Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Transport konfiguračních dat pomocí webových služeb

Lubomír Mokrý

Bakalářská práce

2008

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: 2007/2008

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Lubomír MOKRÝ**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Transport konfiguračních dat pomocí webových služeb**

Zásady pro vypracování:

Cílem bakalářské práce je provést analýzu, návrh řešení a implementaci systému pro přenos konfiguračních dat pomocí technologie webových služeb. V teoretické části budou shrnuty základní mechanizmy a principy používané při práci s webovými službami. Dále bude teoretická část obsahovat analýzu problému přenosu konfiguračních dat informačního systému WinFAS a návrh řešení. Implementační část bude obsahovat dvě aplikace. První aplikace bude řešit serverovou část přenosu dat. Úkolem serverové části je zaslat je klientské části na základě přijatého požadavku odpovídající konfigurační data. Klientská část má za úkol na základě vnějšího volání odeslat požadavek na serverovou část a výsledek předat nadřízené aplikaci k dalšímu zpracování.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

[1]WELLING, Luke, THOMSON, Laura. PHP a MySQL rozvoj webových aplikací. 3. vyd. Praha : SoftPress, 2005. ISBN 80-86497-83-6

[2]BRADLEY, Neil. XML kompletní průvodce. 1. vyd. Praha : Grada Publishing, 2000. 540 s. ISBN 80-7169-949-7

[3]Web Services Activity [online] http://www.w3.org/2002/ws/

[4]Web Services Architecture [online] http://www.w3.org/TR/ws-arch/

Vedoucí bakalářské práce: **Ing. Filip Linsbauer**

Datum zadání bakalářské práce: **30. listopadu 2007**

Termín odevzdání bakalářské práce: **16. května 2008**

Univerzita Pardubice Česká republika

doc. Ing. Stanislav Karamazov, Dr.
děkan

V Pardubicích dne 29. dubna 2008

Poděkování:

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce, panu Ing. Filipovi Linsbauerovi, za odborné vedení a cenné rady, které mi poskytl v průběhu vypracování mé práce.

Souhrn

Cílem této bakalářské práce je návrh řešení a implementace systému pro přenos konfiguračních dat pomocí technologie webových služeb. V první části jsou popsány základní mechanizmy a principy použité při práci s webovými službami. Dále je provedena analýza přenosu konfiguračních dat do informačního systému WinFAS za pomoci webových služeb. Poslední část se zabývá samotnou realizací projektu a obsahuje popis vytvořené webové služby pro přenos konfiguračních dat.

Klíčová slova

webové služby, SOAP, XML, WSDL, WinFAS

Title

Transport configuration data assist in web service

Abstrakt

Objective of this work is to design and implement a system for transfering of configuration data using Web services. The first section describes the basic principles and technologies related to Web services. Second part contains analysis of the transfer of configuration data into an information system WinFAS via Web services. Last part deals with the realization of the project itself and includes a description of implemented Web service for the transfer of configuration data.

Keywords

web servises, SOAP, XML, WSDL, WinFas

Obsah

Seznam obrázků

Seznam zkratek

4GL	programovací jazyk 4. generace
j2EE™	Java 2 Enterprise Editio
RPC	Repote Procedure Call
XML	eXtensible Markup Language
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
WWW	World Wide Web
HTML	HyperText Markup Language
JavaScript	multiplatformní, objektově orientovaný skriptovací jazyk
Java	objektově orientovaný programovací jazyk
PHP	skriptovací programovací jazyk
W3C	World Wide Web Consortium
SSL	Secure Socket Layer
C++	objektově orientovaný programovací jazyk
SMTP	Simple Mail Transfer Protocol
FTP	File Transport Protocol
UDDI	Universal Description, Discovery and Integrarion
WSLI	Web Services Inspection Language
HTML	HyperText Mark Language
SGML	Stardard Generalized Markup Language
CSS	kaskádové styly
XSL	eXtensible Stylesheet Language
PDF	Portable Document Format
UTF-8	UCS Transformation Format
DTD	Document Type Definition

1. Úvod

Organizační kancelář, která je zadavatelem této bakalářské práce, je firma vyvíjející informační systém WinFAS. Tato firma má dlouholeté zkušenosti s vývojem software. Její informační systém WinFAS je komplexní systém zajišťující sběr, udržování, zpracování a poskytování business informací a dat. V rámci systému WinFAS, vznikl požadavek na zpřístupnění souborů konfiguračních dat, poskytovaných Organizační kanceláří koncovým uživatelům systému. Technologie webových služeb byla vybrána jako ideální kandidát na realizaci tohoto úkolu.

Úkolem této práce je návrh a implementace webové služby. Tato služba bude mít za cíl přenést konfigurační data, které bude klient vyžadovat, ze serveru Organizační kanceláře do počítače klienta. Data umožní klientovi nakonfigurovat informační systém WinFAS pro danou činnost. Systém bude rozdělen na dvě části. První z nich je serverová část. Serverová část bude integrována do stávající struktury webu Organizační kanceláře. Bude využívat jako datovou základnu existující databázi a data z ní bude načítat pomocí služeb implementovaných pro tyto účely předchozími projekty. Druhá klientská část musí umožňovat načítání příslušných dat ze serveru jak automaticky (přímo z WinFASu), tak i ručně pro počítače bez přímého přístupu k internetu. Práce je rozdělena do tří částí. V části první je popsán informační systém WinFAS a historie Organizační kanceláře. Také jsou zde sumarizovány technologie, které webové služby využívají a jsou zde vysvětleny pojmy vyskytující se v této problematice. V následující části je provedena analýza a obecný návrh sestavované webové služby. V poslední části je pak popsán způsob implementace jedné serverové a tří klientských částí, kde každá ze tří klientských částí je implementována v jiném vývojovém prostředí.

2. Rozbor problematiky

2.1. Organizační kancelář, s.r.o.

2.1.1. Historie

Organizační kancelář, s.r.o. je následníkem Aplikační skupiny, která vznikla v roce 1976 na Okresní zemědělské správě. Tato skupina zajišťovala výpočetní techniku pro zemědělské podniky v okrese Žďár nad Sázavou.

Další její činností byl sběr dat, která byla následně zpracovávána na sálovém počítači EC1033 v Brně. Jednotlivé podniky měly omezenou možnost předzpracování těchto dat na 8-bitových počítačích MIDO-16. Vývoj programů (agendy hrubých mezd, zvířat, atd.) pro tyto počítače zajišťovala právě ona Aplikační skupina. V roce 1990 přešly všechny tyto agendy na platformu PC a program ASŘ ZpoK.

Po zrušení Aplikační skupiny převzala tyto úkoly Organizační kancelář. Protože program ASŘ ZpoK neumožňoval zpracovávat více firem, rozhodla se v roce 1992 Organizační kancelář, vyvinout vlastní informační systém FAS pod prostředím DOS. Tento systém začala nabízet i nezemědělským zákazníkům a to vedlo k rozšíření funkčnosti programu o další agendy.

V roce 2001 uvedla Organizační kancelář na trh nový informační systém WinFAS, který byl vyvinut pod prostředím Windows. Pro vývoj tohoto systému se rozhodla použít nástroje společnosti Sybase.

2.1.2. Informační systém WinFAS

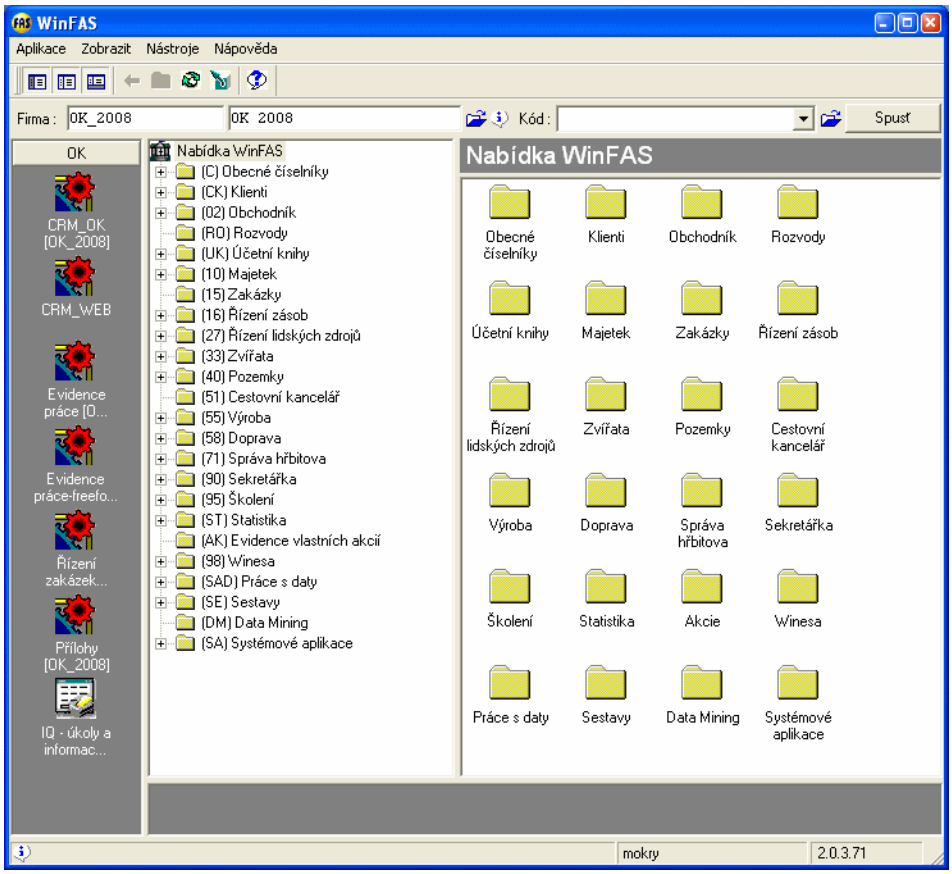
Informační systém WinFAS je systém zajišťující sběr, udržování, zpracování a poskytování informací a dat. Je navržen pro podniky, které díky němu můžou kontrolovat svůj ekonomický růst, automatizovat své výrobní procesy nebo rozšiřovat své podnikání do nových různorodých oblastí.

