Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Webové služby na zvolené platformě

Vít Hlava

Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Akademický rok 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Vít Hlava

obor Informatika

Vedoucí katedry Vám ve smyslu Studijního a zkušebního řádu ČZU v Praze čl. 16 určuje tuto bakalářskou práci.

Název práce: **Webové služby na zvolené platformě**

Osnova bakalářské práce:

1. Úvod
2. Cíl práce a metodika
3. Přehled řešené problematiky
4. Vlastní práce
5. Závěr
6. Seznam použitých zdrojů
7. Přílohy

Rozsah hlavní textové části: 30 - 40 stran

Doporučené zdroje:

1. Ethan, C. Web Services Essentials. 1st ed. O`Reilly, 2002. ISBN 0-596-00224-6
2. Weerawarana, C., Cerbera F., et al. Web Services Platform Architecture. Prentice Hall PTR. 2005.
ISBN 0-13-148874-0
3. Žák, M. XML (začínáme programovat). Grada. 2003. ISBN 80-247-0565-6
4. Benz, B. Oliver, R. Lotus Notes a Domino mistrovství v programování. CP Books. 2005.
ISBN 80-215-0750-7
5. W3C. URL: http://www.w3.org

Vedoucí bakalářské práce: **Ing. Miloš Ulman**

Termín odevzdání bakalářské práce: duben 2011

.....
Vedoucí katedry

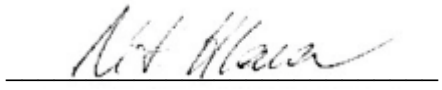
.....
Děkan

V Praze dne: 19. 2. 2010

Evidováno děkanátem PEF pod č.j.: KIT-010-10B

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Webové služby na zvolené platformě" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 21. 3. 2011 

Poděkování

Rád bych touto cestou poděkoval Ing. Miloši Ulmanovi, Ph.D. za vedení mé bakalářské práce, dále Denise Zetkové za jazykové úpravy, Františkovi Bozděchovi za poskytnutí ukázky Webových služeb v .Net frameworku.

Webové služby na zvolené platformě

Web services on the selected platform

Souhrn

Tato práce se zabývá možnostmi použití technologie *webových služeb* na platformě Lotus Notes/Domino a ukazuje konkrétní řešení realizované pomocí webových služeb.

V teoretické části je vymezen pojem webové služby a jsou shrnuty základní standardy, na kterých jsou webové služby postaveny. Všeobecně uznávané standardy umožňují nezávislost systému, který webovou službu poskytuje (resp. systému, který službu konzumuje) na programovacím jazyce, v němž je naprogramován a na operačním systému, v němž je spuštěn. V následující části jsou popsány charakteristika platformy Lotus Notes a Domino firmy IBM a možnosti implementace webové služby v této platformě.

V poslední části je uvedeno praktické využití webové služby v Lotus Notes pro validace poštovních adres, vyhledávání ulic a měst na podkladě databáze UIR-ADR.

Summary

This work deals with the usage of Web Services technology on Lotus Notes and Domino platform. It describes a concrete application based on this technology.

The theoretical part provides definition of the term *web service* and introduces the main standards, that the web service is built for. Standardization ensures the independence on programming language and operating system of system providing and consuming the service. Following part characterizes the platform Lotus Notes and Domino.

The last part is dedicated to realization of application for looking up and validation of postal address based on UIR-ADR database of addresses in Czech Republic.

Klíčová slova: Webová služba, SOA, XML, SOAP, WSDL, poskytovatel (server), konzument (klient), Lotus Notes, Lotus Domino, UIR-ADR registr adres ČR

Keywords: Web services, SOA, XML, SOAP, WSDL, provider (server), consumer (client), Lotus Notes, Lotus Domino, UIR-ADR registry

Obsah

1. Úvod

Jedním z konceptů při návrhu a integraci moderních informačních systémů je architektura zaměřená na služby SOA[1]. V tomto konceptu chápeme procesy a funkce IT systémů (nebo celého podniku) jako služby, které jsou poskytovány jednotlivým uživatelům nebo dalším systémům (organizačním složkám podniku). Důležitým aspektem je myšlenka, že v dynamickém prostředí se rychle mění požadavky kladené na kvalitu poskytovaných služeb. Chce-li být firma konkurenceschopná musí umět rychle na nové požadavky reagovat. Důraz je kladen na volnou vazbu mezi poskytovatelem a konzumentem služby, jinými slovy na možnost snadné výměny nevyhovujícího poskytovatele služby za jiného, na sdílení služby a opakované použití těchto služeb.

Webové služby představují jeden z technologických prostředků, kterými je možné principy SOA realizovat (Newcomer, Lomow, 2004). Jsou postavené na všeobecně uznávaných standardech a čím více se rozšiřuje aplikační podpora této technologie, tím nabývají webové služby na svém významu.

Jednou z aplikací podporujících webové služby je groupwarová platforma firmy IBM - Lotus Notes/Domino. Ačkoli je Lotus Notes/Domino značným počtem firem využíváno pouze jako e-mailový a groupwarový prostředek, nabízí mnohem širší využití, například pro DMS a workflow aplikace, které lze právě na groupwarovou funkcionalitu snadno a s výhodou navázat. Tyto aplikace často potřebují komunikovat s jinými systémy, ať již s externími mimo vnitřní síť podniku nebo interním dostupnými v rámci podniku. Právě webové služby mohou být pro tento účel správnou volbou.

V této práci ukážeme možné praktické využití webových služeb v Lotus Notes/Domino na příkladu propojení tohoto systému se systémem UIR-ADR[2] - registrem adres spravovaným Ministerstvem práce a sociálních věcí (MPSV, 2011).

[1] SOA - Service Oriented Architecture

[2] http://forms.mpsv.cz/uir/popis/popis.jsp

2. Cíl práce a metodika

Cíle práce

Cílem této práce je vymezit pojem webové služby a popsat standardy související s webovými službami včetně rozdílů v různých verzích těchto standardů.

Druhým cílem je charakterizovat platformu Lotus Notes a Domino a ukázat možnosti využití webových služeb na této platformě. Popsat nedostatky podporované funkcionality a navrhnout možnosti využití webových služeb v tomto prostředí. Dílčím cílem je také realizace modulu pro validaci poštovních adres, doplňování názvů měst a ulic použitelného v Lotus Notes aplikacích.

Metodika práce

Teoretická část práce je založena na literární rešerši a analýze získaných poznatků s ohledem na praktickou využitelnost v aplikacích na platformě Lotus Notes a Domino.

Praktická část práce představuje příklad aplikace využívající webových služeb v této platformě, pomocí standardních prostředků tohoto systému.

V závěru práce je provedeno zhodnocení zjištěných poznatků a konkrétního příkladu.

3. Teoretická východiska

3.1. Vymezení pojmu webové služby

Technologie webových služeb byla vyvinuta pro komunikaci mezi dvěma nezávislými, samostatně fungujícími počítačovými systémy. Inteligence a porozumění „živých“ uživatelů, kteří jsou schopni inteligentně klást i zodpovídat dotazy různých forem a tvarů, je nahrazena striktní definicí volání poskytované služby a pevně danou strukturou vstupních parametrů i výstupních dat.

V širším pojetí je webová služba libovolná internetová technologie poskytující služby nezávislé na operačním systému a programovacím jazyce a komunikující pomocí zpráv formátovaných ve standardizovaném XML tvaru (Ethan, 2002, s. 11).

Systém poskytující službu označujeme jako poskytovatele (nebo též server, providera) a systém využívající této služby nazýváme konzumentem (klientem). Poskytovatel čeká na požadavky (zprávy) přicházející od konzumenta webové služby, přicházející zprávy zpracuje a ve většině případů vrací žadateli zprávu s výstupními daty. V praxi požadujeme, aby poskytovaná webová služba byla:

Sebepopisná

Součástí služby je XML dokumentace popisující seznam všech veřejných metod dané služby, jejich argumentů a vrácených hodnot.

Standardizovaná

Dostupná v internetové nebo intranetové síti pomocí standardních protokolů a komunikující pomocí standardizovaného na XML postaveného systému zpráv (SOAP).

Nezávislá

Nezávislá na operačním systému a programovacím jazyce. Nezávislost je posílena právě použitím obecně uznávaných a podporovaných standardů vyžadovaných pro definici samotné služby, přenos i obsah dat.

Vyhledatelná[3]

Musí existovat mechanismus, kterým se zveřejní nově dostupná webová služba a zároveň budou případní zájemci schopni vyhledat službu požadovaných vlastností a získat popis jejího rozhraní.

3.2. XML

Koncept webových služeb je vybudován na několika standardech založených na eXtensible Markup Language (XML). Jedná se o jazyk patřící stejně jako např. HTML do rodiny tzv. značkovacích jazyků. Dokumenty se skládají z vlastního textu, jenž je uzavírán mezi páry značek, které nesou informaci o významu uzavíraného textu nebo způsobu jeho zpracování.

3.2.1. Historie XML

Jazyk byl vyvinut konzorciem W3C již v roce 1996 jako zjednodušení jazyka SGML[4] (W3C, 2006) . Definice XML dokumentu je nazávislá na platformě a zapisovaná v Unicode kódování, podporujícím národní znakové sady. Díky těmto vlastnostem byl velmi rychle přijat a dnes je to široce přijímaný standard pro tvorbu strukturovaných dokumentů. Nemá předdefinovánu žádnou sadu značek, pouze pravidla pro jejich zápis a vnořování. Je to v podstatě metajazyk, pomocí kterého je možné definovat nové jazyky[5]. K tomu si stačí jen definovat povolenou sadu značek, jejich hodnot a strukturu.

Aktuálně konzorcium spravuje paralelně dvě verze XML: verzi XML 1.0 (fifth edition z roku 2008 (W3C, 2008) a verzi XML 1.1 (second edition) z roku 2006 (W3C, 2006). Rozdíl mezi verzemi kromě hlavičky <?xml version="1.0"?>, která je v 1.0 verzi nepovinná, je v používání znaků ve jménech entit. Verze 1.0 striktně předepisuje, které znaky jsou povoleny. Verze 1.1 naopak specifikuje, co povoleno není.

---

[3] Tento princip není dodržován, zvláště když konkrétní realizace webové služby slouží jako integrátor dvou interních podnikových systemů. Podnik tak ani nemá zájem o zveřejnění rozhraní této webslužby. Tento princip má svůj původ v obecnějších požadavcích SOA.

[4] W3C - World Wide Web Consortium , SGML - Standard Generalized Markup Language

[5] Jen na internetu lze nalézt přes 200 značkovacích jazyků založených na XML (http://en.wikipedia.org/wiki/List_of_XML_markup_languages)

3.2.2. Element

Základním stavebním kamenem XML dokumentů jsou elementy.

Element je:

- Dvojice tagů se jménem elementu: otevírací `<dotazy>` a koncový `</dotazy>`. Uvnitř závorek otevíracího tagu je možné definovat atributy ve tvaru jméno = "hodnota". Vícenásobné atributy oddělujeme od sebe mezerou, např: `<dotazy dotaz_pocet="1" tazatel="Vít Hlava">`.
- Případně jeden tzv. „prázdný" tag `<prazdny/>`, který nahrazuje zápis `<prazdny></prazdny>`.
- Nebo jiný speciální výraz: komentář, `<!DOCTYPE ...>, atd.`

Element obsahuje seznam atributů nebo další element – potomka. Potomek je buď:

- element
- textová hodnota (sekvence Unicode znaků)
- komentář `<!-- Komentář -->`
- nebo jiný speciální výraz

Z této rekurzivní definice vyplývá základní pravidlo o vnořování elementů. Jestliže jeden element začíná uvnitř jiného elementu, musí také uvnitř tohoto elementu skončit (elementy se nesmějí překrývat). Jinými slovy tvoří elementy dokumentu stromovou strukturu. Až na kořenový element má každý element právě jednoho rodiče a žadný nebo více potomků.

U názvů elementů i atributů rozlišujeme (narozdíl od HTML) velikost písmen[6], tzn. element `<dotazy>` je jiný než element `<DoTaZy>`. Názvy jsou vždy jednoslovné (bez mezer), mohou obsahovat alfanumerické znaky a kromě tečky, dvojtečky, pomlčky a podtržítka nesmí obsahovat žádné speciální znaky. Kompletní omezení ve jménech elementu je uvedeno ve specifikaci (W3C, 2006).

Vzhledem ke speciálnímu významu určitých znaků bychom při pokusu o jejich použití uvnitř elementu porušili XML syntaxi. Např. při zápisu nerovnice 10<x by zdrojový text vypadal následovně: `<nerovnost>10<x</nerovnost>` a parser by při čtení takového

[6] Jedním z důvodů je poměrně náročný převod Unicode kodování na malá písmena nebo různé převody některých písmen v různých národních abecedách (ačkoli původně měl být „monocase".)

dokumentu reportoval nepovolené znaky „</“ v názvu elementu <x</nerovnost>. Proto bylo potřeba definovat zástupné znaky (speciální entity). Nejdůležitější zástupky jsou v níže uvedené tabulce. Správný XML zápis uvedené nerovnice je tedy <nerovnost>10<x</nerovnost>.

