

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

JAZYK XML
V MICROSOFT VISUAL FOXPRO 9.0

Diplomová práce

Knihovna JU - PF



3115172686

Petra Paletářová
České Budějovice 2006

Anotace

V nových verzích jazyka Microsoft Visual FoxPro se stále více prosazují moderní trendy aplikací databázových technologií a především propojení databází na webové služby. Nejvýznamnějšími technologiemi poslední doby jsou webové služby (WSDL) a formát XML.

Cílem této diplomové práce je zmapovat úlohu a význam jazyka XML a webových služeb založených na tomto jazyku ve světě internetových technologií a především podat přehled současných možností jazyka Microsoft Visual FoxPro 9.0 (dále jen MS VFP 9.0) v oblasti použití a implementace webových služeb.

Annotation

New versions of Microsoft Visual FoxPro increasingly employ modern trends in database technology applications, particularly database links to web services. The most significant technologies have lately been web services (WSDL) and the XML format.

The aim of this diploma work is to map the role and significance of XML and XML-based web services in the field of internet technologies and, above all, to give a survey of the present possibilities provided by Microsoft Visual FoxPro 9.0 (further referred to as MS VFP 9.0) in the field of web services use and implementation.

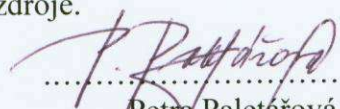
Poděkování

Děkuji vedoucímu této diplomové práce panu Mgr. Miloši Prokýškovi za vysvětlení problematiky webových služeb ve vývojovém nástroji MS VFP 9.0, za cenný čas, který mi věnoval v rámci mých častých dotazů a především za trpělivé vedení, díky němuž nabyla tato práce své konečné podoby.

Mé poděkování patří zároveň panu Ing. Václavu Novákovi, CSc. za cenné rady týkající se problematiky webových služeb v jazyce C#.

V neposlední řadě musím poděkovat všem, kteří mi byli nápomocni radou na portálu <http://www.pandora.cz> a díky nimž jsem postupně odstraňovala nedostatky ve zdrojových kódech svých zkušebních programů, které byly příčinou nekonečných chybových hlášení.

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně, a že jsem uvedla veškerou použitou literaturu i jiné zdroje.


.....
Petra Paletářová

Obsah

1. Úvod.....	8
2. Stručně o XML	10
2.1 Vznik XML	10
2.2 Význam a využití XML	11
2.3 Iniciativy vedoucí ke standardizaci podpory XML.....	14
3. XML a databáze	16
3.1 Úvod.....	16
3.2 Základní druhy XML databázových produktů	18
3.4 Mapování XML do databází	21
4. XML a Microsoft Visual FoxPro 9.0	24
4.1 Úvod.....	24
4.2 Zakomponované funkce pro práci s XML daty	25
4.3 Třídy XMLAdapter, XMLTable a XMLField	29
4.4 Vylepšení MS VFP 9.0	32
5. Webové služby	33
5.1 Úvod do webových služeb	33
5.2 Infrastruktura webových služeb	36
5.3 Budoucnost webových služeb	45
6. Microsoft Visual FoxPro 9.0 a webové služby	47
6.1 Jak webová služba funguje.....	47

6.2 MS VFP 9.0 v úloze konzumenta webové služby.....	49
6.3 Tvorba webových služeb v MS VFP 9.0	50
7. Bezpečnost webových služeb	55
7.1 Úvod.....	55
7.2 Různé úrovně nasazení bezpečnosti.....	56
8. Závěr.....	62
Zdroje.....	64
Přílohy	
Příloha 1: Zadání bakalářské práce	
Příloha 2: Seznam obrázků	

1. Úvod

Technologie XML je v mnoha produktech a řešeních doslova modlou posledních let. V dalším textu je tato problematika probírána s určitým nadhledem ve vztahu k databázovým technologiím a webovým službám. Pozornost je tedy věnována jak úvodu do problematiky jazyka XML a webových služeb na něm založených, tak současně i možnostem vývojového nástroje MS VFP 9.0 v oblasti podpory jazyka XML a v rovině použití a implementace webových služeb. Závěrem jsou zmíněny bezpečnostní mechanismy, které představují klíčovou oblast rozhodující o masovém rozšíření webových služeb.

Jedním z pilířů lidské společnosti jsou informace, většinou zaznamenané formou dokumentů, případně formou více či méně strukturovaných dat v databázích. Tuto skutečnost nástup výpočetní techniky jen potvrdil. Informace a jejich efektivní zpracování mají totiž dnes pro mnoho jednotlivců i firem strategický význam. Ovšem právě výpočetní technika vnesla do zpracování dokumentů značný chaos. Počítače totiž zatím neumí to, co je pro gramotného člověka naprosto přirozené – „automaticky“ rozpoznat takřka libovolnou strukturu dokumentu a na jejím základě správně interpretovat obsažené informace. XML je technologie, která do této problematiky velkou mírou zasahuje a vnáší do ní obrovský pokrok.

V posledních několika málo letech se jako reakce na vznik právě technologie XML stále více hovoří o webových službách a aplikacích na tomto jazyku založených. Lze pozorovat odklon od chápání webu pouze jako média pro poskytování informací a informačních zdrojů k pojetí webu jako nástroje

pro poskytování právě služeb a aplikací. Využití webu tímto způsobem však vyžaduje dobře si rozmyslet účel, pro jaký má být použit, a cíle, jakých má být dosaženo. Z toho pak již vyplývají i nástroje, kterých bude třeba použít pro jejich dosažení.

Tím jasně vystupuje do popředí potřeba sjednocujícího nástroje pro webové aplikace. Takovým nástrojem jsou v současnosti s úspěchem **webové služby s XML rozhraním**, nezávislým na platformách ani firemních formátech. Znamená to, že informační systémy, služby a aplikace na něm založené budou mít dlouhou životnost za podmínky dodržování standardů. Současně lze konstatovat, že vývoj směřuje k využívání distribuovaných služeb a databází, jež využívají právě takováto XML rozhraní pro přístup k nim.

2. Stručně o XML

2.1 Vznik XML

Počítače byly od samého počátku využívány zejména pro přípravu a publikování textu. Jednou vytvořený dokument byl dříve úzce svázán s výstupním zařízením konkrétního výrobce. Tento stav rozhodně nebyl ideální, proto vzniklo několik systémů, které problém nekompatibility různých výstupních zařízení řešily. Mezi nejrozšířenější z těchto systémů patřily bezesporu **troff** a **TEX**. Programy jako TEX se však hodí pouze pro zpracování dokumentů, které se mají ve výsledku tisknout. Vznikla proto potřeba vývoje jazyka, který by umožnil označit význam jednotlivých částí textu, a ne jejich vzhled. [6]

Asi prvním známým značkovacím jazykem byl **GML** (Generalized Markup Language), který vytvořili Charles Goldfarb, Edward Mosher a Raymond Lorie při práci na systému pro uchovávání a následné využití právních textů pro IBM. [6]

V roce 1986 byl v ISO normě 8879 definován jazyk **SGML** (Standard Generalized Markup Language), který umožňoval definici vlastních sad značek a jejich vzájemných vztahů. Standard SGML byl velmi komplexní, jeho úplná implementace však byla velice náročná, nebyl schopný reagovat na požadavky webu a vznikl v době pomalých a drahých počítačů. Aby bylo možné z tohoto robustního standardu získat maximum, byl jazyk SGML vybaven minimalizač-

ními nástroji. Ty měly výsledné textové soubory stlačit na co nejmenší velikost. Důsledkem toho byly složité, drahé a pomalé analyzátory a velká finanční náročnost zavedení SGML standardu do praxe. Během deseti let jeho používání se nutně ukázalo, že v praxi se používá jen část jeho možností a je tedy nutné zjednodušit SGML pro potřeby webu. Tato nejdůležitější podmnožina byla vybrána jako nový jazyk – XML. [1]

Jazyk **XML** (eXtensible Markup Language) vznikl v rámci konsorcia W3C a je touto nezávislou organizací nadále rozvíjen. První verze byla zveřejněna v únoru 1998. Jde o jednoduchý, otevřený formát, nezávislý na operačním systému ani žádné úzce specializované technologii. Jednoduchost spočívá v založení na textovém formátu využívajícím jednoduchá syntaktická pravidla. Otevřenost spočívá ve zveřejnění a volném přístupu ke specifikaci jazyka, což je realizováno na webových stránkách konsorcia W3C.

Princip XML spočívá velmi zjednodušeně řečeno v tom, že značky, obepínající v textovém XML dokumentu určité části textu, označují jejich věcný význam. To, které značky a v jaké struktuře je možné použít, si autor stanoví sám v Definici Typu Dokumentu – DTD a to, jak se má dokument zobrazit, si určí pomocí stylů definic výsledného vzhledu XSL. Výsledkem je důsledné oddělení informačního obsahu od grafického vzhledu.

2.2 Význam a využití XML

Vyjmenujeme-li stručně hlavní výhody XML oproti speciálním formátům dat, jsou to především tyto:

- samopopisný, platformově nezávislý, otevřený textový formát,
- jednoduchá možnost validace a kontroly správnosti dat,

- možno použít různé kódování češtiny (jako znaková sada se používá ISO 10646, která dokáže pojmout všechny znaky dnes používaných jazyků, XML dokument může být v libovolném kódování),
- možnost přenosu i binárních dat,
- rozšiřitelnost s možností definovat vlastní značky a strukturu.

Význam formátu XML je tedy především tam, kde je třeba pořizovat, zpracovávat, ukládat a poskytovat jakákoliv data. Zde se XML ukáže jako vhodný nástroj pro práci s těmito daty a velmi se ocení uvedené výhody. [12]

Celou řadu výhod přináší XML obecně i ve spojení s databázovými platformami, za hlavní přednost je považováno usnadnění tvorby otevřených aplikací využívajících např. elektronickou výměnu dokumentů s rozdílnou strukturou. Podpora otevřenosti vychází z toho, že součástí každého dokumentu XML jsou nejen samotná data, ale také informace popisující jejich přesnou strukturu.

Následuje popis oblastí, ve kterých se XML nejčastěji nasazuje a přináší výhody oproti jiným technologiím.

B2B – business-to-business aplikace

V tomto případě se jedná o výměnu informací mezi obchodními partnery v elektronickém formátu. Použití XML pro tyto účely se jeví jako zcela ideální, protože přidání podpory tohoto formátu do stávajících účetních a ekonomických systémů je velice jednoduché. [6]

„Inteligentní“ webové stránky

S nasazením jazyka XML se počítá především na webu. Možnost definice vlastních značek, které přesně vyznačí význam jednotlivých částí stránky, bude mít pozitivní efekt na přesnost a rychlost vyhledávání informací. [6]

V zásadě lze XML pro tvorbu stránek využít dvěma způsoby. První, více revoluční přístup, počítá s tím, že stránky budou používat zcela vlastní sady značek. Pro mnoho aplikací je však mnohem jednodušší používat již zažité HTML značky a pouze je vhodně doplnit o pár dalších, kterými se označí části stránky důležité pro vyhledávání. [6]

XML se na však nemusí používat jen pro tvorbu stránek. Funkčnost mnoha webových aplikací je dnes rozložena mezi server a prohlížeč. XML je optimální formát i pro výměnu dat mezi serverem a prohlížečem. Klient si vyžádá část dat od serveru a formátování podle různých požadavků je již plně v rukou klienta. [6]

Metadata

Metadata pro dokument představují takové údaje jako autor dokumentu, datum vytvoření, vlastník copyrightu, druh dokumentu apod. Do kategorie metadat patří např. formát Microsoftu CDF, který umožňuje jednoduchou syntaxí, založenou na XML, definovat zajímavé internetové zdroje. [6]

Elektronické publikování

Pokud máme naše dokumenty uloženy v XML, můžeme je pomocí stylů velice snadno zcela automaticky konvertovat do mnoha dalších formátů. XML se proto používá zejména při tvorbě různé technické dokumentace, kdy lze z jednoho zdroje generovat mnoho formátů podle potřeb jednotlivých uživatelů. [6]

Univerzální datový formát

Při psaní větších programů je potřeba, aby si program pamatoval nastavení svých parametrů. Klasicky se parametry ukládají do různých textových nebo INI souborů, do registrů Windows apod. Tento způsob je však poměrně

komplikovaný (program musí načítat jednotlivé parametry z konfiguračního souboru). Pokud by si program ukládal data v XML, bylo by možné načíst je pomocí některé z knihoven pro práci s XML. Do souborů by bylo možné uložit i složité struktury a navíc by byl pořád jen textovým souborem. [6]

Pokud mají informační a komunikační systémy fungovat efektivně, je potřeba, aby se používaly jednotné formáty dat. [6]

2.3 Iniciativy vedoucí ke standardizaci podpory XML

Ke zveřejnění první verze jazyka XML (1.0) došlo v únoru 1998. V říjnu 2000 jsme se dočkali druhé edice. Původní verze XML je však stále v platnosti. To svědčí o její významné stabilitě. Jiná situace se však týká DTD. Od začátku vůči němu byly vznášeny výhrady, zejména proto, že se jedná o jazyk s jinými výrazovými prostředky, než má XML.