Díky dlouholetému vývoji tohoto informačního systému pro různé klienty (obchodní společnosti, výrobní, zemědělské a stavební podniky, účetní firmy apod.) zasahuje informační systém WinFAS do široké oblasti působnosti. Hlavními agendami, které informační systém nabízí, jsou:

- fakturace (obchodník)
- finanční modul
- účetnictví
- evidence klientů
- evidence práce
- mzdy
- personalistika
- řízení zásob
- výroba
- doprava
- majetek
- zvířata
- pozemky

Informační systém WinFAS je modulární systém, což zaručuje klientům využívajících jen části jeho služeb, že je nebude zatěžovat jeho komplexnost a rozsáhlost. Na druhou stranu jsou jednotlivé moduly vzájemně provázány a spolupracují spolu. Díky tomu nemusí klienti využívající služby tohoto informačního systému pořizovat vstupní data do každého modulu zvlášť, ale jednotlivé moduly si tyto data dokáží vzájemně předávat a pracovat s nimi.

Veškerá data udržovaná v tomto informačním systému je možné prohlížet, tisknout či exportovat do různých formátů (XLS, DOC, CSV), databáze (dBase) a dalších aplikací. Data lze také exportovat do výstupních sestav, které lze odesílat emailem nebo ukládat do formátů PDF, EDI. Nad uloženými daty lze také spouštět speciální sestavy, tzv. „IQ-sestavy“, kde si sám uživatel zvolí, která výstupní data jsou pro něj důležitá. Data jsou také zabezpečena přístupovými právy na úrovni jednotlivých aplikací a funkcí. Tyto práva jsou uchována i při archivaci dat. Správnost pořízení dat do informačního systému kontrolují programové i uživatelsky definované kontroly.



Obr. 1) Informační systém WinFAS - zdroj vlastní

Pro vývoj tohoto informačního systému jsou použity nástroje firmy Sybase. Pro ukládání dat je použita databáze SQL Anywhere. Její výhodou je, že může pracovat na jednom počítači, malé síti, ale i víceprocesorových serverech rozsáhlých podnikových sítí. Klientská část systému je programována v 4GL programovacím jazyce PowerBuilder. Programování v PowerBuilderu je realizováno objektově orientovaným jazykem syntakticky podobným Basicu. S počáteční implementací, informačního systému WinFAS v těchto vývojových nástrojích, pomohla Organizační kanceláři jedna z největších současných českých softwarových společností Unicorn, a.s.

Informační systém WinFAS lze individuálně konfigurovat dle jednotlivých potřeb uživatelů pomocí šablon, maker a řídících číselníků. Veškerá uvedená nastavení lze uložit a opětovně použít.

2.1.3. Konfigurační data (šablony)

Šablona je vlastně XML soubor, který nese informace o tom jak mají být nastaveny jednotlivé položky určité části systému. Jako každý XML soubor obsahuje i šablona na prvním řádku XML deklaraci a informaci o použitém kódování. Je zde využito kódování UTF-8. Kořenovým elementem je pak vždy element <winfas_sab>, který říká, že se jedná právě o šablonu IS WinFAS. Poté následuje element <c0063okno_> s informací, pro kterou část systému je šablona určena. Tento kód se kontroluje při nahrávání šablony. Import šablony je povolen pouze ze stejného místa, kde byl proveden export. Následující elementy již nesou informace o nastavení jednotlivých položek. Ukázka IQ šablony je uvedena v příloze č. A.

Pomocí šablon lze nastavovat aplikace, pořízení, či IQ sestavy. V pořízení, kde uživatel zadává vstupní data, lze pomocí šablony např. nastavit, která položka daného pořízení má či nemá být editovatelná, případně daným polím přednastavit určitou hodnotu. U IQ sestavy lze pomocí šablony říci např. která data budou obsažena v konečné výstupní sestavě. V aplikaci se dá do šablony uložit např. které šablony budou použity pro pořízení dat.

2.2. Webové služby (Webservices)

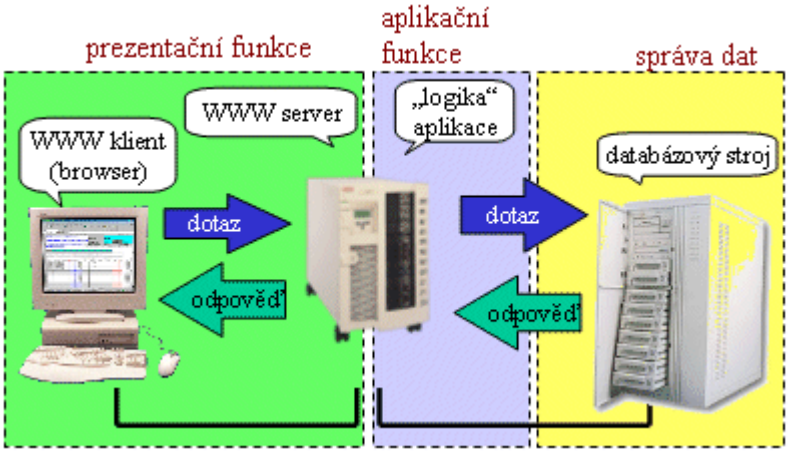
Webová služba je podle definice W3C řešení, jak spolu mohou komunikovat a vyměňovat si informace aplikace přes Internet. Jako protokol pro práci s webovými službami se nejčastěji používá protokol SOAP (viz. kapitola 2.3). K webové službě s protokolem SOAP lze také připojit tzv. WSDL dokument (viz. kapitola 2.4), který popisuje veškerou funkčnost webové služby. Nejpoužívanějším komunikačním kanálem pro přenos informací pro potřeby webových služeb je protokol HTTP.

Velkou výhodou webových služeb je, že vůbec nezáleží na tom v jakém programovacím jazyce, je webová služba napsána nebo z jakého jazyka je volána. Nezáleží také, na jaké platformě služba či klient běží. Jednou z předních vlastností webových služeb je tak i to, že mění současný internet. Většina internetových serverů je nyní převážně souborem HTML stránek, které jsou srozumitelné pouze lidem. Webové služby tento soubor rozšiřují o soubory stránek čitelných pro programy na zcela různých platformách (JavaScript, Java, PHP, mobilní telefony). Díky tomu mohou spolu snadno komunikovat.

Mohlo by se říci, že webové služby fungují na architektuře klient/server, ale je to spíše další vývojové stádium této architektury. Klient byl dříve definován jako program, který přistupuje k vzdálenému serveru, a ten mu poskytuje určitou službu. Výsledky této služby poté klient pouze zobrazí uživateli. Klient webové služby výsledek vrácený serverem pouze nezobrazí, ale dokáže ho díky vlastní inteligenci dále zpracovávat.

Serverem byl nazýván proces nebo systém, který poskytuje nějakou službu dalším programům. Poskytování služby je realizováno pomocí síťových protokolů, například protokolu HTTP. Serverem byl také označován samotný počítač (hardware) vykonávající tyto činnosti. U serveru webové služby však přestává být tato fyzická podstata zřetelná. Vše se zde soustřeďuje na schopnost poskytovat určité definované služby. Kvůli této skutečnosti se u webových služeb někdy hovoří o modelu klient/služba, místo modelu klient/server.

Názorným příkladem může být vyhledávací služba Google. Ta za použití architektury klient/server poskytovala službu pouze pomocí webových stránek. Uživatel použil vyhledávací formulář skrze svůj internetový prohlížeč (browser), kde zadal hledané parametry. Zpět dostal odpověď od serveru opět v podobě WWW stránky, kde se mu zobrazil seznam odkazů na WWW stránky týkající se hledaných parametrů. Tuto stránku již však nemohl výrazně upravovat (viz obrázek 2.). V případě webových služeb si však může uživatel vytvořit úplně jinou aplikaci. Ta si na základě zadaných parametrů vyžádá požadovaná data od služby Google jako své vstupy. Ty si může dále zpracovávat a zobrazovat dle svých představ.



Obr. 2) Třívrstvá architektura klient/server v prostředí webu - zdroj [1]

U webových služeb se také významně změnil vztah mezi klientem a službou. Dříve byla tato vazba daleko pevnější a měla statický charakter. V případě webových služeb je však tato vazba daleko volnější a má dynamický charakter. Klienti (aplikace) a služby existují nezávisle na sobě. Komunikují a spolupracují spolu teprve na základě momentální potřeby. Pro správnou komunikaci je důležité, aby oba (klient i server) dodržovali společné standardy a komunikační protokoly. Tyto standardy vytváří organizace W3C.

2.2.1. Bezpečnost webových služeb[2]

Webové služby bezpečnost moc neřeší. Existují dva způsoby. Bezpečnost na úrovni přenosu (Transport level secutiy) a bezpečnost na úrovni XML zpráv (Message level security).

Bezpečnost na úrovni přenosu se zajišťuje použitím SSL pod HTTP, tj. HTTPS protokolem pro přenos SOAP zpráv. Nevýhoda této metody je, že bezpečnost je zajištěna pouze mezi komunikujícími konci. Nelze zpětně dokázat, že protistrana poslala danou zprávu. U vícevrstvých aplikací nelze zajistit, aby zpráva nebyla změněna mezilehlými vrstvami.

Druhá metoda zabezpečení se snaží využít standardy XML Encryption a XML Signature pro šifrování a podepisování SOAP zpráv, jelikož SOAP je vlastně XML dokument. Standardizační organizace OASIS vydala specifikaci WS-Security, která říká, jak podepisování a šifrování na úrovni SOAP zpráv používat. Bezpečnost na úrovni zpráv má opačné výhody než bezpečnost na úrovni přenosu. Umožňuje

podepisovat zprávy i při vícevrstvých aplikacích. Rychlost zpracování je však značně nižší než při použití SSL. Další nevýhodou je, že toto zabezpečení není zatím důkladně odzkoušené.

2.2.2. Příklad implementace webové služby

Ukázka jednoduché webové služby poskytující aktuální datum. Klientská i serverová část je napsána v programovacím jazyce PHP. Níže uvedený příklad používá extenzi SOAP, která je součástí instalace PHP verze 5.