Tabulka 3.1 Zástupné znaky v XML

<	<	“	"
>	>	‘	'
&	&		

3.2.3. XML dokument

Dokument vždy obsahuje právě jeden kořenový (dokumentový) element. Zápis dokumentu začíná řádkem definujícím verzi, kódování či jiné vlastnosti dokumentu. Dále může obsahovat komentář a další speciální položky, např. odkaz na validační dokument a právě jeden dokumentový element. V našem případě element <dotazy>, viz obr.3.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Komentář -->
<dotazy dotaz_pocet="1" vystup_format="XML">
        <dotaz>
                <hledany_text>Most</hledany_text>
        </dotaz>
</dotazy>
```

Obrázek 3.1 Příklad jednoduchého XML dokumentu.

3.2.4. Správná struktura (Well-Formed) a validita

Další klíčovou vlastností XML dokumentu je oddělení pojmů „správně formátovaný“ dokument a „validní“ dokument. Správně formátovaný dokument je každý, který splňuje syntaktická pravidla. Validní dokument musí navíc splňovat další podmínky kladené na strukturu jeho elementů, jejich hodnoty a atributy. Tyto další podmínky mohou být definovány několika různými způsoby.

Nejjednodušším způsobem, nabízejícím ale pouze základní možnosti kontroly struktury dokumentu, je validace pomocí DTD – Document type definition. Tento způsob je ve XML standardu implementován již od počátku a je podporován snad všemi parsery.

Z výpisu uvedeného na obr. 3-2 může element `<dotazy>` obsahovat jeden nebo více elementů typu `<dotaz>`. Element `<dotaz>` může obsahovat právě jeden element `<hledany_text>`, který již obsahuje parsovaný text (#PCDATA). Navíc element `<dotazy>` musí obsahovat dva atributy `dotaz_pocet` a `vystup_format`.

```
<!ELEMENT dotazy (dotaz+)>
<!ELEMENT dotaz (hledany_text)>
<!ELEMENT hledany_text (#PCDATA)>
<!ATTLIST dotazy dotaz_pocet #REQUIRED vystup_format(XML|HTML)“XML“ >
```

Obrázek 3.2 Příklad DTD pro XML dokument

Dalšími v praxi často používanými způsoby jsou validace pomocí XML Schema[7], anebo Relax NG[8] umožňující definovat i složitější datové struktury a datové typy hodnot elementů. Kromě toho také podporují regulární výrazy a práci se jmennými prostory.

Na obr. 3.3 s výpisem XML Schema je vidět cena, kterou za lepší kontrolu obsahu XML dokumentu zaplatíme - několikanásobně delší a méně přehledný zápis pravidel pro validaci stejného dokumentu, než byl zápis pomocí DTD. Přílišná složitost je zároveň jedním z argumentů odpůrců XML schémat.

V Relax NG jsou povoleny dvě navzájem převoditelné formy zápisu, kompaktní forma podobná DTD a delší forma založená na XML.

Relax NG byl vyvinut konzorciem OASIS[9] spojením dvou předchůdců RELAX a TREX (OASIS, 2001). V současnosti nahradil původní DTD validaci v XML jazyce DocBook verze 5, což je jedna z nejvýznamějších implementací XML vhodná pro psaní počítačové literatury a technické dokumentace (OASIS, 2009).

[7] XML Schema je jazyk založený na XML a standardizovaný organizací W3C (2004a).

[8] Relax NG [ri,læksiŋ].

[9] The Organization for the Advancement of Structured Information Standards.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
 <xs:simpleType name="vystup_format">
  <xs:restriction base="xs:string">
   <xs:enumeration value="XML"/>
   <xs:enumeration value="HTML"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="dotaz">
  <xs:sequence>
   <xs:element name="hledany_text" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:element name="dotazy">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="dotaz" type="dotaz" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="dotaz_pocet" type="xs:int" use="required"/>
   <xs:attribute name="vystup_format" type="vystup_format" default="XML"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Obrázek 3.3 Příklad validačního XML Schema

3.2.5. Jmenné prostory (Namespaces)

Vzhledem k jednoduchosti definice vlastního jazyka založeného na XML vzniká velké množství seznamů jmen elementů a atributů. Pokud bychom v jednom dokumentu zkombinovali více schémat či jiných jazyků, mohlo by se snadno stát, že by byl element daného jména popsán v několika dokumentech a tím by vznikl konflikt při validaci takto pojmenovaných elementů.

Aby se zajistila jednoznačnost jmen všech elementů a atributů, přiřadí se lokální názvy do příslušných jmenných prostorů. Nejprve nadefinujeme jmenný prostor pomocí atributu xmlns kořenového elementu dokumentu. Syntaxe přiřazení je následující: xmlns:jmeno_prostoru="http://pef.czu.cz/priklad" . Názvy elementů a atributů patřících do kontextu tohoto jmenného prostoru opatříme prefixem jmeno_prostoru:název_elementu.

URL tvary pro jména prostorů se používají, protože jsou jednoznačné, čímž je zabezpečena nezaměnitelnost názvů.[10]

3.2.6. Datový model a parsery XML

XML je možno chápat jako zápis obecnějšího datového modelu tzv. XML Infosetu[11]. Zobecněním elementů, atributů a textových hodnot jsou datové komponenty obsahující v případě elementů seznam dětských komponent, seznam komponent atributu a textový obsah elementu. Počítačový program tedy může pracovat s obecnější interpretací XML uloženou ve výhodnější datové struktuře: rychlejší pro zpracování, paměťově úspornější, data nemusí být interně zapsána v Unicode apod. XML Infoset je použit i při popisu dalších standardů W3C, např. SOAP 1.2, WSDL 2.0, WS-Addressing atd.

Převod XML Infosetu do textové XML podoby se nazývá serializace. Serializace musí odpovídat XML 1.0 nebo XML 1.1 specifikaci.

Parsery jsou aplikační prostředky, které umějí kontrolovat, zda textový dokument odpovídá specifikaci XML. Typově můžeme odlišit SAX Parser a DOM Parser. Oba tyto modely jsou podporovány v systému Lotus Notes. V implementaci příkladu je k parsování vstupní zprávy použit DOM Parser, který je díky své hierarchické struktuře přehlednější než SAX Parser. Ten je založen na zpracování událostí typu začátek tagu, konec tagu apod., a proto je lepší (rychlejší) pro zpracování dokumentů s velkým počtem řádků.

3.2.7. DXL

Lotus Domino server má definovaný vlastní jazyk postavený na XML, tzv. Domino Extensible Language – používá se ke XML reprezentaci dat a to nejen dokumentů, ale i všech návrhových prvků jako jsou knihovny, agenti, pohledy a formuláře. Vzhledem k existenci exportovacích a importovacích nástrojů v Lotus Notes Designerovi je možné pomocí DXL upravovat i vlastnosti aplikací, které nejsou podporovány v uživatelském prostředí. Na použití DXL jsou založeny i některé nástroje v Lotus Notes sloužící k lokalizaci aplikací do národních jazyků.

---

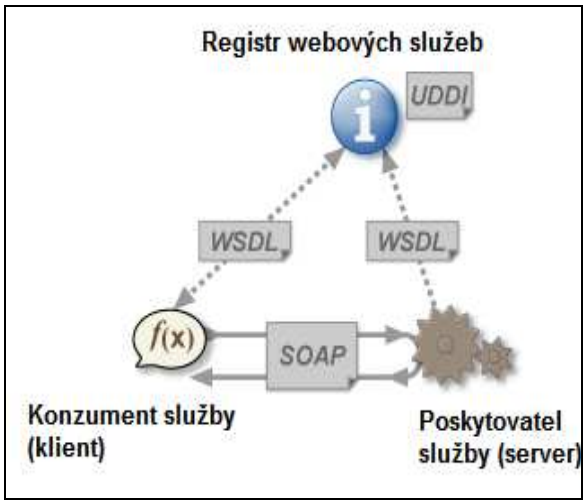
[10] Ve skutečnosti tento URL nemusí ani existovat. Slušní programátoři na URL umísťují dokumentaci vztahující se k danému XML dokumentu.

[11] XML Infoset – XML Information Set. Viz standard W3C (2004b).

3.3. Architektura webových služeb

3.3.1. Klient - server

Webové služby patří svou architekturou mezi tzv. *klient - server* aplikace. Viz obr. 3.4. Serverová aplikace poskytuje své služby jednomu nebo několika klientům. Komunikace je zprostředkována výměnou zpráv založenou na protokolu SOAP.



Obrázek 3.4 Architektura webové služby (Zdroj: Ethan, 2005)

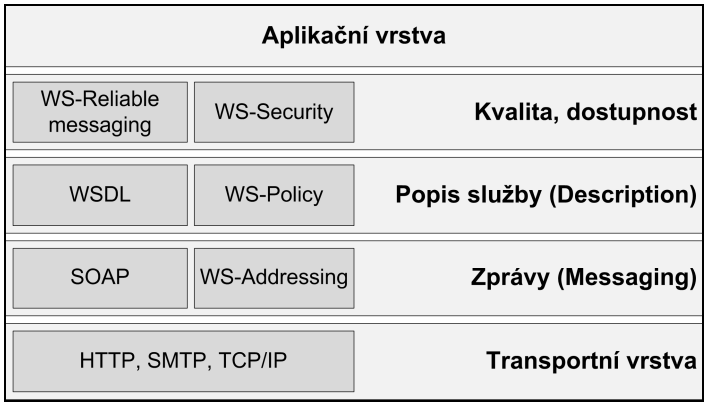
Dalším subjektem v architektuře je centralizovaný registr webových služeb[12], v němž je k dispozici popis webových služeb, odkaz na popis rozhraní – WSDL a případná další omezení.

Pomocí mechanismů UDDI se poskytovatel webové služby při vzniku zaregistruje do registru a stává se tak dostupným pro konzumenty těchto služeb. Naopak potenciální konzument webové služby může v registru vyhledat vhodnou službu a získat síťovou adresu, na které je dostupná a definici jejího rozhraní.

[12] Centralizace na vnitropodnikové úrovni. Vyšší stupně centralizace se neujaly, viz kapitola UDDI.

3.3.2. Vrstevnatá architektura funkčních částí

Z hlediska závislosti funkčních prvků můžeme webovou službu rozdělit do pěti základních vrstev: (Weerawarana, Cerbera, 2005). Viz obr. 3.4.



Obrázek 3.4 Vrstvy webové služby (Zdroj: převzato z Weerawarana, Cerbera, 2005)

Transportní vrstva (Transport)

Funkce na nejnižší úrovni zabezpečují transport zpráv mezi klientskou a serverovou aplikací. Přenos dat může probíhat na libovolném komunikačním protokolu. Nejčastěji se používá protokol HTTP (HTTPS).[13]

Definice zpráv (Messaging)

V této vrstvě se za pomocí W3C standardu SOAP definuje struktura zpráv posílaných mezi klientem a serverem a zároveň způsob adresování (doručení) zprávy na místo určení.

Popis služby (Description)

Na této úrovni se pomocí jazyka WSDL definuje samotná webová služba: seznam funkcí, parametrů a návratových hodnot. Patří sem i další specifikace webových služeb, které nejsou zahrnuty ve WSDL a popisují se pomocí politik (WS-Policy).

Kvalita služeb (Quality Of Service)

Popisuje další obecnější požadavky kladené na webovou službu, jimiž jsou bezpečnost a spolehlivost. Toto rozšíření není v aktuální verzi Lotus Notes/Domino podporováno.

[13] V aktuální verzi Lotus Notes/Domino 8.5.2 nelze výběr protokolu ovlivnit. Podporován je pouze HTTP.

Aplikační vrstva

Představuje již vlastní aplikaci či komponentu poskytující nebo využívající webovou službu. Obsahovat logiku pro zpracování konkrétních funkcí.

3.3.3. SOAP

SOAP je základním stavebním prvkem webových služeb. Je to prostředek pro výměnu zpráv mezi počítači přes síťové rozhraní. Je kompletně založený na XML a právě nezávislost na platformě a programovacích nástrojích byla rozhodující výhodou proti podobným technologiím (CORBA, DCOM) a důvodem jeho rychlého rozšíření.

Původní XML-RPC model

Simple Object Access Protocol byl vyvinut jako nástupce jednoduchého modelu XML-RPC (Remote Procedure Call) v roce 1998 firmou Microsoft. Tento jednoduchý model na přenos dat mezi dvěma systémy je založený na protokolu HTTP, pomocí kterého předává data z klientského počítače na server (HTTP POST request) a zpět (HTTP response).