Norma XML, spolu se SOAP protokolem a WSDL formátem, tvoří základ webových služeb a aplikací. Nicméně uvedené normy nejsou dostatečné pro to, aby byla zaručena jejich bezpečnost, spolehlivost, rozšiřitelnost a správa. Proto IBM a Microsoft přišly s iniciativou **GXA** (Global XML Web Services Architecture). Východiskem pro GXA je pevně definovaný základ (transportní mechanismy, XML a SOAP protokol). Základním cílem je zabezpečit otevřenost, tj. možnost definovat standardy, které umožní vytvářet univerzální stavební prvky pro softwarové řešení, a schopnost zahrnout nové standardy, které si vyžádá reálné nasazování webových služeb.

GXA není zdaleka jedinou iniciativou, která si klade za cíl vytvořit standardy pro podporu XML a aplikací na něm založených. Na půdě W3C konsorcia byla ustanovena pracovní skupina **WSA** (Web Services Architecture), která v dubnu 2002 uveřejnila první pracovní návrh definující požadavky na architekturu webových služeb – Web Services Architecture Requirements.

I v tomto případě se klade za cíl zabezpečení interoperability, spolehlivosti, bezpečnosti, škálovatelnosti a rozšiřitelnosti webových služeb při současném zabezpečení jejich konzistence s webem.

K významným aktivitám ve vytváření dokumentů založených na XML se dochází na půdě společnosti OASIS (Organization for the Advancement of Structured Information Standards). Výsledkem jejího technického výboru UBL (Universal Business Language) je soubor specifikací souhrnně označených jako **ebXML** (Electronic Business XML).

Očekává se, že důležitou úlohu při rozšiřování technologie XML (a webových služeb) sehraje organizace **WS-I** (Web Services Interoperability). Začátkem února 2002 ji založili firmy Microsoft, IBM, BEA Systems a Intel. Dnes jsou jejími členy téměř všechny významné organizace a firmy, které se podílejí na rozvoji, nasazení a použití webových služeb XML.

Lze jen těžko předpovědět, která z uvedených iniciativ se v konečném důsledku prosadí. Jedno je však jisté. XML a technologie na něm založené ovládnou web. Poskytnou možnost vyřešit problémy vzájemné spolupráce systémů různých platforem. Budou prostředkem pro automatizaci nejen obchodních činností, ale i pro správu neustále rostoucího objemu údajů, kterými lidstvo disponuje. A to pro začátek není vůbec málo.

3. XML a databáze

3.1 Úvod

XML je dnes běžně používaným formátem pro popis dat. Bývá použit pro mnoho účelů – aplikace jej využívají k ukládání svých konfiguračních souborů a datových výstupů. Nezastupitelnou roli pak hraje v oblasti integrace a výměny dat mezi heterogenními systémy. Dnes je podpora XML dat integrována i do databázových serverů, avšak v praxi se stále setkáváme s tím, že jen málo vývojářů je schopno tuto podporu využít a zbytečně programují vlastní funkce pro práci s XML.

Současné nasazení XML v databázových aplikacích se odehrává převážně ve dvou rovinách:

- zaměřené na data (datově orientovaná XML data),
- zaměřené na dokument (dokumentově orientovaná XML data).

Datově orientované XML dokumenty

Datově orientované XML dokumenty bývají v literatuře označovány také jako „dokumenty zaměřené na data“. Toto pojmenování představuje XML dokumenty, které plní funkci samopopisné obálky pro přenos či manipulaci s daty. Jsou v drtivé většině určeny pro potřeby jiných programů, XML forma vyjádření takových dat je v podstatě druhotná a dočasná. [20]

Tyto dokumenty se vyznačují poměrně pravidelnou strukturou, jemně granulovanými daty a malým či žádným výskytem smíšeného obsahu v elementech.

Níže je uvedený jednoduchý příklad datově orientovaného XML dokumentu.

```
<objednávka>
  <číslo_o>4711</číslo_o>
  <zákazník>MFF UK</zákazník>
  <datum>2003-03-18</datum>
  <řádek číslo ř="1">
    <zboží>A10</zboží>
    <počet_kusů>12</počet_kusů>
    <cena>1090</cena>
  </řádek>
  <řádek číslo ř="2">
    <zboží>B43</zboží>
    <počet_kusů>600</počet_kusů>
    <cena>400</cena>
  </řádek>
</objednávka>
```

Dokumentově orientované XML dokumenty

U této varianty nasazení se využívá především možnosti sémantického strukturování tímto způsobem napsaného dokumentu. XML nám v tomto případě pomáhá pracovat s dílčími částmi dokumentu podle jejich významu. [20]

Tento typ dokumentu je určen přímo pro čtení nebo zpracování lidmi. Charakterizuje ho méně pravidelná či nepravidelná struktura, hrubá granulace dat a hojné používání smíšeného obsahu v elementech.

Následuje příklad, jak by mohl vypadat jednoduchý dokumentově orientovaný XML dokument.

```
<příručka>
  <nadpis1>Uživatelská příručka</nadpis1>
  <odstavec>Pokud byste se rád(a) zapsala na nějaký náš kurz, musíte se nejprve zaregistrovat do systému. Pouze tak obdržíte své uživatelské jméno a heslo, pod kterým se přihlásíte.</odstavec>
  <nadpis2>Základy práce se systémem</nadpis2>
  <odstavec>Systém rozlišuje, zda jste do něj korektně přihlášen, či nikoliv.</odstavec>
</příručka>
```

3.2 Základní druhy XML databázových produktů

Existence výše uvedených dvou rovin nasazení XML v databázových aplikacích zapříčinil vznik dvou odlišných základních databázových platform. V literatuře se lze však navíc ještě setkat s termíny jako Middleware, XML servery, Wrappery, systémy pro správu obsahu atd. Jedná se však většinou jen o přidanou funkcionalitu, kterou stávající produkt obsahuje.

Databázové systémy zpřístupňující XML data

Tyto platformy podporují technologii XML především prostřednictvím mapování dokumentu XML na relační datové struktury, případně na struktury objektové. Výhodou zůstává obecnost takového řešení – vývojáři mohou využít hlavních předností XML. Nevýhodou zůstává jistá ztráta výkonu – na převod mezi strukturami je nutná jistá časová režie. Navíc běžné mechanismy zvyšující výkon v relačních databázích nejsou příliš použitelné, například stromové indexové struktury jsou mnohem více efektivnější pro nalezení již konkrétní hodnoty, nikoliv prohledávání XML dokumentu. [5]

Podobně fungují také platformy nabízející speciální XML prezentační vrstvu s tím, že struktura dat není relační a v interním formátu co nejvíce odpovídá nárokům na XML. [5]

Variantou uvedených přístupů k zabudování podpory XML je možnost ukládání XML dokumentů do databáze jako celku s podporou například vyhledávání. Ve vyspělejších platformách je podpora XML přímo součástí databázového stroje. V důsledku je tím kompenzován – ale nikdy ne stoprocentně – pokles výkonu. [5]

Podporu XML na těchto úrovních je tedy možno brát jako přidanou hodnotu umožňující maximální využití nejlepšího z nejlepších. Mezi nejzdařilejší databázové platformy z tohoto pohledu patří především Microsoft SQL

Server 2000, Microsoft Access 2002, Oracle 8i, Oracle 9i, IBM DB2 či Sybase ASE 12.5.

Nativní XML databázové systémy

Právě existence relativně nového směru vývoje, který můžeme spatřovat v dokumentově orientovaných XML dokumentech, zapříčinil vznik nového typu databázových systémů. Vedle databázových platform podporujících technologii XML jako své rozšíření (zpřístupňují XML data) existují na trhu také databázové produkty, které používají XML jako nativní formát pro uložení dat.

Pro pojem nativní XML databázový systém neexistuje žádná formální, přesná a jednotná definice. V diskusním archivu na stránkách iniciativy XML:DB se lze setkat se dvěma definicemi.

První definice pochází přímo od iniciativy XML:DB a zní takto: „*Nativní XML databázový systém je databázový systém, který je od základu navržen k ukládání a manipulaci s XML daty. Přístup k datům se odehrává skrze XML a související standardy, např. XPath, XSLT, DOM či SAX. Pod tento pojem spadají všechny databázové systémy, jejichž reprezentace dat zachovává úplnou XML strukturu a související metadata. Formát uložení dat (objektová databáze, relační databáze,...) není důležitý a přístup k datům pomocí jiných prostředků než XML technologií není povolen. Základní jednotkou uložení dat v nativní XML databázi je XML dokument.*“ [19]

Za druhou definici vděčíme Ronaldu Bourretovi: „*Nativní XML databázový systém je databázový systém, který definuje model XML dokumentu a ukládá a získává dokumenty dle tohoto modelu. Model musí obsahovat minimálně elementy, atributy, PCDATA a zachovávat dokumentové pořadí prvků. Příkladem takového modelu je datový model XPath, XML Infoset a modely odvozené z DOM a z událostí SAX 1.0.*“ [19]

Z uvedených definic lze odvodit tři základní charakteristiky nativních XML databázových systémů:

- systém řízení báze dat pracuje s datovým modelem XML,
- základní minimální jednotka uložení dat je XML dokument,
- logická forma uložení dat v databázi není podstatná (může být libovolná).

Hlavním úkolem nativních XML databází je uložit XML dokument včetně jeho logické struktury a podoby. Dokument obdržíme zpět přesně v takové podobě, v jaké jsme jej předali databázi, včetně všech poznámek apod. Nejčastěji ukládají objektový model (DOM) XML dokumentu, který mapují do svého databázového schématu, přičemž databázové schéma může být libovolné – relační, objektové, hierarchické... DOM (Document Object Model) znázorňuje strukturu objektů, které tvoří XML dokument.

Výhodou takových řešení je především vyšší výkon aplikací využívajících XML, protože odpadá nezbytný mezikrok převodu z/do formátu XML. Tyto platformy nabízí drtivou většinu z toho, co je známo u „klasických“ databázových produktů. [5]

Mnohem podstatnější jsou další funkce, které nám nativní XML databáze přináší. Patří mezi ně například možnost indexování uložených dokumentů pro výrazné zvýšení výkonu a možnost provádění dotazů napříč sadou dokumentů umístěných v takzvané kolekci. [20]

Předpoklad úspěchu nativních XML platforem je zejména ve specializovaných aplikacích, případně v oblastech, kde je potřeba ukládat velké množství XML dokumentů. Podle známého databázového odborníka Mikea Stonebrakera nelze očekávat jejich masové rozšíření do konce této dekády. K podobnému závěru došli také účastníci panelové diskuse na semináři Databázový svět 2003, který byl pořádán ve Zlíně. Nativní XML databázové systémy tedy v nejbližší době nečeká žádná velká revoluce. Důvodem může být

zejména to, že databázové platformy, které „pouze“ zpřístupňují XML data, rozšiřují svou podporu XML tak rychle, že během několika málo let se z nich mohou stát plnohodnotné nativní XML databáze.

Za nejzdařilejší platformy tohoto druhu lze považovat především produkty Tamino od firmy Software AG, eXist od Wolfganga Meiera, Xindice od Apache Software Foundation a mnoho dalších.

3.3 Mapování XML dokumentů do databází

Mapováním XML dokumentů do databází se rozumí proces načtení, konverze a uložení dat XML dokumentu do databáze.

V dalším textu budou velice stručně zmíněny dva hlavní přístupy: mapování do tabulek a objektově-relační mapování. Důležitou vlastností obou přístupů je, že jsou obousměrné. Mohou být tedy použity k přenosu dat jak z XML do databáze, tak i z databáze do XML.

Mapování do tabulek

Mapování XML dokumentu do tabulek spočívá v tom, že na XML dokument nahlédneme jako na tabulku nebo množinu tabulek. Problém lze vysvětlit na jednoduchém příkladu. Mějme níže uvedený XML dokument obsahující data o zaměstnancích podniku.

```
<zamestnanci>
  <zamestnanec>
    <jmeno>Miroslav</jmeno>
    <prijmeni>Dostál</prijmeni>
    <pohlavi>muž</pohlavi>
  </zamestnanec>
  <zamestnanec>
    <jmeno>Marie</jmeno>
    <prijmeni>Bláhová</prijmeni>
    <pohlavi>žena</pohlavi>
  </zamestnanec>
</zamestnanci>
```

Tento XML dokument bude zmapován do relační tabulky zaměstnanci takto:

jmeno	prijmení	pohlaví
Miroslav	Dostál	muž
Marie	Bláhová	žena

Aby bylo možné transformovat XML dokument do tabulky, musí mít daný dokument určitou strukturu. Příklad vhodné obecné struktury je uveden dále.

```
<tabulka>
  <radek>
    <sloupec_1>...</sloupec_1>
    ...
    <sloupec_n>...</sloupec_n>
  </radek>
  ...
  <radek>
    <sloupec_1>...</sloupec_1>
    ...
    <sloupec_n>...</sloupec_n>
  </radek>
</tabulka>
```

Mapování XML dokumentů do tabulek však s sebou nese několik vážných nevýhod:

- je použitelný pouze na malou množinu XML dokumentů s konkrétní strukturou,
- z dokumentu nelze přenést jeho fyzickou strukturu, metainformace, komentáře a instrukce pro zpracování.