Klient

```
//příprava parametrů klienta do pole
//location určuje  kde se daná služba nachází
//uri určuje identifikátor služby
$opts= array( 'location' => 'http://192.168.1.51/soap/','uri' =>
'urn:pc_SOAP_return_time');
//vytvoření instance třídy SOAPClient s připravenými parametry
$client = new SOAPClient(null, $opts);
// zavolání webové služby a uložení výsledku do proměnné
$result = $client->__soapCall('return_time', array());
//vypsání výsledku
echo 'Vraceny cas'.$result;
```

Server

```
// vlastní výkonná funkce pro službu
function return_time()
//vracíme pouze aktuální datum ve formátu rok-měsíc-den
{return date("Y-m-d");}
// inicializace serverové části vytvořením instance třídy SOAPServer s daným
názvem
$server = new SOAPServer (null, array ('uri'=>'urn:pc_SOAP_return_time'));
```

```
// registrace výkonné funkce k instanci serveru
$server->addfunction('return_time');
//spuštění instance serveru
$server->handle();
```

2.3. SOAP

Simple Object Access Protocol (jednoduchý protokol pro přístup k objektům) je, jak již z názvu vyplívá, protokol pro posílání zpráv XML a je základem pro webové služby. Je založený na HTTP a XML, což mu zajišťuje nezávislost na platformě. Ostatní standardy jako WSDL a registr UDDI vznikly až po něm a rozšiřují jeho možnosti. SOAP umožňuje zasílání zpráv mezi dvěma aplikacemi a pracuje na principu peer-to-peer. Zpráva je jednosměrný přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv můžeme pomocí SOAPu snadno implementovat běžné komunikační scénáře. [3]

Prvotnímu vývoji protokolu SOAP se začaly věnovat v roce 1998 firmy Microsoft, DevelopMentor a UserLand, které mu daly i jeho název. Protokol navazoval na o rok mladší a jednodušší protokol XML-RPC. Posléze se k nim připojilo mnoho dalších firem včetně IBM. Jiné firmy vyvinuly podobné protokoly jako např. WDD nebo XMI. W3C konsorcium vzalo SOAP 1.1 na vědomí v průběhu roku 2000 a posléze ustavilo pracovní skupinu (XML Protocol Workong Group) pro vývoj sjednocujícího protokolu. V prosinci roku 2002 vznikl protokol SOAP 1.2, který tato skupina vyprodukovala.

SOAP se nejčastěji používá k vzdálenému volání procedur RPC. První aplikace pošle v XML zprávě požadavek. Druhá aplikace tento požadavek zpracuje a opět v XML zprávě první aplikaci odpoví. Při předešlém příkladu volání bývá webová služba vyvolána druhou aplikací, tedy webovým serverem, který vyčkává na požadavky klientů. V okamžiku, kdy pomocí HTTP přijde soapová zpráva s požadavkem, spouští server webovou službu a předává jí obdržený požadavek. Výsledek služby pak server odešle klientovi, který jej zpracuje.

V současné době existuje několik různých implementací SOAP na nejrůznějších platformách od C/C++, Javu, .NET, Perl, PHP až po JavaScript v prohlížeči Mozilla.

2.3.1. Struktura zprávy

SOAP zpráva je jednoduchý XML dokument. Jeho kořenovým elementem je element s názvem Envelope (obálka). Ten dále obsahuje dva elementy, Header (hlavička), která je nepovinná a nese pomocné informace jako například identifikaci uživatele, autentizační informace (jméno, heslo) atd. Druhý elementem obsaženým v Envelope je element Body (tělo). V těle jsou obsaženy nejdůležitější informace SOAP zprávy. Jsou zde informace identifikující volanou službu, předávané parametry nebo návratové hodnoty služby. SOAP používá jmenné prostory pro identifikaci jednotlivých částí XML zprávy. Obálka, hlavička a tělo zprávy jsou součástí jmenného prostoru http://schemas.xmlsoap.org/soap/envelope/. [3] Ukázka struktury SOAP zprávy je uvedena v příkladu 2.3.3 a 2.3.4.

2.3.2. Transportní mechanismy

Původním protokolem pro přenos SOAP zpráv byl transportní protokol HTTP. Později byla však skupina protokolů, které mohou být k přenosu použity rozšířena o další přenosové protokoly, např. SMTP, FTP.

Jelikož se SOAP převážně používá pro RPC volání, využívá se pro přenos požadavku/odpovědi stejně nejčastěji právě transportní protokol HTTP. Hlavním důvodem jeho častého použití je jeho široká podpora v různých aplikacích. Navíc webovou službu lze nahrát přímo na webový server, který jednotlivé požadavky předává odpovídající webové službě ke zpracování. Výhodou je také to, že stávající síťová struktura dovoluje téměř neomezenou komunikaci na portu vyhrazeném pro HTTP (TCP port 80). Tím je možné používání webových služeb bez zásahu do konfigurace aktivních síťových prvků jako jsou například firewally.

SOAP požadavek se nachází v těle HTTP. Používá se metoda POST, která umožňuje posílání dat v těle HTTP. Požadavek musí obsahovat HTTP hlavičku SOAPAction (viz příklad 2.3.3), která identifikuje, že jde o SOAP požadavek. Tato

hlavička je využita i firewally k filtrování. Také může obsahovat URI s identifikací služby, která se má vyvolat.

2.3.3. Příklad SOAP požadavku zaslaného přes http

Ukázka požadavku byla vygenerována klientem dříve uvedené webové služby na zjištění aktuálního data.

POST /soap/ HTTP/1.1

Host: 192.168.1.51

Connection: Keep-Alive

User-Agent: PHP-SOAP/5.2.3

Content-Type: text/xml; charset=utf-8

SOAPAction: "urn:pc_SOAP_return_time#return_time"

Content-Length: 394

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelop

*xmlns:SOAP-ENV=*http://schemas.xmlsoap.org/soap/envelope/

xmlns:ns1="urn:pc_SOAP_return_time"

*xmlns:xsd=*http://www.w3.org/2001/XMLSchema

*xmlns:SOAP-ENC=*http://schemas.xmlsoap.org/soap/encoding/

SOAP-ENV:encodingStyle=

"http://schemas.xmlsoap.org/soap/encoding/">

<SOAP-ENV:Body>

<ns1:return_time/>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

2.3.4. Příklad SOAP odpovědi přenesené přes http

Zde je uvedena odpověď serveru, který byl dotazován klientem na aktuální datum.

HTTP/1.1 200 OK

Date: Thu, 08 Nov 2007 14:53:44 GMT

Server: Apache/2.2.4 (Win32) PHP/5.2.3 SVN/1.4.4 DAV/2

X-Powered-By: PHP/5.2.3

Content-Length: 534

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope

*xmlns:SOAP-ENV=*http://schemas.xmlsoap.org/soap/envelope/

xmlns:ns1="urn:pc_SOAP_return_time"

*xmlns:xsd=*http://www.w3.org/2001/XMLSchema

*xmlns:xsi=*http://www.w3.org/2001/XMLSchema-instance

*xmlns:SOAP-ENC=*http://schemas.xmlsoap.org/soap/encoding/

SOAP-ENV:encodingStyle=

"http://schemas.xmlsoap.org/soap/encoding/">

<SOAP-ENV:Body>

<ns1:return_timeResponse>

<return xsi:type="xsd:string">2007-11-08</return>

</ns1:return_timeResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

SOAP umožňuje i chybové návratové zprávy, jež jsou obdobou vyjímek v programovacích jazycích. Taková zpráva poté místo těla obsahuje element Fault. Ten pak dále obsahuje tři části. Element Faultcode, který říká kde došlo k chybě. Element Faultstring obsahující chybové hlášení srozumitelné lidem a element Detail, který může obsahovat libovolně složitě strukturované XML.

2.4. WSDL

Možnost vzdáleného volání funkcí pomocí SOAP by ztrácelo význam pokud bychom nevěděli jaké funkce s jakými parametry se dají volat a jaké jsou jejich návratové hodnoty. Tento problém řeší jazyk WSDL (Web Services Description Language). Tento jazyk je založený na XML a hojně využívá standardy XML Namespace a XML Schéma.

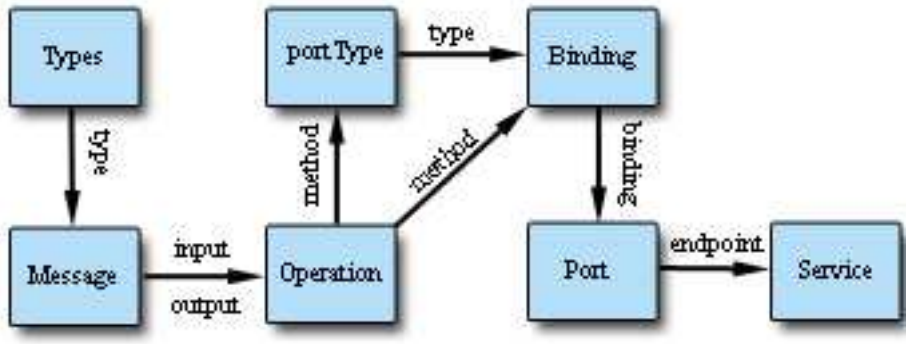
Jazyk WSDL vznikl jako společná práce firem Microsoft a IBM, které si uvědomili potřebu sjednocujícího jazyka pro popis webových služeb. WSDL 1.1 navazuje na předchozí jazyky NASSL, SCL a SDL, které sloužily k těmto účelům dříve něž vznikl právě WSDL jazyk. W3C konsorcium vzalo WSDL 1.1 na vědomí a přijalo ho jako „W3C Note“, tj. vzala jej na vědomí a zveřejnila ho na svých webových stránkách, ale bez další podpory. Po té byla vytvořena pracovní skupina, která pracovala na jeho novější verzi. Ta v 26.červnu roku 2007 vydala verzi WSDL 2.0 jako „W3C Recommendation“, tj. doporučení W3C.

