Servicii Web. O perspectivă SOAP & WSDL & UDDI

Alexandru Dovlecel

12 decembrie 2002

Prefata

Bine aţi venit în lumea Serviciilor Web! Bine aţi venit în lumea în care programarea distribuită, interoperabilitatea, independenţa de platformă şi limbaj de programare îşi spune cuvântul, îşi cere drepturile. Sper ca această lucrare să vă ajute să găsiţi reperele de bază şi să navigaţi cu plăcere printre aceste idei si concepte. Şi, de ce nu, chiar să vă aduceţi propria contribuţie la dezvoltarea unei ramuri a informaticii care a luat naştere de curând şi se află într-o continuă expansiune.

Există şi funcţionează?

În ultimul timp tehnologia serviciilor web a apărut tot mai des în atenţia presei de specialitate. Un număr mare de produse legate de programarea internet a inceput să invadeze piaţa IT. Tot mai multe companii ce îşi desfăşoară activitatea cu ajutorul aplicaţiilor Internet îşi indreaptă atenţia către această ramură a tehnologiei. Cât despre adevăraţii lideri in domeniul producerii de software, aceştia au creat companii specializate in specificarea, dezvoltarea şi folosirea acestor noi protocoale şi tehnologii. Despre toate acestea vom vorbi mai încolo. Dar din aceste observatii putem deduce că acesta este un câmp de activitate nou, dinamic si plin de viitor.

Deşi de scurt timp în activitate, serviciile web au început să prindă contur. Iar pentru a continua dezvoltarea într-o manieră cât mai organizată, a fost necesară prezenţa unor standarde adoptate de comun acord de majoritatea producătorilor ce au un cuvant de spus în acest domeniu. Din această nevoie au luat naştere diverse standarde, în mare parte specificate de consortiul W3C, dintre care vom prezenta: eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Hypertext Transfer Protocol(HTTP), Web Service Description Language (WSDL) şi Universal Description Discovery and Integration (UDDI). Aceste standarde oferă regulile de bază ale serviciilor si vor fi prezentate în amănunt pe parcursul lucrării.

Pentru cei ce nu sunt familiarizaţi cu aceste denumiri, sa aruncăm o scurtă privire peste ele:

- XML oferă utilizatorului un mod independent de platformă de a încapsula si organiza datele. În acelaşi timp permite crearea elementelor de sintaxă pentru definirea unor noi vocabulare si a unor noi structuri de date.
- SOAP este un vocabular XML ce permite computerelor să interactioneze prin intermediul reţelei. Acest protocol de organizare şi încapsulare a informaţiei este următorul pas în domeniul Remote Procedure Call (RPC).
- HTTP este protocolul de comunicare cel mai des folosit de serviciile web pentru a transmite informaţiile. Este extrem de eficient datorită posibilităţii de a trece de firewall, în acelaşi timp permiţând utilizatorului să examineze datele transmise.

- WSDL este de asemenea un vocabular XML prin care computerul descrie în limbaj 'propriu' serviciile web pe care le expune, oferind programatorului posibilitatea de a genera cod de conectare în mod automat, prin intermediul unor tooluri disponibile pe piaţă (unele free, altele comerciale).
- UDDI a apărut datorită nevoii utilizatorilor de a găsi anumite servicii care să le îndeplinească cerinţele. Bazat pe XML, este un mod de publicare a serviciilor web.

Un alt argument ce ar putea fi decisiv pentru a studia acest domeniu îl reprezintă utilizarea tehnologiei SOAP de către Microsoft in noua platforma .NET. SOAP este protocolul pe care se bazează comunicarea dintre componente cât şi un suport puternic pentru dezvoltarea serviciilor distribuite.

Despre lucrare

După ce v-am prezentat toate aceste argumente extrase din lumea ce ne inconjoara (lumea giganţilor IT), ar fi timpul să spunem câteva cuvinte despre această lucrare. Ea a fost realizată din două motive: nevoia personală de a înţelege aceste noi tehnologii şi dorinţa de a prezenta cât mai amănunţit, concis si didactic cu putinţă cunoştinţele pe care le-am acumulat în acest domeniu.

Aici veti putea gasi informaţii despre naşterea si evoluţia internet în paralel cu nevoia de a folosi metode de programare cat mai puternice (OOP, programarea paralelă, programarea distribuită). După care urmatorul mare pas va fi explicarea cât mai concisă a fiecărui protocol mentionat mai sus, punând accentul în mod firesc asupra celor trei protocoale ce formează subiectul acestei lucrări: SOAP & WSDL & UDDI. Toate aceste informaţii vor fi completate cu exemple elocvente şi cu descrierea unor produse ce oferă programatorului posibilitatea de a implementa cât mai uşor şi rapid propriile servicii web. Ultima parte a lucrării de faţă este un studiu de caz al unei aplicaţii vaste, aplicatie ce are ca scop oferirea de servicii informaţionale printr-o interfaţă WEB.

CONTINUTUL IN DETALIU:

PARTEA I ...

PARTEA A II-A ...

PARTEA A III-A ...

Cui i se adresează această lucrare?

Această lucrare se adresează în mod special celor ce doresc să înteleagă ceea ce află în spatele ideii de serviciu WEB. Deşi extrem de vast, am încercat să modularizez acest subiect astfel încât să fie uşor de parcurs (de la un capăt la altul) cât şi usor de accesat în cazul căutării unor informaţii legate de un produs anume. Datorită dorinţei de a împărtăşi propriile cunoştinţe, această lucrare are şi un caracter didactic, oferind o parcurgere gradată a subiectului, pornind de la lucruri de bază, idei fundamentale şi ajungând la concepte avansate si tehnici mai putin folosite.

Pentru a intelege mai bine unele aspecte referitoare la organizarea, serializarea si transmiterea informatiei cu ajutorul SOAP, cititorul trebuie sa detina cunostinte de baza despre XML. O scurta prezentare a XML poate fi gasita in anexele acestei lucrari. Dar trebuie mentionat ca XML nu face subiectul central al acestei prezentari, deci este posibil ca informatia sa nu fie suficienta.