Výhodou tohoto přístupu je jeho jednoduchost.

Objektově-relační mapování

Princip tohoto přístupu je takový, že XML dokument je vyjádřen jako strom objektů a tyto objekty jsou posléze mapovány do databáze. Tento druh mapování se nejvíce používá ve spojitosti s relačními databázovými systémy.

Celý mapovací proces lze opět předvést na jednoduchém příkladu. Mějme níže uvedený XML dokument.

```
<zamestnanec>
  <jmeno>Miroslav</jmeno>
  <prijmeni>Dostál</prijmeni>
  <pohlavi>muž</pohlavi>
</zamestnanec>
```

Objekt vytvořený mapováním vypadá takto:

```
objekt zamestnanec {
  jmeno = "Miroslav"
  prijmeni = "Dostál"
  pohlavi = "muž"
}
```

Objektově-relační mapování je vhodné pro datové XML dokumenty. Mapovat obsah smíšených elementů, které se hojně objevují v dokumentových XML dokumentech objektově-relačním způsobem je neefektivní.

Toto mapování je založeno na propracované metodě, která s sebou přináší řadu specifik, např. mapování DTD do objektových schémat, mapování komplexních typů elementů, smíšeného obsahu a další. Popis těchto problémů je však nad rámec této práce, proto pro zájemce o tuto problematiku odkazují na internetové stránky Ronalda Bourreta: <http://rpbouret.com>.

V další kapitole bude pozornost věnována problematice podpory XML dokumentů v jazyce MS VFP 9.0. Jedná se o objektově-orientovaný relační databázový systém, který zpřístupňuje práci s XML daty jako své rozšíření, tzn. že mapuje XML dokumenty na relační nebo objektové datové struktury.

4. XML v Microsoft Visual FoxPro 9.0

4.1 Úvod

MS VFP 9.0 podporuje práci s XML daty (import nebo export XML dokumentů) prostřednictvím odpovídajících zakomponovaných funkcí nebo objektů typu XMLAdapter, XMLField a XMLTable. [8]

Správně strukturované XML dokumenty

Základním předpokladem pro další zpracování XML dokumentu je, aby byl správně strukturovaný („well-formed“), tzn. aby splňoval následující základní pravidla:

- každý XML dokument musí být celý obsažen v jednom (kořenovém) XML elementu,
- každý počáteční tag se shoduje s ukončovacím, rozlišují se velká a malá písmena,
- pro každý počáteční tag existuje odpovídající ukončovací tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem,
- elementy se nesmí překřížovat,
- rezervované znaky jsou součástí XML syntaxe a nejsou proto interpretovány při použití v datové části XML dokumentu (jedná

se o znaky &, <, >, “, ‘). Pro jejich zápis je nutné použít znakové entity (&, <, >, ", ').

Následuje příklad správně strukturovaného XML dokumentu.

```
<?xml version="1.0"?>
<Data>
<ORDER>
  <CUSTOMER>Mary Baker</CUSTOMER>
  <ITEM>Coho Winery&apos;s Chablis</ITEM>
  <PRICE>$10.00</PRICE>
  <QUANTITY>1 Bottle</QUANTITY>
</ORDER>
</Data>
```

Podpora XML schémat

XML schémata (XSD) definují pravidla a odpovídající struktury pro XML dokumenty a zároveň velice komplexně řeší podporu různých datových typů. Jednak přímo obsahují několik nejběžnějších datových typů, z nichž jsou odvozeny jemnější typy, navíc je možné definovat i typy vlastní.

MS VFP 9.0 interpretuje XML dokumenty s nebo bez použití těchto schémat. Použití XML schématu zajišťuje, že jakýkoliv XML dokument použitý pro import nebo export dat obsahuje specifická data a řídí se definovanou strukturou. [8]

4.2 Zakomponované funkce pro práci s XML daty

MS VFP 9.0 umožňuje výměnu dat mezi aplikacemi převodem Visual FoxPro dat do XML a naopak. Pro tento účel jsou k dispozici funkce **CURSORTOXML()**, **XMLTOCURSOR()** a **XMLUPDATEGRAM()**. K jejich použití je nutné mít nainstalován Microsoft XML Core Services (MSXML) parser.

Funkce CURSORTOXML ()

```
CURTORTOXML(nWorkArea | cTableAlias, cOutput [, nOutputFormat  
[, nFlags [, nRecords [, cSchemaName [, cSchemaLocation [, cNameSpace  
]]]]]])
```

Funkce převádí VisualFoxPro kurzor do XML.

První parametr (**nWorkArea**) specifikuje pracovní oblast tabulky, ze které má být vytvořen řetězec XML. MS VFP 9.0 použije aktuální pracovní oblast, pokud zadáme 0 nebo žádnou hodnotu.

Druhý parametr (**cTableAlias**) určuje alias tabulky, ze které bude vytvořen XML řetězec.

Třetí parametr (**cOutput**) popisuje cestu a název souboru nebo paměťové proměnné, do které budou uloženy výsledky. Pokud je parametr **nFlags** nastaven na 0, znamená to, že MS VFP 9.0 vrátí výsledky do paměťové proměnné. Jestliže proměnná neexistuje, je automaticky vytvořena. Pokud je parametr **nFlags** nastaven na hodnotu 512, jsou výsledky uloženy do souboru. Pokud neexistuje, je vytvořen, pokud již takový soubor existuje, je přepsán.

Parametr **nOutputFormat** specifikuje formát XML řetězce. Nabývá hodnot 1, 2 nebo 3. Popis těchto hodnot lze najít v nápovědě k MF VFP 9.0.

Parametr **nFlags** udává formátování a cíl XML řetězce, který je vytvořen. Nabývá hodnot 0, 1, 2, 4, 8, 16, 32, 512, 4096 a 32768. Pro podrobnější informace opět odkazují na nápovědu.

Další parametr (**nRecords**) specifikuje počet záznamů, které jsou vráceny ve výsledku. Výchozí hodnota je 0, která udává navrácení všech záznamů.

Parametr **cSchemaName** určuje jméno a umístění XML schématu. Hodnotou může být konkrétní název schématu, např. „MySchema.xsd“, nebo „1“ (pro inline schéma). Pokud je hodnotou prázdný řetězec, není použito žádné schéma.

Parametr **cSchemaLocation** definuje místo, kde má aplikace, která čte XML data, hledat soubor s odpovídajícím schématem.

Poslední parametr (**cNamespace**) udává jmenný prostor XML nebo schématu a jeho výchozí hodnotou je prázdný řetězec.

Návratovou hodnotou funkce **CURSORTOXML()** je počet bytů, který je zapsán do soubor nebo paměťové proměnné.

Funkce **XMLTOCURSOR()**

```
XMLTOCURSOR(eExpression | cXMLFile [, cCursorName [, nFlags]])
```

Funkce převádí XML text do kurzoru nebo tabulky Visual FoxPro.

Parametr **eExpression** specifikuje XML text nebo výraz, který zahrnuje platná XML data. Hodnotou může být paměťová proměnná, obsah memo pole, výsledek HTTP požadavku, návratová hodnota metody SOAP nebo ADO řetězec. Funkce generuje chybu, pokud hodnota parametru **eExpression** není nalezena nebo neobsahuje platná XML data.

Druhý parametr (**cXMLFile**) určuje název nebo (volitelně) cestu ke XML souboru, který je umístěn na lokálním počítači nebo v síti.

Třetí parametr **cCursorName** definuje název kurzoru, do kterého jsou uloženy výsledky a vytváří daný kurzor v aktuální pracovní oblasti. Pokud kurzor už existuje nebo je právě otevřen, MS VFP 9.0 ho zavře a vytvoří nový v dosud nepoužité pracovní oblasti. Pokud je hodnotou parametru prázdný řetězec, je vytvořen kurzor s názvem „XMLRESULT“.

Parametr **nFlags** určuje, jak je zacházeno s hodnotou parametru **eExpression**. Nabývá hodnot 0, 4, 512, 1024, 2048, 4096, 8192, 32768, 65536 a 131072. Více informací lze nalézt v nápovědě.

Návratovou hodnotou funkce **XMLTOCURSOR()** je počet vytvořených záznamů.

Funkce XMLUPDATEGRAM()

```
XMLUPDATEGRAM([ cAliasList [, nFlags [, cSchemaLocation]])
```

Funkce odráží změny provedené nad vyrovnávanou tabulkou nebo kurzorem a vrací řetězec znaků (tzv. XML UpdateGram). XML UpdateGram reprezentuje předchozí a následné stavy změněných částí tabulek nebo kurzorů ve VFP 9.0.

V případě použití této funkce je nutné mít nastaveno SET MULTILOCKS na hodnotu ON a zpřístupnit pesimistické nebo optimistické vyrovnávání tabulek. Je také vhodné nastavit vlastnost KeyFieldList funkce CURSORSETPROP() na seznam odpovídajících klíčových atributů, se kterými se bude dále pracovat.

Parametr **cAliasList** udává seznam otevřených tabulek nebo kurzorů (oddělují se čárkou) seřazený buď podle názvu nebo čísel pracovních oblastí v jakékoliv kombinaci, který má být zahrnut ve výsledném XML UpdateGramu. V případě, že hodnota tohoto parametru je nastavena na prázdný řetězec, použije VFP 9.0 všechny tabulky nebo kurzory otevřené v aktuální pracovní oblasti, nad kterými byly provedeny změny.

Parametr **nFlags** určuje, zda má být vrácen formátovaný soubor.

Parametr **cSchemaLocation** udává název a umístění XML schématu (pokud existuje).

Pro ilustraci uvádím jednoduchý příklad. Pomocí následujících příkazů vytvoříme kurzor Test s atributy mField a cField. Do kurzoru vložíme čtyři záznamy, nastavíme vyrovnávání kurzoru a atribut cField nastavíme jako klíčový pro další zpracování. Poté změníme záznamy, které mají hodnotu atributu cField menší než 3.

```
SET MULTILOCKS ON
CREATE CURSOR Test (mField M, cField I AUTOINC)
INSERT INTO Test (mField) VALUES ("123456789")
INSERT INTO Test (mField) VALUES ("23456789")
```

```

INSERT INTO Test (mField) VALUES ("3456789")
INSERT INTO Test (mField) VALUES ("456789")
CURSORSETPROP("Buffering",5)
CURSORSETPROP("keyfieldlist",'cField')
UPDATE Test SET mField = "XXXXXXX" WHERE cField < 3
? XMLUPDATEGRAM()

```

Výsledný XML UpdateGram je uveden níže.

```

<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
  <updg:sync>
    <updg:before>
      <test>
        <mfield>123456789</mfield>
        <cfield>1</cfield>
      </test>
    </updg:before>
    <updg:after>
      <test>
        <mfield>XXXXXXX</mfield>
        <cfield>1</cfield>
      </test>
    </updg:after>
    <updg:before>
      <test>
        <mfield>23456789</mfield>
        <cfield>2</cfield>
      </test>
    </updg:before>
    <updg:after>
      <test>
        <mfield>XXXXXXX</mfield>
        <cfield>2</cfield>
      </test>
    </updg:after>
  </updg:sync>
</root>

```

4.3 Třídy XMLAdapter, XMLTable a XMLField

VFP 9.0 zvyšuje svou funkčnost při práci s XML daty pomocí podpory formátů XML DiffGram a DataSets (.NET Framework ADO .NET). K tomuto účelu VFP 9.0 zavádí třídy **XMLAdapter**, **XMLField** a **XMLTable**.

Třída XMLAdapter

Pomocí třídy **XMLAdapter** lze importovat XML za účelem vytváření tabulek a z těchto tabulek lze opět zpětně vytvořit a generovat XML data. [8]

Objekt typu XMLAdapter uchovává objektové odkazy na XML schéma a uzly typu Document Object Model (DOM). Navíc tento objekt obsahuje kolekci objektů typu XMLTable. Každý objekt typu XMLTable může obsahovat další objekty typu XMLTable nebo, v častějším případě, kolekci objektů typu XMLField. [8]

Použitím této třídy je možné:

- interpretovat formáty schémat jako např. W3C XML schémata (XS), XDR a ADO Recordset schémata,
- obdržet z XML zdroje XML data a odpovídající schémata,
- vytvořit kurzor založený na schématu a naplnit ho daty reprezentovanými objekty typu XMLTable,
- vložit objekty typu XMLTable do XMLAdapter objektu,
- generovat z jedné nebo více tabulek a kurzorů XML dokument se schématem ve formátu W3C XML.