WSDL dokument popisuje rozhraní webové služby na syntaktické úrovni, stejně jako .h soubory v jazyce C++ nebo interface v jazyce Java. Tedy jako seznam jmen funkcí/metod, zde zvaných operace, spolu s jmény a typy parametrů a návratových hodnot. (viz příklad 2.4.2) Na takto definované WSDL dokumenty existují automatizované nástroje, které z nich dokáží vygenerovat zástupný kód pro volání dané služby ve zvoleném programovacím jazyce. WSDL popis může kromě popisu rozhraní služby obsahovat i její umístění, tedy specifikovat protokol a adresu, kde je dosažitelná. [5]

2.4.1. Struktura WSDL dokumentů [3]

WSDL jazyk je velice obecný, aby byla zachována co nejvyšší platformová nezávislost. Skládá se zejména z následujících elementů, které jsou základními částmi každého WSDL

- types
  - obsahuje definice datových typů využitých ve zprávách
  - k definici lze teoreticky připojit libovolný typový systém, nejčastěji se však používá XML schéma
- message
  - definuje formát předávaných zpráv pomocí dříve definovaných datových typů
  - fungují jako vstupně/výstupní struktury pro operace
  - každá zpráva může být složena z několika logických částí s vlastním datovým typem
- operation
  - abstraktní definice operací, které jsou službou podporovány
  - definuje jaké má vstupy a výstupy, ty jsou popsány již existující zprávou (message)
- portType
  - souhrn operací, odpovídá interface(Java) nebo header souboru (C)
- binding
  - definuje možný způsob přístupu různými protokoly
- port
  - jeden koncový bod služby definovaný jako kombinace síťové adresy a dříve definované vazby (binding)
- service
  - sdružuje několik koncových bodů (portů) do jedné služby



Obr. 3) Struktura WSDL zdroj [6]

2.4.2. Příklad WSDL dokumentu

Dokument popisuje dříve definovanou webovou službu na vrácení aktuálního data.

```
<?xml version='1.0' encoding="utf-8">
<definitions name="Date" targetNamespace="urn:Date"
 xmlns:typens="urn:Date"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

 <message name="return_time">
 </message>
 <message name="return_timeResponse">
        <part name='response' type='xsd:char'/>
 </message>
```

```
<portType name="DatePortType">
        <operation name="return_time">
                <input message="typens:return_time"></input>
                <output message="typens:return_timeResponse"></output>
        </operation>
</portType>

<binding name="DateBinding" type="typens:DatePortType">
<soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http">
</soap:binding>
        <operation name="return_time">
                <soap:operation
                        soapAction="urn:DateAction">
                </soap:operation>
                <input>
                        <soap:body namespace="urn:Date" use="encoded"
                                encodingStyle=
                                "http://schemas.xmlsoap.org/soap/encoding/">
                        </soap:body>
                </input>
                <output>
                        <soap:body namespace="urn:Date" use="encoded"
                                encodingStyle=
                                "http://schemas.xmlsoap.org/soap/encoding/">
                        </soap:body>
                </output>
```

```
        </operation>
    </binding>

    <service name="DateService">
        <port name="DatePort" binding="typens:DateBinding">
            <soap:address location='http://localhost/wsdl/'/>
        </port>
    </service>
</definitions>
```

2.5. Mechanizmy pro nalezení služeb

Jedná se o systémy, jejichž prostřednictvím lze definovat a posléze vyhledávat nabízené služby a jejich vlastnosti. V oblasti nalézání webových služeb se používají dva mechanismy. Historicky starší UDDI (Universal Description, Discovery and Integrarion) nebo mladší WSIL (Web Services Inspection Language).

2.5.1. UDDI

Nabízí jakýsi telefonní seznam, do kterého poskytovatelé webových služeb mohou ukládat popisy služeb a uživatelé ji prohledávat. UDDI funguje jako velký adresář, který obsahuje informace o subjektech (firmách) a jimi poskytovaných službách. Tento registr rovněž pracuje jako webová služba a komunikace s ní probíhá pomocí SOAPu. UDDI registr obsahuje čtyři druhy informací:

- podnikatelské entity (business entity)
    - u každé firmy je zaznamenán název, stručný popis a kontaktní údaje
    - každé firmě mohou být přiřazeny identifikátory, které určují oblast podnikání a geografickou polohu
- informace o službě (business service)
    - zde jsou uloženy seznamy služeb, které firma poskytuje
    - každá služba je popsána a obsahuje seznam vazeb, které ukazují na technické údaje nutné pro využití služby

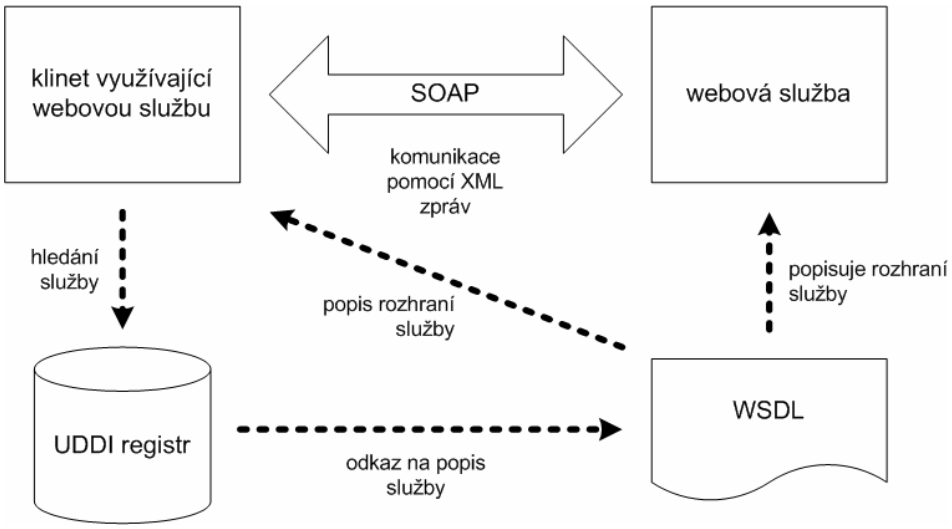
- šablony vazeb (Binding template)
    - popisují jak a kde je možné se službou komunikovat
    - každá šablona také odkazuje na typ služby, který implementuje
- typy služeb(service type)
    - definuje abstraktní službu (několik firem může nabízet stejný druh služby se stejným rozhraním i typem služby)

Práce s UDDI probíhá tak, že uživatel prohledá registr, najde si služby, které se mu budou hodit. Získá pro ně popis WSDL. UDDI ovšem nemusí obsahovat popis služby pouze ve WSDL dokumentu, ale lze do něj ukládat v jakémkoli formátu. [3]

V roce 2005 byl registr UDDI přijat pod organizaci OASIS, která vydala verzi 3.0.2 jako svůj standard. V roce 2006 však bylo zjištěno, že celé dvě třetiny záznamů v UDDI rejstříku byly neplatné až nesmyslné, protože záznamy mohl přidávat kdokoli. Proto byly v lednu roku 2006 tyto veřejné rejstříky vypnuty. Uvádí se, že UDDI se nyní používá vnitropodnikově.

2.5.2. WSLI

WSLI je jazyk pro popis služeb poskytovaných nějakou firmou či organizací. WSLI dokument je vždy nazván inspection.wsli a je umístěn v kořenovém adresáři web serveru příslušné instituce. Vyhledávání pomocí WSLI funguje jinak než u registru UDDI. Nejdříve si najdete poskytovatele služby, kterému důvěřujete a z WSLI dokumentu pak zjistíte jaké služby instituce poskytuje. Ty pak můžete využívat.



Obr. 4) Vztah technologií (SOAP, WSDL a UDDI) webových služeb [3]

2.6. XML

eXtensible Markup Language je značkovací jazyk, který byl vyvinut a standardizován W3C konsorciem. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. [7]

XML je jednoduchý textově založený formát, který můžeme editovat v libovolném textovém editoru. S XML formátem umí pracovat řada programů. Není vázaný na žádný software ani operační systém. Data z XML se dají snadno převést do spousty dalších formátů (PDF, MS Word, MS Excel,…).

XML je velice podobný jazyku HTML. To je dáno společným předchůdcem jazykem SGML. Rozdíl je v tom, že XML je podmnožina SGML, kdežto HTML je jeho aplikace, která má definovanou sadu značek. Ty mají různé významy a definují způsob zobrazování dat. HTML tedy slouží pouze k definici vzhledu pomocí striktně definovaných tagů. Pro XML si sadu značek každý vytváří sám. Tím umožňuje XML popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí. Prezentace dokumentu se definuje pomocí stylových jazyků (CSS, XSL, …). Tím je oddělen obsah od způsobu jeho prezentace. Pro jeden XML může být definováno několik možností vzhledu a pro více dokumentů XML jeden společný vzhled.

2.6.1. Struktura XML dokumentu

Každý XML dokument začíná deklarací XML:

<?xml version="1.0" encoding="utf-8"?>

Atribut version udává verzi použitého XML dokumentu. Další atribut encoding nemusí být uveden. V tom případě je kódování nastaveno na implicitní hodnotu UTF-8. Pro použití jiného kódování (windows-1250, iso-8859-2) musí být toto kódování uvedené právě za atributem encoding.

XML patří mezi značkovací jazyky. Důležité části dokumentu se tedy označují pomocí značek. V XML terminologii se jednotlivým označeným částem dokumentu říká elementy. Elementy do sebe mohou být navzájem vnořené a tím dle potřeby zachycovat strukturu informací uložených v dokumentu. [8] Kdyby například XML dokument nesl elektronický obsah knihy, skládal by se z elementů kapitola. Dále by pak v každém elementu kapitola byl vnořen element nadpis a potřebný počet elementů odstavec. Jiným příkladem struktury XML dokumentu může být uložení databázové tabulky. Dokument bude obsahovat odpovídající počet elementů jednotlivých záznamů (řádků) tabulky. Každý z těchto elementů pak bude obsahovat další elementy odpovídající jednotlivým položkám tabulky.

