Universitatea Politehnica din Bucuresti. Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Ingineria software orientată pe servicii web

Grupa: 441A

Studenti : Șerbănescu George Daniel
Lăcătușu Raluca

-- Bucuresti 2013 --

Cuprins

Impartire cuprins pe studenti :

Lăcătușu Raluca : capitolele 1,2,5

Șerbănescu George: capitolele 3,4,6

1. Servicii web. Definitie

Prin serviciu web se intelege un serviciu pus la dispozitie utilizatorilor pe Internet. Standardele disponibile si protocoalele au creat posibilitatea comunicarii intre aplicatii aflate la distante mari, cu acces la Internet. Astfel, exista sisteme ce ofera servicii de informare si procesare a informatiilor care in general sunt independete de platforma hardware, accesul la acestea facandu-se prin servicii web.

Exista mai multe definitii general valabile in ceea ce priveste un web service.

A. O componenta software descrisa printr-un modul WSDL care ofera posibilitatea sa fie accesata folosind protocoale de retea standard de genul SOAP si HTTP.

B. Un software care se pune la dispozitie pe Internet si care foloseste un sistem de mesaje standardizat baza pe XML. Pentru a gasi serviciul dorit si interfata publica a acestuia trebuie sa existe mecanisme simple.

C. O colectie de protocoale si standarde folosite pentru schimbul de date intre aplicatii sau sisteme. Aplicatiile software scrise in limbaje de programare diferite si care ruleaza pe diverse platforme pot folosi serviciile web pentru a face schimb de date in reatea, intr-o maniera oarecum asemanatoare comunicarii intre procesele de pe un singur calculator. Interoperabilitatea se datoreaza folosirii unor standarde publice adecvate.

In plus, alte caracteristici care se doresc de la un serviciu Web:

- sa se auto-descrie: un serviciu publicat ar trebui sa aiba publicata si o interfată publica a sa. De obicei si interfata e descrisa in format XML (ex pentru servicii SOAP);
- sa poata fi descoperit: sa fie publicat, sa existe mecanisme prin care aplicatiile interesate sa poata descoperi serviciul si localiza interfata acestuia.

Dacă se continuă în direcţia unor servicii care se pot uşor descoperi, care se auto-descriu şi care respectă standardele bine stabilite, atunci se va putea ajunge la integrarea automată a aplicaţiilor (care nu necesită intervenţia unui programator / utilizator uman):

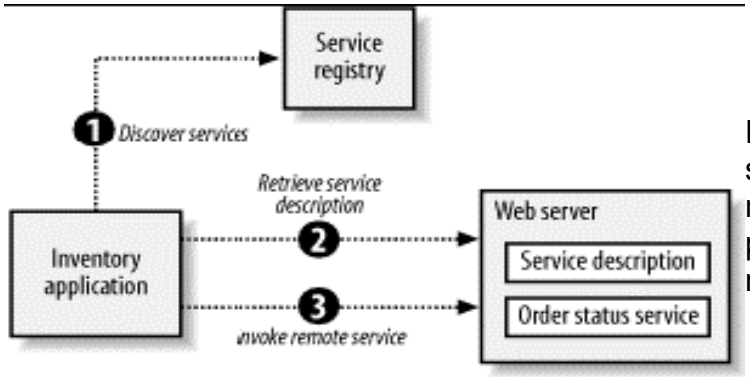

Fig1. Intelegerea rezultatelor si selectarea unui serviciu adecvat necesita inca interventia umana. In plus, exista aspecte ale relatiilor care nu se pot automatiza.

Există mai multe framework-uri propuse pentru servicii Web, care însă partajează un set comun de tehnologii: SOAP, WSDL, UDDI.

Un exemplu des intalnit in Web este RSS, Really Simple Sindication, care se doreste a fi un serviciu prin care utilizatorii pot afla stirile lor facvorite in timp real de la cei care le furnizeaza. RSS-urile sunt lar acceptate sunt interpretate de orice platforma

conectata la internet, fie ca este vorba de un laptop sau un telefon mobil. RSS-urile sunt calea instatnta pentru stiri.

Clienții trebuie să fie în totalitate compatibili cu serviciul pe care vor să îl folosească. Compatibilitatea în acest context nu înseamnă doar că, clienții trebui să înțeleagă formatul mesajelor și ordinea în care acestea trebuie transmise, ci și că ei trebuie să fie compatibili cu alte cerințe importante, cum ar fie faptul că informația trebuie criptată sau că trebuie verificat faptul că nu s-a pierdut informație în timpul transmisiei. În ceea ce privește serviciile, cerințele non-funcționale sunt definite prin intermediul regulilor, care nu sunt scrise ca și parte a documentației unui serviciu.

