaka chyba sluzby");
    return "Naše firma";
}
catch (Exception e)
{
    throw new FaultException(e.Message);
}
```

Příklad 14: Ukázka vyvolání FaultException



Obrázek 15: Výpis chyby služby na straně klienta pomocí FaultException

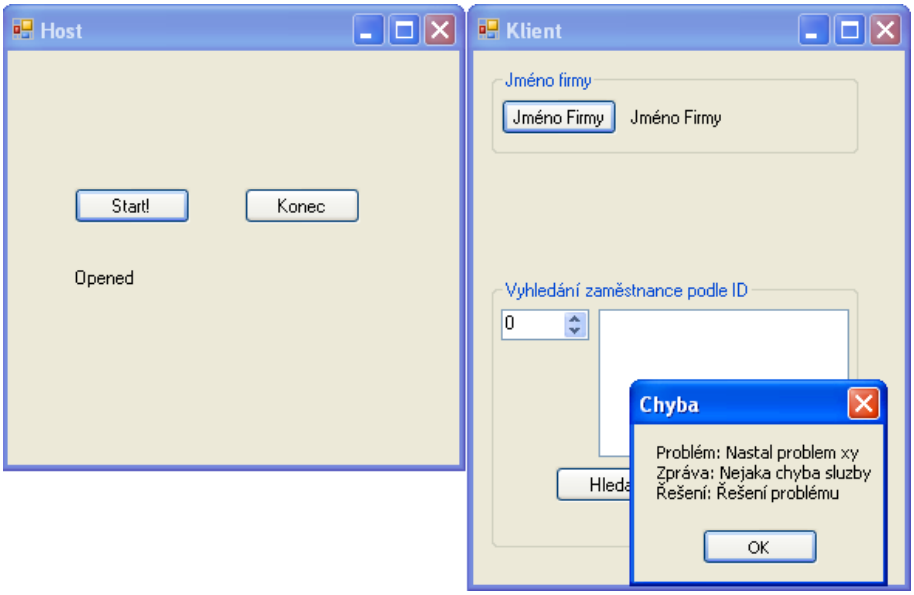
3.8.3 Generická FaultException<T> chyba

Pokud je ve WCF potřeba odeslat nějak strukturovaný popis vzniklé chyby, či je-li potřeba mít ještě větší kontrolu nad popisem chyb než nabízí klasická FaultException, existuje řešení v podobě generické FaultException. V dalším příkladě je znázorněno i toto řešení, pro jehož využití je potřeba si vytvořit novou třídu se strukturou dat s hlášením o chybě. Ukazuji, jak tuto třídu implementovat a opatřit ji potřebnými atributy. Dále se zájemce dozví, jak ji zavést do rozhraní s kontraktem služby k definici jednotlivých hlaviček metod, ve kterých se bude používat pro popis vzniklé chyby. Zavádí se ve formě FaultContractu, kterému se předá jako typ. Vysvětluji i to, že lze nadefinovat více těchto FaultContract atributů jedné operaci.

```
[ServiceContract]
public interface IFirmaService
{
    [OperationContract]
    [FaultContract(typeof(FirmaFault))]
    string JmenoFirmy();

    [FaultContract(typeof(...))]
    [FaultContract(typeof(...))]
    [OperationContract]
    void PridatZamestnance(string jmeno, string funkce);
}
```

Příklad 15: Definice FaultContract atributu

Nyní se již ve službě v ošetřovacím bloku catch nevytváří výjimka typu FaultException, ale generická FaultException<T>. Navíc je zde třeba vytvořit instanci objektu obsahujícího strukturu popisu chyby a zinicializovat jeho jednotlivé proměnné. Na straně klienta se pak přistupuje k proměnným tohoto objektu a získávají se z nich informace o chybě. V lekci samozřejmě ukazuji výsledek ošetření vyvolané výjimky, aby zájemce na vlastní oči viděl, jak by se měla jeho aplikace při chybě správně zachovat. Dávám mu tím také návod, jak aplikaci otestovat, či jak jednoduše otestovat správnou funkci své vlastní aplikace.