Jednotlivé elementy se v textu vyznačují pomocí tzv. tagů. Většině elementů odpovídají dva tagy, počáteční a ukončovací.[8]

<jmeno_elementu>Obsah elementu</jmeno_elementu>

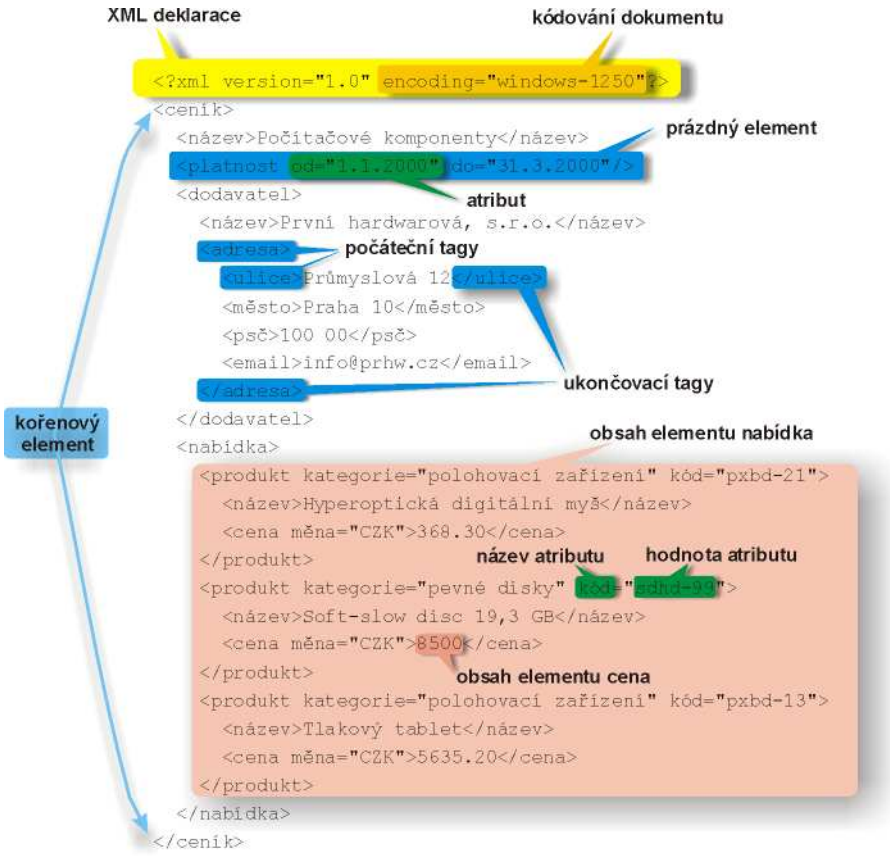
Ukázka ukazuje právě jeden element. Počáteční element je <jmeno_elementu> a ukončovací </jmeno_elementu>. Názvy tagů se zapisují mezy znaky „<“ a „>“. Ukončovací tag má před svým názvem ještě znak „/“, aby se snadno odlišil od počátečního. [8] Element, který nenese žádnou hodnotu se dá zapsat dvěma způsoby. Buď se hned za počáteční element uvede koncový a nebo se za jméno počátečního elementu uvede znak „/“ a ukončovací tag se vynechá. Podmínkou správné syntaxe XML dokumentu je, že každý počáteční tag musí mít ukončovací tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem.

Každý počáteční element může obsahovat libovolné množství atributů. Tyto atributy slouží k upřesnění významu elementu a k přidání dalších důležitých informací.

<jmeno_elementu autor=”Jan Novák” zabezpečení=”důvěrné”>

Obsah elementu </jmeno_elementu>

Hodnota atributu musí být uzavřena do uvozovek nebo apostrofů. Jednotlivé elementy se oddělují mezerou.



Obr. 5) Struktura XML dokumentu [9]

Každý XML dokument musí být celý obsažen v jednom elementu. Následující ukázka tedy není správný XML dokument, protože se skládá z několika samostatných elementů. [8]

<nadpis>Pokusný nadpis</nadpis>

<odstavec>První odstavec</odstavec>

<odstavec>Druhý odstavec</odstavec>

<odstavec>Třetí odstavec</odstavec>

Přidáme-li však jeden kořenový element obalující všechny výše uvedené, je vše v pořádku.

<článek>
<nadpis>Pokusný nadpis</nadpis>
<odstavec>První odstavec</odstavec>
<odstavec>Druhý odstavec</odstavec>
<odstavec>Třetí odstavec</odstavec>
</článek>

Chceme-li kdekoli v XML dokumentu použít řídící znak „<“ nebo „>“ mimo počátečního a ukončovacího tagu, musíme k tomu použít tzv. znakovou entitu. Různým znakům tedy odpovídají následující entity:

<	<
>	>
“	"
‘	'
&	&

2.6.2. XML Namespace

XML je metajazyk, pomocí něhož můžeme definovat nové značkovací jazyky. Může se stát, že je nutné v jednom XML dokumentu použít značky z více jazyků. To se stává například v dokumentech popisujících transformaci dokumentu XML. Zde jsou smíchány značky transformačního a cílového jazyka, do kterého se transformuje. Pokud by se v těchto jazycích vyskytovala stejná značka, došlo by k nejednoznačnosti.

Kvůli těmto možným problémům existuje XML Namespace, která umožňuje značky z různých jazyků rozlišovat. Značky každého jazyka jsou ve vlastním jmenném prostoru, který je identifikován pomocí jistého URI. [5] Například značky jazyka XHTML jsou identifikovány v URI „http://www.w3.org/1999/xhtml“. Značky jsou pak se jmenným prostorem svázány pomocí prefixu, zapsaného před značkou a odděleného dvojtečkou. Prefix musí být před svým použitím definován společně s příslušným URI.

xmlns:prefix="URI"

Použitý prefix není podstatný, důležité je URI jemu přidělené. Pro zjednodušení nemusí mít jeden ze jmenných prostorů definovaný prefix. V tom případě jsou všechny značky uvedené bez prefixu zařazeny do tohoto tzv. implicitního jmenného prostoru.

2.6.3. XML Schéma

Již bylo uvedeno, že XML je metajazyk, s jehož pomocí lze vytvářet nové značkovací jazyky kladoucí určitá omezení na dokumenty v daném jazyku. Původně se pro zápis struktury definovaného jazyku používal jazyk DTD. Jeho nevýhodou bylo, že měl jinou syntaxi než XML a neumožňoval zadat typy dat. Tato nevýhoda se projevovala zejména při zpracování dat databázového charakteru. Proto byl standardizován nový definiční jazyk XML Schema, který umožňuje popsat strukturu daného jazyka včetně typových omezení. Nevýhodou XML Schema je, že poskytovaný typový systém, který je vhodný pro definici struktury dokumentů, lze jen velmi obtížně přenést do objektově orientovaných programovacích jazyků. Výhodou nových jazyků popsaných pomocí XML Schema však je, že kontrolu správnosti načítání dat může místo aplikace provádět již knihovna pro rozbor XML. Tím se tvorba aplikace výrazně zjednodušuje.

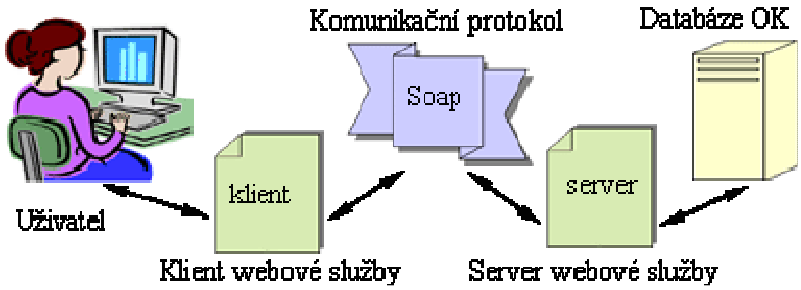
3. Analýza

3.1. Návrh webové služby pro Organizační kancelář

Jak již bylo uvedeno Organizační kancelář používá ke konfiguraci svého software WinFAS takzvané šablony. Základním požadavkem na webovou službu je zpřístupnění šablon uložených na serveru Organizační kanceláře zákazníkům. Vzhledem k tomu, že přenášená data (šablony) neobsahují citlivé údaje, tak není třeba zabezpečovat vlastní přenos dat. Kvůli usnadnění použití bude použit protokol HTTP, který je většinou přístupný a nakonfigurovaný na většině PC připojených k internetu. Vzhledem ke stávající infrastruktuře webových stránek Organizační kanceláře bude serverová část napsána v jazyce PHP. Jako zdroj informací bude použit stávající systém pro připojení k databázi. Webová služba bude obstarávat dvě úlohy. Za prvé bude mít za úkol podle zadaného identifikátoru vyhledat všechny dostupné šablony přiřazené k dané aplikaci. Jednoznačný identifikátor uživatel jednoduše zjistí při spuštění dané aplikace v informačním systému WinFAS. Při automatickém zjišťování dostupných šablon má klientská aplikace tento kód k dispozici jako globální proměnnou. Druhým úkolem, který bude webová služba obstarávat je případné poskytnutí vybrané šablony klientovi. Samozřejmě pouze v případě, že první část webové služby nějaké šablony k dané aplikaci najde. K realizaci klientské části bude použito tří technologií. První bude PHP klientská webová aplikace. Tato aplikace bude sloužit zejména pro klienty, jež nemají na počítači s WinFASem přístup na internet a budou si šablony stahovat tzv. offline. Stahováním offline je myšleno stažení dat na jiném PC s připojením na internet, uložení obdržených dat do souboru. Soubor s daty lze pak použít na počítači s WinFASem pro nastavení systému. Druhá technologie bude implementována netradičně pomocí interních nástrojů databáze SybaseAnywhere a bude navazujícími projekty použita pro automatický import šablon přímo z WinFASu. Vlastní realizace spočívá v sestavení zápisu pro SOAP upravené syntaxe pro uložené procedury. Poslední, spíše demonstrativní aplikací bude klient v prostředí Microsoft .NET verze 2.0.