```
POST /target HTTP 1.0
User-Agent: identifikátor softwaru
Host: internetová adresa volajícího
Content-Type: text/xml
Content-Length: délka požadavku v bytech
```

Obrázek 3.5 XML-RPC HTTP hlavička volání funkce

```
<?xml version="1.0"?>
<methodCall>
        <methodName>GetListOfStreets</methodName>
        <params>
                <param>
                        <value><string>Most</string></value>
                </param>
        </params>
</methodCall>
```

Obrázek 3.6 XML-RPC – obsah volání funkce

Volaná funkce i odpověď jsou organizované jako jednoduchý XML dokument. Požadavek začíná kořenovým elementem <methodCall>, uvnitř kterého se definuje jméno funkce a její parametry v elementu <params>. Odpověď začíná elementem

<methodResponse> a v elementu <params> obsahuje hodnotu výstupu. Hodnoty parametrů se uzavírají do elementu <value> a jsou jedním ze šesti základních datových typů: int, double, boolean, string, datetime, base64 nebo jedním ze dvou složených typů: array a struct.

XML-RPC neobsahuje žádné další mechanismy na kontrolu routování, zabezpečení, validaci nebo publikování služeb. Hlavním nedostatkem XML-RPC modelu je neexistence prostředku pro definování struktury zpráv, což snižuje jeho flexibilitu, brání automatickému volání a vyhledávání služeb a nesplňuje tak požadavky SOA paradigmatu.

Historie SOAP modelu

První verze SOAP[14] byla publikována konzoricem W3C v roce 2000 jako verze 1.1 (W3C, 2001). Záměrem bylo vytvořit standardizovaný, snadno rozšiřitelný model komunikace mezi počítačovými systémy, který odstraňuje nedostatky předchozího modelu XML-RPC. V roce 2003 bylo vydáno SOAP Version 1.2 jako W3C Recommendation. Označení „SOAP“ přestalo být vnímáno kvůli nejednoznačnosti verzí jako zkratka původního názvu. Aktuální je verze 1.2 (Second Edition) z roku 2007 (W3C, 2007a).

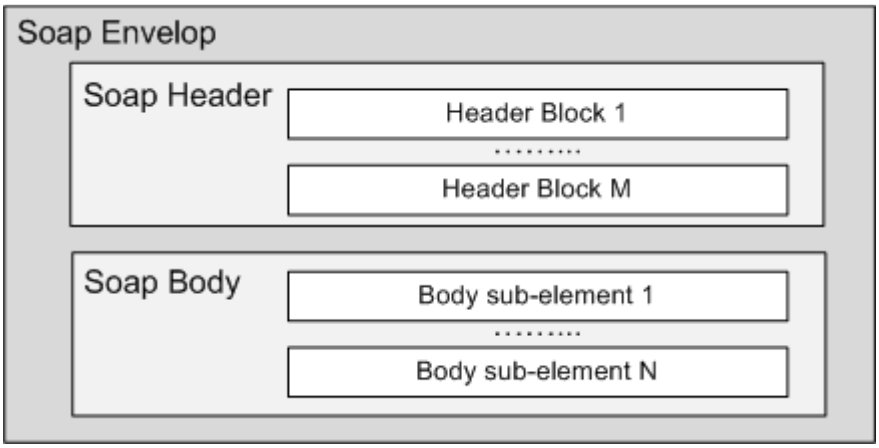
Definice SOAP

SOAP je souhrn pravidel pro tvoření správně formátovaných zpráv a jejich zpracování. Tato pravidla určují chování jednotlivých SOAP uzlů, přes které je zpráva doručována ke svému koncovému příjemci. Uzly jsou zařízení nebo aplikace, jež obsahují logiku potřebnou pro vytvoření, přijetí, zpracování nebo přeposlání SOAP zprávy.

[14] Simple Object Access Protocol (pouze do verze 1.1)

Struktura SOAP zprávy

Struktura SOAP zprávy je složitější než u předchozího modelu, podporuje použití XML Schema, jmenné prostory a další doplňky jako WS-Addressing. Zpráva se skládá z obálky (Envelop), která obsahuje hlavičku (Header) a tělo (Body). Hlavička je rozdělena na bloky (nula, jeden nebo více), které obsahují informace o jednotlivých příjemcích zprávy: adresu příjemce, identifikátor zprávy, časovou značku a případně další informace dle implementace SOAP. Tělo obsahuje vlastní data, tj. volání funkce nebo XML dokument ke zpracování.



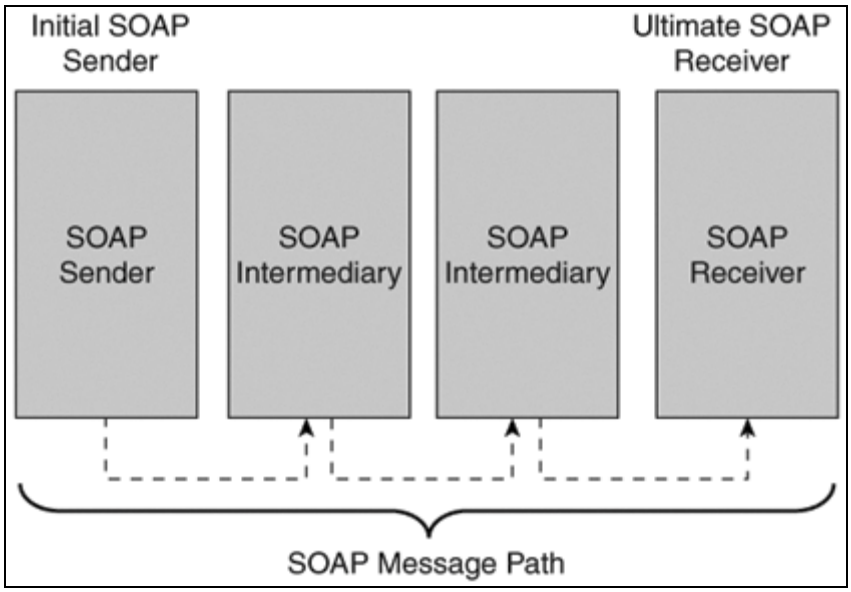
Obrázek 3.7 Struktura SOAP (Zdroj: Weerawarana, Cerbera, 2005)

Dokument versus RPC

SOAP podporuje dva způsoby předání vlastních dat v těle zprávy. První způsob je předání samostatného XML dokumentu (document literal), příjemce dokument zpracuje a v odpovědi vrací nový XML dokument. Druhým způsobem je RPC (remote procedure call), kdy tělo zprávy, tedy element <env:Body>, obsahuje jediný vnořený element s názvem voláné funkce a v něm parametry funkce (anlogicky jako v XML-RPC modelu). V těle odpovědi se pak vrací výsledek volané funkce. Ve webových službách preferujeme předávání celých dokumentů před programově zaměřeným voláním funkcí.

Zpracování SOAP zprávy přijímacím uzlem

SOAP zpráva je vytvořena v primárním uzlu (Initial SOAP Sender) a postupně předávána jednotlivým spojovacím uzlům (SOAP Intermediaries) až je nakonec doručena ke koncovému uzlu (Ultimate SOAP Receiver). Viz obr. 3.8. V nejjednodušším případě neobsahuje cesta žádné spojovací uzly - je tvořena pouze odesílajícím a přijímajícím uzlem.



Obrázek 3.8 Předávání SOAP zprávy (Zdroj: Weerawarana, Cerbera, 2005)

Zpracování SOAP zprávy se děje na základě rolí. Každý uzel zaujímá roli ve vztahu k přijímané zprávě. Každá zpráva má v hlavičce bloky, jež mohou obsahovat nepovinný atribut role. Pokud je role uzlu zpracovávajícího danou zprávu shodná s rolí bloku v hlavičce zprávy, je tento uzel zodpovědný za zpracování bloku. Specifikace definuje tři základní role identifikované pomocí URI[15], ke kterým mohou být přidány libovolné další podle potřeb aplikace (viz následující tabulka 3.2).

Tabulka 3.2 SOAP role (Zdroj: Weerawarana, Cerbera, 2005)

next	"http://www.w3.org/2003/05/soap-envelope/role/next"
none	"http://www.w3.org/2003/05/soap-envelope/role/none"
ultimateReceiver	"http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"

[15] Uniform Resorce Identifier – Řetězec znaků identifikující zdroje uvnitř počítačových sítí.

Role *Next* (ve významu další uzel na cestě zprávy) je zaujímána každým přijímajícím uzlem, role *UltimateReceiver* je zaujímána pouze koncovým uzlem a role *None* je vyhrazena pro vyznačení bloku hlaviček, které nemají být zpracovány žádným uzlem. Každý uzel může zaujímat i více rolí. Tak je to například u koncového uzlu, jenž má jak roli *Next*, tak roli *UltimateReceiver*. Při shodě rolí uzlu a bloku hlavičky je daný blok hlavičky uzlem zpracován a při přeposlání dalšímu uzlu je ze zprávy vypuštěn. Toto chování lze upravit dalšími nepovinnými atributy bloku. Koncový uzel má navíc povinnost zpracovat tělo SOAP zprávy. Detailně jsou role popsány ve specifikaci SOAP 1.2.

Komunikační vzory SOAP

SOAP je ze své podstaty bezstavový, jednosměrný systém předávání zpráv od odesílatele k příjemci. Tuto základní funkcionalitu lze rozšířit a docílit komunikace typu *požadavek – odpověď*, typu *odpověď* (analogicky HTTP GET request) nebo složitějších komunikačních vzorů. Rozšíření je možné provést pomocí doplňků implementovaných v hlavičce SOAP zprávy (WS-Addressing), nebo využitím transportního protokolu, kterým je požadavek přenášen (např. použití transportního protokolu HTTP), nebo přímo na aplikační úrovni. Nejčastěji je přenos zprávy zabezpečen pomocí HTTP.

Přenos SOAP zprávy pomocí HTTP

Specifikace SOAP 1.2 dokumentuje dva způsoby využití HTTP k přenosu zpráv.

Prvním je vložení SOAP zprávy do těla HTTP POST requestu a výsledné SOAP zprávy do těla HTTP response. HTTP POST request a HTTP response jsou přenášeny za použití jediného TPC připojení a tedy přirozeně fungují jako komunikační vzor typu *požadavek – odpověď*. Jak vypadá struktura požadavku je znázorněno na obr. 3.9.

```
POST /GetListofStreets HTTP/1.1
Host: nomoney.cz
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>  ...  </env:Header>
  <env:Body>  ...  </env:Body>
</env:Envelope>
```

Obrázek 3.9 Schéma HTTP POST požadavku

Strukturu odpovědi ukazuje obr. 3.10.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>  ...  </env:Header>
  <env:Body>  ...  </env:Body>
</env:Envelope>
```

Obrázek 3.10 Schéma HTTP response

Druhým specifikovaným způsobem je metoda volání GET, která odpovídá komunikačnímu vzoru *odpověď*. V hlavičce HTTP GET se uvede, že výsledek má být předán v těle HTTP response jako SOAP zpráva a to nastavením parametru Accept: application/soap+xml.

WS-Addressing [16]

Při použití této technologie se do hlavičky SOAP zprávy vkládají další elementy sloužící k identifikaci cílového uzlu <To>, akce <Action> , odesílatele <From> ,zprávy <MessageID>, adresy pro odpověď <ReplyTo>. V odpovědi může hlavička navíc obsahovat identifikátor původní zprávy v elementu <RelatesTo>, viz příklad na obr. 3.11.

```
Request:
<env:Header>
   <wsa:MessageID>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff</wsa:MessageID>
   <wsa:ReplyTo>
      <wsa:Address>http://nomoney.cz/pc</wsa:Address>
   </wsa:ReplyTo>
   .....
</env:Header>

Response:
<env:Header>
   <wsa:MessageID>uuid:uuuuvvvv-wwww-xxxx-yyyy-zzzzzzzzzzzz</wsa:MessageID>
   <wsa:RelatesTo>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff</wsa:RelatesTo>
   .....
</env:Header>
```

Obrázek 3.11 WS Addressing - hlavička SOAP zprávy

[16] WS Addressing – Web Services Addressing (W3C, 2004c)

Rozdíly mezi SOAP 1.1 a SOAP 1.2

SOAP verze 1.1 je popsána v jediném dokumentu, ve kterém je definován model zpracování SOAP zprávy, vzory pro volání funkcí RPC, vazba na HTTP protokol a kódování dat.

Specifikace Verze 1.2 je rozdělena do tří dokumentů. Nevyžaduje již serializovaný XML dokument, ale pracuje s obecnějším datovým modelem - XML Infoset. V první části je přehled standardu včetně popisu přenosu příloh. Ve druhé části je definováno vlastní zprácování zpráv a možná propojení SOAP zpráv s jinými transportními protokoly. Ve třetí části je specifikace volitelných rozšíření komunikačních vzorů včetně ukázky komunikačního vzoru typu *požadavek – odpověď* pomocí protokolu HTTP.

Vzhledem k rozšíření SOAP 1.1 se nepředpokládá převod aplikací kompatibilních s verzí 1.1 na verzi 1.2.

3.3.4. WSDL

Web Service Description Language - WSDL je jazyk, který zavádí standardizovaný způsob popisu webových služeb. WSDL dokument popisuje syntaxi a strukturu služby, uvádí výčet jednotlivých funkcí včetně struktury příslušných vstupních a výstupních parametrů a umístění služby, aniž by rozkrýval vnitřní uspořádání, metody výpočtů, programovací jazyk či platformu.[17] Konzument služby je pak po předložení tohoto dokumentu schopen předepsaným způsobem zaslat webové službě zprávu v předepsaném formátu a tím volat její jednotlivé funkce. Konzument navíc není závislý na jednom poskytovateli služby a veškeré vnitřní změny poskytovatele jsou pro něj transparentní. Může přejít na novější verzi nebo i na úplně jiného poskytovatele s odpovídajícím WSDL dokumentem (Weerawarana, Cerbera, 2005).