Obrázek 16: Detilnější výpis chyby služby na straně klienta pomocí generické FaultException<T>

3.8.4 Hierarchie obsluhy výjimek

V lekci také nezapomínám i na dosti důležitou a často opomíjenou skutečnost, kterou je pořadí ošetřování výjimek. V lekci věnuji pro jistotu této problematice pár řádků, protože zájemce nemusí znát strukturu dědění výjimek. Předcházím tím možným problémům při překladu aplikace, pokud je potřeba ošetřovat různé typy výjimek. Pokud by totiž došlo k prohození pořadí některých ošetřujících bloků catch, tak by například výjimka dědící od výjimky předchozí nemusela být nikdy ošetřena. V lekci je samozřejmě uvedeno správné a funkční pořadí bloků catch.

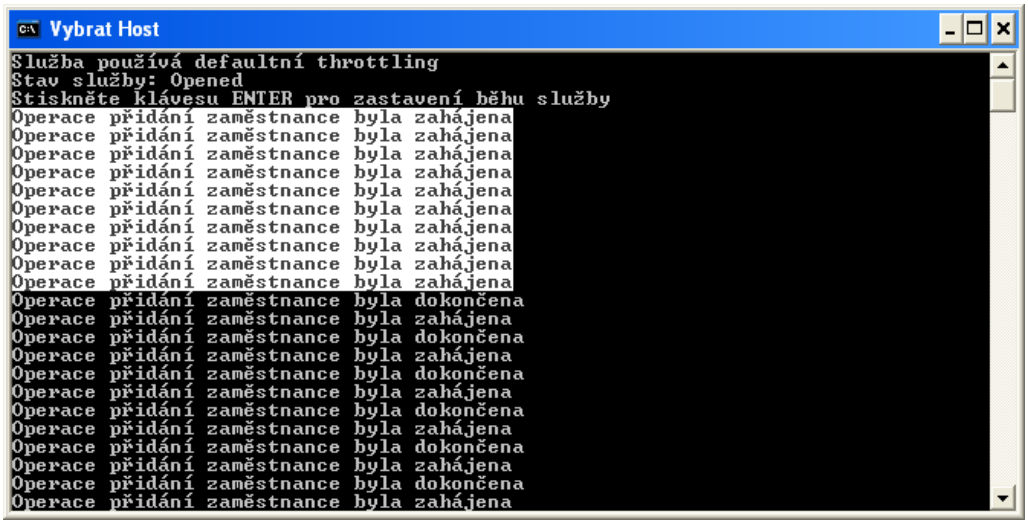
3.9 Lekce techniky Throttling

Throttling (škrcení) je velice užitečná technika pro správné a optimalizované řízení chodu služby. Díky throttlingu lze například zařídit, aby se služba nepřetížila pod náporem dotazů od klientů. Pomocí této snadno pochopitelné metody, lze velice lehce regulovat počet volání, instancí, či relací služby. Lekci jsem do výukového materiálu zařadil, protože omezování zatížení služby je příliš důležitá kapitola na to, abych ji vynechal.

Nejprve zájemci popisuji základní pojmy týkající se škrcení služeb, ve zkratce vysvětluji i některé z interních chování služby pro ještě lepší pochopení dané problematiky. Popisuji funkci všech třech možných vlastností objektu serviceThrottle tedy MaxConcurrentCalls, MaxConcurrentInstances a MaxConcurrentSessions.

Postupně lekce přechází do části seznamující zájemce s přiloženým příkladem, který bude sloužit k výuce. Dále na tomto příkladě ukazuji nastavení vlastností pro omezení počtu volání a počtu relací služby. Zvolil jsem právě tyto dva parametry, protože u nastavení InstanceContextMode na PerSession, které využívám v ukázkovém příkladě, odpovídá počet instancí počtu relací a pozbývá smyslu měnit nastavení třetí vlastnosti tedy MaxConcurrentInstances. Navíc problematika Throttling není příliš složitá, tudíž nastavení poslední vlastnosti si zájemce lehce odvodí z mého popisu. Tímto úmyslným vynecháním části nastavení opět nechávám prostor pro samostatné řešení ze strany zájemce. Na konci lekce jsem vypsal i určitá doporučení pro nastavení hodnot vlastností pro škrcení služby.