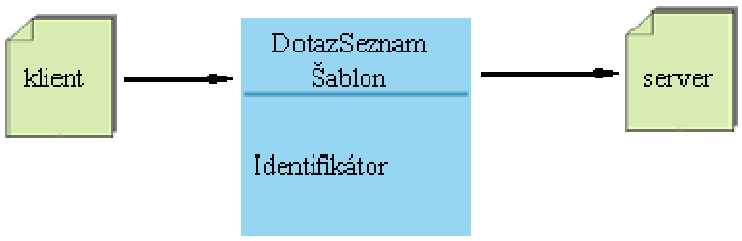
3.1.1. Získání dat pomocí webové služby

Data z databáze Organizační kanceláře, budou webovou službou vybrána podle přijatých parametrů a dále budou distribuována pomocí SOAP zprávy klientské aplikaci. V klientské aplikaci si bude moci klient vybrat požadovanou šablonu, případně si ji stáhnout. Jako klient webové služby může vystupovat uživatel pracující přes webové rozhraní, nebo i aplikace přistupující ke službám serveru přímo.



Obr. 6) Získání informací z databáze Organizační kanceláře pomocí webové služby [vlastní]

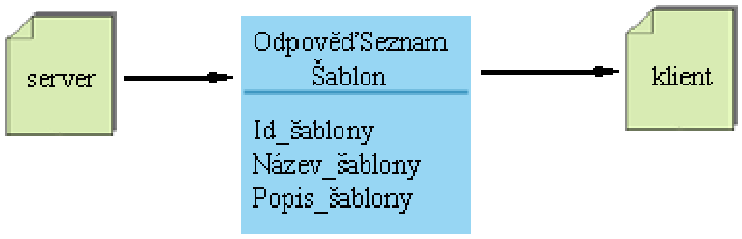
Podoba prvního dotazu na seznam dostupných šablon bude vypadat dle níže uvedeného obrázku. Uživatel zde posílá jako parametr pouze identifikátor, podle kterého pak vyhledá webová služba příslušné šablony.



Obr. 7) Dotaz na seznam šablon [vlastní]

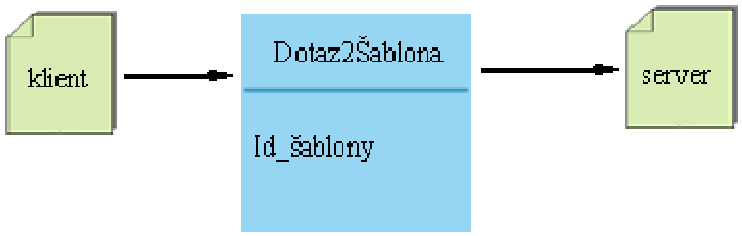
V případě, že se v databázi budou nacházet šablony spojené s daným identifikátorem bude odpověď obsahovat XML data s následujícími elementy:

- Id_šablony
- Název_šablony
- Popis_šablony



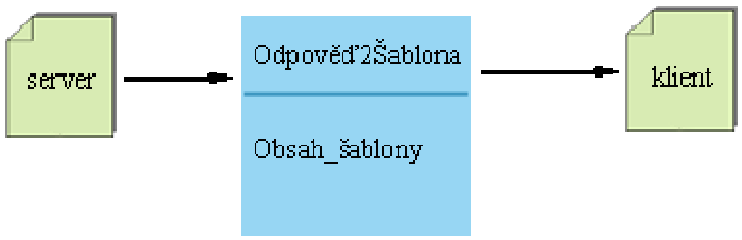
Obr. 8) Odpověď na seznam šablon [vlastní]

Pokud tedy odpověď služby na seznam šablon nevrátí prázdný seznam, uživatel si dle názvu a popisu šablon vybere požadovanou šablonu a odešle na webovou službu druhý požadavek, kde bude jako parametr použit Id_šablony jak ukazuje níže uvedený obrázek.



Obr. 9) Dotaz na konkrétní šablonu [vlastní]

Webová služba poté dle Id_šablony vrátí uživateli obsah konkrétní šablony.



Obr. 10) Odpověď na konkrétní šablonu [vlastní]

Návrh webové služby je proveden zcela obecně a pro konkrétní implementaci je možné použít jakýkoli programovací jazyk. Jedinou podmínku je, že zvolený programovací jazyk musí umět pracovat s webovými službami.

4. Implementace řešení

Jak již bylo uvedeno ke konkrétní implementaci webové služby může být použit jakýkoli programovací jazyk, který umí pracovat s XML a webovými službami. K implementaci serverové části byl zvolen programovací jazyk PHP, který umí pracovat s databází Organizační kanceláře a jsou v něm vyvinuty všechny webové aplikace této firmy. Pro řešení klientské části byly vybrány tři možné varianty. První variantou je taktéž programovací jazyk PHP, druhou variantou budou uložené procedury v databázi SQL Anywhere, jež webové služby také podporuje. Třetí a poslední klientskou částí bude samostatná aplikace vytvořená v programovacím jazyku C#.

4.1. Serverová část webové služby

Tato část obsahuje tři hlavní soubory index.php, server_function.class.php a server.wsdl. Dále je zde využíván soubor xml_generator.class.php. Tento soubor obsahuje definici třídy c_xml_generator jež slouží k vytváření XML dokumentů. Tato třída mnou nebyla vyvinuta. Její kód je volně přístupný a dá se získat na této adrese: http://podklady.interval.cz/peprnicek/685/php_xml_generator.zip.

4.1.1. server_function.class.php

V tomto souboru se nachází několik funkcí. Dvě z nich jsou funkce, které bude využívat webová služba, a další tři jsou pomocné funkce prvních dvou.

Funkce s názvem WinfasSOAP_GetSeznam je první funkcí webové služby, která podle vstupního parametru *kod* vyhledá v databázi seznam šablon spojených s tímto identifikátorem. Vyhledání tohoto seznamu je provedeno pomocí dotazu v databázi SQL Anywhere. Tato funkce dále využívá funkci CreateXML, která zabalí vrácená data do XML podoby. To se provádí kvůli jednoduchému zpětnému dekódování seznamu v klientské části. Do XML je zakomponován jako atribut i vstupní parametr *kod*. Funkce WinfasSOAP_GetSeznam ještě před vrácením výsledku zakóduje návratovou hodnotu (XML dokumentu) do datového formátu base64.

Druhou funkcí webové služby je funkce WinfasSOAP_GetSablona. Tato funkce pomocí vstupního parametru *pk* získá obsah konkrétní šablony, které jsou uloženy na serveru Organizační kanceláře. Funkce používá funkci GetFile, která

slouží k otevření souboru, který se nachází na místě vstupního parametru *path* této funkce. Stejně jako první funkce WinfasSOAP_GetSeznam i tato funkce před vrácením návratové hodnoty výsledek zakóduje do datového formátu base64.

Soubor ještě obsahuje funkci CreateXMLfile. Tato funkce ukládá *data*, což je jeden ze vstupních parametrů do souboru s názvem *nazev_soboru*(druhý parametr) a s příponou .xml. Tato funkce byla využívána při tvorbě a testování správné funkčnosti předešlých funkcí.

4.1.2. server.wsdl

Tento soubor, jak jíž název napovídá, je popisem webové služby. Webová služba by fungovala i bez tohoto popisu, jelikož každá webová služba tento popis mít nemusí. Tento popis však významně ulehčuje tvorbu klientské části v programovacím jazyce C#.

Jak již bylo řečeno, tento soubor popisuje webovou službu. Jeho obsah byl z části automaticky vygenerován pomocí PHP WSDL Generatoru. Tento generátor dokáže z třídy, kde jsou popsány funkce, jež bude webová služba využívat, vytvořit tento popis. Produkt PHP WSDL Generátor pro generování WSDL souborů patří pod licenci GNU Lesser General Public Licence a je dostupná z http://www.phpclasses.org/browse/package/3509.html.

Obsah vygenerovaného WSDL tímto produktem však není zcela korektní. Je nutné v něm udělat drobné úpravy. Jelikož je produkt PHP WSDL Generator napsán v jazyce PHP, který je beztypový, je nutné do vygenerovaného wsdl dokumentu dopsat typy vstupních a výstupních parametrů. Další věcí, která ve vygenerovaném wsdl dokumentu chybí je adresa, kde webová služba běží. Poslední drobnou úpravou, kterou však již není nutné provádět je zpřehlednění vygenerovaného wsdl dokumentu, jelikož obsah dokument vygenerovaného PHP WSDL Generatorem není odřádkován ani odsazen. Vygenerovaný WSDL dokument je uvedený v příloze B.

4.1.3. index.php

Soubor index.php je hlavním souborem serverové části. Na úvod je sem pomocí direktivy require_once připojen soubor core_start.php, který slouží k počáteční inicializaci jádra (připojení databáze apod.), nad kterým běží všechny webové skripty v Organizační kanceláři. Vývoj tohoto jádra však není obsahem mé bakalářské práce, proto se o něm více nebudu zmiňovat. Dále je do souboru index.php pomocí include direktivy vložen již popsaný soubor server_function.class.php. Z něho jsou volány dvě hlavní funkce WinfasSOAP_GetSeznam a WinfasSOAP_GetSablona. Hlavní funkcí je však inicializace serveru webové služby, kde jí je přiřazen již zmíněný soubor server.wsdl, který ji popisuje. Dále jí je přiřazeno jméno a jsou jí přidány dvě výše uvedené funkce, které má obsluhovat.

4.2. Klientská část webové služby vytvořená v PHP

Tato klientská část webové služby bude uživatelům přístupná přes webové rozhraní Organizační kanceláře. Skládá se ze dvou soborů clientu.ph a client.php.

4.2.1. ClientU.php

Tento soubor je pouze grafickým vstupem do klientské části webové služby. Uživatel pomocí tohoto souboru vidí jednoduchý formulář, kde musí vybrat o jakou funkční část informačního systému WinFAS se jedná. Zda jde o „Aplikaci“, „IQ sestavu“ nebo o „Pořízení“. Dále zde musí vyplnit již zmíněný identifikátor, pomocí kterého se vyhledává seznam šablon příslušející dané aplikaci informačního systému WinFAS. Zadané vstupní parametry tohoto formuláře jsou pak zpracovávány v souboru client.php.