De asemenea este necesara cunoasterea limbajului java (sintaxa si nu numai) cat si idei de baza despre organizarea unui Application Server pentru partea a doua. Partea a treia necesita cunostinte avansate de java, avand in vedere ca se vor folosi urmatoarele tehnologii: parsare de xml, conectare la baza de date folosind JDBC, multithreading.

Este recomandata cunoasterea elementelor de baza ale UML (Unified Modelling Language) pentru a putea intelege cu usurinta modularizarea si structurarea pe clase a serverelor SOAP studiate in aceasta lucrare. Aceste cunostinte sunt foarte utile in parcurgerea studiului de caz oferit de partea a treia.

In concluzie, sper că acest ghid va fi uşor de utilizat, plăcut ca exprimare si concis ca şi conţinut.

Navigare placută în continuare...

Partea I

Protocolul SOAP

Capitolul 1

Introducere in lumea Serviciilor Web

Dupa cum cu totii stim, ultimele decenii au adus o adevarata explozie de produse si tehnologii noi. Industria IT a cunoscut o dezvoltare extrem de rapida, de multe ori ducând la modificari radicale privind arhitectura si dezvoltarea produselor software. Un astfel de moment a sosit si acum, când conceptul de Serviciu Web (Web Service) se apropie de maturitate.

1.1 Inainte de Servicii Web (un pic de istorie)

Serviciile Web nu au luat nastere din neant. Ele au aparut ca o nevoie naturala de extindere a tehnologiei informatiei si se bazeaza pe experienta acumulata de la inceputul erei informationale. Aceasta experienta s-a concretizat, in diverse etape, prin tehnologii noi, noi moduri de organizare, salturi informationale. Pentru a întelege cum si de ce au aparut Serviciile Web, trebuie sa ne intoarcem privirea catre trecut, spre a gasi punctele de baza ale evolutiei tehnologiei.

Programarea, in primii sai pasi, a fost dificila si plina de capcane ascunse, datorate in mare parte utilizarii programarii nestructurate. Din aceasta cauza multe proiecte software au intâmpinat greutati în dezvoltarea produsului dar în mod special în încercarea de a-l întretine sau de a aduce îmbunatatiri. Programatorii aveau nevoie de ordine si din aceasta nevoie a luat nastere conceptul de programare structurata.

Urmatorul hop ce a fost depasit tine mai mult de organizarea si reprezentarea modelelor. Desi programatorul avea la dispozitie un model de organizare a produsului, folosirea programarii structurate nu era îndeajuns de clara. Îi lipsea naturaletea organizarii datelor. Astfel a fost foarte normal sa apara un nou model de programare, denumit obiect orientat. De ce pe obiecte? Trebuie doar sa aruncam o privire în jur si o sa vedem ca suntem înconjurati de obiecte: masini, oameni etc. Realitatea este o suma de obiecte ce interactioneaza unele cu altele, de evenimente ce cauzeaza sau au efect asupra obiectelor. Astfel se poate scrie cod într-o maniera mult mai elenganta si mai apropiata de modelul ce trebuie exprimat.

În aceeasi perioada, o noua lume a început sa prinda viata. Omenirea avea nevoie de un mod de comunicare eficient si rapid. Telefonul nu mai era de ajuns, nu oferea suficienta informatie si conectarea era limitata. Utilizatorul dorea sa acceseze informatii variate fara un efort considerabil. Solutia a fost evidenta: o retea globala de calculatoare legate permanent între ele:*INTERNET*. La început simplu proiect de cercetare, începând cu anii '80 a cunoscut o puternica expansiune, aducând tot mai multe inovatii. Acestea idei noi s-au dovedit a fi adevarate mine de aur, motiv pentru care marile companii nu au stat pe gânduri si s-au

implicat in propagarea acestora prin intermediul produselor proprii. Rezultatul a fost clar: un flux tot mai mare de date si informatii a împânzit lumea virtuala.

Primul pas a fost reprezentat de prezentarea informatiei prin intermediul paginilor html. În scurt timp acestea nu au mai fost de ajuns. Utilizatorul dorea o interfata cât mai prietenoasa dar si utila. Avea nevoie de dinamism. Astfel a luat nastere ideea de scripting. Pentru început a fost doar client-side scripting. Exemplele cele mai relevante sunt JavaScript si JScript.

Dar lucrurile nu s-au oprit aici. Cantitatea crescânda de informatie cerea foarte multa munca de întretinere si administrare a paginilor deja existente. Schimbarile si actualizarile erau (si înca mai sunt) costisitoare. Aceste inconveniente au creat server-side scriptingul. La inceput sub forma CGI, a evoluat treptat catre tehnologii mult mai usor de utilizat, cum ar fi ASP (Active Server Pages, solutie propusa de Microsoft), JSP (Java Server Pages, standard propus de SUN si adoptat de comun acord pentru scrierea de aplicatii web în java) si PHP. Noile metode se sprijina tot pe vechiul HTML, dar conceptele sunt noi: bazat pe cererile HTML executate de client, serverul construieste raspunsul tot ca pagina HTML pe baza informatiilor pe care le detine (în baza de date, în sesiunea utilizatorului, în fisiere de configurare aflate pe server samd).

Datorita impulsului si perspectivelor oferite de dezvoltarea programarii orientate pe obiecte un nou fenomen a luat nastere: *programarea distribuita*. Când se creaza aplicatii mari, uneori este mai eficient sau chiar necesar ca anumite procese sa fie executate pe sisteme diferite, crescând astfel performanta solutiei. Pentru a functiona corect, un astfel de sistem trebuie sa permita comunicarea între componentele sale. Pentru aceasta trebuie stabilit un protocol de transmitere si receptionare a informatiei. Ca urmare, fiecare mare firma de dezvoltare software a incercat sa isi impuna propria viziune, creand solutii complexe precum: CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model) de la Microsoft, RMI/IIOP (Remote Method Invocation over Internet Inter-Orb Protocol) de la Sun. Dar aceasta diversitate de produse ridica urmatoarele probleme:

- *Lipsa unor standarde oficiale.* Desi multe dintre solutii au fost prezentate unor organizatii abilitate în crearea si promulgarea de standarde (cum ar fi W3C), ele nu au fost aprobate. De aceasta problema s-a lovit standardul CORBA, care, în faza actuala este folosit cu succes, dar versiunile difera (mai mult sau mai putin) de la un producator la altul.
- *Lipsa de interoperabilitate.* Fiecare solutie are un mod diferit de a incapsula si transmite informatia prin retea. De aici si imposibilitatea de comunicare dintre ele. Totusi exista încercari de a trece peste acest obstacol, un exemplu bun fiind COM/CORBA bridge (o punte între COM si CORBA). Dar aparitia oricarei schimbari în vreunul dintre protocoalele implicate duce la schimbari (uneori chiar majore) ce trebuie aduse acestei solutii.
- *Comunicarea între diverse limbaje* este foarte dificil de obtinut, daca nu chiar imposibil. Un exemplu elocvent ar fi RMI/IIOP, implementat si folosit doar in java datorita modului de serializare specific cat si protocolului de transport folosit.