Regulile sunt un set de declarații care pot fi interpretate de o aplicație și care oferă informații despre cerințe precum securitatea și fiabilitatea. Regulile pot fi înglobate în contractul unui serviciu sau pot fi stocate într-o baza de date specializată, de unde pot fi obținute dinamic în timpul rulării.

Contractele bazate pe reguli pot fi privite ca și o parte a documentației serviciului, dar în același timp ele pot fi folosite de utilitare pentru a genera cod compatibil atât pentru partea de client cât și pentru cea de serviciu. De exemplu, o regulă de securitate poate fi folosită pentru a genera cod care va verifica, că mesajele primite sunt criptate, va decripta mesajele și le va trimite către aplicația serviciu.

Separarea regulilor de contracte permite aplicațiilor client să se adapteze dinamic pentru a respecta cerințele unui anumit serviciu. Acest lucru se va dovedi foarte util pe măsură ce serviciile sunt standardizate și sunt oferite de diferiți furnizori.

Sunt bazate pe o structura tipizata in XML care contine, titlul, link si descriere, alaturi de informatiile de timp si data.

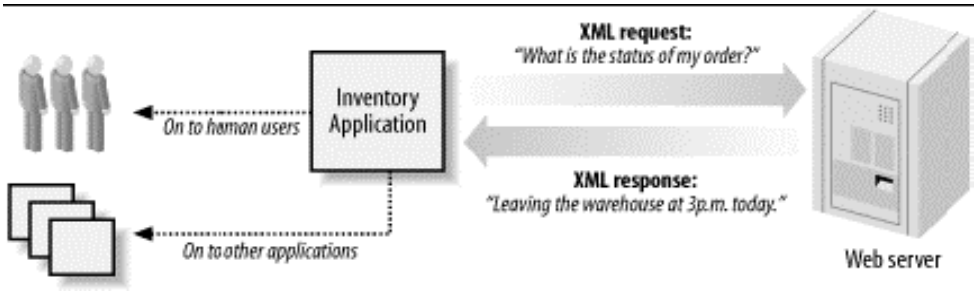

Fig2.

2. Istoric

Documentul de fata trateaza problema technologiei obiectelor distribuite care poate fi inteleasa ca web services.

De la aparitia internetului, prima retea creata de ARPANET in anul 1969 si care a conectat calculatoarele de la Stanford Research Institute, University of California Los Angeles, University of California Santa Barbara si University of Utah, se poate discuta de perioada pre – SOA (Service – Oriented Arhitecture).

SOA a aparut ca un concept in anii 1990, s-a suprapus cu dezvoltarea CORBA, limbraj de programe obiect orientata.

Arhitecturile pw servicii Web reprezinta ultimul pas in dezvoltarea tehnologiilor middleware. Aceste tehnologii rezolva problema inter-operabilitatii si ofera baza pentru dezvoltarea de aplicatii Internet de mari dimensiuni.

Tehnologiile middleware care permit integrarea aplicatiilor sunt folosite în diferite scopuri, de la interconectarea componentelor unei aplicatii desktop sau Web, până la dezvoltarea de sistem software care se intind peste Internet. Tehnologiile traditionale precum serverele de aplicatii J2EE sau cele bazate pe mesaje reprezinta soluții ideale pentru dezvoltarea de sistem software care ruleaza în cadrul unei singure organizatii.

Totusi, ele sunt destul de limitate cand se pune problema interconectarii sistemelor

software folosite de organizatii diferite, care sunt conectate prin Internet. Serviciile Web si arhitecturile bazate pe servicii sunt proiectate tocmai pentru a satisface aceste nevoi.

In multe feluri tehnologiile bazate pe servicii Web nu reprezintă o noutate. La fel ca restul tehnologiilor si arhitecturilor care oferă suport pentru calcul distribuit, principalul scop al tehnologiilor bazate pe servicii este acela de a oferi suport pentru un sistem software de a invoca functionalități oferite de un alt sistem software (asa cum J2EE permite clientilor Java sa apeleze metode implementate de componente J2EE).

Principala diferenta consta în faptul că accentul in cazul arhitecturilor bazate pe servicii se pune accentul pe interoperabilitate si rezolvarea problemelor ridicate de folosirea de platforme si limbaje diferite. Desi se poate dezvolta o arhitectură bazata pe servicii utilizand orice tehnologie care permite calcul distribuit, numai serviciile Web oferă un grad de interoperabilitate nelimitat.