Historie

Jazyk WSDL vznikl v roce 2000 kombinací dvou svých předchůdců - NASSL firmy IBM a SDL firmy Microsoft[18]. První verze WSDL 1.1 byla vydaná organizací W3C z roku

[17] *„...WSDL carefully does not cross into describing semantics of Web services. A WSDL document tells, in syntactic or structural terms, what messages go in and come out from a service...“* (Weerawarana, Cerbera, 2005).

[18] NASSL (Network Application Service Specification Language), SDL (Service Description Language)

2001 (W3C, 2001) a je dodnes v podstatě uznávána jako standard popisu webových služeb. Neschopnost a nejednoznačnost zápisu některých vlastností dala později vzniknout verzi WSDL 2.0, která nedostatky předchozí verze odstraňuje (W3C, 2007b). Vzhledem k tomu, že většina existujících aplikací podporuje a pracuje s WSDL 1.1, bude přechod i přes veškeré nedostatky verze 1.1 trvat mnoho let, což platí i ve světě Lotus Notes/Domino. Proto budou pro srovnání popsány hlavní prvky paralelně v obou systémech.

Obecná struktura WSDL dokumentu

Tento jazyk je založený na XML, a proto na WSDL dokumenty můžeme aplikovat standardní nástroje pro validaci: DTD, XML Schema apod. Pro popis datových typů se obvykle používá definice datových typů z XML Schema, ale s ohledem na budoucí potřeby nebo potřeby složitějších aplikací je povoleno i libovolné rozšíření. Pro služby určené širokému okruhu konzumentů se doporučuje validace využívat. Validační schéma usnadňuje potenciálním tvůrcům pochopení a vývoj klientských systémů.

WSDL dokument se obecně skládá ze dvou logických částí uzavřených do kořenového elementu <definitions> :

- Abstraktní část– popisující seznam funkcí, které služba poskytuje. Tato část odpovídá na otázku: „*Co?*“ služba poskytuje. (ve WSDL 1.1. je odpověď obsažena v elementech <type>, <message> a <portType> a ve WSDL 2.0 uvnitř tagu <interface> .)
- Konkrétní – popisující konkrétní způsob transportu zpráv, jejich formát a adresu webové služby. Tato část odpovídá na otázku: „*Jak?*” se lze k webové službě připojit, pomocí jakého protokolu a odpovídá elementu <binding>. Dále odpovídá na otázku: „*Kde?*“, na jaké adrese je služba dostupná (element <service> a <port> ve verzi 1.1. resp. <endpoint> ve verzi 1.2).

Struktura WSDL 1.1

Nejdůležitější elementy dokumentu jsou uvedeny na obr. 3.11. Kompletní popis lze nalézt v příloze A4 specifikace W3C (W3C, 2001).

```
<definitions>
        <types> ... </types>
        <message name="..."> ... </message>
        <portType name="..."> ... </portType>
        <binding name="..."> ... </binding>
        <service name="..."> ... </service>
</definitions>
```

Obrázek 3.12 Struktura WSDL 1.1

První element <types> definuje datové typy použité ve webové službě. Definují se zpravidla pomocí XML Schema, ačkoli lze použít i jiné technologie: DTD, Relax NG i technologie, jež umí popisovat dokumenty, které nejsou ve formátu XML.

Element <message> popisuje jednosměrné zprávy, jež si klient a poskytovatel služby vyměňují. Lze jej přirovnat k volání funkcí v běžných programovacích jazycích. Každá zpráva může obsahovat více částí pomocí tagu <part>, ve kterém lze dodefinovat i typ předávaných dat. Element <part> tedy odpovídá parametrům funkce.

Nejdůležitějším elementem sloužícím pro popis nabízené funkcionality je <portType>. Popisuje množinu operací, které webová služba dokáže provádět a zprávy, jež k tomu využívá. Je možné jej přirovnat ke knihovně funkcí nebo třídě ze standardních programovacích jazyků. Operace jsou ve WSDL 1.1 buď párové (*požadavek – odpověď*) nebo nepárové (jednosměrné, *one-way*). Obecně lze definovat směr libovolně, ale z pohledu poskytovatele služby nemá odchozí požadavek příliš velký smysl, pokud poskytovatel nemá informaci, kam požadavek směřovat. (Možnost, jak dodat informaci o směru nabízí např. rozšíření WS-Addressing.)

Konkrétní implementace přenosu dat pro každý jednotlivý port (portType) se popisuje elementem <binding>. Obsahuje potřebné informace o protokolu, kterým jsou zprávy přenášeny. Tento tag obsahuje dva atributy: *name* je jednoznačný identifikátor a *type* nabývá hodnoty atributu name elementu portu, k němuž se vztahuje. Specifikace WSDL 1.1 nepředepisuje konkrétní prostředek, kterým má být služba přenesena, ale naopak nabízí tři možné, nikoliv jediné způsoby: SOAP, HTTP a MIME včetně příkladů.

Struktura WSDL 2.0

Strukturu WSDL dokumentu popisuje specifikace konzorcia W3C rozdělená do dvou dokumentů. První popisuje vlastní jazyk WSDL 2.0 a druhý ukazuje jeho možná rozšíření. Ve specifikaci WSDL 2.0 je narozdíl od verze 1.1 použitý obecnější popis pomocí XML Infosetu. Specifikace definuje webovou službu jako soubor komponent, jež popisují jednotlivé její části. Komponenty jsou kolekce datových typů a vlastností reprezentující konkrétní části. Základní struktura WSDL dokumentu serializovaného do XML je znázorněna na obr. 3.12. Kořenový element <description> obsahuje povinně element definující datové typy <types> a nepovinně jeden nebo více elementů popisujících vlastnosti služby <interface>, <binding> a <service>. Výsledný WSDL dokument lze validovat podle XSD Schema (W3C, 2007c).

```
<description>
        <types> ... </types>
        <interface name="..."> ... </interface>
        <binding name="..."> ... </binding>
        <service name="..." interface="..."> ... </service>
</description>
```

Obrázek 3.13 Struktura WSDL 2.0

3.3.5. UDDI

Jedním z principů konceptu SOA je možnost dynamicky měnit poskytovatele služeb, tj. nejen při návrhu, ale i za běhu aplikace. UDDI je možná implementace tohoto přístupu, která byla publikována v roce 2000 společnostmi Microsoft, IBM a Ariba. Později byla předána do správy konzorcia OASIS. Poslední verzí je standard UDDI 3.02 z roku 2004 (OASIS, 2004).

Universal Description Discovery & Integration (UDDI) je specifikace psaná v XML, jež definuje nástroje pro publikování a vyhledávávání webových služeb na internetu. Službu popsanou standardním způsobem (např. WSDL) zpřístupní poskytovatel v registru služeb. Konzumenti mohou vyhledat v tomto registru vhodnou službu a získat internetovou adresu, na níž je služba poskytována a dostatečný technický popis pro využití dané funkce.

UDDI nabízí základní nástroje pro vytvoření jak veřejného registru služeb, tak registru služeb dostupného pouze interně v rámci jedné organizace. Vzhledem k tomu, že reálný zájem o veřejně poskytované služby byl minimální (firmy preferují spíše ověřené služby

s partnerem, s nímž mají určitou obchodní historii nebo servisní smlouvu), zastavily zakládající firmy v roce 2006 své centrální registry a práce na standardu UDDI (spravovaným konzorciem OASIS) byla pozastavena (SOA World Magazine, 2005). Dnes se využívají převážně registry služeb na podnikové úrovni.

3.4. Shrnutí

Technologie webových služeb zaznamenala od svého vzniku poměrně mnoho změn, které byly způsobeny vývojem jednotlivých standardů. Vedle XML 1.0 vzniklo XML 1.1, o něco větším vývojem prošel standard SOAP. Novější SOAP 1.2 oproti původnímu SOAP 1.1 neomezuje přenos zpráv pouze na protokol HTTP. Standard používá k popisu tzv. XML Infoset a umožňuje tak aplikacím použití úspornější reprezentace dat v neserializovaném tvaru a tím zrychlení přenosu i zpracování dokumentů. Rozšiřuje možnosti adresování webových zpráv, možnosti přenosu příloh a další.

Možná největší změny doznal standard WSDL pro vlastní popis poskytovaných služeb, který v nové verzi 2.0 odstranil nejednoznačnosti v popisu služeb ve verzi 1.1. Verze 2.0 navíc poskytuje více volnosti při definici komunikačních vzorů a umožňuje práci s neserializovanými dokumenty. Nicméně vzhledem k tomu, že existuje mnoho aplikací podporující pouze WSDL 1.1 se předpokládá, že obě verze budou do budoucna stále podporovány.

Budoucnost webových služeb je do značné míry zajištěna jejich stávajícím rozšířením, konceptem standardů nezávislých na operačním systému a programovacím jazyce, které navíc umožňují nová rozšíření, aniž by byla ohrožena zpětná kompatibilita.

4. Lotus Notes a Lotus Domino

Produkty Lotus Notes a Lotus Domino představují širkou softwarovou platformu pro firemní komunikaci, sdílení, publikování dokumentů, tvorbu dokumentově zaměřených aplikací apod. Často se uvádí, že se jedná o groupwarové řešení, což úplně nevystihuje veškeré přednosti této platformy, hlavně možnost jednoduché tvorby vlastních aplikací. Jedná se o aplikaci typu klient – server. Klientská část se nazývá *Lotus Notes* a serverová část od verze 4 *Lotus Domino*.

Historie

První verze systému Lotus Notes byla uvedena na trh v roce 1989 firmou Iris Software (Benz, Oliver, 2005, s. 41). V roce 1995 koupila tuto firmu společnost IBM, která má největší podíl na masivním rozšíření této platformy po celém světě. Aktuální verze Lotus Notes i Lotus Domina je verze 8.5.2., vydaná v srpnu 2010.[19]

4.1. Datový model

Systém Lotus Notes/Domino pracuje s dokumenty uspořádanými v souborech formátu .NSF[20]. Jedná se o speciální datový kontejner obsahující různé typy dokumentů:

Administrativní dokumenty – které vytvářejí a upravují administrátoři systému, ukládají do nich informace o zabezpečení a replikaci.

Dokumenty návrhů a kolekcí návrhů - které vytvářejí vývojáři (designéři), aby uživatelům umožnily práci s datovými dokumenty. Mezi návrhové dokumenty patří *Forms* (WYSIWYG[21] komponenty uživatelského rozhraní pro prostředí Lotus Notes a pro webové prostředí), *Script Libraries* - knihovny funkcí s aplikační logikou, *Agents* – funkce spouštěné manuálně z menu nebo automaticky při definované události, *Views*, *Pages* a *XPages* – návrhové prvky pro tvorbu webových aplikací, *Framesets* a další. Vzhledem k tomu, že databáze v sobě obsahují i funkcionalitu, odkazuje se na ně jako na *aplikace*.

Datové dokumenty – představují vlastní data databáze. Prvky v dokumentech nejsou popsané žádným schématem a nemusí splňovat žádné dopředu dané typové omezení jako

[19] Podle IBM používá systém Lotus Notes/Domino přes 140 miliónů uživatelů.

[20] NSF - Notes Storage Facility

[21] WYSIWYG – What You See Is What You Get

je tomu např. u relačních databází. Tento často vytýkaný nedostatek Lotus Notes je ale naopak základním kamenem architektury Lotus Notes a je chápán jako výhoda.

NSF soubor kromě dokumentů obsahuje ještě záhlaví, kde je objektové úložiště příloh a vnořených objektů nebo také informace o replikacích a o přečtených dokumentech (Benz, Oliver, 2005, s. 60).

Základní předností Lotus softwaru je tzv. *replikace* databází. Jedná se o funkcionalitu, která zajišťuje synchronizaci aplikací (databází) mezi servery nebo serverem a klientem. Každá databáze nese jednoznačný identifikátor, tzv. *Replica ID*, pomocí něhož systém pozná, které databáze patří k sobě a mohou vzájemně synchronizovat svůj obsah. Databáze se shodným Replica ID se nazývají *repliky*. Vytvořením repliky aplikace na Lotus Notes klienta umožňujeme uživateli použití aplikace off-line (bez přístupu k serveru). Využitím replikace databází na více serverů a klientů lze vytvářet rozsáhlé aplikace umožňující plně využít groupwarové podstaty systému Lotus Notes a Domino.

Vzhledem k tomu, že i návrh aplikace je uložený v dokumentech, je replikace zároveň nástroj pro šíření úprav, oprav a nových funkcí v jednotlivých aplikacích. Pomocí šablony (databáze obsahující pouze návrhové dokumenty) se zaktualizuje návrh cílové databáze, rozšíření na všechny servery a počítače všech uživatelů je potom ponecháno automatické replikaci.

4.2. Lotus Domino

Již od počátku je serverová část podporována celou řadou operačních systémů: Windows, x86 Linux, Solaris, AIX, iSeries, zLinux, z/OS. Tvoří centrální bod komunikace mezi uživateli systému a zabezpečuje další systémové funkce. Podobně jako u systému Windows je možné více serverů spojovat do větších celků – domén a tím dosáhnout i obrovských sítí s desetitisíci uživateli.