1.2 De ce Serviciile Web?

Chiar daca abia au început sa apara implementari de Servicii Web, conceptul a luat nastere cu ceva timp în urma. Prima încercare a apartinut firmei Hewlett-Packard care în anul 1999 a lansat produsul *eSpeak*. Acesta permitea utilizatorilor sa construiasca aplicatii asemanatoare cu serviciile web de acum. Dar aceasta solutie nu s-a raspândit foarte rapid datorita caracterului proprietar pe care il are, depinzând în mare masura de echipamente si tehnologii oferite de firma producatoare.

Urmatorul pas a fost facut de Microsoft prin intermediul noii platforme pe care a lansat-o: .NET. Aceasta are ca si componenta cheie conceptul de Web Service. Din acest moment toate marile companii au început o cursa de încorporare si extindere a acestei noi tehnologii.

Si totusi, ce este un Serviciu Web?. Un serviciu web este o interfata de comunicare oferita de server, prin care clientii (programe aflate pe alte sisteme) pot solicita diverse informatii. Clientul poate varia, poate fi prezent pe acelasi calculator cu serverul, poate fi in aceeasi retea locala sau se poate afla in partea opusa a globului. Este o implementare de Remote Procedure Call. Si nu numai.

Dar acesta este doar un concept. Si pentru a nu sfarsi ca si alte tehnologii care au pornit de la o idee buna dar s-au extins in tot mai multe directii si fara a se focaliza catre anumite probleme, a fost nevoie de un standard pe baza caruia sa se contruiasca serviciile web. Altfel zis un set de reguli care sa fie strict respectate de produsele dedicate acestei arii. Ele se bazeaza pe experienta anterioara cat si pe nevoile programatorului. In consecinta, un Serviciu Web trebuie:

- *sa fie usor de extins si refolosit* in aplicatii noi. Acest lucru este realizat prin adoptarea programarii obiect orientate, cat si prin folosirea modularizarii. Un serviciu poate fi vazut ca un modul, un obiect. Clientul nu trebuie sa stie ca serverul se afla pe alta masina ci doar sa apeleze o metoda a serviciului ca si cand acesta este un obiect ce apartine propriului program.
- *sa ofere interoperabilitate*, indiferent de limba, platforma si limbajul de programare folosit. Aceasta problema foarte delicata a fost rezolvata prin decizia de a folosi XML (eXtended Markup Language) pentru a transmite datele prin retea. Dar XML fiind un vocabular foarte general, s-a hotarat realizarea unei particularizari a acestuia, rezultand SOAP (Simpe Object Access Protocol). Acesta doar impune niste reguli de formatare a mesajului XML reprezentand informatia transmisa.
- *sa fie transmis prin cat mai multe cai posibile prin retea.* Acest lucru este necesar deoarece multe companii doresc sa foloseasca doar anumite protocoale de transport pentru o mai buna securitate. Aici trebuie recunoscut faptul ca cea mai buna solutie este folosirea protocolului HTTP (HyperText Transfer Protocol) datorita posibilitatii de a nu fi blocat de firewall.
- *sa fie usor de descris si creat clienti.* Pentru aceasta importanta problema ridicata de cei ce dezvolta programele client ale serviciilor web s-a gasit o solutie ingenioasa: dezvoltarea unui nou subset XML denumit WSDL prin care, prin elemente prestabilite sa se informeze clientul asupra structurilor (obiectelor) folosite de serviciu, asupra metodelor expuse si asupra modului de accesare a acestora (parametrii, tipul parametrilor -intrare, iesire, intrare si iesire-, protocoalele de transport admise, stilul si tipul de incodare samd). Datorita caracterului fix pe care il are acest limbaj, a fost posibila realizarea unor tooluri externe pentru crearea suportului de comunicare (necesar unui client pentru a invoca un serviciu) doar pe baza fisierului WSDL ce descrie acel serviciu.
- *sa fie usor de gasit pe internet.* In alte cuvinte, daca un proiect are nevoie de o anumita functionalitate, cei ce il creaza sa poata cauta cu usurinta pe internet serviciul respectiv si sa vada daca intr-adevar indeplineste conditiile necesare pentru a putea fi folosit. Acest lucru este realizat prin folosirea unor registrii UDDI.

1.3 Avantaje si dezavantaje ale Serviciilor Web

Si daca tot am aratat ca Serviciile Web au luat nastere din nevoia de intercomunicare cat si pe baza experientei oferite de produsele anterioare, este timpul sa prezentam avantajele aduse:

- Respecta un Standard Deschis (Open standard), adica permit comunicarea intre componente scrise in limbaje diferite sau care ruleaza pe platforme diferite.
- Au un caracter modular. Prin urmare companiile pot refolosi acelasi serviciu in alte proiecte, fara a depune nici un efort.
- Sunt destul de usor de implementat iar costurile sunt reduse deoarece se bazeaza pe o infrastructura deja existenta (reteaua de comunicare si protocoalele folosite de aceasta).
- Reduce costul de integrare al aplicatiilor internet cat si comunicarea intre aplicatii B2B (Business to Business), fiind o alternativa profitabila pentru companiile ce se folosesc de aceste tipuri de aplicatii.
- Permit o implementare incrementala, gradata, evitand schimbari bruste de tehnologii in interiorul unei organizatii.