Výstup v ukázkovém příkladě je jak v hostující, tak v klientské aplikaci řešen přehledně jako výpis v konzolové aplikaci, aby zájemce viděl přesné chování služby i klienta. Vždy při zahájení vykonávání operace ve službě je v hostující aplikaci vypsána řádka signalizující start operace. Pokud je spuštěna nějaká aktivita na straně klienta, tak se v jeho okně zobrazí právě prováděná operace a číselná identifikace klienta vykonávajícího tuto operaci. Odesílání požadavků od klienta probíhá v intervalu deseti sekund, což má simulovat nějakou časově náročnou operaci, jako je například uložení informací do vzdálené databáze. Upozorňuji zájemce i na možnou TimeoutException, která může nastat po překročení přiděleného času na vyřízení žádosti klienta.



Obrázek 17: Výpis činnosti služby při volání operací klienty

3.9.1 Nastavení Throttling v konfiguračním souboru

Nejprve ukazuji chování služby ve výchozím nastavení, kdy se k ní připojuje asynchronně celkem deset klientů. Poté měním nastavení pomocí WCF Service Configuration Editoru, který provede změnu v konfiguračním souboru aplikace, konkrétně tedy v souboru App.config. Opět je možno toto nastavení kdykoliv měnit ručně a ne za pomocí zmíněného konfigurátoru. Měním postupně nastavení vlastnosti pro omezení počtu volání a ukazuji zájemci vliv na chod služby při řešení požadavků od deseti klientů.

```
<behaviors>
  <serviceBehaviors>
    <behavior name="FirmaServiceBehavior">
      <serviceMetadata httpGetEnabled="true" />
      <serviceThrottling maxConcurrentCalls="16"
                         maxConcurrentSessions="10"
                         maxConcurrentInstances="26"/>
    </behavior>
  </serviceBehaviors>
</behaviors>
```

Příklad 16: Definice škrcení služby v konfiguračním souboru

3.9.2 Nastavení Throttling přímo v kódu aplikace

Zájemce si osvojí i možnost změny těchto vlastností přímo v kódu aplikace, nikoliv tedy v konfiguračním souboru. Toto nastavení má občas své opodstatnění a má své určité výhody a nevýhody. Důrazně upozorňuji na nutnost provést změnu těchto parametrů ve zdrojovém kódu aplikace ještě před samotným hostováním služby, jinak se změny neprojeví a nebudou mít žádný vliv, WCF totiž použije výchozí hodnoty. Nastavení hodnot vlastností ponechávám stejné jako v předchozím případě, aby bylo vidět, že lze dosáhnout stejných výsledků, ať už se zájemce rozhodne pro jakýkoliv ze dvou uvedených způsobů. Opět zájemce uvidí naprosto stejné chování služby jako v předchozím případě.

```
public void StartHost()
{
    ServiceHost host = new ServiceHost(typeof(FirmaService));
    ServiceThrottlingBehavior throttleBehavior = new
ServiceThrottlingBehavior();
    throttleBehavior.MaxConcurrentCalls = 16;
    throttleBehavior.MaxConcurrentInstances = 26;
    throttleBehavior.MaxConcurrentSessions = 10;
    host.Description.Behaviors.Add(throttleBehavior);
    host.Open();
}
```

Příklad 17: Definice škrcení služby přímo v kódu aplikace

4 Evaluace výsledků

4.1 Zkoumaný problém a cíle výzkumu

Mým úkolem bylo vytvořit schopný výukový materiál pro výuku technologie WCFoundation od firmy Microsoft. Hlavním problémem ostatních materiálů zabývajících se touto problematikou je, že buď nejsou v českém jazyce, a pokud ano, tak obsahují pouze povrchní způsob podání požadovaných informací. Proto bych chtěl na menší skupině respondentů se zájmem o tuto technologii vyzkoušet kvalitu mnou vytvořeného výukového materiálu. Zajímá mě hlavně kvalita jednotlivých elementů tohoto materiálu, jako je text výukových lekcí, vzorové příklady či ozvučené videoukázky.