4.2.2. client.php

Soubor client.php je hlavním souborem klientské části aplikace vyvinuté v programovacím jazyce PHP. Nejprve je zde ověřeno, že ze vstupního formuláře opravdu přišla požadovaná data. Po té je zde inicializována klientská část webové služby. Je jí nastavena adresa, kde se volaná služba nachází, jak se jmenuje a kde se nachází její popis v podobě WSDL souboru. Po inicializaci je zavolána první metoda webové služby WinfasSOAP_GetSeznam na vrácení seznamu šablon dle identifiká-

toru. Návratová hodnota je rozkódována z datového formátu Base64. Po té je ověřeno, zda návratová hodnota obsahuje nějaký seznam šablon. V kladném případě je seznam z XML podoby přetransformován do HTML a zobrazen uživateli. V záporném případě je vypsáno hlášení, že nebyla nalezena ani jedna šablona patřící k zadanému identifikátoru.

V případě, že však byl nějaký seznam šablon vrácen, je tento seznam uživateli zobrazen. Po jeho kliknutí na název šablony je zavolána druhá metoda webové služby WinfasSOAP_GetSablona s parametrem vybrané šablony. Její návratová hodnota je opět rozkódována z datového formátu Base64 a následně její obsah zobrazen v prohlížeči jako XML soubor. Z prohlížeče si může uživatel zobrazený obsah uložit jako XML soubor na svůj pevný disk a následně naimportovat do informačního systému WinFAS do dané aplikace.

4.3. Klientská část webové služby vytvořená v SQL Anywhere

Klientská část webové služby v SQL Anywhere je naprogramována pomocí uložených procedur. Tyto procedury samy o sobě neumožňují přehledné zobrazení nebo výběr dané šablony ze seznamu šablon vráceného webovou službou. To musí být ošetřeno v nadřazeném programovacím jazyce, který bude tyto uložené procedury využívat. Tímto nadřazeným jazykem je v Organizační kanceláři vývojové prostředí PowerBuilder, v kterém je celý systém WinFAS vytvořen a díky těmto procedurám do něj bude integrován přenos konfiguračních dat. Implementace nadřazené části ve vývojovém prostředí PowerBuilder již však nebylo součástí mé práce a proto zde nebude podrobně popsáno.

4.3.1. Metoda na vrácení seznamu šablon

V uložené proceduře obsluhující klientskou část webové služby je nutné dodat, že se jedná o webovou službu, což je zajištěno pomocí klíčového slova „SOAP“, dále se uvede kde webová služba běží a určí se typ vstupního parametru. Uložená procedura je velice jednoduchá a vypadá následovně:

ALTER PROCEDURE "DBA"."WinfasSOAP_GetSeznam"(in kod char(30))

result(response long varchar) url 'http://www.winfas.cz/soap/' type 'SOAP'

Nevýhodou muže být, že název uložené procedury se musí shodovat s názvem metody poskytované serverem webové služby. Tato uložená procedura také nedokáže zjistit, že je služba nedostupná nebo obsloužit jiné chybové kódy. To je zajištěno tím, že tato jednoduchá procedura je zaobalena v další proceduře, která tyto chybové stavy ošetřuje a nadřazenému vývojovému prostředí již zasílá chybové stavy.

4.3.2. Metoda na vrácení konkrétní šablony

Uložená procedura volající druhou metodu webové služby vypadá velice obdobně.

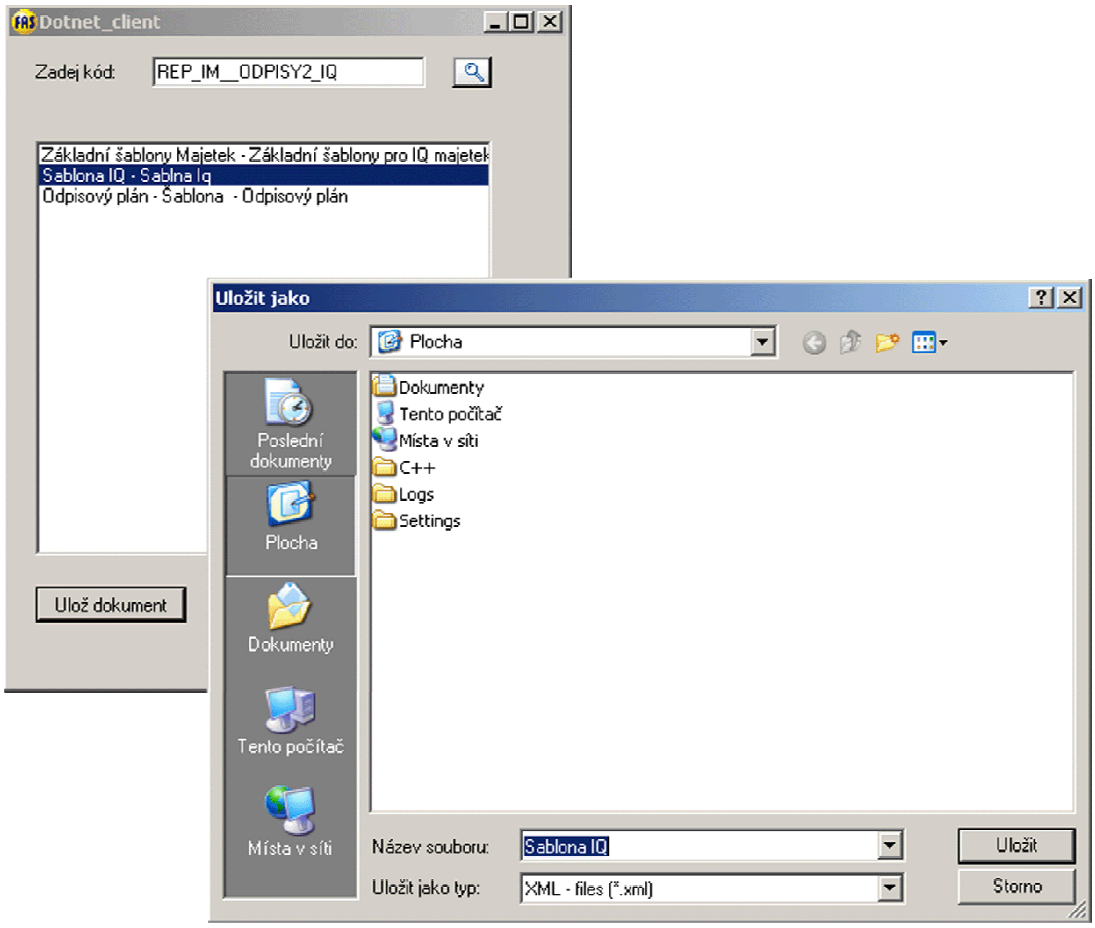
ALTER PROCEDURE "DBA"."WinfasSOAP_GetSablona"(in pk integer)

result(response long varchar) url 'http://www.winfas.cz/soap/' type 'SOAP'

Tato metoda je obdobně jako předchozí metoda obalena ošetřující procedurou. Tyto ošetřující procedury jsou uvedeny v příloze C.

4.4. Klientská část webové služby vytvořená v C#

Klientská aplikace vyvinutá ve vývojovém prostředí C# je samostatně běžící program. Při vývoji tohoto programu na obsluhu klienta webové služby se dá s úspěchem použít již dříve připravený WDSL popis. Pokud při tvorbě zadáme adresu kde se tento WSDL popis nachází, tak vývojové prostředí C# si pak z tohoto popisu již zjistí, jaké metody webová služba nabízí, jaké mají vstupní a výstupní parametry, jaké návratové hodnoty atp. Pro volání se na základě WSDL souboru vytvoří třída obalující metody webové služby. Po vytvoření instance třídy, pak můžeme jednotlivé funkce webové služby volat jako metody.Program stejně jako klientská část vyvinutá v PHP obsahuje pole, kam se zadá identifikátor, podle kterého se budou vyhledávat příslušné šablony pro danou aplikaci. Při nalezení seznamu šablon se tento seznam zobrazí v „listBoxu“, kde bude uveden identifikátor šablony, její název a dále její popis. Uživatel vybranou šablonu označí například kliknutím myši a po té stiskne tlačítko „Ulož dokument“ jež zavolá druhou webovou metodu na stažení obsahu šablony. V případě úspěšné návratové hodnoty se zobrazí uživateli dialogové okno pro uložení dokumentu. Tam si uživatel vybere, kam chce šablonu uložit. Z daného místa si ji pak jednoduše naimportuje do informačního systému WinFAS.



Obr. 11) Klient webové služby vyvinutý v C# [vlastní]

5. Závěr

Hlavním cílem této bakalářské práce bylo vytvoření webové služby pro přenos konfiguračních dat informačního systému WinFAS.

V průběhu práce byla provedena analýza, která definovala základní rámec toho, jak bude daná webová služba vypadat. Pro implementaci serverové části webové služby byl vybrán programovací jazyk PHP. Důvodem proč právě PHP bylo to, že tento jazyk obsahuje extenzi SOAP, která významně usnadňuje použití této technologie na platformě PHP. Klientská část webové služby byla implementována třemi způsoby. Jako nejjednodušší se ukázalo využití vývojového prostředí C#, které dokáže velice dobře využít popisný dokument WSDL, díky kterému nám vývojové prostředí Microsoft Visual Studio umožní automaticky vytvořit obalující třídu, která nás zcela odstíní od technologie webových služeb. Jediným, avšak z pohledu cílové skupiny uživatelů poměrně závažným nedostatkem je nutnost instalace .NET frameworku na počítač uživatele. Zbývající dvě klientské aplikace již tento WSDL popis takto efektivně využít nedokáží. Další z klientských částí byla vyvinuta stejně jako serverová část v jazyce PHP. Tato implementace slouží jako „tenký“ klient a pro použití stačí pouze internetový prohlížeč. Navíc lze takto stahovat šablony na jakémkoliv počítači připojeném k internetu. Uložená procedura SQL Anywhere implementovaná jako třetí řešení slouží pouze k volání daných funkcí webové služby z nadřízené aplikace. I přes značná omezení, které svazuje programátora při využití této varianty, se toto řešení ukázalo jako provozuschopné a pro danou aplikaci více než dostatečné.

Technologie Webových služeb je nesmírně flexibilní a pro dané zadání velmi vhodná. Zadání práce se podařilo v plném rozsahu naplnit. Nelze říci, které řešení je nejlepší. Každá z použitých technologií má svoje klady i zápory a každé řešení má jinou cílovou skupinu uživatelů. Všechna implementovaná řešení byla otestována v reálném provozu a byla plně funkční.