Dar nu se poate ca totul sa fie perfect. Astfel ca serviciile web mai prezinta si unele dezavantaje. Cel mai mare este faptul ca inca nu exista o specificatie finala pentru standardele folosite, ea fiind inca in lucru. Cu toate acestea exista o forma preliminara deja publicata, dar cu anumite paragrafe neclare (printre care se remarca tratarea si incapsularea exceptiilor de pe server catre client).

1.4 Rularea serviciilor Web

Protocoalele de baza au fost stabilite. Dar acest lucru nu este suficient pentru a putea sa rulam servicii web. Pentru aceasta mai raman cativa pasi importanti despre care trebuie sa vorbim.

Este necesara existenta unei componente numita *motor SOAP (SOAP engine)*. Aceasta componenta este responsabila cu deserializarea cererilor de la clienti in obiecte standard si cu serializarea raspunsurilor pentru clienti conform specificatiilor SOAP. Aceasta componenta poate sa fie integrata in platforma folosita (cum este cazul cu .NET) sau poate fi separata.

Rularea unui serviciu poate sa fie facuta in doua moduri: independent (standalone) sau in contextul unui server.

1.4.1 Rularea serviciului standalone

Acest tip de rulare este permis de platforma .NET. In realitate aceasta nu este in intregime independenta. Acest tip de serviciu este compus din doua clase principale: clasa care contine interfata oferita de serviciu (cat si implementarea) si clasa ce lanseaza in executie serviciul.

Clasa ce contine interfata si functionalitatea serviciului este obisnuita si trebuie sa se conformeze anumitor standarde cerute de platforma .NET. Ea poate sa foloseasca diverse alte module instalate pe platforma respectiva.

Clasa responsabila cu lansarea serviciului reprezinta de fapt serverul si are o forma standard. Ea anunta platforma .NET ca doreste sa inregistreze un serviciu la o anumita adresa a sistemului si la un anumit port. Apoi trebuie sa dea informatii referitoare la serviciu: ce tip de serviciu este (singleton sau one per request), care este clasa care ii defineste interfata si implementarea cat si metodele expuse. Apoi ramane in ashteptare (daca am incheia executia metodei principale, ar fi echivalent cu a opri serverul).

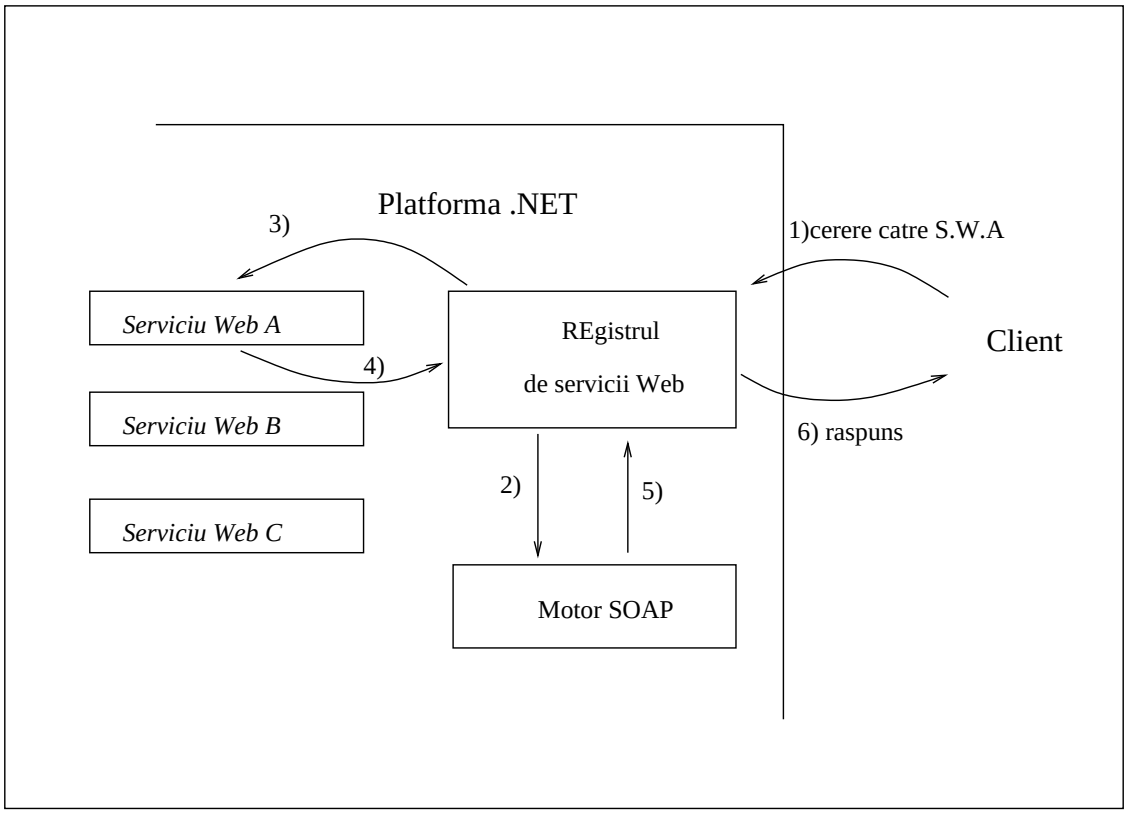

Figura 1.1: Traseul unei cereri SOAP catre un serviciu standalone pe o platforma .NET

In acest caz platforma .NET se comporta ca un quasiserver. Figura 1.1 prezinta traseul pe care il urmeaza un mesaj SOAP de la client catre server si inapoi. Etapele sunt:

pasul 1) Platforma .NET primeste cererea de la client sub forma de mesaj SOAP si il trimite mai departe la registrul de servicii web.

pasul 2) Registrul deserializeaza mesajul prin intermediul Motorului SOAP.

pasul 3) Pe baza mesajului decodat, registrul incarca serviciul dorit de client si apeleaza metoda ceruta.

pasul 4) Registrul primeste rezultatul metodei cerute.

pasul 5) Registrul serializeaza rezultatul metodei prin intermediul Motorului SOAP.

pasul 6) Registrul trimite clientului rezultatul serializat in format SOAP.