Třída XMLAdapter zahrnuje tyto vlastnosti: CodePage, DeclareXMLPrefix, DisableEncode, ForceCloseTag, FormattedOutput, IsDiffGram, IsLoaded, IXMLDomElement, MapBinary, MapN19_4ToCurrency, MapVarchar, NoCpTrans, PreserveWhiteSpace, RespectCursorCP, RespectNesting, SelectionNamespaces, SOM, Unicode, UseCodePage, UTF8Encoded, WrapCharInCDATA, WrapMemoInCDATA, XMLConstraints, XMLName, XMLNameIsXPath, XMLNamespace, XMLPrefix a XMLSchemaLocation.

Použit lze následující metody: AddTableSchema(), Attach(), LoadXML(), ReleaseXML() a ToXML().

Třída **XMLTable**

Kolekce tabulek v objektu typu **XMLAdapter** je reprezentována pomocí objektů typu **XMLTable**. Každý objekt typu **XMLTable** může obsahovat další objekty tohoto typu nebo kolekci objektů typu **XMLField**. [8]

Třída **XMLTable** zahrnuje tyto vlastnosti: **Alias**, **ChildTable**, **DeclareXMLPrefix**, **FirstNestedTable**, **NestedInto**, **NextSiblingTable**, **ParentTable**, **XMLAdapter**, **XMLConstraints**, **XMLName**, **XMLNameIsXPath**, **XMLNamespace** a **XMLPrefix**.

K dispozici jsou tyto metody: **ApplyDiffgram()**, **ChangesToCursor()**, **Nest()**, **ToCursor()** a **Unnest()**.

Třída **XMLField**

Každý objekt typu **XMLField** reprezentuje strukturu, která je pro VFP 9.0 významná z hlediska interpretace XML dat. Informace uložené v každém objektu typu **XMLField** zahrnují potřebné detaily (správný datový typ, velikost a jiné) k vytvoření konkrétního pole v kurzoru. Pokud VFP 9.0 generuje XML data z kurzoru, poskytují objekty typu **XMLField** informace k vytvoření XML dokumentu a schématu. [8]

Třída **XMLField** zahrnuje tyto vlastnosti: **Alias**, **DataType**, **DisableEncode**, **FractionDigits**, **IsAttribute**, **IsBase64**, **IsBinary**, **IsNull**, **Keyfield**, **MaxLength**, **NoCpTrans**, **Unicode**, **WrapInCDATA**, **XMLName**, **XMLNameIsXPath**, **XMLTable**, **XMLType**, **XSDfractionDigits**, **XSDmaxLength**, **XSDtotalDigits** a **XSDtype**.

Bližší popis všech uvedených vlastností a metod je součástí nápovědy.

4.4 Vylepšení MS VFP 9.0

Jazyk Microsoft Visual FoxPro prošel od původní verze až k dnešní verzi 9.0 zásadním vývojem. Jsou navrhovány stále nové a nové programátorské prvky, které zlepšují produktivitu práce. Ve verzi 9.0 jde zejména o následující vylepšení:

- generátoru sestav a tiskových sestav obecně,
- datových charakteristik a podpory XML,
- jazyka SQL,
- tříd,
- programovacího jazyka Visual FoxPro,
- průvodců a návrhářů,
- další (nová rezervovaná slova, nová funkčnost verze 9.0)

Podpora XML se objevuje teprve ve verzi 7.0 prostřednictvím funkcí `CURSORTOXML()`, `XMLTOCURSOR()` a `XMLUPDATEGEAM()` pro práci s XML daty. Zde jsou však uvedené funkce dost omezené co se týče možných hodnot parametrů a dalších prvků.

Verze 8.0 navíc vedle funkcí pro práci s XML daty podporuje stejně jako verze 9.0 třídy `XMLAdapter`, `XMLTable` a `XMLField`. Po této stránce jsou si tyto verze v zásadě dost podobné. Odlišnost se týká především drobných vylepšení jednotlivých metod v rámci uvedených tříd.

5. Webové služby

5.1 Úvod do webových služeb

Akceleraci rozvoje webových služeb umožnily otevřené standardy. Bez nich by Internet ani žádná jeho služba nemohla existovat. Potřeba těchto otevřených standardů si vynutila i současnou existenci formátu XML. A právě XML a webové služby na něm založené se stávají nástrojem pro integraci webových aplikací. Chceme-li správně pochopit jejich podstatu a principy, je vhodné začít pohledem zpět – jak to vypadalo s provozováním aplikací dříve.

Nejstarší aplikace provozované ve světě počítačů měly monolitický charakter. To znamená, že byly vytvořeny jako jediný a dále nedělitelný celek. Provozovat se daly a dodnes dají jak na samostatných, tak i na sdílených počítačích vybavených terminály. V prostředí sítě (ať už lokální nebo rozlehlé) se dají používat také, a to v režimu **host/terminál**. Tradičně tyto monolitické aplikace běží na počítačích označovaných jako „hostitelé“ a uživatelé vzdálených počítačů (v roli terminálů) k těmto aplikacím přistupují pomocí mechanismů vzdáleného přihlašování. Nověji byl tento model zdokonalen tak, že umožňuje provozovat i aplikace s grafickým výstupem.

Dalším vývojovým stádiem bylo rozdělení původně monolitických aplikací na dvě samostatné části (model **klient/server**). Zatímco u modelu host/terminál víceméně nezáleží na tom, zda pracuje na izolovaných počítačích s terminály nebo v síťovém prostředí, model klient/server již se síťovým pro-

středím přímo počítá (předpokládá se, že klient i server budou provozováni na různých uzlech sítě s možností vzájemné komunikace). Další významnou charakteristikou je těsná vazba mezi klientem a serverem. Tyto části tvoří dohromady nějakou aplikaci a jsou spolu svázány zejména pravidly vzájemné komunikace, formáty předávaných dat atd. V praxi to znamená, že ke každé službě implementované v architektuře klient/server musí existovat nejen specifické servery, ale také specifickí klienti, uzpůsobení povaze příslušné aplikace.

Později nutně došlo k tomu, že se všechny původně samostatné aplikace přesunuly na platformu webu. Staly se z nich jakési nadstavby nad webem a uživatelé s nimi mají možnost pracovat skrze své klientské programy. Ty komunikují s webovým serverem, za kterým stojí jiný server (ten přísluší již dané specifické aplikaci a zajišťuje, to, co je pro ni vlastní – tzv. aplikační logiku). Sám má obvykle za sebou ještě další server, který řeší práci s daty (tzv. databázový server). Situace je znázorněna na obrázku 5.11. Hlavní předností takového **třívrstvé architektury** je skutečnost, že všem uživatelům stačí jediný klientský program pro práci s různými aplikacemi. To je dáno tím, že tento zmíněný klientský program realizuje pouze univerzální prezentační funkce, zatímco to, co je pro danou aplikaci specifické, realizuje aplikační server.

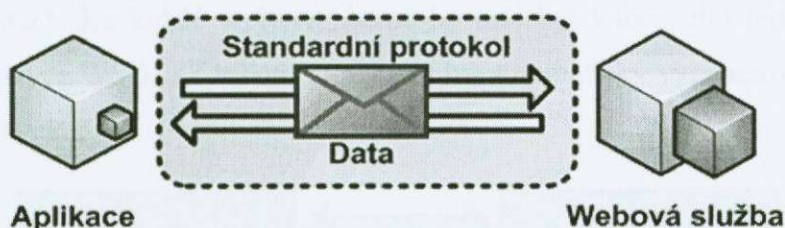


Obr. 5.11

Pokud přeskočíme nepřilíš úspěšné další stádium vývoje webových služeb – éru **tenkých klientů** (tzv. network computing), kdy jsou aplikace sice umístěné na síti, ale pro jejich běh dochází k jejich instalaci na klientské počítače, dostáváme se již k samotné definici webové služby.

Pojem **webová služba** není příliš starý, jeho idea je však známa již delší dobu. Na internetových stránkách <http://www.uddi.org> lze nalézt poměrně výstižnou definici pojmu webová služba: „*Webová služba je sebepopisující, zcela samostatná, modulární jednotka aplikační logiky, která poskytuje určitou business funkcionalitu jiným aplikacím prostřednictvím Internetu. Aplikace přistupují k webovým službám prostřednictvím všeobecně rozšířených webových protokolů a datových formátů (HTTP, XML), aniž by se musely starat o to, jak je daná služba implementována. Webové služby mohou být kombinovány a párovány s jinými webovými službami, aby prováděly řízení rozsáhlejších informačních toků nebo business transakce.*“

Webová služba je prostě web bez viditelného rozhraní (HTML stránek), ke kterému přistupují jen aplikace, nikoliv uživatel (jedná se o interakci stroj – stroj, které spolu komunikují na základě nějakého standardizovaného protokolu). Tuto situaci názorně ukazuje obrázek 5.12.



Obr. 5.12

Namísto URL webové stránky je volána URL metody vzdáleného objektu. Ze serveru jsou místo HTML stránky zaslány datové typy XSD, zabalené do zpráv XML.

Webové služby se dají použít v libovolném webovém prostředí, ať už jde o Internet, intranet nebo extranet. Pro vyhledání a přístup k webové službě stačí její URL. Teoreticky tuto URL mohou využít různé protokoly, v praxi je to prozatím téměř vždy HTTP.

Existence různých webových služeb umožňuje programátorům a tvůrcům informačních systémů tvořit nové webové aplikace tyto služby využívající. Jestliže pak budou webové služby založeny na formátech XML pro rozhraní

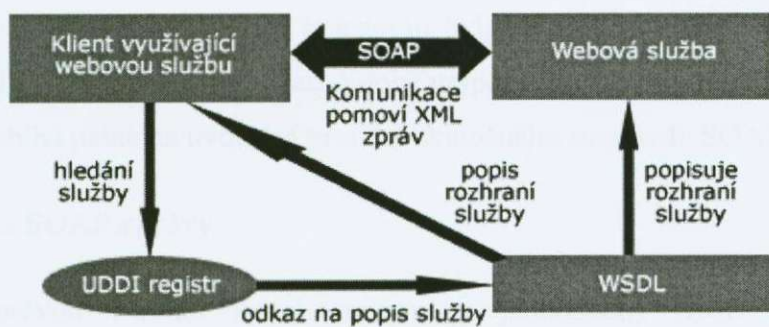
umožňující přístup k nim, bude možné budovat nové aplikace, které budou využívat předností těchto XML rozhraní, jako je například nezávislost na platformě. Tím se webové služby XML skutečně stávají integrujícím nástrojem pro budování webových aplikací.

5.2 Infrastruktura webových služeb

Celá infrastruktura webových služeb je založena na třech základních technologiích:

- **SOAP protokol (Simple Object Access Protocol)**
- **WSDL (Web Service Description Language)**
- **UDDI (Universal Description, Discovery and Integration)**

Vzájemné vztahy mezi těmito třemi technologiemi jsou zachyceny na obrázku 5.21. Ke každé webové službě by měl být k dispozici její formální popis v jazyce WSDL. Z tohoto popisu již lze automaticky vygenerovat soapový požadavek.



Obr. 5.21

SOAP protokol

SOAP je protokol pro posílání zpráv XML a je základem webových služeb. Ostatní standardy jako WSDL a UDDI vznikly až po uvedení SOAP protokolu a jen dále rozšiřují jeho možnosti a snadnost použití. SOAP umožňu-

je zaslání XML zprávy mezi dvěma aplikacemi (pracuje na principu peer-to-peer). Zpráva je jednosměrný přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv lze pomocí tohoto protokolu snadno implementovat běžné komunikační scénáře.

Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek/odpověď (request/response). Jedna aplikace pošle v XML zprávě požadavek jiné aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde soapová zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď.

První verze protokolu SOAP (1.0) vznikla na konci roku 1999 jako výsledek společné práce firem DevelopMentor, Microsoft, UserLand Software a Lotus Development Corporation, které chtěly vytvořit protokol pro vzdálené volání procedur (RPC) založený na XML. Protokol navazoval na o rok mladší, jednodušší a méně flexibilní protokol XML-RPC. V roce 2000 se k podpoře přihlásila i firma IBM a W3C konsorciu byla zaslána nová verze protokolu SOAP 1.1. Tato verze je v současné době nejpoužívanější. Na půdě W3C konsorcia probíhá práce na uvolnění prvního skutečného standardu SOAP 1.2.

Struktura SOAP zprávy

Zprávou v tomto smyslu rozumíme jednoduchý XML dokument, s kořenovým elementem **<Envelope>**. Tento element musí být ve zprávě SOAP obsažen. Jeho součástí jsou deklarace jmenných prostorů a dodatečných atributů. V této obálce jsou pak uzavřeny dva elementy **<Header>** (hlavička) a **<Body>** (tělo). Hlavička je nepovinná a používá se k přenosu pomocných informací pro zpracování zprávy (identifikace uživatele, autentizační informace apod.). Podrobnější informace o této problematice jsou uvedeny v kapitole 5,

kteřá se tývá bezpečnosti webových služeb. O to nejdůležitější se stará tělo zprávy, v němž se přenášejí informace identifikující volanou službu a předávané parametry, resp. návratové hodnoty služby. SOAP používá jmenné prostory pro identifikování jednotlivých částí XML zprávy. Obálka, hlavička a tělo zprávy patří do jmenného prostoru `http://schemas.xmlsoap.org/soap/envelope`.