4.2 Plán výzkumu

Odpovědi budu získávat pomocí rozdaných dotazníků mezi jednotlivé respondenty. Cílovou skupinou jsou zájemci přibližně v mé věkové hladině tedy kolem 20 let, většinou studenti Jihočeské univerzity. Skupina čítá celkem 21 osob, z toho většina jsou muži, přesně 20 mužů a 1 žena. Respondenti disponují různými znalostmi v oblasti programování, převážně jsou to začátečníci či mírně pokročilý a jeden profesionál. S technologií WCF se zatím nikdo z respondentů nikdy nesetkal. Dotazníky budou rozdány v tištěné formě, tudíž očekávám jejich 100 % návratnost. Na dodání vyplněných dotazníků nemají respondenti stanovený přesný termín, přičemž po obdržení výukového materiálu s dotazníkem zabere nejvíce času samotné nastudování problematiky právě z výukového materiálu, vyplnění dotazníku pak zabere maximálně 10 minut.

4.3 Předpoklady

Po vyhodnocení výsledků předpokládám spíše průměrnou či lehce nadprůměrnou úroveň mnou vytvořených výukových materiálů. Nepředpokládám, že by průměr hodnocení kvality některé části při známkování jako ve škole dosáhl vynikající hodnoty kolem čísla 1 (tedy výborně). Myslím si to vzhledem

k porovnání s profesionálními zpracováními, kdy se ve většině případů na tvorbě výukového materiálu podílí celý tým specialistů, který může nabídnout kvalitnější výsledky, protože se každý člen týmu zabývá pouze svým specifickým okruhem odpovídajícím jeho zaměření. Dále předpokládám, že s pomocí tohoto výukového materiálu zvládne úspěšně vyřešit obsažené příklady většina respondentů.

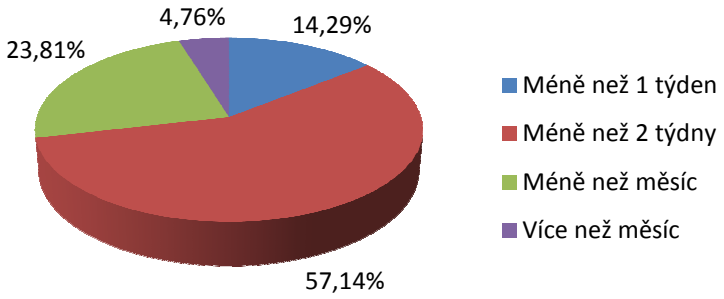
4.4 Výsledky

Celá skupina dotazovaných respondentů výukový kurz kompletně dokončila. To znamená, že si prošla veškeré lekce a krok za krokem si zkusila řešit problematiku obsaženou v lekcích za pomoci veškerých poskytnutých prostředků. Tato skutečnost je velice důležitá pro jakékoliv posuzování získaných výsledků, pokud by totiž všichni respondenti nedokončili kompletně celý výukový kurz, nebylo by možné považovat výsledky za platné ukazatele kvality výukového materiálu.

Otázka:

Kolik času Vám přibližně zabralo pochopení problematiky obsažené ve výukovém materiálu?

Výsledek:



Graf 1: Čas potřebný pro pochopení problematiky obsažené ve výukovém materiálu

Otázka:

Ohodnoťte celkovou srozumitelnost podání informací, obsažených v tomto výukovém materiálu (výukových lekcích), o WCF:*

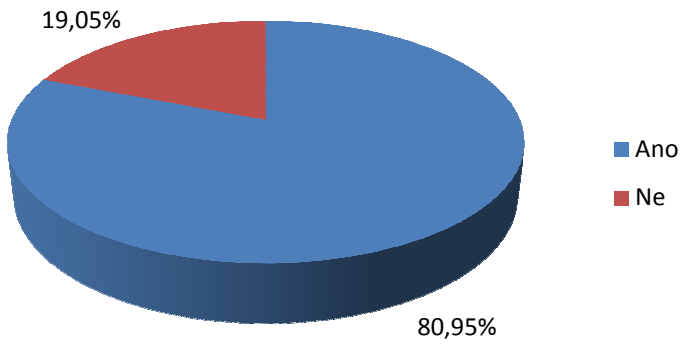
Výsledek:

Zprůměrování výsledků nabývá přibližné hodnoty 1,86.