1.4.2 Rularea serviciului cu server

Rularea serviciul prin intermediul unui server confera acestuia o mai mare putere (beneficiind de mecanisme specifice unui server pentru repartizarea optima a cererilor, optimizarea conexiunilor, folosirea de sesiuni). In acest caz este posibila folosirea unor fisiere de configurare, unor fisiere de instalare (deploy) si de dezinstalare (undeploy) a serviciului astfel incat aceste operatii de rutina sa se desfasoare in mod automat. Aceste fisiere speciale sunt specifice fiecarui server in parte, astfel ca pentru a afla mai multe informatii trebuie consultat ghidul serverului.

Acest tip de rulare poate fi de doua feluri: rularea intr-un server ce suporta servicii (solutia este IIS si servicii .NET) sau rularea serviciului prin intermediul unei aplicatii web instalata pe un application server (de ex. Apache AXIS ruland pe un server tomcat sau orion).

Solutia cea mai des folosita pentru *servere ce suporta servicii* este combinatia IIS cu servicii .NET (oferita de Microsoft) sau Apache Server cu modul de .NET si servicii .NET. In aceste cazuri, serverul are la dispozitie (deobicei extern) facilitatile oferite de platforma .NET, adica Motorul SOAP nu este incapsulat in server. Spre deosebire de varianta standalone, in acest caz serverul face toata partea de management a serviciilor cat si alegerea acestora, inlocuind rudimentara varianda cu un Registru De Servicii Web.

A doua solutie este mai complexa, fiind necesara instalarea unui Application Server (de exemplu tomcat). In interiorul acestui app server se va instala o aplicatie web (web application) care tine locul serverului de servicii. Prin intermediul acestei aplicatii se face managementul de servicii instalate cat si incarcarea si executarea acestora. In acest caz este necesara prezentarea pas cu pas a traseului pe care il capata o cerere SOAP:

pasul 1) App Serverul primeste o cerere de la client. Nu are relevanta ce tip de cerere este (ar putea sa fie HTTP Post sau un mesaj SOAP). de servicii web.

pasul 2) Pe baza URL-ului pe care il adreseaza, mesajul este trimis la aplicatia Web ce este inregistrata pe acea locatie. Sa presupunem ca am primit un mesaj SOAP adresat aplicatiei web MySoap instalata pe app server. In acest caz mesajul este trimis la punctul de intrare al aplicatiei (in cazul nostru procesorul de cereri).

pasul 3) Procesorul de cereri deserializeaza mesajul cu ajutorul Motorului SOAP continut de aplicatia MySoap.

pasul 4) Procesorul de cereri localizeaza serviciul ce trebuie folosit, il incarca si executa metoda ceruta de client.

pasul 5) Procesorul de cereri primeste rezultatul executarii metodei.

pasul 6) Procesorul de cereri serializeaza rezultatul metodei folosind Motorul SOAP

pasul 7) Procesorul de cereri trimite app serverului raspunsul deja serializat (sub forma unui Stream)

pasul 8) app serverul trimite catre client raspunsul.

Datorita faptului ca majoritatea Application Server-elor nu suporta (inca) instalarea de servicii SOAP, este necesara folosirea unei aplicatii secundare care are ca scop implementarea functionalitatii cerute de specificatiile SOAP. Aceasta aplicatie Web este in schimb scutita de a se mai ocupa de nivelul de transport, acesta fiind responsabilitatea serverului. Despre aceasta insa in partea a doua, cand vom detalia cateva produse de acest tip (Apache SOAP si Apache AXIS).

1.5 Dezvoltarea si viitorul serviciilor Web

1.5.1 Tool-uri

Dupa cum am mentionat mai devreme, deja exista pe piata mai multe tooluri care au ca scop crearea si intretinerea cu efort minim a serviciilor web. Acum avem prilejul sa va enumeram unele dintre ele si sa adaugam cateva cuvinte despre ele.

- Platforma .NET. Dezvoltata de Microsoft. De asemenea se gaseste o implementare Ximian pentru Linux.
- Visual Net. Dezvolatat de Antarctica. Modul pentru Apache Server, printre altele ofera si servicii .NET.
- Apache SOAP. Dezvoltat de Apache Organisation. In mare parte suporta specificatiile SOAP 1.2. Considerat inchis, versiunile ulterioare o sa faca doar bug-fixing, nu si adaugare de functionalitate.
- Apache AXIS. Dezvoltat de Apache Organisation, este urmarea proiectului SOAP. Ofera mult mai multa functionalitate, design refacut, viteza de lucru si folosirea memoriei mai buna.
- kSOAP. Dezvoltat de Enhydra. Ofera suport doar pentru client.

O lista mai mare de astfel de implementari poate fi studiata in partea a doua.

1.5.2 Exemple de servicii Web

Pentru a demonstra ca intr-adevar serviciile Web reprezinta o solutie viabila in lumea tehnologiei informationale, vom prezenta pe scurt cateva servicii ce au fost deja implementate si facute publice pentru utilizatori din orice colt al lumii.

- Universitatea Berkeley din California foloseste servicii Web pentru a integra si imbunatati sistemul de comunicatii. Mai exact pentru a unifica emailul, voice-mailul si mesajele fax in casute postale individuale ce pot fi accesate de proprietar prin intermediul telefonului celular, PDA-ului (Personal Digital Assistant), telefon sau client mail. In acest moment proiectul este in faza de dezvoltare. Pentru mai multe informatii consultati urmatorul URL: http://telecom.berkeley.edu/uc/.

- Foarte multe companii aeriene, de turism, de transport, de inchirieri se folosesc de servicii Web nu numai pentru a oferi clientilor informatii de ultima ora, cat si pentru a integra cat mai usor sisteme diferite. Este cazul firmelor Dollar Rent a Car si Southwest Airlines, care au fondat un parteneriat prin care firma de transport aerian dorea sa dea posibilitatea clientilor sai sa inchirieze o masina prin intermediul siteului propriu. Problema a aparut datorita diferentei de sisteme de operare: compania aeriana foloseste UNIX pe cand cea de inchirieri Windows. Solutia aleasa de Dollar este clara: implementarea unui serviciu Web prin intermediul caruia sa se poata face rezervari pentru masinile sale. Cu aceasta ocazie, acest serviciu va putea fi folosit nu numai de Southwest Airlines dar si de catre alti parteneri.