Seznam použité literatury

[1] *Ovládnou web služby – LUPA*, November 22, 2002 [online], Jiří Peterka, [cit. 2008-05-09]. Dostupné z WWW: http://www.lupa.cz/clanky/ovladnou-web-sluzby

[2] *Tutoriál web services*, Martin Kuba, [cit. 2008-05-09]. Dostupné z WWW: http://www.ics.muni.cz/~makub/soap/tutorial.html#cojsou

[3] *Využití webových služeb a protokolu SOAP při komunikaci*, March 01, 2003 [online], Jiří Kosek, [cit. 2008-05-10]. Dostupné z WWW: http://www.kosek.cz/diplomka/html/websluzby.html

[4] *Web services*, April 23, 2008 [online], Martin Kuba, [cit. 2008-05-10]. Dostupné z WWW: http://www.ics.muni.cz/zpravodaj/articles/269.html

[5] *Web services*, October 14 – October 17, 2006 [online], Martin Kuba, [cit. 2008-05-10]. Dostupné z WWW: http://www.ics.muni.cz/~makub/soap/MartinKuba_WebServices_Datakon2006_clanek.pdf

[6] *Řetězení webových služeb v prostředí open source GIS*, 2007 [online], Martin Prager, [cit. 2008-05-10]. Dostupné z WWW: http://postgis.vsb.cz/GISacek2007/sbornik/prager_gisacek07.pdf

[7] *Extensible Markup Language*, May 01, 2008 [online], [cit. 2008-05-11]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Extensible_Markup_Language

[8] *Lehký úvod do XML*, Jiří Kosek, [cit. 2008-05-11]. Dostupné z WWW:http://www.cstug.cz/slt/01/plne_texty/13.pdf

[9] *Základy jazyka XML*, 2001 [online], Jiří Kosek, [cit. 2008-05-11]. Dostupný z WWW: http://www.kosek.cz/clanky/swn-xml/syntaxe.html

[10] *Slabikář XML - úvod do problematiky - INTERVAL*, March 23, 2002 [online], Marek Soldát, [cit. 2008-05-11]. Dostupný z WWW: http://interval.cz/clanky/slabikar-xml-uvod-do-problematiky/

[11] *Vše o XML, 5.díl - web services - LINUXZONE*, July 07, 2002 [online], Vojtěch Patrný, [cit. 2008-05-11]. Dostupný z WWW: http://www.linuxzone.cz/index.phtml?idc=311&ids=2

[12] *Webové služby v PHP: XML-RPC a SOAP*, August 14, 2007 [online], Jakub Vrána, [cit. 2008-05-09]. Dostupné z WWW: http://php.vrana.cz/webove-sluzby-v-php-xml-rpc-a-soap.php

[14] XML eXtensible Markup Language, May 28, 1999 [online], Jiří Kosek, [cit. 2008-05-10]. Dostupné z WWW: http://www.kosek.cz/clanky/xml/index.html

[15] Historie Organizační kanceláře, 2008 [online], [cit. 2008-05-09]. Dostupný z WWW: http://www.winfas.cz/index.php?pk_d9101__=543

[16] WELLING, Luke, THOMSON, Laura. PHP a MySQL rozvoj webových aplikací. 3. vyd. Praha : SoftPress, 2005. ISBN 80-86497-83-6

[17] BRADLEY, Neil. XML kompletní průvodce. 1. vyd. Praha : Grada Publishing, 2000. 540 s. ISBN 80-7169-949-7

Příloha A Konfigurační šablona

```
<?xml version="1.0" encoding="utf-8"?>
<winfas_sab>
<c0063okno_>VYBREP_POLOZKY</c0063okno_>
 <c0063>
  <c0063_row>
   <c0063ryvyb>VZOR</c0063ryvyb>
   <c0063nazev>Opravné pol.</c0063nazev>
   <c0064>
    <c0064_row>
     <c0064udaj_>zobr_n0010dokla</c0064udaj_>
     <c0064typud>1</c0064typud>
     <c0064hodno>1</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>1</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>zobr_n0026castd</c0064udaj_>
     <c0064typud>1</c0064typud>
     <c0064hodno>1</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>1</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>zobr_n0026evouc</c0064udaj_>
     <c0064typud>1</c0064typud>
     <c0064hodno>1</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>1</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>@VYBREP_TECKY</c0064udaj_>
     <c0064typud>0</c0064typud>
     <c0064hodno>0</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>0</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>@VYBREP_NORMALNI</c0064udaj_>
     <c0064typud>0</c0064typud>
     <c0064hodno>2</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>0</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>zobr_c0030klien</c0064udaj_>
     <c0064typud>1</c0064typud>
     <c0064hodno>1</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>1</c0064activ>
    </c0064_row>
    <c0064_row>
     <c0064udaj_>zobr_n0026datso</c0064udaj_>
     <c0064typud>1</c0064typud>
     <c0064hodno>0</c0064hodno>
     <c0064prika>0</c0064prika>
     <c0064activ>1</c0064activ>
    </c0064_row>
    <c0064_row>
```

<c0064udaj_>zobr_n0026datuz</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>0</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026nazdr</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026castk</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@CRM</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno>0</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026procd</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_c0077nazkn</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>0</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_c0031nazad</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026datko</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026kodpa</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>

<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0011nazpo</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>0</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0011castk</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026procn</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026castn</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@VYBREP_GRUPOVANI</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno>kl_id,kl_nazad,cislo_faktury,predpis_pohl</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@LEVOHORE</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@AUTOSIZE</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno>-1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_n0026evouy</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@VYBREP_SORTOVANI</c0064udaj_>

<c0064typud>0</c0064typud>
<c0064hodno>kl_id A,kl_nazad A,cislo_faktury A,predpis_pohl A,kod_dr A,druh_op A,proc_dan A,cast_dan A,proc_nedan A,cast_nedan A,castk_pred A,dat_konk A,sto_tvo A,obdobi_op A</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@OPEN_DIALOG</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>zobr_c0077kniha</c0064udaj_>
<c0064typud>1</c0064typud>
<c0064hodno>1</c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>1</c0064activ>
</c0064_row>
<c0064_row>
<c0064udaj_>@POPIS_UPRAV</c0064udaj_>
<c0064typud>0</c0064typud>
<c0064hodno></c0064hodno>
<c0064prika>0</c0064prika>
<c0064activ>0</c0064activ>
</c0064_row>
</c0064>
<c0065/>
</c0063_row>
</c0063>
</winfas_sab>

Příloha B WSDL soubor webové služby

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions
 name="ServerFunction"
 targetNamespace="urn:ServerFunction"
 xmlns:typens="urn:ServerFunction"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

 <message name="WinfasSOAP_GetSablona">
  <part name="pk" type="xsd:int"></part>
 </message>

 <message name="WinfasSOAP_GetSablonaResponse">
  <part name='response' type='xsd:char'/>
 </message>

 <message name="WinfasSOAP_GetSeznam">
  <part name="kod" type="xsd:char"></part>
 </message>

 <message name="WinfasSOAP_GetSeznamResponse">
  <part name='response' type='xsd:long varchar'/>
 </message>

 <portType name="ServerFunctionPortType">
  <operation name="WinfasSOAP_GetSablona">
   <input message="typens:WinfasSOAP_GetSablona"></input>
   <output message="typens:WinfasSOAP_GetSablonaResponse"></output>
  </operation>
  <operation name="WinfasSOAP_GetSeznam">
   <input message="typens:WinfasSOAP_GetSeznam"></input>
   <output message="typens:WinfasSOAP_GetSeznamResponse"></output>
  </operation>
 </portType>

 <binding name="ServerFunctionBinding" type="typens:ServerFunctionPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"></soap:binding>

  <operation name="WinfasSOAP_GetSablona">
   <soap:operation soapAction="urn:ServerFunctionAction"></soap:operation>
   <input>
    <soap:body namespace="urn:ServerFunction" use="encoded"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
   </input>
   <output>
    <soap:body namespace="urn:ServerFunction" use="encoded"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
   </output>
  </operation>

  <operation name="WinfasSOAP_GetSeznam">
   <soap:operation soapAction="urn:ServerFunctionAction"></soap:operation>
   <input>
    <soap:body namespace="urn:ServerFunction" use="encoded"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
```

```
    </input>
    <output>
      <soap:body namespace="urn:ServerFunction" use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"></soap:body>
    </output>
  </operation>
 </binding>

 <service name="ServerFunctionService">
  <port name="ServerFunctionPort" binding="typens:ServerFunctionBinding">
    <soap:address location='http://localhost/soap/'/>
  </port>
 </service>
</definitions>
```

Příloha C Ošetřující procedury

```
ALTER PROCEDURE "DBA"."sp_d9101q02"(in @kod char(30))
begin
  declare @trans SD_TRANSAKCE_;
  declare response long varchar;
  declare err_notfound exception for sqlstate value '02000';
  declare curThisCust dynamic scroll cursor for select response from WinfasSO-
        AP_GetSeznam(@kod);
  call sp_z_trans_number(@trans);
  open curThisCust;
  CustomerLoop: loop
    fetch next curThisCust into response;
    if sqlstate = err_notfound then
      leave CustomerLoop
    end if;
    insert into w0003( w0003blob_,w0003trans) select BASE64_DECODE(response),@trans;
    select convert(integer,@@identity) as response
  end loop CustomerLoop;
  close curThisCust
exception
  when others then
    select-1 as response
end

ALTER PROCEDURE "DBA"."sp_d9101q03"(in @pk integer)
begin
  declare @trans SD_TRANSAKCE_;
  declare response long varchar;
  declare err_notfound exception for sqlstate value '02000';
  declare curThisCust dynamic scroll cursor for select response from WinfasSOAP_GetSablona(@pk);
  call sp_z_trans_number(@trans);
  open curThisCust;
  CustomerLoop: loop
    fetch next curThisCust into response;
    if sqlstate = err_notfound then
      leave CustomerLoop
    end if;
    insert into w0003( w0003blob_,w0003trans) select BASE64_DECODE(response),@trans;
    select convert(integer,@@identity) as response
  end loop CustomerLoop;
  close curThisCust
exception
  when others then
    select-1 as response
end
```