Otázka:

Myslíte si, že byly naplněny stanovené cíle výuky?

Výsledek:



Graf 2: Naplnění stanovených cílů výuky

Otázka:

Ohodnoťte kvalitu zpracování doprovodných výukových textů:*

Výsledek:

Zprůměrování výsledků nabývá přibližné hodnoty 2,29.

Otázka:

Ohodnoťte kvalitu zpracování připravených ukázkových příkladů pro výuku:*

* Hodnotí se jako ve škole, tedy pomocí stupnice 1 – 5.

Výsledek:

Zprůměrování výsledků nabývá přibližné hodnoty 2,48.

Otázka:

Ohodnoťte kvalitu zpracování doprovodných ozvučených videoukázek:*

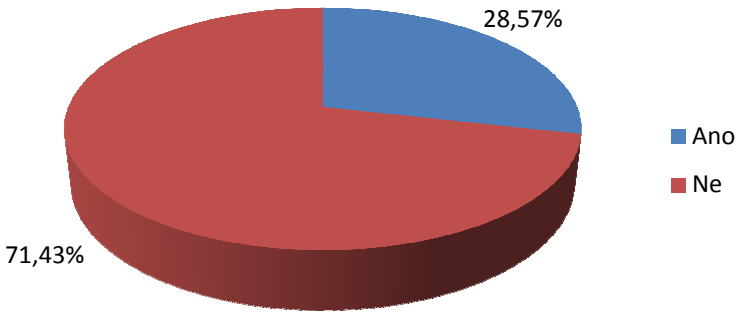
Výsledek:

Zprůměrování výsledků nabývá přibližné hodnoty 1,62.

Otázka:

Zaujala Vás technologie WCF natolik, že se jí budete věnovat i nadále?

Výsledek:



Graf 3: Vyjádření respondentů k dalšímu studiu technologie WCF

* Hodnotí se jako ve škole, tedy pomocí stupnice 1 – 5.

5 Závěr

Cílem mé práce bylo vytvořit výukový materiál technologie Windows Communication Foundation, který by seznámil zájemce s určitými aspekty této technologie. Snažil jsem se velice srozumitelným způsobem vysvětlit jak naprosté základy pro úplné začátečníky, tak i záležitosti pro pokročilejší zájemce o tuto technologii. Mým hlavním cílem bylo tyto informace podat krok za krokem s názornou ukázkou na ukázkovém příkladě, doplněné pro ještě lepší názornost i videoukázkou s komentářem k dané problematice. Myslím si, že tento způsob výuky technologie WCF v českém jazyce chybí. České zdroje bývají většinou pouze povrchní, málo srozumitelné a většinou i málo obsáhlé. Tento problém byl mou motivací k tvorbě práce na toto téma.

Vytvořil jsem výukové lekce zasazené do prostředí webových stránek, které zaručují bezproblémovou kompatibilitu na každém počítači s webovým prohlížečem. Lekce obsahují výukový text s právě probíranou problematikou doplněný názornými obrázky, zdrojovými kódy ukázkových příkladů, které je možno i stáhnout do svého počítače a již zmíněnými videoukázkami.

Z výsledků, které mi respondenti poskytli, vyplývá, že mnou vytvořený výukový materiál je vcelku schopnou pomůckou pro vývojáře se zájmem o komunikační technologie. Dalo by se říci, že dokonce překonal má očekávání. Neočekával jsem hodnoty průměru hodnocení kvality pod hranici 3 (tedy dobře), přičemž hodnocení kvality dosáhlo hodnot značně lepších. Dále respondentům trvalo pochopení učiva obsáhlého v lekcích ve většině případů méně než 2 týdny, pokud bych tento čas porovnal s časem stráveným mým vlastním studiem, tak mé studium bylo nesrovnatelně delší.