Následuje ukázka velmi jednoduchého SOAP požadavku na zjištění posledního známého kurzu akcie s kódem MOT. Ukázková zpráva pro jednoduchost neobsahuje hlavičku, ale pouze tělo. V něm je požadavek na vyvolání vzdálené funkce `GetLastTradePrice` s parametrem `symbol` s hodnotou MOT (kód akcií firmy Motorola).

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="urn:x-example:services:StockQuote">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Dále uvedený příklad naznačuje, jak by mohla vypadat SOAP zpráva s výsledkem.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse
      xmlns:m="urn:x-example:services:StockQuote">
      <Price>14.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Kódování dat

Ve výše uvedených ukázkách SOAP zpráv je kromě jmenného prostoru `http://schemas.xmlsoap.org/soap/envelope` uveden ještě jeden prostor, kterým je `http://schemas.xmlsoap.org/soap/encoding`. Ten identifikuje způsob **kódování přenášených dat do XML**. Se SOAP protokolem je možné použít libovolný způsob serializace, standard protokolu jeden rovnou definuje. Standardní

serializace je schopna do XML převést graf obsahující otypované objekty. V praxi se nejčastěji používají běžné skalární datové typy (čísla, řetězce apod.), které definují XML schémata. Navíc kódování SOAP protokolu definuje způsob serializace složených datových objektů – seznamů skalárních hodnot (polí) a struktur. Lze serializovat i reference na objekty.

Obecně platí, že hodnoty se ukládají vždy jako obsah elementů, jedna hodnota do jednoho elementu. Datové typy mohou být definovány buď v externím XML schématu nebo přímo v XML zprávě. Druhý způsob je jednodušší a tedy i používanější. Např. osobní údaje by mohly být zakódovány do XML pro potřeby SOAP protokolu následujícím způsobem:

```
<jméno xsi:type="xsd:string">Petra Paletářová</jméno>
<email xsi:type="xsd:string">petra.paletarova@seznam.cz</email>
<věk xsi:type="xsd:int">22</věk>
```

Odpovídající samostatné XML schéma by pak vypadalo následovně:

```
<xsl:element name="jméno" type="xsd:string"/>
<xsl:element name="email" type="xsd:string"/>
<xsl:element name="věk" type="xsd:int"/>
```

Transportní mechanismy

Protože se dnes SOAP protokol typicky používá pro volání typu požadavek/odpověď (request/response), uplatňuje se nejčastěji pro přenos protokol HTTP. Důvodem je zejména jeho široká podpora v různých aplikacích. Navíc webovou službu lze nahrát přímo na běžný webový server, který jednotlivé požadavky předá odpovídající webové službě ke zpracování. Výhoda použití protokolu HTTP spočívá také v tom, že stávající síťová infrastruktura (zvláště ve firemní sféře) dovoluje téměř neomezenou komunikaci na portu vyhrazeném pro HTTP (TCP port 80).

SOAP požadavek (request) se zasílá v těle HTTP požadavku. K tomu účelu se používá metoda POST, která dovoluje posílat data v těle HTTP požadavku. Požadavek musí obsahovat HTTP hlavičku SOAPAction, která identifikuje SOAP požadavek. Tuto hlavičku mohou používat jednak firewally k

filtrování požadavků a jednak může obsahovat URI s identifikací služby, která se má vyvolat. Pokud je obsahem hlavičky prázdný řetězec, služba ke spuštění je identifikována přímo adresou, na kterou směřuje požadavek.

Následuje příklad SOAP požadavku zaslaného přes HTTP.

```
POST /StockQuote HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
SOAPAction: ""

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="urn:x-example:services:StockQuote">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Odpověď na požadavek nemá při přenosu pomocí HTTP žádné přidávané informace, obsah odpovědi musí být identifikován jako XML dokument. Konkrétní podoba odpovědi je uvedena níže.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse
      xmlns:m="urn:x-example:services:StockQuote">
      <Price>14.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WSDL specifikace

Jazyk **WSDL** slouží k popisu síťových služeb jako množiny koncových bodů zpracovávajících zprávy. Operace a zprávy jsou zpracovávány na abstraktní úrovni a teprve poté jsou svázány s konkrétním síťovým protokolem a datovým formátem. To umožňuje snadné vytvoření popisu rozhraní, které na-

bízí jednu službu několika způsoby. V praxi WSDL popisy nejčastěji definují služby, které si posílají zprávy pomocí formátu SOAP a protokolu HTTP.

WSDL jazyk vznikl jako společná iniciativa firem Microsoft a IBM, které si uvědomily potřebu sjednocení jazyka používaného pro popis rozhraní webových služeb. Navazuje tak na předchozí aktivity, zejména na jazyky NASSL (Network Accessable Service Specification Language), SCL (SOAP Contract Language) a SDL (Service Description Language).

WSDL soubor s definicí rozhraní služby je XML dokument. Skládá se zejména z následujících elementů, které tvoří základní části každého WSDL popisu.

Element **<types>** obsahuje definici datových struktur používaných ve zprávách. K definici lze použít teoreticky libovolný typový systém, ale nejčastěji se používají XML schémata. Nástroje webové služby se starají o mapování datových typů podle XML schémat na nativní datové typy použitého jazyka.

Element **<message>** definuje formát předávaných zpráv pomocí dříve definovaných datových typů. Zprávy fungují jako vstupní nebo výstupní struktury pro operace. Každá zpráva se může skládat z několika logických částí s vlastním datovým typem. Při použití SOAP protokolu pro vzdálené volání procedur (RPC) odpovídá jedna část zprávy jednomu parametru vzdálené metody.

Element **<operation>** tvoří abstraktní definice operací, které jsou službou podporovány. U operací se definují jejich vstupy a výstupy. Vstup a výstup je poslán již existující zprávou. V modelu vzdáleného volání procedur (RPC) při použití SOAP protokolu odpovídá jedna operace jedné metodě.

Element **<portType>** sdružuje dohromady několik operací.

Element **<binding>** slouží pro navázání určitého typu portu na konkrétní protokol a formát přenosu zpráv.

Element **<port>** určuje jeden koncový bod služby definovaný jako kombinace síťové adresy a dříve definované vazby.

Element **<service>** sdružuje několik koncových bodů (portů) do jedné služby.

Následuje ilustrativní příklad WSDL souboru, který definuje rozhraní výše popsané služby (v části věnující se SOAP protokolu) pro zjišťování aktuálního kurzu zadané akcie.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions name="StockQuote"
  targetNamespace="urn:x-example:services:StockQuote"
  xmlns:tns="urn:x-example:services:StockQuote"
  xmlns:xsd="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="GetLastTradePrice">
        <complexType>
          <all>
            <element name="symbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="GetLastTradePriceResponse">
        <complexType>
          <all>
            <element name="Price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd:GetLastTradePrice"/>
  </message>

  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd:GetLastTradePriceResponse"/>
  </message>

  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>

  <binding name="StockQuoteSoapBinding">
```

```

        type="tns:StockQuotePortType">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
        <soap:operation soapAction=""/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>

<service name="StockQuoteService">
    <documentation>Moje první služba</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
        <soap:address location="http://example.com/StockQuote"/>
    </port>
</service>

</definitions>

```

UDDI protokol

UDDI protokol nabízí mechanismy pro registrování, kategorizování a vyhledávání webových služeb. Cílem UDDI je poskytnout uživateli jak klasická data (především informace o poskytovateli služby), tak metadata (způsob použití služby). Funguje jako velký adresář, který obsahuje informace o subjektech (firmách) a jimi poskytovaných službách. Samotný registr pracuje rovněž jako webová služba a komunikace s ním probíhá na základě SOAP protokolu.

První specifikace UDDI pochází ze září roku 2000 a označuje se jako 1.0. Následovala verze 2.0 z března roku 2001 a 3.0 z prosince téhož roku. U zrodu projektu stály firmy IBM, Microsoft a Ariba.

UDDI pracuje na principu registrace webových služeb jejich poskytovateli. V současné době existuje implementace The UDDI Business Registry, kde producenti služeb registrují svou firmu (popis, druh služeb) a dále konkrétní nabízené webové služby (obvykle formou URL odkazů). Zákaznické podniky zde pak mohou tyto služby vyhledávat. Systém je provozován jako distribuovaná aplikace se společným rozhraním a průběžnou datovou synchronizací.

UDDI umožňuje:

- vyhledání webové služby podle definovaného rozhraní,
- vyhledání producenta podle zvolených klasifikačních znaků,
- zjištění podporovaných protokolů dané webové služby,
- vyhledání webové služby podle klíčových slov,
- uložení zjištěných informací a jejich online update za běhu.

K tomu slouží následující čtyři druhy entit:

- **podnikatelské entity (businessEntity):** popisují poskytovatele služeb. U každé firmy v registru jsou zaznamenány základní údaje (název, stručný popis) a konkrétní údaje. Každé firmě mohou být přiřazeny klasifikační identifikátory, které určují oblast jejího podnikání a geografickou polohu.
- **služby (businessService):** ke každé firmě jsou v registru uloženy seznamy služeb, které firma poskytuje. Každá služba je popsána a obsahuje seznam šablon vazeb, které ukazují na technické údaje nutné pro využití služby. Každá služba je podřízena jediné podnikatelské entitě.
- **šablony vazeb (bindingTemplate):** popisují, jak a kde je možné se službou komunikovat. Typicky je tato informace popsána odkazem na WSDL soubor s definicí rozhraní služby. Každá šablona odkazuje na typ služby, který implementuje.
- **technický model (tModel):** představuje abstraktní vlastnost (atribut, rozhraní) jiného UDDI objektu. Každá detailní specifikace, každý typ přenosu, protokol nebo jmenný prostor jsou dány jedním tModelem. Každá bindingTemplate pak obsahuje odkazy na relevantní tModely, které popisují jednotlivé vlastnosti dané webové služby a v souhrnu vlastně určují její typ. Webové služby jsou stejného typu, pokud obsahují odkazy na stejné

tModely. Z hlediska fyzické realizace je podstatné, že tModel obsahuje pouze odkazy na definiční dokumenty, nikoliv přímo definici daného rozhraní.

V současnosti existuje globální implementace UDDI registru The UDDI Business Registry, který poskytuje specifikované UDDI služby libovolnému uživateli zdarma.

Základní myšlenkou UDDI specifikace je nezávislost na struktuře vkládaných dat a metadat a na přenosových protokolech registrovaných webových služeb. Pro popis jsou podporovány standardy XML a WSDL, ale třeba i volný text. Komunikace s registrem probíhá pomocí HTTP a protokolu SOAP. Vnitřní datové struktury mají objektový charakter.

V oblasti webových služeb lze v nejbližší době očekávat prudký rozvoj. Specifikace UDDI však již nabyla poměrně ucelené podoby a její zásadní změny pravděpodobně nelze očekávat. Důvodem je fakt, že již existuje globální implementace této služby a má se dále rozšiřovat. Možnosti UDDI jsou však v každém případě obrovské.

5.3 Budoucnost webových služeb

Webové služby jsou hitem posledních let. Důkazem tohoto tvrzení ostatně může být fakt, že celá řada špičkových firem (Microsoft, IBM, Novell) vyvíjí nástroje pro jejich implementaci a v podstatě každý nový softwarový produkt tuto formu využití Internetu podporuje. V opačném případě by byl totiž při současných podmínkách považován za zastaralý. Přestože se podobných trendů či témat v posledních letech objevila celá řada, mnohé zanikly stejně rychle, jako se objevily na trhu, v případě webových služeb jsme se však doposud něčeho podobného nedočkali a ani se s největší pravděpodobností takového vývoje obávat nemusíme.

Potenciál webových služeb je obrovský. Dochází zde k maximálnímu využití toho, že jazyk HTML byl od samého počátku vývoje nezávislý na používané platformě. Proto byl vybrán jako nejvhodnější základ pro univerzální řešení, které by umožnilo naplnit filozofii webových služeb. Nejzajímavější a nejvyhledávanější budou v budoucnosti služby podporující tvorbu webových aplikací zajišťujících pomocí webu činnosti, které dnes zatím nejsou možné. Příkladem může být kalendářová služba umožňující sjednat si s kýmkoliv schůzku podle jeho kalendáře, který formou služby vystaví na web, a podle vlastního kalendáře, aniž by bylo nutné cokoli osobně vyřizovat.