Servery lze instalovat v několika režimech podle toho, jaké funkce od serveru vyžadujeme. Mezi nejdůležitější funkce patří:

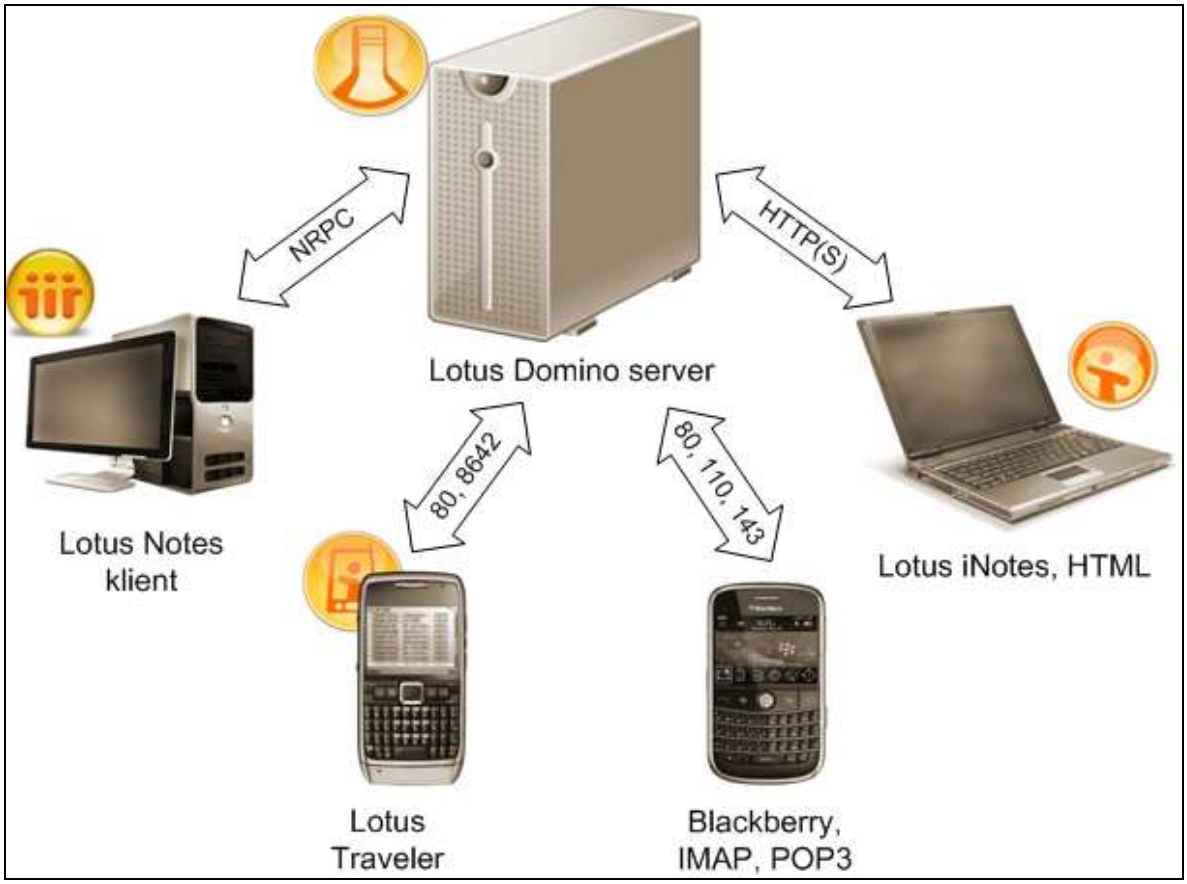
Databázový server - Funkce zajištující přístup uživatelů k aplikacím a datům uloženým v serverových databázích pomocí klienta , synchronizaci dat mezi více servery, konzistenci databází atd.

Poštovní server - Server pro příjem a odesílání e-mailových zpráv. Nejčastěji se pro komunikaci s mailovou schránkou používá přímo klient Lotus Notes, nicméně uživatel

může využít i další metody - POP3, IMAP nebo speciální konektory, např. pro MS Outlook.

Webový server - Pro tuto práci je významná možnost přístupu k databázím pomocí protokolu HTTP (HTTPS). To znamená, že vhodně navržené aplikace je Domino server schopen převést do HTML podoby a používat je i bez klienta Lotus Notes pomocí internetových prohlížečů. Podporuje práci s XML soubory, validaci a transformaci dat, využití AJAXu a dalších technologií pro vytváření dynamických webových stránek. Novinkou od verze 8.5 je možnost využití široké škály webových nástrojů, např. Dojo frameworku v novém vývojovém elementu nazvaném XPages.

Ostatní funkce – Po instalaci doplňků je možné přistupovat k poště pomocí mobilních zařízení (Lotus Notes Traveler) či Blackberry (BES for Lotus Domino), je možné využívat funkcí okamžitých zpráv (Lotus Sametime) včetně přenosu souborů, audia a videa (obdoba aplikací ICQ, Skype na firemní úrovni). Pomocí četných konektorů je možné integrovat datové zdroje z jiných databázových zdrojů, jako jsou SAP, DB2, MS SQL server a další. Domino server může fungovat také jako LDAP pro autentikaci do dalších systémů atd.



Obrázek 4.1. Komunikace Lotus klientů s Lotus Domino serverem (Zdroj : Autor, 2011)

4.3. Lotus Notes

Program Lotus Notes je klientská část systému Lotus Notes/Domino. Aktuální verze Lotus Notes 8.5.2 byla naprogramována v Javě v prostředí Eclipse a je podporována na operačních systémech Windows, Mac OS X a Linux. Ilustrační obrázek prostředí Lotus Notes je uveden v obrazové příloze (obr. 8.1) .

Lotus Notes může fungovat zcela samostatně bez připojení k Lotus Notes doméně a serveru[22], nebo je součástí domény a má nakonfigurováné připojení k serveru Lotus Domino. Jednotlivé aplikace se spouští poklikáním na ikonu na pracovní ploše nebo výběrem aplikace ze záložek v menu. Aplikace je otevírána buď ze serveru, nebo z lokálního počítače. Pro off-line práci musí existovat lokální replika databáze. Soubor .nsf je uložen na disku počítače a není potřeba internetového připojení.

[22] Podobně jako MS Outlook nemusí být napojen na Exchange server.

Kromě typických Lotus Notes aplikací (NSF) umožňuje klientské prostředí využívat mnoho dalších funkcí, např. chatování a hlasovou komunikaci mezi uživateli pomocí Lotus Sametime, k prohlížení webových stránek, jako RSS čtečku, lze do něj integrovat Javové aplikace (widgets) a další.

4.4. Lotus Domino Designer

Designer je program sloužící k vývoji aplikací pro platformu Lotus Notes a Domino. Novou aplikaci je možné vytvářet buď úplně od začátku (blank) nebo pomocí existujících šablon dodávaných přímo výrobcem, příkladem je mailová databáze uživatele, nebo šablon získaných z jiných zdrojů. Ilustrativní obrázek prostředí je v příloze 8.2.

Prostředí Lotus Domino Designera umožnuje programátorům použití dvou proprietárních jazyků: Jazyk formulí a LotusScript. Jazyk formulí obsahuje základní funkce pro práci s dokumenty, prvky dokumentů, validaci hodnot, výpočet hodnot v pohledech atd. Ačkoli lze pomocí jazyka formulí vytvořit plnohodnotnou aplikaci, je nezbytné pro složitější úkoly použít LotusScript. Tento jazyk umožňuje členění kódů do knihoven, umožňuje volat externí API funkce, umožňuje práci se složitějšími datovými strukturami a obsahuje prvky objektově orientovaného programování ve smyslu použití a dědění tříd. Nicméně neumožňuje např. polymorfismus, přetěžování funkcí, ani nepodporuje možnost rozdělení programu do více vláken. Ukázková webová služba pro validaci adres, popsaná v následující kapitole, je psaná právě v jazyce LotusScript.

Pokud nestačí základní prostředky tvorby aplikace, je možné ve vybraných designových elementech použít Javu, popřípadě JavaScript v aplikacích psaných pro web.

4.5. Lotus Domino Administrator

Pro správu Lotus Domino serverů je určen program Lotus Domino Administrator. Administrátoři pomocí tohoto programu konfigurují jednotlivé funkce Domino serverů, registrují uživatele, spravují certifikáty, nastavují přístupy k jednotlivým databázím, provádí kompaktaci a aktualizaci návrhů databází. (Ilustrativní obrázek serverové konzole domino Administratora je součástí přílohy 8.3.)

4.6. Nevýhody a výhody Lotus Notes a Domino

Jasnou nevýhodou je nedostatečný výkon databází pro vyšší počet dokumentů. Databáze jsou v podstatě omezeny pouze systémovými prostředky, ale indexování pohledu

a odezva databáze se prodlužuje, když se počet dokumentů začne blížit k hranici jednoho milionu.

V určitých případech je nevýhodou, že nad databázemi nelze provádět operaci spojení, používanou v relačních databázových systémech. Pokud máme například seznam klientů v jedné a uzavřené smlouvy v druhé databázi (tabulce), nelze standardními prostředky zobrazit zároveň data z obou tabulek. Toto omezení je obcházeno nesystémovým propisováním stejných hodnot na více dokumentů nebo exportem dat mimo Lotus Notes. Tento problém ve webovém prostředí částečně řeší nový návrhový element XPages.

Ačkoli se uvádí, že vývoj Lotus aplikací je až pětinásobně rychlejší než u jiných systémů, tak lze za další nevýhodu považovat určité nepohodlí programátora tvořícího rozsáhlejší aplikace v Lotus Domino Designerovi v porovnání s tvorbou aplikací v moderních komerčních i nekomerčních nástrojích typu Visual Studio, Eclipse a jiných. Chybí zde propracovanější průzkumník tříd, nedokonalé debugování a správa projektů, přestože se tato slabá místa od verze 7 výrazně vylepšila.

Výhodou je pokročilý nástroj replikace databází. Používá se pro zvýšení dostupnosti dat pro domény s více servery, pro dostupnost dat na počítači uživatele – práce off-line, pro šíření funkčních úprav nebo zálohování.

Další výhodou je poměrně rychlý vývoj aplikací. Možnost tvorby pohledů a agentů standardními uživateli aplikace.

Zásadní předností je i vysoké zabezpečení dokumentů uložených v aplikacích. Na každém dokumentu lze definovat, kdo ho má právo ho číst a editovat. Systém umožňuje šifrování dat až na úroveň jednotlivých polí dokumentu. Pole zašifrovaná uživatelem nedokáže přečíst ani administrátor systému.

5. Implementace systému validace adres

Obsahem této kapitoly je ukázka implementace webových služeb v systému Lotus Notes a Domino na praktickém příkladu. Definicí problému je usnadnění aplikace zadávání poštovních adres svých klientů uživatelům Lotus Notes. Vzhledem k využití adresy pro zasílání zboží či případný výjezd technika je kladen důraz na správnost zadané adresy. Navržený systém se opírá o veřejně dostupný registr adres UIR-ADR instalovaný v lokální síti. Informace o tomto systému je možné nalézt na internetových stránkách Ministerstva práce a sociálních věcí (MPSV, 2011).

5.1. Návrh řešení

5.1.1. Výchozí stav

Výchozím stavem pro validaci adres v Lotus Notes aplikaci byla jiná - zastaralá aplikace v Lotus Notes, obsahující seznam PSČ, seznam obcí podle PSČ a seznam ulic v obcích. Data v této databázi byla jednorázově naimportovaná data z číselníku obcí a PSČ České pošty a neúplného seznamu ulic staženého z nedůvěryhodného internetového zdroje. Zásadním nedostatkem se po čase ukázala neúplnost, nesprávnost a čím dál větší neaktuálnost dat v této pomocné databázi.

5.1.2. Výběr aktualizovaného zdroje dat

Za vhodnou platformu pro aktuální a pravidelně se aktualizující data byl zvolen registr adres UIR-ADR spravovaný Ministerstvem práce a sociálních věcí. Ten je možné využívat on-line zasláním správně formátovaného XML požadavku nebo off-line vyžádáním instalačního CD poskytovaného zdarma, které zahrnuje mimo jiné:

- Instalaci MS SQL serveru s připravenou databází registru adres
- Program na prohlížení, vyhledávání a ověřování adres
- Program na aktualizaci dat ze stažených změnových souborů

Vzhledem k regulaci počtu dotazů do on-line registru, k riziku výpadku internetového připojení a negarantované dostupnosti byla vybrána off-line varianta i za cenu nutnosti pravidelného stahování změnových souborů a aktualizaci databáze.

5.1.3. Způsob přístupu ke zdroji dat

Dále bylo potřeba zvolit technologii přístupu k tomuto zdroji dat. Přímé připojení uživatelů k SQL serveru bylo zavrženo. Vyžadovalo instalaci příslušného SQL driveru a konfiguraci datového zdroje na každém uživatelském počítači. Navíc by se musel vyřešit dostatečně bezpečný přístup k tomuto serveru z vnější sítě tak, aby tuto funkcionalitu mohli využívat i uživatelé mimo vnitřní síť podniku.