  Puteti verifica navigand la urmatoarea adresa (pagina pentru inchirieri de masini): http://www.iflyswa.com/cgi-bin/carItinerary.

- Companiile din domeniul financiar au mereu nevoie de valori la zi legate de stocuri, de actiuni samd. Pentru a facilita accesul la astfel de informatii pentru agentii de bursa, a implementat un serviciu Web care returneaza valorile curente ale cotelor. Acest lucru a dus la o mai buna organizare a informatiilor si astfel la un lucru mult mai eficient din partea angajatilor firmei.

- Simplu serviciu folosit pentru traducerea unui text dintr-o limba in alta poate fi gasit la urmatoarea adresa: http://babelfish.altavista.com/.

- Si daca totusi aceasta lista nu v-a convins atunci puteti incerca sa intrati pe urmatorul link si nu veti fi dezamagiti. XMethods este un site cu o lista impresionanta de servicii Web dintre cele mai variate. http://www.xmethods.com/

1.5.3 Noi domenii de explorat

Dupa cum am zis, serviciile Web sunt doar la inceput. Desi foarte promitatoare, ele inca mai au mult pana sa ajunga la deplina maturitate. Pentru aceasta exista organizatii care se ocupa de dezvoltarea de noi specificatii, de noi protocoale si noi metode de utilizare a acestor noi si puternice tehnologii.

Odata cu aparitia serviciilor web, foarte multe lucruri ce pareau greu de obtinut au devenit realizabile.

- accesarea unui stateless EJB (Enterprise Java Bean) prin intermediul SOAP este o realitate. Pentru a accesa un astfel de element (vezi documentatia referitoare la java 2 Enterprise Edition) nu mai este nevoie de complicate manevre de mapare folosind multiple tehnologii, ci doar de un simplu apel de metoda din partea clientului spre a primi rezultatul cerut. Datorita acestei realizari, aplicatiile web bazate pe EJB ofera acum o noua interfata mult mai usor de folosit (fiind posibil accesul chiar si de pe telefoane mobile). Si asta fara a modifica deloc codul initial, doar prin realizarea unei mapari corespunzatoare in fisierele de configurare si de deploy ale Aplicatiei SOAP.

- Conectarea la baze de date folosind direct protocolul SOAP. Cu ajutorul acestei tehnologii nu va mai fi nevoie de incarcare de drivere specifice, de folosire de protocoale sau de punti de conectare (ODBC JDBC bridge, drivere JDBC). Accesul se va face direct, executand comanda SQL ceruta si primind inapoi un rezultat deja mapat ca obiect.

1.6 EXTRA: Un pic de politica

Acest subcapitol are ca scop prezentarea unor informatii despre stadiul in care se afla specificatiile pentru protocoalele ce fac subiectul acestui studiu cat si despre organizatiile ce lucreaza la ele. Dupa care ne vom focaliza asupra problemei credibilitatii organizatiei WS-I si conflictului (mai mult sau mai putin fatis) dintre firmele mamut Microsoft si Sun (evindent, cu referire directa la aceasta problema). Dupa cum probabil v-ati dat seama, decizia de a citi acest subcapitol este lasata la latitudinea cititorului, nefiind prezentate informatii relevante despre protocolul SOAP ci mai mult parerile unor programatori de pe lista oficiala de discutii a Apache AXIS.

1.6.1 Specificatiile

Dupa cum am precizat si intr-un paragraf anterior, stadiul in care se afla specificatiile este una dintre problemele cele mai stringente cu care se confrunta lumea serviciilor Web. Conceptul de serviciu Web a fost pentru prima oara aprofundat de Microsoft prin intermediul produsului .NET. In acelasi timp o alta firma concurenta, IBM a lucrat in stabilirea unor reguli de comunicare pentru servicii Web. Datorita efortului depus in aceasta noua ramura a IT, acesti doi giganti au devenit in scurt timp autaritati in acest domeniu. Iar ca rezultat al acestui efort si al muncii lor de dezvoltare, intelegere si extindere a acestui concept a luat nastere un document care este considerat baza serviciilor web de acum: Specificatiile SOAP 1.1. In plus, pentru a realiza o imagine cat mai completa asupra subiectului in cauza, acelasi grup a dezvoltat specificatiile pentru WSDL 1.1. In schimb specificatiile pentru UDDI au fost si sunt in continuare dezvoltate de organizatia OASIS.

De ce aceste documente reprezinta baza serviciilor Web? Pentru ca toate tool-urile care au fost implementate dupa aparitia acestor specificatii au incercat pe cat posibil sa respecte regulile enuntate in ele.

Desi bine intentionat, grupul de cercetatori (dintre care mare parte erau angajati IBM si Microsoft) se afla la inceput de drum. Ei trebuiau sa construiasca totul de la zero, sa gaseasca formula cea mai potrivita pentru serializarea datelor cat si modul cel mai elocvent de exprimare. Astfel ca au aparut unele probleme: paragrafe nu tocmai clare, care pot duce la diverse interpretari ale textului, lipsuri in exprimare, lispuri in specificarea anumitor parametrii. Datorita acestor probleme (care par marunte la prima vedere) a aparut un numar destul de mare de probleme de interoperabilitate intre diversele implementari ce au iesit pe piata.

Aceste specificatii au fost depuse la consortiul W3C pentru validare si promulgare. W3C a hotarat sa nu valideze aceste specificatii, pe care le-a pastrat sub forma unor note ce pot fi gasite la urmatoarele adrese: `http://www.w3.org/TR/SOAP/` (SOAP 1.1) si `http://www.w3.org/TR/WSDL/wsdl.html` (WSDL 1.1). Chiar daca nici una dintre ele nu a fost acceptata ca standard, fiind singurele specificatii in domeniul SOAP, au devenit regulile de baza pentru adaugarea de suport pentru servicii.

Poate ca ati observat (cu stupoare, cel putin asta a fost reactia noastra) ca Sun (deobicei sinonim cu Java) nu s-a implicat deloc in dezvoltarea serviciilor web prin intermediul SOAP. Chiar au afirmat ca sunt sceptici in privinta acestui nou protocol si au hotarat sa ramana la dezvoltarea de aplicatii si extinderea platformei Java Web Start.