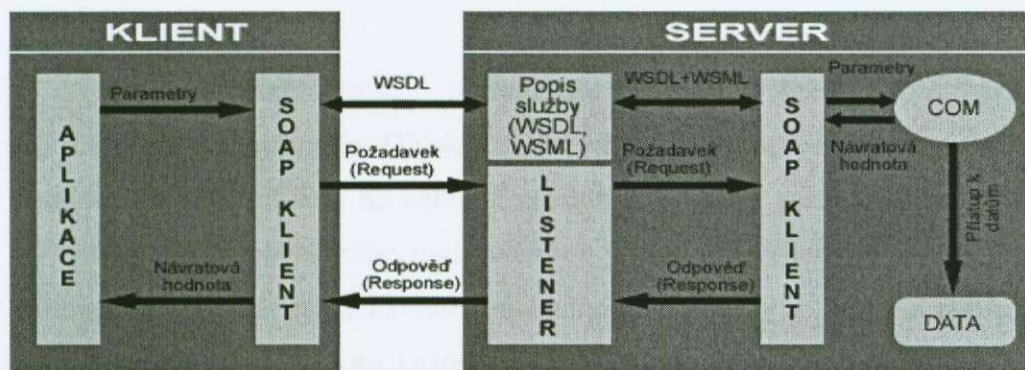
V oblasti geografických informačních systémů (GIS), geodézie a zeměměřičství se jako velmi zajímavé a perspektivní do budoucna jeví rozvoj webových služeb pro poskytování mapových dat v prostředí Internetu či intranetu, sloužících jako podklad pro webové aplikace vyžadující polohovou/prostorovou složku. Jejich hlavním přínosem je umožnění sdílení dat GIS v distribuovaném prostředí Internetu. Nevýhodou může být snad jen nutnost online připojení k mapovému serveru, jehož služby jsou využívány.

Funkčnost webových služeb není v současnosti dostatečná pro všechny myslitelné scénáře. Jako reakce na toto úskalí vznikla iniciativa GXA (Global XML Web Services Architecture), která do specifikace XML webových služeb přidává infrastrukturou často řešené rysy (bezpečnost, transakce atd.). Představuje jakýsi rámec pro rozšiřování funkčnosti těchto služeb.

6. Microsoft Visual FoxPro 9.0 a webové služby

6.1 Jak webová služba funguje

Funkčnost webové služby není závislá na tom, v jakém jazyce je klient nebo server implementován. Vlastní implementace však už na použitém jazyce závisí, proto se v dalších odstavcích zaměřím na problém webových služeb v prostředí MS VFP 9.0.



Obr. 6.11

Komunikaci klienta a serveru znázorňuje obrázek 6.11. Ten vychází z případu, kdy se použije **Microsoft SOAP Toolkit** na straně klienta i na straně serveru a kdy webovou službu poskytuje komponenta COM. SOAP je předpis, kterým se řídí vlastní komunikace mezi klientem a serverem. Komunikace probíhá přes protokol HTTP a je uskutečňována výměnou dat XML ve speciálním tvaru. O tuto záležitost se stará zprostředkovatel SOAP, např. Microsoft SOAP

Toolkit. Toolkit se skládá ze dvou základních částí: serveru SOAP a klienta SOAP. Každá tato část se na své straně sítě stará o zpracování dat SOAP.

Celý proces začíná požadavkem ze strany klientské aplikace. Ta kontaktuje klienta SOAP, předá mu parametry pro webovou službu a sdělí mu její URL adresu. Adresa služby je ve skutečnosti adresou souboru WSDL.

Klient SOAP si stáhne ze souboru WSDL informace o službě. Pomocí získaných dat ověří, zda daná služba opravdu obsahuje metodu, kterou aplikace požaduje a ověří datové typy získaných parametrů. Dále sestaví požadavek SOAP (SOAP request) a odešle jej v těle požadavku HTTP (HTTP request) na zjištěnou adresu listeneru, který přijímá požadavky pro webovou službu. Listener může být dvojího typu. Jedná se buď o server ISAPI nebo stránku ASP.

Server SOAP z přijatého požadavku zjistí jméno metody komponenty COM, kterou má zavolat a jaké parametry jí má předat. Nakonec ze souboru WSDL načte poslední informace potřebné k vytvoření objektu komponenty COM.

Poté dochází k volání požadované metody objektu COM spolu s požadovanými parametry. Objekt COM požadavek zpracuje a vrátí výsledek zpět serveru SOAP. Ten na základě návratové hodnoty sestaví odpověď ve formátu XML, odpověď vrátí listeneru a ten ji vrátí zpět klientovi SOAP. Ten nakonec z odpovědi zjistí návratovou hodnotu a vrátí ji aplikaci. Z pohledu uživatele je objekt COM při každém požadavku vytvořen a znovu zrušen. Ve skutečnosti se však objekt COM z důvodu lepší výkonnosti ponechává v paměti i pro případná další volání.

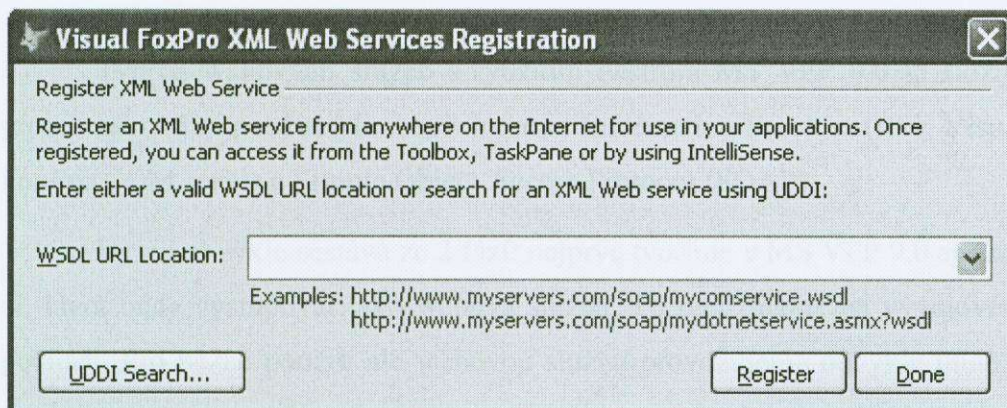
6.2 MS VFP 9.0 v úloze konzumenta webové služby

Základním předpokladem, aby aplikace MS VFP 9.0 mohla webovou službu použít, je, že je nainstalován Microsoft SOAP Toolkit, nebo alespoň

knihovna DLL. Ta obsahuje klienta SOAP. Toolkit se standardně dodává na CD s MS VFP 9.0 nebo je možné jej stáhnout ze stránek společnosti Microsoft.

Pro použití konkrétní webové služby je možné (pokud neznáme konkrétní adresu na webovou službu) použít protokol UDDI, který umožňuje rychlé vyhledávání v seznamu všech dostupných webových služeb. Alternativou k UDDI je hledání například ve specializovaném registru.

Pokud známe konkrétní adresu dané webové služby (například <http://www.foxcentral.net/foxcentral.wsdl>), můžeme nakonfigurovat v MS VFP 9.0 IntelliSense tak, aby se správně nainstaloval klient SOAP. Z nabídky Tools spustíme IntelliSense Manager. Přejdeme na záložku Types a stiskem tlačítka Web Services otevřeme dialog (Obr. 6.21), ve kterém se přímo zadává adresa k WSDL souboru. V tomto kroku je nutné být připojen k Internetu a jako aktivní stránku mít zobrazenou tu se souborem WSDL.



Obr. 6.21

Tlačítko Register vygeneruje skript pro IntelliSense. Tlačítkem Done IntelliSense Manager opustíme a skript využijeme v nově založeném programu. V editoru programu napíšeme WS a klávesou Enter nebo mezerníkem přimějeme IntelliSense k zobrazení okna pro výběr požadované služby. Zvolíme službu a potvrdíme. IntelliSense vygeneruje kód pro konfiguraci klienta SOAP. V programu můžeme nahradit řádek „* Call your XML Web service here...” například voláním

```
? loWSFoxCentral.GetItems({^2004-12-10},0,0,"ALL")
```

Získaná data ve formátu XML lze převést do kurzoru DBF použitím funkce `XMLTOCURSOR()`.

Kromě použití IntelliSense Manageru můžeme klienta SOAP vyvolat přímo z MS VFP 9.0 příkazy:

```
oSOAP = CREATEOBJECT("MSSOAP.SoapClient")
oSOAP.MSSOAPInit(http://www.foxcentral.net/foxcentral.wsdl)
? oSOAP.GetItems({^2004-12-10},0,0,"ALL")
```

V tomto případě musíme obsluhu (názvy metod a způsoby volání) zvládnout sami.

6.3 Tvorba webových služeb v MS VFP 9.0

Úvod

Tvorba webových služeb s využitím nástroje MS VFP 9.0 je složitý proces zahrnující použití Microsoft Internet Information Services (IIS), Visual FoxPro COM server a Simple Object Access Protocol (SOAP).

Vývoj obvykle sestává ze 2 fází: nejprve tvoříme v MS VFP 9.0 aplikaci, která bude vystupovat jako webová služba. To realizujeme na vývojovém počítači. Pro běžné použití ale webovou službu provozujeme obvykle jinde – na webovém serveru. Na vývojovém počítači je nutné mít nainstalován Microsoft SOAP Toolkit, který umožňuje automatizované generování souborů pro webovou službu, a MS VFP 9.0. Vývojový počítač rozšíříme o funkci webového serveru nejjednodušeji tak, že na něm rozchodíme Microsoft Internet Information Server. Pro samotnou implementaci webové služby na produkčním webovém serveru potřebujeme na server jednak nainstalovat foxovskou aplikaci, která bude vystupovat jako webová služba, a navíc je potřeba aplikaci zaregistrovat, protože ji dodáváme ve formě DLL. Dále potřebujeme zajistit, aby IIS

rozpoznal naši aplikaci jako webovou službu. Veškeré tyto postupy jsou rozebrány dále.

Vytvoření projektu

Projekt v MS VFP 9.0 zahrnuje všechny komponenty, které se mohou umístit na server. Pokud chceme sestavit webovou službu, bez projektu se neobejdeme.

Vytvoření COM serveru

Protože webová služba je soubor s příponou .dll nebo .exe, je nutné použít COM server jako základ. COM server je v MF VFP 9.0 soubor s příponou .dll nebo .exe zahrnující třídu sestavenou jako OLEPUBLIC. Objekty specifikované jako OLEPUBLIC jsou přístupné automatizovanými klienty.

Součástí praktické části této diplomové práce je jednoduchá webová služba přístupující k tabulce Employees databáze Northwind na SQL Serveru. OLEPUBLIC třída má tuto podobu:

```
DEFINE CLASS Zamestnanci AS CUSTOM OLEPUBLIC
FUNCTION VratZamestnance() AS STRING
  LOCAL spojeni AS INTEGER, retezecSpojeni AS STRING
  spojeni = 0
  retezecSpojeni = „DRIVER=SQL Server; SERVER=PETRA-958X9X6SW;
  UID=paletp00; PWD=tp001635; APP=Microsoft Visual FoxPro;
  WSID=PETRA-958X9X6SW; DATABASE=Northwind“
  STORE SQLSTRINGCONNECT(retezecSpojeni) TO spojeni
  IF spojeni>0 THEN
    IF SQLEXPRESS(spojени,“SELECT * FROM Employees WITH(nolock)“,
      „kurzor“) > 0 THEN
      CURSORTOXML(„kurzor“,„xml“,1,2,0,„1“,„“,„“)
    ENDIF
    SQLDISCONNECT(spojени)
  ENDIF
  RETURN xml
ENDFUNC
ENDDFINE
```

Definice třídy je součástí programu zamestnanci.prg zahrnutého v projektu zamestnanci.pjx. Funkčnost programu je vhodné před sestavením komponenty COM otestovat například následujícím způsobem:

```
objekt = NEWOBJECT(„Zamestnanci“, „zamestnanci.prg“)
? objekt.VratZamestnance()
```

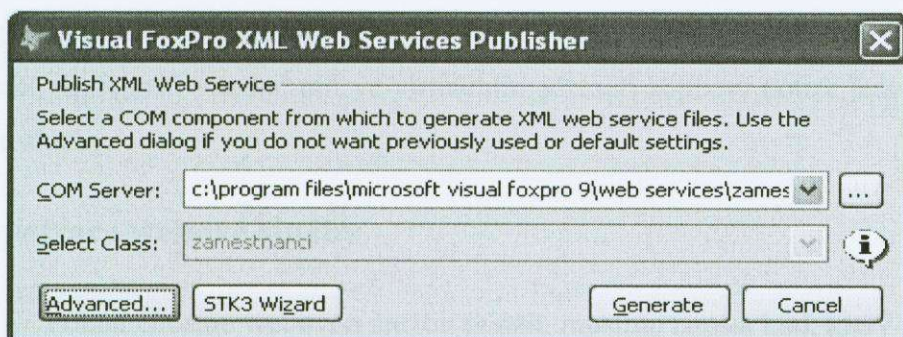
Dalším krokem je sestavení komponenty COM, kterou MS VFP 9.0 při vytváření automaticky zaregistruje do Windows. Použijeme např. příkaz

```
BUILD MTDLL "zamestnanci" FROM "zamestnanci.pjx"
```

nebo v Project Manageru označíme program zamestnanci.prg a stiskneme tlačítko Build... V dialogovém okně vybereme volbu Multi-threaded COM server (dll) a potvrdíme.

Pokud chceme program zamestnanci.prg upravit, musí se změny promítnout i do komponenty COM, proto musíme komponentu znovu sestavit. Pokud dojde k chybovému hlášení, je zpravidla nutné restartovat službu IIS.