Jasnou volbou bylo využití technologie webových služeb a to hned z několika důvodů:

- Webovou službu je možné využít v budoucnosti i z jiných systémů a aplikací fungujících v podniku.
- Možnost využití služby i z jiných operačních systémů než Microsoft Windows.
- Klienta webové služby je možné na lokálním počítači realizovat v rámci Lotus Notes bez nutnosti instalace nebo konfigurace jakékoli další komponenty.
- Pro přístup k webové službě je možné využít standardních prostředků systému Windows nebo internetových prohlížečů[23].

5.1.4. Platforma pro poskytovatele webové služby

Nezávislost poskytovatele a konzumenta na programovacím jazyce a operačním systému dovoluje implementaci poskytovatele požadované webové služby na libovolné platformě. Jediným omezením je napojení na databázi UIR-ADR běžící na MS SQL serveru, ale vzhledem k tomu, že existuje mnoho konektorů na MS SQL server i z jiných operačních systémů, lze odstranit i toto omezení.

Vzhledem k existenci infrastruktury Lotus Notes a Domino v podniku bylo použití Lotus Domino serveru jako platformy pro implementaci poskytovatele webové služby přirozeným výběrem.

5.1.5. Alernativní použití jiných platforem

Kromě použití Lotus Domino serveru pro poskytování webové služby bylo možné v tomto řešení využít jiných systémů. Mezi nejpoužívanější lze zařadit např. univerzální Apache (Apache Axis2) nebo webový server IIS[24]firmy Microsoft.

---

[23] MSXML2.XmlHttp, Microsoft.XmlHttp - ActiveX komponenty využitelné pro volání webové služby.

[24] IIS - Internet Information Services

Apache Axis2/Java

Apache Axis2 je balík programových prostředků pro podporu technologie webových služeb, protokolu pro přenost zpráv SOAP a popisu služeb WSDL. Je implementován v programovacím jazyce Java. Axis2/Java systém je podporován na platformách Windows XP, Linux, Mac OS X, Fedora core, Ubuntu, Gentoo s nainstalovaným JDK[25] verze 1.5 a vyšší (Apache, 2010). Lze ho použít jak pro implementaci konzumenta webové služby, tak pro implementaci poskytovatele webové služby a obsahuje i funkcionalitu WS-Addressing, WS-Security, WS-SecurityPolicy, WS-ReliableMessaging a programátorské nástroje jako SOAP Monitor. Dokonce umožňuje přenos zpráv i přes protokoly SMTP a POP3.

Apache Axis2 zároveň obsahuje vlastní webový server zprostředkující dostupnost poskytovatele webové služby, popřípadě může být integrován do libovolného webového serveru kompatibilního s J2EE. Nejčastěji se integruje s Apache serverem. Je potřeba doinstalovat Axis2 WAR[26] soubor do Servlet Containeru, tj. adresáře obsahujícího javové aplikace určené pro běh ve webovém prostředí. Po instalaci je možné otevřít v prohlížeči webové rozhraní Axis2. Zde je možné spravovat, validovat a nahrávat nové webové služby, které lze připravit např. v programovacím prostředí Eclipse.

Microsoft IIS/.Net

Microsoft IIS je služba zprostředkující hostování webových aplikací včetně webových služeb na operačních systémech Microsoft Windows. Nejnovější verze je 7.5 pro operační systém Windows Server 2008 a Windows 7.

.Net framework obsahuje podobně jako Axis knihovny pro implementaci poskytovatele a konzumenta webové služby. Vlastní aplikaci webové služby je možné naprogramovat např. ve vývojovém prostředí Visual Studio v programovacím jazyce C. Pro vývoj postkytovatele webové služby v tomto prostředí je k dispozici šablona, která vygeneruje kostru kódu. Aplikaci (projekt) uložíme s příponou asmx a vypublikujeme do virtuálního adresáře IIS serveru.

Použitím WCF[27] rozšíření .Net frameworku umožnuje využít i většinu rozšíření jako jsou WS-Addressing, WS-Security, WS-Policy a další.

---

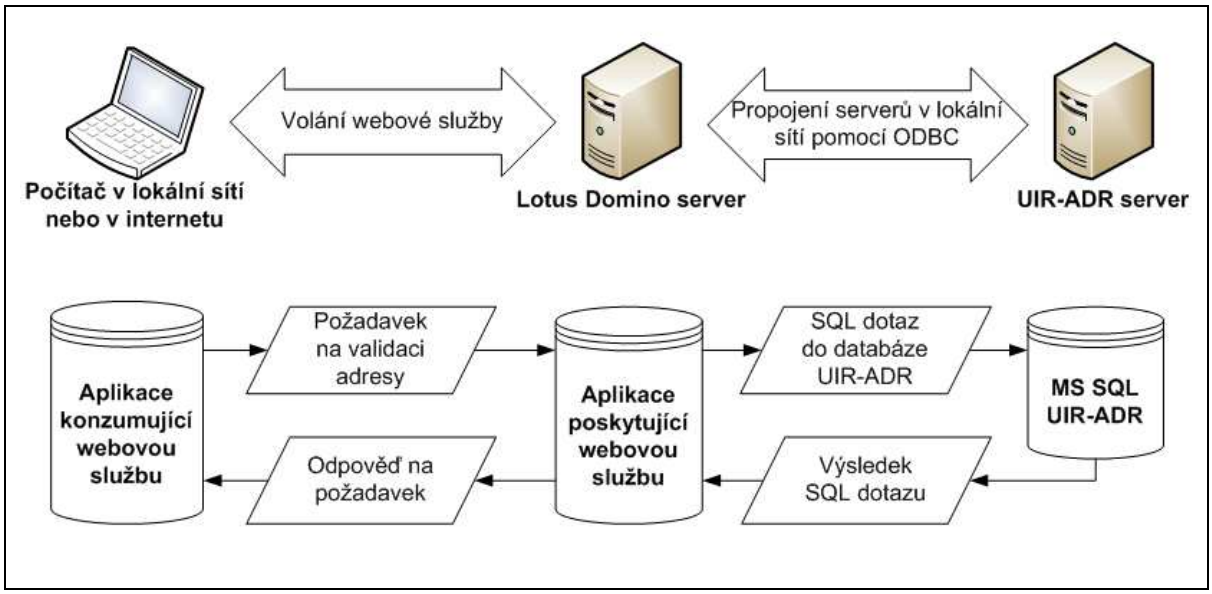
[25] JDK - Java Development Kit

[26] WAR – Web application archive

[27] WCF – Windows Communication Foundation

5.1.6. Struktura řešení

Popis infrastruktury je na obrázku 5.1. MS SQL server s databází systému UIR-ADR je nainstalován na samostatném počítači označeném UIR-ADR server. Na druhém počítači ve stejné Windows doméně je nainstalován Lotus Domino server s nakonfigurovaným datovým zdrojem napojeným na SQL server. Na Lotus Domino serveru běží webový server (HTTP task) přístupný na standardním portu, přes který je dostupná aplikace, jež poskytuje webovou službu.



Obrázek 5.1 Schéma sytému validace adres (Zdroj: Autor, 2011)

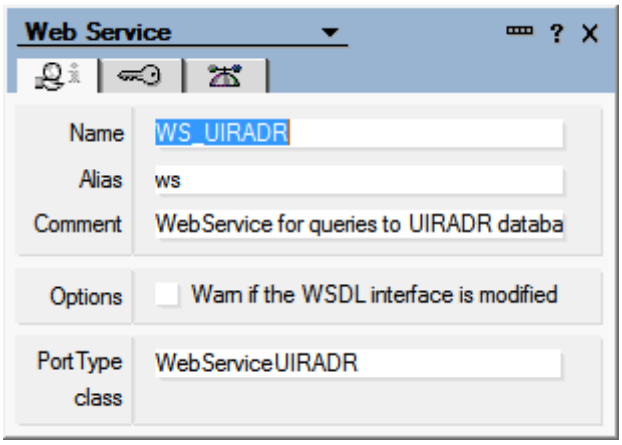
Základní webová služba poskytuje následující funkce:

- Seznam měst pro zadané PSČ a část města. Výstupem je pole názvů měst vyhovujících zadaným vstupním parametrům.
- Seznam ulic v zadaném městě, pro zadané PSČ a část názvu ulice. Výstupem je pole názvů ulic.
- Validace zadané adresy – vstupní parametry této metody jsou město, PSČ, ulice a číslo popisné nebo orientační. Výstupem je zpráva oznamující, zda je adresa v pořádku nebo není, popřípadě zda je špatně zadané číslo nebo PSČ.

5.2. Implementace v Lotus Domino Designerovi

5.2.1. Vytvoření webové služby

Poskytovatele webové služby je možné v Lotus Domino Designerovi napsat buď v LotusScriptu nebo v Javě. Po výběru programovacího jazyka v průvodci vytvoření nové webové služby je potřeba zadat název (Name), popřípadě Alias webové služby a poznámku (Comment) identifikující návrhový prvek v rámci aplikace. Toto označení se stane součástí cílové adresy URL volání této webové služby.



Obrázek 5.2 Vlastnosti webové služby (Zdroj: Lotus Domino Designer, IBM, 2010)

Jako další krok je potřeba založit novou veřejnou třídu (Class), jež ponese jméno webové služby a bude obsahovat všechny funkce, které webová služba poskytuje. Jméno této třídy je potřeba ručně přepsat do pole *PortType Class* ve vlastnostech webové služby, viz obr. 5.2. Tím určíme, která ze tříd (je-li jich v deklaracích více) obsahuje webovou službu. Od této chvíle můžeme webovou funkci uložit a nechat zobrazit WSDL soubor, jenž je automaticky generovaný vývojovým prostředím.

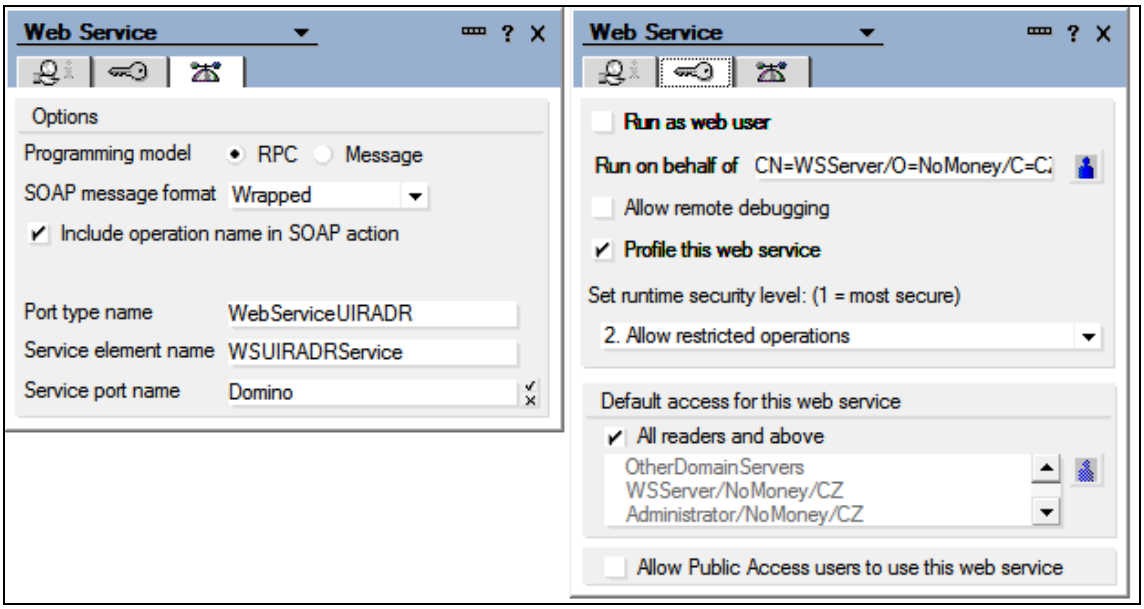
V příloze 8.4 je LotusScriptový kód webové služby. Obsahuje třídu WebServiceUIRADR a pomocnou třídu pro předání výstupní proměnné. Uvnitř hlavní třídy jsou definovány funkce, které webová služba poskytuje.

Lotus Notes podporují pouze verzi WSDL 1.1, čili struktura vygenerovaného WSDL odpovídá obrázku 3.12. Název třídy koresponduje s elementem portType ve výsledném WSDL soubouru: <portType name="**WebServiceUIRADR**">. Uvnitř tohoto elementu jsou v elementech <operation> uvedeny všechny tři poskytované funkce. Element portType tak

v podstatě tvoří ekvivalent knihovny funkcí. Viz obr. 8.5 s kompletním WSDL vygenerovaným pro tuto webovou službu.

Aby se předešlo nechtěné změně rozhraní webové služby při úpravě vlastního kódu webové služby, je možné nastavit automatickou kontrolu modifikace WSDL. Toto nastavení je doporučeno povolit až po dokončení a odladění webové služby. Prostředí pak neumožní uložit změny, které by měly za důsledek změnu WSDL a tím znehodnocení případných konzumentů již využívané webové služby.

Další vlastnosti ovlivňující tvar výsledného WSDL jsou na obr. 5.3 (vlevo). Nastavuje se typ SOAP zprávy a *name* atributy elementů <service name="**WSUIRADRService**">, <port name="**Domino**" ...> a atribut *type* elementu <binding type="**WebServiceUIRADR**" ...>.