In acelasi timp Sun lucreaza la un alt protocol pentru Enterprise Businnes, numit ebXML (electronic businnes using eXtensible Markup Language), reprezentand un sistem de mesage ce are la baza vocabularul XML pentru serializarea si deserializarea datelor transmise. Specificatiile, create de OASIS (organizatie asemanatoare cu W3C) sunt in faza de conceptie, dar exista proiecte ce implementeaza sau se folosesc de functionalitatea descrisa in draftul de specificatie, dintre care trebuie mentionate unele dintre cele mai

importante: Oracle9i Application Server, BEA WebLogic Integration, Fujitsu INTERSTAGE, IONA Orbix E2A Collaborate si multe altele. Mai multe informatii la siteul oficial ebXML: `http://www.ebxml.org`.

Dar sa ne intoarcem la subiectul acestei lucrari: SOAP. Desi standardul original a fost respins, acesta a fost adoptat de un numar foarte mare de dezvoltatori. Majoritatea se gasesc in USA si Europa. Datorita numarului mare de servicii existente bazate pe acest protocol, SOAP 1.1 este considerat standardul de facto in acest domeniu. Piata asiatica este reluctanta in adoptarea combinatiei SOAP&WSDL&UDDI in favoarea variantei sustinute initial de Sun, ebXML.

Ca urmare a acestei situatii, s-au format doua directii in dezvoltarea si imbunatatirea SOAP. In acest subcapitol vom vorbi in mod special despre prima directie: W3C, desi a respins propunerea venita sub forma standardului SOAP 1.1, a inceput dezvoltarea noii specificatii SOAP 1.2. W3C incearca sa lamureasca problemele ce au aparut in urma primei specificatii. Acest efort se bazeaza pe mai multi factori:

- Studiul atent al specificatiilor SOAP 1.1
- Raspunsul venit de la programatori in urma folosirii toolurilor implementate pe baza specificatiilor initiale
- Intrebarile si sesizarile ridicate de utilizatori la citirea specificatiilor initiale.
- Problemele de interoperabilitate aparute in diverse situatii si intre diferite produse.

1.6.2 SOAPBuilers si WS-I

Aceste doua organizatii reprezinta cea de-a doua directie aleasa de catre comunitatea programatorilor de servicii Web. Ea nu contrazice nu nimic prima directie, ci doar incearca sa transforme interoperabilitatea dintre toolurile SOAP la un punct final.

SOAPBuilders este o organizatie neoficiala care are ca scop transformarea visului interoperabilitatii in realitate. Desi in linii mari interoperabilitatea este realizata, exista unele mici discrepante intre produse. Si acesta este campul in care lucreaza SOAPBuilders. Formata din programatori sau coordonatori ce lucreaza la aproape toate toolurile ce implementeaza specificatiile SOAP, incearca sa stabileasca principalele puncte nevralgice ale comunicarii. Mai mult, se testeaza mereu noile versiuni ale produselor pentru a se verifica ce a fost rezolvat si ce nu din punct de vedere al interoperabilitatii.

WS-I este o alta organizatie oficiala care are un rol asemanator: incearca sa stabileasca niste reguli ce trebuie urmate de programatori pentru a reusi o implementare cat mai portabila si cu un grad sporit de interoperabilitate cu celelalte tooluri. Despre ea vom spune mai multe.

Numele este o abreviere a Web Service Interoperability Organisation si se autocaracterizeaza ca fiind *un efort deschis al companiilor avand ca scop direct promovarea interoperabilitatii serviciilor Web, indiferent de platforme, aplicatii si limbaje de programare. Organizatia inglobeaza mare parte din liderii serviciilor Web cu scopul de a oferi utilizatorilor indrumari, practici recomandate (recomended practices) si resurse pentru realizarea de servicii cu grad sporit de interoperabilitate.*

Organizatia a fost fondata in luna ianuarie a anului 2002 de patru mari companii din domeniul IT: Microsoft, IBM, BEA Systems si Intel. Scopul declarat a fost de a oferi indicatii si sfaturi practice in realizarea unor servicii interoperabile. Odata cu maturizarea ei, WS-I a adunat un numar mare de firme producatoare de software si a inceput o campanie activa de sprijinire a dezvoltarii serviciilor web catre o mai buna colaborare.

Spre deosebire de SOAPBuilders, WS-I se focalizeaza mai mult asupra codului incercand sa identifice niste modele standard de a implementa servicii cat mai stabile. In urma acestor cercetari, au inceput redactarea unui document intitulat WS-I Basic Profile. La momentul redactarii acestei lucrari, documentul era inca in stadiul de draft.

Dupa cum se si vede, aceasta companie este independenta. Desi multe alte surse au interpretat implicarea Microsoft in crearea ei ca fiind o incercare de acaparare a specificatiilor serviciilor Web si de promulgare a propriilor lor protocoale, acest lucru nu este adevarat. Ca dovada, organizatia este 'condusa' de un numar de noua oameni (la ora actuala) fiecare fiind angajat al unei companii ce s-a implicat in dezvoltarea SOAP. Microsoft nu detine decat un singur scaun.

Adevarata problema a aparut datorita faptului ca Sun nu a fost invitata sa adere la aceasta organizatie. Acest lucru a fost apoi explicat de organizatie prin modul indiferent cu care Sun trata protocolul SOAP, incercand pe cat de mult cu putinta sa favorizeze propriul protocol (ebXML). Acest lucru a fost realizat si prin interventia Microsoft, care a conditionat prezenta sa in consortiu de excluderea participarii firmei Sun. (marturie stau documentele din procesul antitrust intentat companiei Microsoft: `http://news.zdnet.co.uk/story/0,,t288-s2110205,00.html`).

Acest lucru este pe cale sa fie repatat, Sun afirmand ca este de acord sa faca parte din membrii ce contribuie la WS-I. In acelasi timp si-a afirmat intentia de a candida pentru cele doua noi locuri adaugate la comitetul director. Pentru mai multe detalii: `http://www.aspnews.com/news/article/0,,4191_1488041,00.html`.