Dalším krokem je přetavení komponenty COM do webové služby. Nej-
snazší je použití průvodce. Z nabídky Tools vybereme položku Wizards – Web Services. Správné dialogové okno v MS VFP 9.0 je patrné z obrázku 6.31.

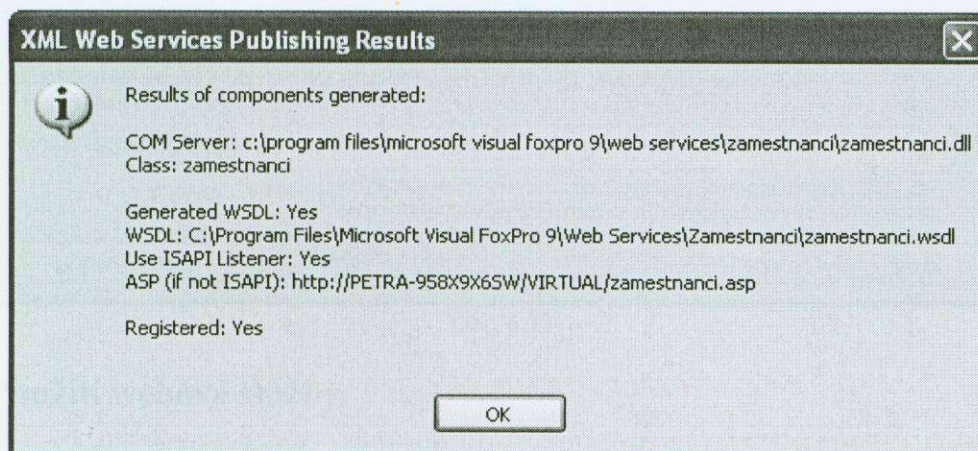


Obr. 6.31

Přes rozbalovací seznam „COM Server“ dohledáme přes trojtečku vpravo komponentu COM – zamestnanci.dll.

Pro správnou funkčnost webové služby je nutné nakonfigurovat virtuální adresář IIS serveru. Ačkoliv jde o adresář virtuální, musí k němu existovat i adresář skutečný (pro jednoduchost adresář, ve kterém je uložený projekt zamestnanci.pjx). Oba adresáře nadefinujeme stiskem tlačítka Advanced. Zde vyplníme potřebné údaje a opustíme výše uvedený dialog tlačítkem Generate.

O vygenerovaných souborech se dozvíme v závěrečném oznamovacím dialogu (Obr. 6.32).

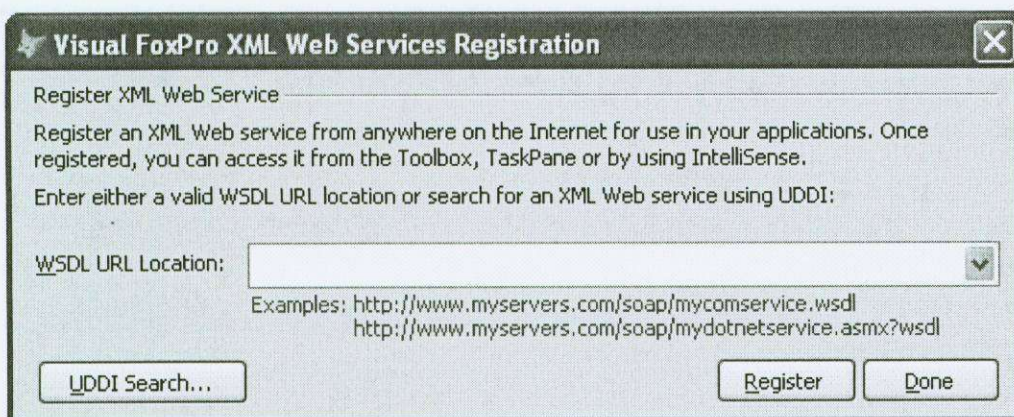


Obr. 6.32

Nutnou podmínkou úspěšnosti publikace webové služby je položka Registered v dialogovém okně, která musí mít hodnotu Yes. V opačném případě nám položka Registered oznamuje zhruba toto: „There was a problem accessing the Listener URI specified for the WSDL file“. V takovém případě je z mé vlastní zkušenosti vhodné mít ve virtuálním adresáři soubory pouze jedné webové služby.

Registrace webové služby

Pokud chceme webovou službu použít, musíme napsat kód, který vyvolá klienta SOAP. V MS VFP 9.0 je to zjednodušeno zaregistrováním webové služby přes IntelliSense Manager. Ten vyvoláme výběrem položky IntelliSense Manager v nabídce Tools. Na záložce Types vybereme tlačítko Web Services. V dialogovém okně (viz Obr. 6.33) vybereme přes rozbalovací seznam odpovídající WSDL soubor. Stiskem tlačítka Register poskytne IntelliSense danou webovou službu jako možnou volbu, kdykoliv v kódu použijeme proměnou WS a stiskneme například mezerník.



Obr. 6.33

Použití webové služby

Webová služba v MS VFP 9.0 se používá stejně jako jakýkoliv COM server k poskytnutí funkčnosti jedné aplikace jinému programu nebo aplikaci. Například lze vytvořit program, který webovou službu zavolá:

```
LOCAL x AS Zamestnanci
x = CREATEOBJECT („MSSOAP.SoapClient“)
x.MSSOAPInit („http://localhost/VIRTUAL/zamestnanci.wsdl“)
? x.vratZamestnance()
```

Jinou možnost představuje již výše zmíněné použití proměnné WS, kdy dojde k vygenerování odpovídající kódu přes IntelliSense Manager.

7. Bezpečnost webových služeb

7.1 Úvod

Praktické nasazení webových služeb na veřejném Internetu zaostává za představami tvůrců. Klíčovou oblast rozhodující o masovém použití webových služeb představují jejich bezpečnostní standardy. Reálné aplikace webových služeb musejí pracovat s ověřitelnou identitou uživatele, s ověřitelnými elektronickými podpisy a s daty zabezpečenými proti nežádoucí manipulaci. Těžko si lze představit použití webových služeb při obchodních transakcích typu bankovního příkazu nebo nákupu akcií, aniž by byly použity náležité bezpečnostní mechanismy.

Není-li zajištěna bezpečnost webových služeb, hrozí následující základní bezpečnostní rizika spadající do klasických oblastí zabezpečení:

- **Neautorizovaná transakce:** např. klient banky posílá zprávu SOAP požadující vyzvednutí peněz a transakce je neautorizovaná.
- **Zpráva SOAP je veřejně čitelná (bez kryptování):** zpráva je tím pádem k dispozici narušiteli, může jít např. o číslo bankovního účtu a jeho stav. Řešením je konvertování textu do nečitelné podoby zašifrováním.
- **Zpráva SOAP je modifikovatelná (bez integrity):** v tomto případě hrozí, že narušitel zprávu nelegálně modifikuje, např. změní číslo účtu.

V dalším textu se zaměřím na základní tři úrovně nasazení bezpečnosti webových služeb a konkrétně bezpečnost na úrovni hlavičky SOAP protokolu doplním podrobnějšími informacemi o specifikaci WS-Security, která jistě v budoucnu zaujme na poli bezpečnosti webových služeb své zásadní místo.

7.2 Různé úrovně nasazení bezpečnosti

Existují tři úrovně nasazení bezpečnosti webových služeb:

- na úrovni HTTP protokolu,
- na úrovni hlavičky SOAP protokolu,
- na úrovni těla SOAP protokolu.

Níže je uvedený příklad volání jednoduché webové služby s názorným odlišením výše uvedených úrovní nasazení.

<pre>POST /salesreport/SalesReportService.asmx HTTP 1.1 Host:expo-star Content-Type: text/xml; charset=utf-8 Content-Length: length SOAPAction: „xmlnet/cs/0735618011/GetSalesReport“</pre>	HTTP protokol
<pre><?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"></pre>	
<pre><soap:Header>... ... </soap:Header></pre>	Hlavička SOAP protokolu
<pre><soap:Body> <GetSalesReport xmlns="xmlnet/cs/0735618011"> <theYear>int</theYear> </GetSalesReport> </soap:Body></pre>	Tělo SOAP protokolu
<pre></soap:Envelope></pre>	

Bezpečnost na úrovni HTTP protokolu

Bezpečnost webových služeb na úrovni HTTP protokolu je zajišťována pouze po dobu vlastní komunikace (tzv. Point-to-Point zabezpečení). Jedná se o pragmatické řešení, ve spojení s webovými službami má však své nevýhody.

Z bezpečnostních prostředků, které mohou být použity na úrovni HTTP protokolu, lze zmínit restrikcí IP adres a autentizaci.

V případě **restrikce IP adres** se jedná o velice slabý způsob autorizace a autentizace. Nicméně v kombinaci s jinými postupy (Firewall, Routing and Remote Access Service) lze toto řešení označit za velmi účinné.

Další možností je využití **autentizace**. Existuje mnoho druhů autentizace, pro názornost zmíním např. Windows Integrated autentizaci (bezpečnost, šifrovanost, neopakovatelnost, funguje pouze u Internet Exploreru, problémy při přechodu přes Firewallly – používá jiné porty než 80 v závislosti na použité autentizaci), Basic/Clear text autentizaci (podpora ve všech prohlížečích, v heterogenních systémech, katastrofální bezpečnost – heslo pouze kódované, snadno odposlechnutelné, dobrá volba pro internetové scénáře v kombinaci se SSL), Digest autentizaci (nedostatečná podpora v HTTP klientech a serverech) a serverový nebo klientský certifikát SSL (šifrování veškeré komunikace mezi klientem a serverem).

Bezpečnost na úrovni hlavičky SOAP protokolu

Jedná se o zabezpečení typu End-to-End. Nevýhodou je, že takového zabezpečení není standardizováno. Je tedy nutná implementace na různých klientech, což je sice možné, nicméně jedná se o velmi pracný a neinteroperabilní přístup. Navíc je nutná odborná expertiza v případě šifrování dat nebo digitálního podpisu.

Nicméně v budoucnosti se jako s možným východiskem počítá se specifikací WS-Security.

Specifikace WS-Security vznikla ve spolupráci se společností Microsoft, IBM a VeriSign a používá standardizovaný způsob rozšíření protokolu SOAP pomocí tzv. SOAP extensions. Specifikace definuje rámec, na jehož základě mohou webové služby využívat existující bezpečnostní modely a jejich implementace – šifrování, digitální podpisy, bezpečnostní tokeny, důvěryhodné domény a další. WS-Security tedy není definicí nových bezpečnostních modelů a šifrovacích algoritmů, ale použitím stávajících postupů v kontextu webových služeb a protokolu SOAP.

WS-Security definuje bezpečnost v kontextu jedné zprávy, jednoho konkrétního volání webové služby nebo jednoho vrácení výsledku webové služby. Nezabývá se bezpečnostním kontextem posloupnosti zpráv. Bezpečnostní protokol pro posloupnost zpráv je samozřejmě možné definovat s využitím WS-Security jako základního prvku a právě takový postup využívají další připravované protokoly pro webové služby, které jsou uvedeny na konci této podkapitoly.

Korektní implementace WS-Security zabezpečí uživatelům následující důležité rysy bezpečné komunikace:

- **Utajení dat před třetí osobou (důvěryhodnost):** důvěryhodnost chrání obsah zprávy před čtením neautorizovaným příjemcem. Povoleno je kryptování těla, hlavičky a přílohy. Ke kryptování je potřeba klíč a algoritmus.
- **Odhalení manipulace s daty třetí osobou (integrita):** integrita chrání zprávu před neoprávněnou modifikací. Z klíče, který má k dispozici pouze odesílatel, vytváří za použití určeného algoritmu digitální podpis XML na základě obsahu části zprávy. Pokud dojde ke změně zprávy, není již kompatibilní s digitálním

podpisem XML. Příjemce rovněž vytvoří z obdržené zprávy podpis. Jsou-li podpisy různé, je odesílateli zasláno hlášení o chybě, aniž by došlo k zpracování zprávy příjemcem.

- **Předání bezpečnostních informací odesílatele (bezpečnostní token):** v zabezpečené zprávě je uvedena kombinace uživatelské jméno/heslo (username/password), tzv. základní autentizace. Webová služba je zpracována pouze v případě, že výše uvedená kombinace je platná. Alternativou je např. použití digitálního podpisu.

XML elementy používané v návrhu WS-Security

Protokol SOAP je definován s využitím jazyka XML, bezpečnostní rozšíření navrhovaná v rámci WS-Security se nacházejí ve jmenném prostoru <http://schemas.xmlsoap.org/ws/2002/04/secext/>. Návrh standardu WS-Security dále využívá pro šifrování částí XML dokumentu standard XML Encryption definovaný ve jmenném prostoru <http://www.w3.org/TR/xmlenc-core/> a pro podpis částí XML dokumentu standard XML Signature definovaný ve jmenném prostoru <http://www.w3.org/TR/xmldsig-core/>.