Necesitatea pentru interoperabilitate izvoraste din faptul ca majoritatea companiilor mari dețin un mix de sisteme software in ceea ce priveste limbajele de programare și platformele utilizate. In conditiile in care reimplementarea acestor sisteme pe o singura platformă este mult prea costisitoare si prea riscant este evidenta necesitatea pentru tehnologii middleware care sa permita comunicare între aceste sisteme software. Nu de putine ori se intampla ca sistemele software care trebuie sa comunice sa fie dezvoltate pentru platforme incompatibile, de aceea este nevoie de interoperabilitate.

Serviciile Web si arhitecturile bazate pe servicii reprezinta o solutie care permite integrarea diferitelor tehnologii.

3. Cum functioneaza ?

Toate aplicatiile de tipul tehnologiilor de integrare, inclusiv serviciile Web si alternativele la acestea precum J2EE sau CORBA, ofera de fapt doar patru functii de bază care permit urmatoarele:

- descoperirea serviciului potrivit (utilizand fie UDDI fie un alt serviciu de localizare);
- descoperirea interfetei unui serviciu (utilizand WSDL);
- trimiterea unei cereri către un serviciu (utilizand SOAP) ;
- utilizarea serviciilor precum securitatea (utilizand standardul WS-*);

SOAP, WSDL și UDDI au fost primele standarde publicate, dar ele nu indeplinesc decat cerinte de baza. Nu ofera suport pentru securitate, tranzactii, fiabilitate si alte functii importante. Acest vid a fost insa umplut prin definirea unor standarde intitulate “WS-*”, primele astfel de standarde au fost definite de către IBM și Microsoft.

Serviciile Web sunt standarde bazate pe XML. Serviciile sunt definite utilizand XML, aplicatiile cer servicii trimitand mesaje XML, practic serviciile Web ca standard folosesc o

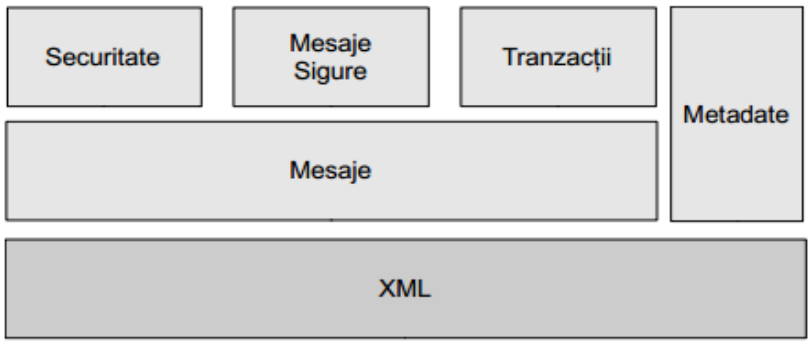

Fig. Standarde definite pentru serviciile Web.

serie de standarde XML existente. In ansamblu serviciile sunt cat de simple se poate,

avand in vedere faptul ca ele trebuie sa ofere suport pentru securitate, robustete si interoperabilitate. De asemenea exista o serie de utilitare si librarii, care ofera suport pentru aceste standarde, astfel ca programatorii trebuie sa înteleaga doar capabilitatile acestor standarde si nu detalii de sintaxa XML. În figura de mai jos sunt prezentate principalele tipuri de standarde definite pentru serviciile Web.

Unul dintre principiile care sta la baza serviciilor Web este acela ca diferitele campuri si atribute care sunt folosite pentru a suporta funcționalități precum securitatea sau fiabilitatea sunt total independente unul de celalalt. Astfel aplicatiile trebuie sa includa doar acele atribute care sunt importante pentru functionalitatea dorita, celelalte putand fi ignorate.

Un alt obiectiv al serviciilor Web este acela de a oferi suport pentru arhitecturi de sisteme care folosesc “intermediari”. Astfel, in loc sa se presupuna ca un client trimite întotdeauna un mesaj direct către un serviciu, modelul cu intermediari presupune că aceste mesaje pot să traverseze un lanț de aplicații până să ajungă la destinație. Aceste aplicații intermediare pot să facă o serie de operații cu mesajele primite, de exemplu, pot să logheze informația, să verifice securitatea sau chiar să modifice mesajele.

Standardele existente pentru serviciile Web oferă suport pentru arhitecturi bazate pe intermediari în diverse feluri.