Obrázek 5.3 Nastavení vlastností webové služby (Zdroj: Lotus Domino Designer, IBM, 2010)

Dále je možné definovat prava, s jakými bude služba na serveru spouštěna, viz obr. 5.3 (vpravo). Účet, pod kterým služba poběží, musí mít v konfiguraci serveru Lotus Domino nastavené oprávnění přistupovat k systémovým prostředkům a tedy i k datovému zdroji definovanému v operačním systému.

Konzument webové služby může být opět napsán pomocí jazyků LotusScript nebo Java. Pro webové aplikace je možné použít systémové objekty volané z JavaScriptu. Prostředí dokáže automaticky analyzovat zadané WSDL a předpřipravit třídu s metodami pro volání jednotlivých funkcí webové služby. Kód konzumentské aplikace, je uveden v příloze 8.6. Jednotlivé metody, jsou pak volány z jiných částí aplikace.

6. Závěr

Cílem této práce bylo charakterizovat technologii webových služeb. Webové služby řeší zásadní problém při komunikaci dvou a více počítačových systémů mezi sebou. Hlavní předností narozdíl od jiných mechanismů pro výměnu dat je nezávislost na programovacím jazyce a operačním systému. Všechny části webových služeb jsou standardizovány. Standardy byly vytvořeny za spolupráce předních světových softwarových firem a díky tomu jsou podporovány mnoha softwarovými produkty.

Vzhledem k existenci více verzí jednotlivých standardů se podpora jednotlivých součástí webových služeb v různých systémech liší. V sytému Lotus Notes a Domino je podpora webových služeb ve srovnání s ostatními vývojovými platformami na nižší úrovni. Lotus Domino Designer podporuje pouze starší stadardy pro definici poskytovaných služeb WSDL 1.1, nepodporuje rozšíření popisu WS-Policy, adresování WS-Addressing, nebo zabezpečení WS-Security. Webové služby lze zabezpečit pouze využítím nástrojů přenosového protokolu HTTP případně jeho šifrované podoby HTTPS.

Poskytovatele i konzumenty webových služeb lze na serveru Lotus Domino implementovat také pomocí externích javových aplikací a tak dosáhnout funkcionality srovnatelné s Axis2. Avšak toto řešení je velice pracné a v praxi špatně spravovatelné.

Širší uplatnění najdou webové služby na této platformě hlavně jako nástroj pro integraci s jinými systémy uvnitř podnikové sítě. Samotná implementace webové služby pomocí vlastních nástrojů aplikace Lotus Domino Designer je velmi rychlá a svou jednoduchostí srovnatelná s implementací webových služeb v rozhraní .Net Framework. Programovací prostředí se samo stará o vygenerování WSDL a jeho zpřístupnění. Datové typy parametrů jsou automaticky mapovány do podoby vyhovující standardu.

Dílčím cílem této práce byl i návrh a implementace praktického využití webové služby v Lotus Notes a Dominu. Byl vytvořen aplikační modul v Lotus Notes, který využívá webové služby pro doplňování a validaci poštovních adres zadávaných do vlastní aplikace. Dále byla vytvořena aplikace poskytující tuto službu na serveru Lotus Domino, která byla propojena s relačním databázovým zdrojem UIR-ADR, který lze aktualizovat pomocí pravidelně vydávaných změnových souborů. Navržené řešení tak odstraňuje zásadní nevýhody původního řešení s úplností a aktuálností podkladových dat.

7. Seznam použitých zdrojů

Tištěné dokumenty:

Benz, B., Oliver, R. *Lotus Notes a Domino mistrovství v programování*. 1. vyd. Brno: CP Books, a.s., 2005. 966 s. ISBN 80-215-0750-7.

Ethan, C. *Web Services Essentials*. 1st ed. O'Reilly, 2002. 304 p. ISBN 0-596-00224-6.

Newcomer, E., Lomow, G. *Understanding SOA with Web Services.* 1st ed. Addison Wesley Professional. 2004. 480 p. ISBN 0-321-18086-0.

Weerawarana, C., Cerbera F., et al. *Web Services Platform Architecture.* 1st ed. New Jersey: Prentice Hall PTR. 2005. 456 p. ISBN 0-13-148874-0.

Žák, M. *XML (začínáme programovat)*. Praha: Grada. 2003. 204 s. ISBN 80-247-0565-6.

Internetové zdroje:

Apache Software Foundation. *Apache Axis2*.[on-line]. Version 1.5.4. 2010. (HTML). [cit 2011-03-10]. Dostupné z WWW: <http://axis.apache.org/axis2/java/core/index.html>.

IBM. *Lotus Domino Designer 8.5* [on-line]. 2011. (HTML). [cit. 2011-02-12]. Dostupné z WWW: <http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp>.

MPSV. *Informace o registru UIR-ADR* [on-line]. Verze 4.2. Praha: Ministerstvo práce a sociálních věcí. 2011 [cit. 2011-1-24]. Dostupné z WWW: <http://forms.mpsv.cz/uir/popis/popis.jsp>.

OASIS. *The DocBook Schema Version 5.0: OASIS Standard* [on-line]. 2009. (HTML). [cit. 2010-12-07]. Dostupné z WWW: <http://www.docbook.org/specs/docbook-5.0-spec-os.html>.

OASIS. *RELAX NG Specification: Committee Specification* [on-line]. 2001. (HTML). [cit. 2010-12-07]. Dostupné z WWW: <http://relaxng.org/spec-20011203.html>.

W3C. *Extensible Markup Language (XML)* [on-line]. W3C Working Draft. W3C, 1996. (HTML). [cit. 2010-11-04]. Dostupné z WWW: <http://www.w3.org/TR/WD-xml-961114.html>.

W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition): W3C Recommendation* [on-line]. 2008. (XHTML). [cit. 2010-11-04]. Dostupné z WWW: <http://www.w3.org/TR/REC-xml/>.

W3C. *Extensible Markup Language (XML) 1.1 (Second Edition):W3C Recommendation* [on-line]. W3C, 2006. (XHTML). [cit. 2010-11-04]. Dostupné z WWW: <http://www.w3.org/TR/2006/REC-xml11-20060816/>.

W3C. *Simple Object Access Protocol (SOAP): W3C Note* [on-line]. 2000. (HTML). [cit. 2011-01-15]. Dostupné z WWW:<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.

W3C. *XML Schema Part 0: Primer Second Edition: W3C Recommendation* [on-line]. 2004a. (HTML). [cit. 2010-10-05]. Dostupné z WWW: <http://www.w3.org/TR/xmlschema-0/>.

W3C. *XML Schema Part 1: Structures Second Edition: W3C Recommendation* [on-line]. 2004a. (HTML). [cit. 2010-10-05]. Dostupné z WWW: <http://www.w3.org/TR/xmlschema-1/>.

W3C. *XML Schema Part 2: Datatypes Second Edition: W3C Recommendation* [on-line]. 2004a. (HTML). [cit. 2010-10-05]. Dostupné z WWW: <http://www.w3.org/TR/xmlschema-2/>.

W3C. *XML Information Set (Second Edition): W3C Recommendation* [on-line]. 2004b. (HTML). [cit. 2011-02-26]. Dostupné z WWW: <URL:http://www.w3.org/TR/xml-infoset/>.

W3C. *SOAP Version 1.2 Part 0: Primer (Second Edition): W3C Recommendation* [on-line]. 2007a. (HTML). [cit. 2011-01-15]. Dostupné z WWW: <http://www.w3.org/TR/soap12-part0/>.

W3C. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition): W3C Recommendation* [on-line]. 2007a. (HTML). [cit. 2011-01-15]. Dostupné z WWW: <http://www.w3.org/TR/soap12-part1/>.

W3C. *SOAP Version 1.2 Part 2: Adjuncts (Second Edition): W3C Recommendation* [on-line]. 2007a. (HTML). [cit. 2011-01-15]. Dostupné z WWW: <http://www.w3.org/TR/soap12-part2/>.

W3C. *Web Services Addressing (WS-Addressing): W3C Member Submission* [on-line]. 2004c. (HTML). [cit. 2011-01-15]. Dostupné z WWW: <http://www.w3.org/Submission/ws-addressing/>.

W3C. *Web Services Description Language (WSDL) 1.1: W3C Note* [on-line]. 2001. (HTML). [cit. 2011-01-16]. Dostupné z WWW: <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>.

W3C. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language: W3C Recommendation* [on-line]. 2007b. (HTML). [cit. 2011-01-16]. Dostupné z WWW: <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>.

W3C. *Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts: W3C Recommendation* [on-line]. 2007b. (HTML). [cit. 2011-01-16]. Dostupné z WWW: <http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/>.

W3C. *XML Schema for WSDL 2.0. (Latest version).* [on-line]. 2007c. (XSD). [cit. 2011-01-16]. Dostupné z WWW: <http://www.w3.org/2007/06/wsdl/wsdl20.xsd>.

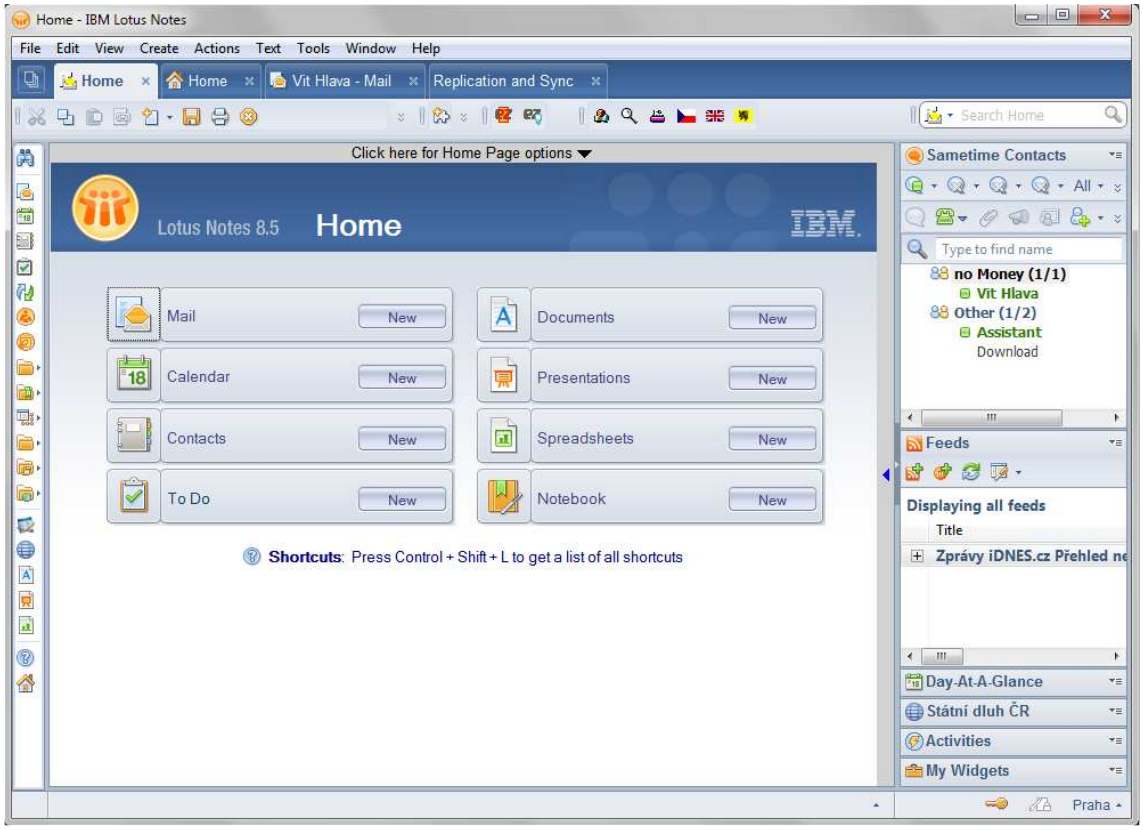
OASIS. *UDDI Version 3.0.2: UDDI Spec Technical Committee Draft* [on-line]. 2004. (HTML). [cit. 2011.02.02]. Dostupné z WWW: <http://uddi.org/pubs/uddi_v3.htm>.

SOA World Magazine. *Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort* [on-line]. 2005. (HTML). [cit. 2011.02.02]. Dostupné z WWW: <http://soa.sys-con.com/node/164624>.