Avšak původně jsem měl v plánu obsáhnout většinu aspektů komunikační technologie WCF, avšak tato technologie je velice obsáhlá, tudíž z časových a rozsahových dispozic jsem se musel uskromnit a obsáhnout ve výukovém materiálu méně témat. Navíc jsem nesehnal nikoho, kdo by již technologii WCF ovládal a pomohl mi s přípravou a otestováním pouze pokročilých výukových

lekcí. Myslím si, že z tohoto hlediska nedošlo k úplnému naplnění mých předpokladů. Proto jsem zvolil kompromis v podobě rozdělení výukového materiálu na dvě části, první pro začátečníky a druhou pro mírně pokročilé. Problematiku pro začátečníky jsem podle mého názoru obsáhl dostatečně, obsahuje návod jak vytvořit službu, jak ji hostovat a jak si vytvořit klientskou aplikaci. Pro pokročilejší zájemce jsem obsáhl témata přenosu kolekcí, správu výjimek a škrcení služeb ve WCF.

Bibliografie

[1] BUSTAMANTE, Michele Leroux. *Learning WCF*. 1st edition. California (Sebastopol) : O\'Reilly Media, Inc., 2007. 608 s. ISBN 978-0596101626.

[2] KLEIN, Scott. *Professional WCF Programming .NET Development with the Windows Communication Foundation.* 1st edition. Indianapolis (Indiana) : Wiley Publishing, Inc., 2007. 430 s. ISBN 978-0-470-08984-2.

[3] LÖWY, Juval. *Programming WCF Services.* John Osborn; Robert Romano and Jessamyn Read. 2nd edition. California (Sebastopol) : O\'Reilly Media, Inc., 2008. 750 s. ISBN 978-0-596-52130-1.

[4] MCMURTRY, Craig, MERCURI, Marc, WATLING, Nigel. *Microsoft Windows Communication Foundation: Hands-on.* 1st edition. Indianapolis (Indiana) : Sams, 2006. 560 s. ISBN 978-0-672-32877-0.

[5] MCMURTRY, Craig, et al. *Windows Communication Foundation Unleashed*. Indianapolis (Indiana) : Sams, 2007. 720 s. ISBN 978-0672329487.

[6] NAGEL, Christian, et al. *Professional C# 2008*. Indianapolis (Indiana) : Wiley Publishing, Inc., 2008. 1848 s. ISBN 978-0470191378.

[7] PEIRIS, Chris, MULDER, Dennis, et al. *Pro WCF : Practical Microsoft SOA Implementation.* 1st edition. New York : Apress, 2007. 512 s. ISBN 978-1590597026.

[8] RESNICK, Steve, CRANE, Richard, BOWEN, Chris. *Essential Windows Communication Foundation: For .NET Framework 3.5.* 1st edition. Boston (Massachusetts) : Addison-Wesley Professional, 2008. 608 s. ISBN 978-0-321-44006-8.

[9] SHARP, John. *Microsoft Windows Communication Foundation Step by Step.* 1st edition. Redmond (Washington) : Microsoft Press, 2007. 448 s. ISBN 978-0-7356-2336-1.

[10] SMITH, Justin. *Inside Microsoft Windows Communication Foundation.* Redmond (Washington) : Microsoft Press, 2007. 304 s. ISBN 978-0-7356-2306-4.

[11] KUMAR, Saravana. *Saravanakumar'S WCF Tutorial.net* [online]. 2009 [cit. 2010-02-18]. WCF tutorial. Dostupné z WWW: <http://wcftutorial.net/>.

[12] LACKO, Ľuboslav. *Nová vlna technológií pre Visual Studio 2008 a .Net Framework 3.5* [online]. [s.l.] : [s.n.], 2008 [cit. 2010-02-10]. Dostupné z WWW: <http://download.microsoft.com/download/a/5/d/a5d67517-57f3-46cb-b0a1-1a451b680b92/Visual_Studio_8_Net%20Framework_3_5.pdf>.

[13] LARYŠ, Kryštof. *.NetStudent* [online]. 2.0.4. 2009 [cit. 2010-02-18]. WCF (Windows Communication Foundation) pro začátečníky – 0. díl. Dostupné z WWW: <http://www.netstudent.cz/Články/tabid/56/articleType/ArticleView/articleId/222/Default.aspx>.