Dupa cum se stie, pentru a obtine interoperabilitatea este nevoie nu numai de specificatii cat mai clare, dar si de companii mature, care sa fie in stare sa treaca peste divergentele de opinie si peste mandrie si sa se alature in incercarea de a realiza produse compatibile si usor de folosit de catre programatori.

Capitolul 2

Ce este SOAP? Primii pasi (teorie si introducere)

ce inseamna soap

cum a aparut

celelalte RPC si encoding

avantaje si dezavantaje soap

encodyng soap si tipuri de acces (trebe documentat mult, interesant din mailing lists)

modelul de baza (client, soap, server cu imagine si subliniere Transport layer, independent!!!)

din ce este compus (XML de fapt si mapare de obiecte)

2.1 EXTRA

un exemplu de mesaj soap.

Capitolul 3

XML folosit in SOAP

de ce acest capitol. ? de ce a fost folosit xml ? structura de baza a doc xml specificarea formei unui document xml (DTD, XSM) XML PARSING : DOM si SAX models + extension pentru kxml. XML si soap (cu exemple dar nu foarte amanuntit explicate) explicarea atenta a namespace folosite in soap (pe exemple)

EXTRA:

link la appendix de XML detailed (care va avea xml, dtd, xslt basics)

XML parsing with java

Capitolul 4

Basic soap (Hai sa SOAP)

+ explica ce inseamna SOAP calls si tipurile lor: rpc sau messaging (tre cautat) + detaliaza ideea de rpc si ideea de messaging (+comparare cu CORBA si RMI) + Structura unui mesaj soap si detaliere ce contine fiecare + maparea tipurilor de baza (din soap specs) + maparea tipurilor complexe + maparea array

Capitolul 5

SOAP advanced

(trebe cautat in specs ce este mai complicat si ce ridica probleme) :o)

Capitolul 6

In primul rand interoperabilitatea

si acum limbaje: interop? Cum si de ce + detaliez interop si de ce este utila. + adaug concept de .NET si independenta de limbaj + cum este obtinuta interop de SOAP : text si XML, mapari independente de limbaj + probleme datorate specificatii slabe + problemele curente de interoperabilitate

Capitolul 7

de la interop la WSDL

+ cum de a aparut ==> interop + de cine este dezvoltat (inceput si actual) Microsoft + IBM , acum W3C + detalierea cu scheme cum este WSDL in WEB services + detaliem ce inseamna un fisier wsdl

Capitolul 8

Pe scurt UDDI

+ daca tot le avem descrise, cum le publicam si cum gasim? + starea UDDI si proiecte asemanatoare + pe scurt UDDI

Partea II

IMPLEMENTARI SOAP

Capitolul 9

Structura generala a unui server si client SOAP

+ despre arhitectura unui server + despre arhitectura pentru client + studiu comparativ si apoi ce module folosite impreuna shamd

Capitolul 10

Lista de implementari

+ in primul rand limbajul + apoi o lista de implementari si cu cat mai multe informatii despre

Capitolul 11

Sa introducem apache

scurt capitol despre proiectele apache. (Avand in vedere ca aceasta lucrare este axata pe produsele server realizate de organizatia Apache, sa spunem cateva cuvinte despre ei) .

Capitolul 12

APACHE SOAP

+ ce ofera + de unde se downloadeaza si linkuri la documentatie si lista de discutii + cum se instaleaza si pe ce... tomcat, orion and so on + simplu exemplu de cum se dezvolta un serviciu si cum se deploy pe tomcat + un serviciu mai complicat care pastreaza si session tracking !!! (sper sa mearga)

Capitolul 13

Apache AXIS pentru inceput

+ spun la inceput ca se trage din apache soap. Si spun ca starea lui nu este chiar buna :o) lol + ce ofera (detaliat) + de unde se downloadeza si linkuri catre docs (greu aici ca nu prea au) + instalarea + cum se face un serviciu simplu (jws) + deploy + cum se face un serviciu normal + deploy (fisier wsdd pe scurt) + cum sa folosim wsdl din axis EXTRA: + ce merge bine + ce nu este prea solid

Capitolul 14

Axis advanced

+ arhitectura axis + conceptul de handler + fisierele WSDD detailed (deploy files) + Session management cu axis : cu cookies (Default) si cu SOAP headers (nu este inca spec, dar suporta) + Exemplu pentru session management din perspectiva client si server + EJB binding !!! habar nu am cum se face dar merge !!! EXTRA: + ce probleme apar si unde + de ce este bun

Capitolul 15

Si acum clientii: ksoap.

+ detaliere despre ksoap & enhydra (poate si appendix cu enhydra platform) + structura ksoap + cum merge si pe j2me si schimbari pentru SE + exemplu (care se conecteaza la apache soap, cat si la apache axis EXTRA: cum modificam ksoap pentru java2xml ;o) kxml pe scurt

Partea III

PITA system

Capitolul 16

Problema generala

Capitolul 17

Solutia aproximativa (design) pentru problema generala

Capitolul 18

Problema particulara: interactziunea cu web

Capitolul 19

Solutiile propuse

+ implementarea incrementala a proiectului + intai versiunea statica + descrierea pe scurt a ceea ce trebuie sa faca versiunea statica + versiunea dinamica + noile cerinte adaugate de versiunea dinamica

Capitolul 20

PITA basics

+ inceputul designului + gasirea modulelor de baza + diagrama modulelor + explicarea lor si intercomunicarea dintre ele

Capitolul 21

WEB interface

Detalierea web interface

Cum am ajuns la solutia asta

Cum arata implementarea

Clasele (diagrama)

Capitolul 22

Modulul de obtinere Route (routeplanner)

Capitolul 23

modulul de Agents

+ cind si cum se lanseaza + ce trebuie sa faca un agent + cum face + structura de clase (UML)

Capitolul 24

Modulul thread de verificare a delay (include si adaptoare de interogare)

Capitolul 25

Module auxiliare

+ modulul DB + modulul de atentionare user (User notifyer)

Capitolul 26

Baza de date: structura si decizii de stocare

Capitolul 27

Clientul pita

+ cum trebuie implementat un client PITA + succesiunea de metode : login abia apoi restul + session tracking + detaliere despre cum am implementat + class design