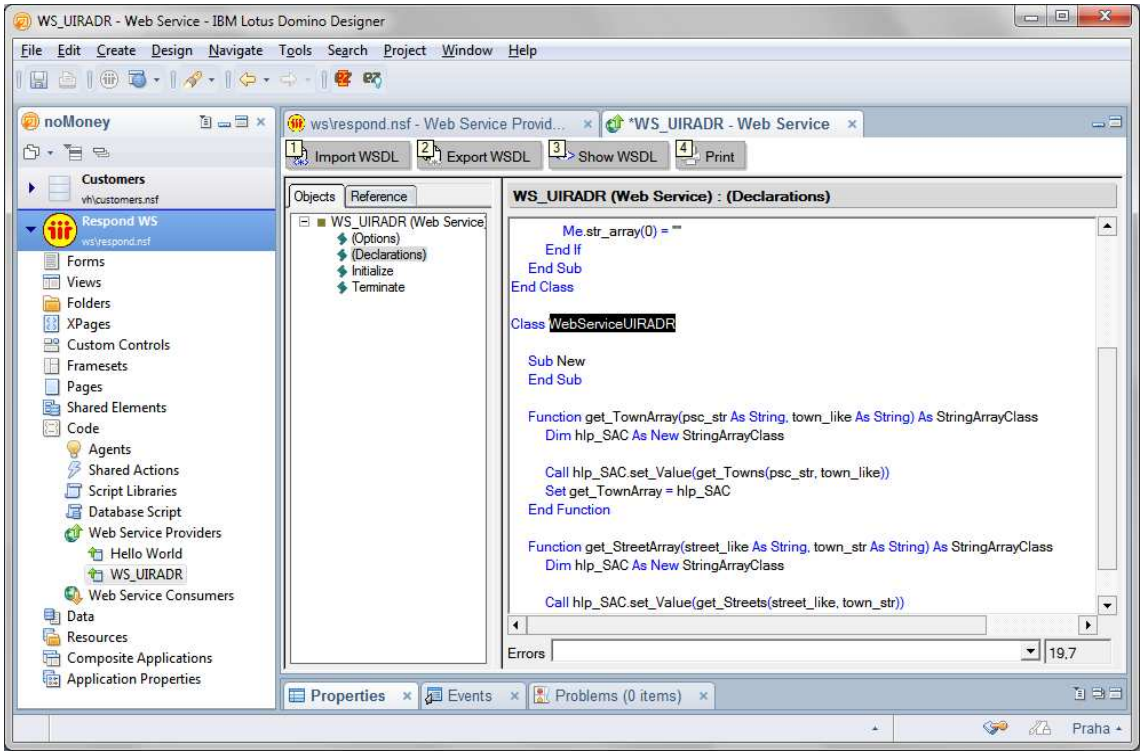
7.1. Seznam obrázků:

7.2. Seznam tabulek

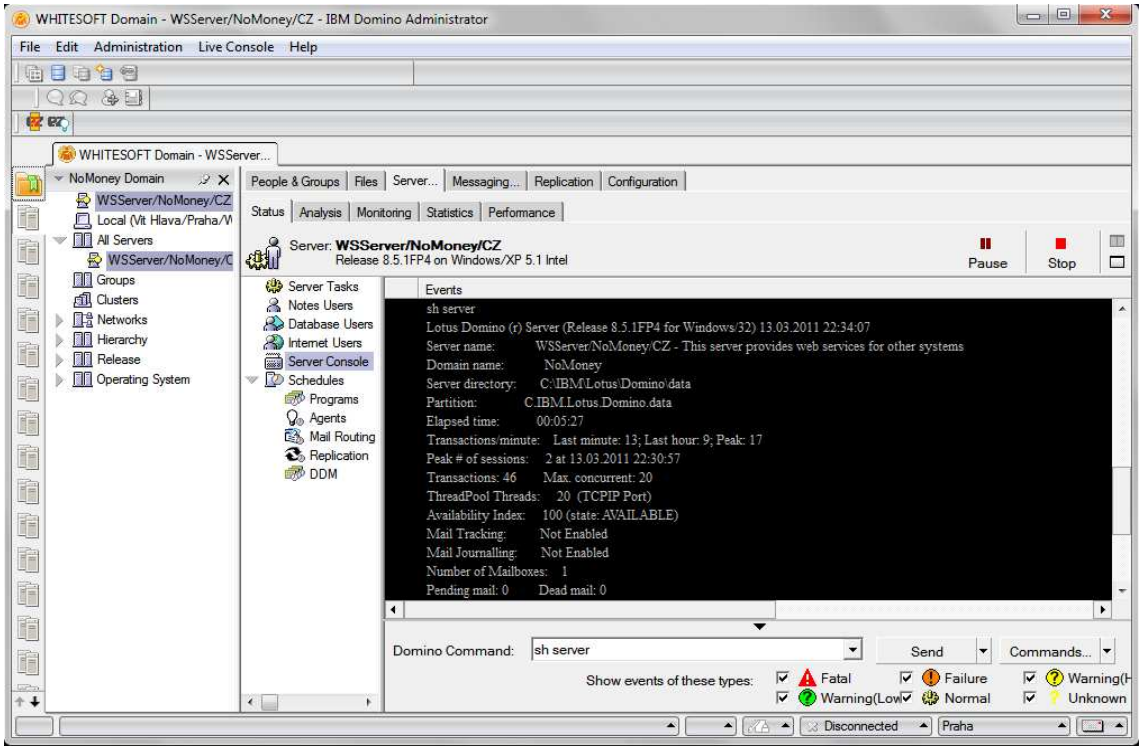
8. Přílohy



Obrázek 8.1 Prostředí Lotus Notes (Zdroj: Lotus Notes 8.5, IBM, 2010)



Obrázek 8.2 Prostředí Lotus Domino Designer (Zdroj: Lotus Notes 8.5, IBM, 2010)



Obrázek 8.3 Lotus Domino Administrator (Zdroj: Lotus Notes, IBM, 2010)

```
Class StringArrayClass

  Public str_array () As String

  Sub New
  End Sub

  Sub set_Value(source_var As Variant)
    Dim i As Integer

    If Isarray(source_var) Then
      Redim Me.str_array(Lbound(source_var) To Ubound(source_var)) As String
      For i = Lbound(source_var) To Ubound(source_var)
        Me.str_array(i) = source_var(i)
      Next
    Else
      Redim Me.str_array(0) As String
      Me.str_array(0) = ""
    End If
  End Sub

End Class

Class WebServiceUIRADR

  Sub New
  End Sub

  Function get_TownArray(psc_str As String, town_like As String) As StringArrayClass
    Dim hlp_SAC As New StringArrayClass

    Call hlp_SAC.set_Value(get_Towns(psc_str, town_like))
    Set get_TownArray = hlp_SAC
  End Function

  Function get_StreetArray(street_like As String, town_str As String) As StringArrayClass
    Dim hlp_SAC As New StringArrayClass

    Call hlp_SAC.set_Value(get_Streets(street_like, town_str))
    Set get_StreetArray = hlp_SAC
  End Function

  Function validate_Address(psc_str As String, town_str As String, street_str As String, _
              no_str As String) As String
    validate_Address = validate_PostAddress(psc_str, town_str, street_str, no_str)
  End Function

End Class
```

Obrázek 8.4 Kód poskytovatele webové služby v LotusScriptu (Zdroj: Autor, 2011)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions  targetNamespace="urn:DefaultNamespace"
              xmlns="http://schemas.xmlsoap.org/wsdl/"
              xmlns:apachesoap="http://xml.apache.org/xml-soap"
              xmlns:impl="urn:DefaultNamespace"
              xmlns:intf="urn:DefaultNamespace"
              xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
              xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema targetNamespace="urn:DefaultNamespace"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="GET_TOWNARRAY">
    <complexType>
     <sequence>
      <element name="TOWN_LIKE" type="xsd:string"/>
      <element name="PSC_STR" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="GET_TOWNARRAYResponse">
    <complexType>
     <sequence>
      <element name="GET_TOWNARRAYReturn"
               type="impl:STRINGARRAYCLASS"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="STRINGARRAYCLASS">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0"
              name="STR_ARRAY" type="xsd:string"/>
    </sequence>
   </complexType>
   <element name="GET_STREETARRAY">
    <complexType>
     <sequence>
      <element name="STREET_LIKE" type="xsd:string"/>
      <element name="TOWN_STR" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="GET_STREETARRAYResponse">
    <complexType>
     <sequence>
      <element name="GET_STREETARRAYReturn"
               type="impl:STRINGARRAYCLASS"/>
     </sequence>
    </complexType>
   </element>
```

```
 <element name="VALIDATE_ADDRESS">
  <complexType>
   <sequence>
    <element name="PSC_STR" type="xsd:string"/>
    <element name="TOWN_STR" type="xsd:string"/>
    <element name="STREET_STR" type="xsd:string"/>
    <element name="NO_STR" type="xsd:string"/>
   </sequence>
  </complexType>
 </element>
 <element name="VALIDATE_ADDRESSResponse">
  <complexType>
   <sequence>
    <element name="VALIDATE_ADDRESSReturn" type="xsd:string"/>
   </sequence>
  </complexType>
 </element>
</schema>
</wsdl:types>
<message name="GET_STREETARRAYRequest">
 <part element="impl:GET_STREETARRAY" name="parameters"/>
</message>
<message name="GET_TOWNARRAYRequest">
 <part element="impl:GET_TOWNARRAY" name="parameters"/>
</message>
<message name="VALIDATE_ADDRESSRequest">
 <part element="impl:VALIDATE_ADDRESS" name="parameters"/>
</message>
<message name="GET_TOWNARRAYResponse">
 <part element="impl:GET_TOWNARRAYResponse" name="parameters"/>
</message>
<message name="GET_STREETARRAYResponse">
 <part element="impl:GET_STREETARRAYResponse" name="parameters"/>
</message>
<message name="VALIDATE_ADDRESSResponse">
 <part element="impl:VALIDATE_ADDRESSResponse" name="parameters"/>
</message>
<portType name="WebServiceUIRADR">
 <operation name="GET_TOWNARRAY">
  <input message="impl:GET_TOWNARRAYRequest"
         name="GET_TOWNARRAYRequest"/>
  <output message="impl:GET_TOWNARRAYResponse"
          name="GET_TOWNARRAYResponse"/>
 </operation>
 <operation name="GET_STREETARRAY">
  <input message="impl:GET_STREETARRAYRequest"
         name="GET_STREETARRAYRequest"/>
  <output message="impl:GET_STREETARRAYResponse"
          name="GET_STREETARRAYResponse"/>
 </operation>
```

```
 <operation name="VALIDATE_ADDRESS">
  <input message="impl:VALIDATE_ADDRESSRequest"
         name="VALIDATE_ADDRESSRequest"/>
  <output message="impl:VALIDATE_ADDRESSResponse"
          name="VALIDATE_ADDRESSResponse"/>
 </operation>
</portType>
<binding name="DominoSoapBinding" type="impl:WebServiceUIRADR">
 <wsdlsoap:binding style="document"
                   transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="GET_TOWNARRAY">
  <wsdlsoap:operation soapAction="GET_TOWNARRAY"/>
  <input name="GET_TOWNARRAYRequest">
   <wsdlsoap:body use="literal"/>
  </input>
  <output name="GET_TOWNARRAYResponse">
   <wsdlsoap:body use="literal"/>
  </output>
 </operation>
 <operation name="GET_STREETARRAY">
  <wsdlsoap:operation soapAction="GET_STREETARRAY"/>
  <input name="GET_STREETARRAYRequest">
   <wsdlsoap:body use="literal"/>
  </input>
  <output name="GET_STREETARRAYResponse">
   <wsdlsoap:body use="literal"/>
  </output>
 </operation>
 <operation name="VALIDATE_ADDRESS">
  <wsdlsoap:operation soapAction="VALIDATE_ADDRESS"/>
  <input name="VALIDATE_ADDRESSRequest">
   <wsdlsoap:body use="literal"/>
  </input>
  <output name="VALIDATE_ADDRESSResponse">
   <wsdlsoap:body use="literal"/>
  </output>
 </operation>
</binding>
<service name="WSUIRADRService">
 <port binding="impl:DominoSoapBinding" name="Domino">
  <wsdlsoap:address location="http://localhost"/>
 </port>
</service>
</definitions>
```

Obrázek 8.5 Vygenerované WSDL (Zdroj: Lotus Domino Designerem, IBM, 2010)

```
%INCLUDE "lsxsd.lss"
Class STRINGARRAYCLASS As XSD_ANYTYPE

  Public STR_ARRAY() As String

  Sub NEW
  End Sub

End Class

Class WebServiceUIRADR As PortTypeBase

  Sub NEW
    Call Service.Initialize ("UrnDefaultNamespaceWSUIRADRService", _
       "WSUIRADRService.Domino",_
       "http://localhost:80/ws/respond.nsf/WS_UIRADR?OpenWebService", _
       "WebServiceUIRADR")
  End Sub

  Function GET_TOWNARRAY(TOWN_LIKE As String, PSC_STR As String) _
              As STRINGARRAYCLASS
    Set GET_TOWNARRAY = Service.Invoke("GET_TOWNARRAY", _
              TOWN_LIKE, PSC_STR)
  End Function

  Function GET_STREETARRAY(STREET_LIKE As String, TOWN_STR As String) _
             As STRINGARRAYCLASS
    Set GET_STREETARRAY = Service.Invoke("GET_STREETARRAY", _
             STREET_LIKE, TOWN_STR)
  End Function

  Function VALIDATE_ADDRESS(PSC_STR As String, TOWN_STR As String, _
             STREET_STR As String, NO_STR As String) As String
    Let VALIDATE_ADDRESS = Service.Invoke("VALIDATE_ADDRESS", _
             PSC_STR, TOWN_STR, STREET_STR, NO_STR)
  End Function

End Class
```

Obrázek 8.6 Kód konzumenta webové službyLotusScriptu (Zdroj: Autor, 2011)