[14] Microsoft Corporation. *Windows Communication Foundation* [online]. Microsoft Corporation, c2007 , 2009-12-09 [cit. 2010-01-01]. Text v angličtině. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/ms735119.aspx>.

[15] Microsoft Corporation. *Přehled architektury .NET* [online]. [s.l.] : [s.n.], 2002 [cit. 2010-02-10]. Dostupné z WWW: <http://download.microsoft.com/download/8/6/c/86c09926-affc-4e14-bec0-3c45cd989436/Prehled_architektury_NET.pdf>.

[16] Microsoft Corporation. *Příručka vývojáře pro přechod na platformu Microsoft .NET* [online]. [s.l.] : [s.n.], 2004 [cit. 2010-02-10]. Dostupné z WWW: <http://download.microsoft.com/download/8/6/c/86c09926-affc-4e14-bec0-3c45cd989436/DevelopersGuide2Moving2MicrosoftNET.pdf>.

[17] *Galin Iliev [Galcho] Blog!* [online]. 2008 [cit. 2010-02-10]. .NET Layer Cake. Dostupné z WWW: <http://www.galcho.com/Blog/PermaLink.aspx?guid=cdd4b3a0-e4e1-4b3f-9219-2046929e07d4>.

[18] Microsoft Corporation. *MSDN Microsoft* [online]. 2010 [cit. 2010-03-15]. BizTalk Server and WCF Integration. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/bb973215.aspx>.

[19] Microsoft Corporation. *Získání verze Internet Information Server (IIS)* [online]. Microsoft Corporation, c2010 , 2009-12-09 [cit. 2010-01-01]. Text v češtině. Dostupný z WWW: <http://support.microsoft.com/kb/224609>.

Rejstřík

Přílohy

Příloha č. 1:

Dotazník k výukovému materiálu WCF

Vážený respondente, studuji na Pedagogické fakultě Jihočeské univerzity v Českých Budějovicích třetí ročník bakalářského oboru „Výpočetní technika“. V současné době zpracovávám bakalářskou práci na téma „Služby ve Windows Communication Foundation“. Mám za úkol vytvořit výukový materiál pro tuto technologii a rád bych otestoval, jak se mi práce zdařila. Touto cestou bych Vás chtěl požádat o anonymní vyzkoušení výukového materiálu a o podání zpětné vazby o kvalitě vyplněním tohoto dotazníku. Získané výsledky zpracuji a uveřejním ve své práci. Zvolené odpovědi prosím zakroužkujte.

Identifikace respondenta

Pohlaví: Muž Žena

Věk: .....

Ohodnoťte své dosavadní programátorské schopnosti v oblasti objektově orientovaného programování:

Nezkušený Začátečník Pokročilý Profesionál

Setkali jste se již někdy s technologií WCF? Ano Ne

Dokončili jste celý výukový kurz? Ano Ne

Otázky

Kolik času Vám přibližně zabralo pochopení problematiky obsažené ve výukovém materiálu?

Méně než 1 týden Méně než 2 týdny Méně než měsíc Více než měsíc

Ohodnoťte celkovou srozumitelnost podání informací, obsažených v tomto výukovém materiálu (výukových lekcích), o WCF:*

1 2 3 4 5

Myslíte si, že byly naplněny stanovené cíle výuky?

Ano Ne

Ohodnoťte kvalitu zpracování doprovodných výukových textů:

1 2 3 4 5

Ohodnoťte kvalitu zpracování připravených ukázkových příkladů pro výuku:

1 2 3 4 5

Ohodnoťte kvalitu zpracování doprovodných ozvučených videoukázek:

1 2 3 4 5

Zaujala Vás technologie WCF natolik, že se jí budete věnovat i nadále?

Ano Ne

* Poznámka: Hodnotí se jako ve škole, tedy ve stupnici od 1 – 5.

Příloha č. 2:

Obsah přiloženého CD