Nositelem bezpečnostních dat je XML element **<Security>**, který se nachází v hlavičce protokolu SOAP. V rámci tohoto elementu jsou definovány důležité prvky (XML elementy) bezpečné komunikace:

- **<UsernameToken>**, **<BinarySecurityToken>** nebo **<SecurityTokenReference>**,
- **<EncryptedData>**,
- **<Signature>**.

XML element **<UsernameToken>** popisuje entitu, která odeslala zprávu protokolu SOAP. V nejjednodušším případě je obsahem tohoto elementu uživatelské jméno. V takovém případě se předpokládá, že příjemce už něja-

kým způsobem ověřil totožnost volajícího a uvedenému uživatelskému jménu důvěřuje. Obsahem elementu může být i uživatelské jméno a heslo, v takovém případě může příjemce ověřit platnost uživatelského jména a hesla. WS-Security nedefinuje šifrování hesla. [12]

Zajímavější a pro WS-Security v mnohém podstatnější je XML element **<BinarySecurityToken>**. Obsahem tohoto elementu jsou libovolná binární data, která pomocí zvoleného autentizačního mechanismu definují identitu uživatele. [12]

XML element **<SecurityTokenReference>** popisuje případ, kdy binární reprezentace identifikačních dat není obsažena v každé zprávě. Tento element tedy namísto identifikačních dat obsahuje odkaz na jejich URL.

Definice WS-Security využívá existujících standardů, příkladem je element **<EncryptedData>**, převzatý ze standardu XML Encryption. XML element **<EncryptedData>** obsahuje podle očekávání zašifrovaná data. Parametr *Id* definuje šifrovanou část XML dokumentu, SOAP zprávy.

Standardy WS-Security a XML Encryption nedefinují, jaký algoritmus a šifrovací klíč byl pro šifrování dat použit. Předpokládá se, že odesílatel a příjemce buď mají sdílený tajný klíč (symetrické šifrování) nebo šifrovaný dočasný klíč (session klíč) je obsažen ve vnořeném elementu **<EncryptedKey>**. V takovém případě je XML elementem **<EncryptionMethod>** definován použitý algoritmus a XML elementem **<KeyInfo>** je definován veřejný klíč, který byl k šifrování dočasného klíče použit. Příjemce má typicky k dispozici privátní klíč, pomocí kterého získá dočasný klíč a zašifruje data zprávy. V další komunikaci už mohou odesílatel a příjemce používat dočasný klíč, který už nadále nemusí být součástí zprávy ani v šifrované podobě.

Digitální podpis části SOAP zprávy je definován XML elementem **<Signature>**, který je převzatý ze standardu XML Signature. Podobně jako v případě elementu **<EncryptedData>** nepředepisuje WS-Security použití kon-

krétních šifrovacích algoritmů. Vnořený element **<SignatureMethod>** odkazuje na použitý algoritmus elektronického podpisu, vnořený element **<KeyInfo>** definuje stejně jako ve výše uvedeném případě použitou dvojici veřejného a soukromého klíče. Samotný podpis SOAP zprávy nebo její části definované parametrem **Id** je obsažen ve vnořeném elementu **<Signature-Value>**. [12]

Rozšíření WS-Security

Norma WS-Security definuje bezpečnost a integritu jedné zprávy protokolu SOAP. Implementace bezpečné komunikace protokolem SOAP mezi subjekty s různým stupněm zabezpečení a různými oprávněními vyžaduje víc než zabezpečení jedné zprávy, navíc WS-Security řeší jen část bezpečnostních služeb (nepracuje např. s auditem, garancí neodmítnutelnosti apod.). Vznikají tedy další standardy pro bezpečnost webových služeb používající WS-Security jako svůj základ, mimo jiné:

- **WS-Policy:** definice požadavků, které webová služba klade na volajícího (zda může webovou službu volat anonymně a pokud ne, jaký autentizační mechanismus má použít). [12]
- **WS-Trust:** delegace a přenos důvěry mezi násobnými účastníky komunikace protokolem SOAP. [12]
- **WS-Federation:** správa identity uživatele a jednotné přihlášení k webovým službám. [12]

Bezpečnost na úrovni těla SOAP protokolu

Tento typ bezpečnosti je opět označován jako End-to-End. Jeho implementace je pro vývojáře velmi jednoduchá, lze ho však použít pouze pro nenáročné scénáře. Nevýhody jsou v tomto případě stejné jako u hlavičky SOAP protokolu, tzn. nutnost implementace na různých klientech a nutnost odborné expertizy.

Závěr

Na úvodních stránkách jsou v anotaci uvedeny dva základní cíle, které by tato diplomová práce měla splnit. Konkrétně se jedná o:

- zmapování úlohy a významu jazyka XML a webových služeb založených na tomto jazyku ve světě internetových technologií a
- podání přehledu současných možností jazyka Microsoft Visual FoxPro 9.0 v oblasti použití a implementace webových služeb.

Práci jsem tedy celkově pojala jako stručný průvodce jazykem XML a webovými službami, které jsou na XML založené s důrazem na použití těchto technologií v databázových systémech a především v jazyce MS VFP 9.0. Jedná se o velmi rozsáhlou problematiku, snažila jsem se však informace co nejvíce zjednodušit a dosáhnout tak jejich stravitelnosti i pro vesměs nezkušeného a s danou problematikou jen zběžně seznámeného čtenáře. Na druhou stranu bylo mou snahou uvést veškeré důležité aspekty, které s tématem souvisí.

Oba dané cíle jsem tedy podle mého úsudku v rámci svých možností a omezeného rozsahu diplomové práce víceméně splnila a nechávám na vedoucím, oponentuře a dalších znalcích, do jaké míry se mi to podařilo.

V této závěrečné kapitole bych ráda ještě dodatečně a zcela stručně zhodnotila efektivnost a možnosti práce s XML daty a implementace webových služeb v MS VFP 9.0. Jako srovnávací element jsme po odborné poradě s panem Ing. Václavem Novákem, CSc. zvolili jazyk C# v prostředí nástroje .NET Framework (Microsoft Visual Studio .NET 2003).

Zásadní rozdíl mezi zvolenými nástroji tkví ve způsobu podpory XML dat. Ve třetí kapitole je uvedeno, že MS VFP 9.0 zpřístupňuje práci s XML daty jako své rozšíření, mapuje XML dokumenty na relační nebo objektové datové struktury. Funkce a možnosti tohoto nástroje jsou proto značně omezené. MS VFP 9.0 neinterpretuje totiž na rozdíl od .NET Framework XML jako nativní datový typ. Nejsou zde tedy dost dobře nebo vůbec zakomponovány určité operace s XML daty jako je např. serializace XML (to lze provádět např. uložením stavu objektu do textového souboru a následným načtením do XML a naopak) a jiné.

Na druhé straně prostředí Microsoft .NET Framework poskytuje komplexní sadu tříd pro práci s dokumenty XML a příbuznými technologiemi na různých úrovních a v přísném souladu s nejnovějšími standardy a doporučeními organizace W3C. Podpora XML v .NET Framework zahrnuje XML 1.0, jmenné prostory XML, model DOM Level 2 Core, jazyk XSD, transformace XSLT a výrazy XPath. Základní třídy XML jsou navíc úzce integrovány s dalšími klíčovými součástmi prostředí .NET Framework, včetně přístupu k datům, serializaci a konfiguraci aplikací.

MS VFP 9.0 spolupracuje s .NET Framework do určité míry na základě podpory tříd XMLAdapter, XMLTable a XMLField (více v kapitole 4). Tyto třídy řeší interakci s objekty ADO.NET DataSets z prostředí .NET Framework.

Co se týče webových služeb, zdá se mi jejich podpora v obou nástrojích vcelku podobná. Lze možná poukázat na drobné rozdíly, ty jsou ale z mého globálního pohledu na problematiku vcelku bezpředmětné.

MS VFP 9.0 tedy stále zůstává především databázovým nástrojem pro práci s různorodými formáty dat a podpora XML je jeho velice užitečným a už teď do značné míry propracovaným rozšířením, které se určitě bude v budoucnu stále více a více zdokonalovat.

Zdroje

- [1] Buranský, Imrich: **XML a webové služby.**
Prienik do XML cez Microsoft .NET a Murphyho zákony.
- [2] Computer Press: **Microsoft Visual FoxPro 6.0.**
První vydání, Praha, Computer Press, 1999. ISBN 80-7226-149-5.
- [3] Esposito, Dino: **XML. Efektivní programování pro .NET.**
První vydání, Praha, GRADA Publishing, 2003. ISBN 80-247-0775-6.
- [4] Kirsten Wolfgang; Ihringer, Michael; Kühn, Mathias; Röhrig Bernhard:
Caché. Databáze postrelačního typu a tvorba aplikací.
První vydání, Brno, CP Books, a. s., 2005. ISBN 80-251-0491-5.
- [5] Kocan, Marek: **Databáze a XML? A proč ne!**
URL: <http://dbsvet.cz/view.php?cisloclanku=2004030901>
- [6] Kosek, Jiří: **XML pro každého. Podrobný průvodce.**
První vydání, Praha, GRADA Publishing, 2000. ISBN 80-7169-860-1.
- [7] Lacko, Ľuboslav: **Vývoj aplikácií pre mobilné zariadenia.**
Praha, Microsoft, s. r. o.
- [8] **Microsoft Visual FoxPro 9.0 Help**
- [9] Peterka, Jiří: **Ovládnou web služby?**
URL: <http://www.lupa.cz/clanky/ovladnou-web-sluzby/>

- [10] Schmuller, Joseph: **Myslíme v jazyku UML.**
První vydání, Praha, GRADA Publishing, 2001. ISBN 80-247-0029-8.
- [11] Sláma, Martin: **Automatická bezpečnost.**
Connect!, září 2004
- [12] Stoklasa, Jan: **Budoucnost XML webových služeb – bezpečnost.**
<http://objekty.pef.czu.cz/2003/sbornik/Stoklasa2003.pdf>
- [13] Talich, Milan: **Webové služby a aplikace XML.**
INFORUM 2004: 10. konference o profesionálních informačních zdrojích, Praha, 25. – 27. 5. 2004
URL: <http://www.inforum.cz/inforum2004/pdf/Talich-Milan.pdf>
- [14] **UDDI aneb jak sehnat webovou službu?**
URL: <http://nb.vse.cz/~zelenyj/it380/eseje/xtomm19/uddi.htm>
- [15] Vít, Igor: **Visual FoxPro a webové služby I.**
URL: <http://dbsvet.cz/view.php?cisloclanku=2005010501>
- [16] Vít, Igor: **Visual FoxPro a webové služby II.**
URL: <http://dbsvet.cz/view.php?cisloclanku=2005011201>
- [17] Vít, Igor: **Visual FoxPro a webové služby III.**
URL: <http://dbsvet.cz/view.php?cisloclanku=2005011901>
- [18] Vít, Igor: **Visual FoxPro a webové služby IV.**
URL: <http://dbsvet.cz/view.php?cisloclanku=2005012601>
- [19] **XML:DB iniciativa** (diskusní archiv)
<http://archive.xmldb.org/xmldb/2001/Jul/msg00066.html>
- [20] Žižka, Petra: **Nativní XML databáze – nástin teorie.**
URL: <http://interval.cz/clanky/nativni-xml-database-nastin-terorie>

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
Akademický rok 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petra Paletářová**
Studijní program: **B1802 Aplikovaná informatika**
Studijní obor: **Výpočetní technika**

Název tématu: **Jazyk XML v prostředí MS Visual FoxPro 9.0**
XML in MS Visual FoxPro 9.0

Zásady pro vypracování:

V nových verzích jazyka Visual FoxPro (VFP) se stále více prosazují moderní trendy aplikací databázových technologií a především propojení databází na webové služby. Nejvýznamnějšími technologiemi poslední doby jsou webové služby (WSDL) a formát XML.

Úkolem řešitele je podat přehled současných možností jazyka VFP v oblasti internetových technologií a aplikovat je při řešení konkrétního databázového systému.

Rozsah práce: **60 stran**
Rozsah příloh:
Forma zpracování: **tištěná**
Seznam odborné literatury:
Pokorný, J.: Dotazovací jazyky, SCIENCE, 1994, ISBN 80-901475-2-6
<http://www.w3.org/TR/wsdl>

Vedoucí bakalářské práce: **Mgr. Miloš Prokýšek**
Katedra informatiky

Datum zadání bakalářské práce: **26. dubna 2005**
Termín odevzdání bakalářské práce: **28. dubna 2006**

SEZNAM OBRÁZKŮ

Obrázek 5.11 Třívrstvá architektura

Obrázek 5.12 Komunikace s webovou službou

Obrázek 5.21 Vztahy mezi SOAP protokolem, WSDL formátem a UDDI registrem

Obrázek 6.11 Komunikace klienta a serveru

Obrázek 6.21 Registrace existující webové služby v MS VFP 9.0

Obrázek 6.31 Přetavení komponenty COM do webové služby

Obrázek 6.32 Úspěšná publikace webové služby v MS VFP 9.0

Obrázek 6.33 Registrace webové služby v MS VFP 9.0