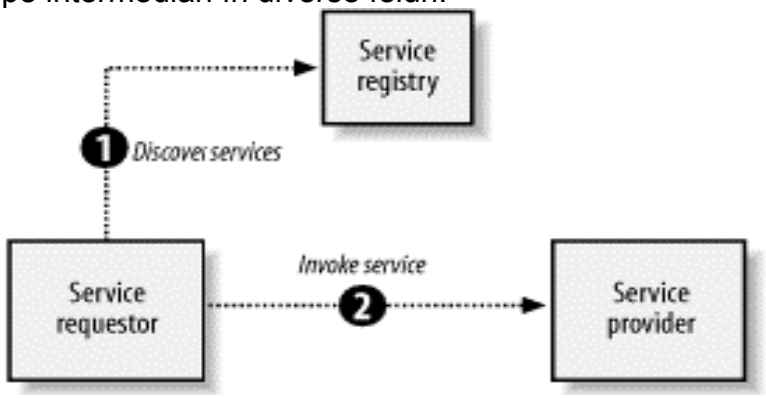

Fig3. Arhitectura serviciilor Web

I. Din perspectiva rolurilor participanţilor :

-- Furnizorul serviciului – implementeaza serviciul si-l face disponibil în Internet ;
-- Solicitantul serviciului – utilizeaza un serviciu Web, pe care-l solicita deschizand o conexiune şi trimitând o cerere XML ;
-- Registru de servicii .

II. Din perspectiva stivei de protocoale implicate in servicii Web:

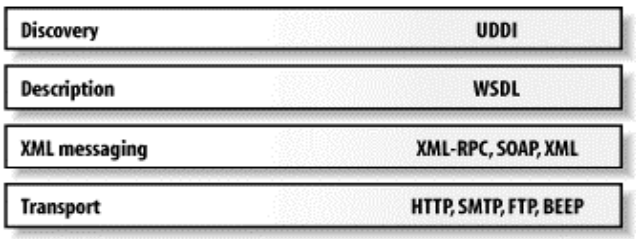

Fig4

--Descoperire servicii: responsabil cu centralizarea serviciilor într-un registru comun, şi furnizarea unei funcţionalitati de publicare / regasire facile; via Universal Description, Discovery, and Integration (UDDI) (http://www.xmethods.net/) ;
--Descrierea serviciilor: responsabil cu descrierea interfetei publice a unui anumit serviciu; via Web Service Description Language (WSDL).

--Transmiterea mesajelor în format XML: responsabil cu codarea mesajelor într-un format XML comun, astfel incat acestea sa fie intelese de furnizor si solicitant; via XML-RPC şi SOAP ;
--Transport: responsabil cu transportul mesajelor între aplicatii; HTML, SMTP, FTP .

Etapele crearii si consumarii serviciilor Web:

A. Din perspectiva furnizorului serviciului
-- Dezvoltarea nucleului funcţional al serviciului Web care va fi oferit;
-- Dezvoltarea unui wrapper (SOAP de exemplu) pentru nucleul serviciului;
-- Furnizarea descrierii serviciului (un fişier WSDL pentru serviciile SOAP);
-- Desfăşurarea (deployment-ul) serviciului – poate consta în integrarea cu un server Web;
-- Publicarea existenţei şi specificaţiilor serviciului respectiv – de obicei, se publică într-un director UDDI global, public sau privat.

B. Din perspectiva clientului
-- Identificarea si descoperirea serviciilor relevante aplicatiei client – prin cautarea în registre (directoare) de servicii publice/private;
-- Odata identificat serviciul dorit, se localizeaza descrierea acestuia (daca e serviciu SOAP, atunci exista un document WSDL).;
-- Crearea aplicatiei client;
-- In final, executarea aplicatiei client pentru a invoca efectiv serviciul Web.

4. Protocoale utilizate

4.1 WSDL

WSDL-ul este folosit pentru a descrie serviciile Web, si anume interfata, metodele si parametrii. WSDL-ul este suportat de medii de dezvoltare precum Visual Studio sau WebSphere. Aceste utilitare pot genera o descriere WSDL direct din codul sursa, respectiv pot genereze cod care apeleaza serviciul pe baza unei astfel de descrieri.

Serviciile bazate pe standardul SOAP sunt descrise de documente WSDL (Web Service Description Language) si pot fi localizate cautând intr-un director UDDI (Universal Description, Discovery and Integration). Serviciile pot sa specifice cerinte precum securitatea si fiabilitatea prin asa numitele declaratii policy, definite prin intermediul platformei WS-Policy și prin declaratii policy specializate, cum ar fi de exemplu WS-SecurityPolicy. Aceste declaratii pot fi atasate unei definitii de serviciu WSDL sau pot fi stocate într-o baza de date separata de unde pot fi accesate utilizând WS-MetadataExchange.

Exemplu de pentru WSDL:

```
<?xml version="1.0"?>
<!-- root element wsdl:definitions defines set of related services -->
<wsdl:definitions name="EndorsementSearch"
  targetNamespace="http://namespaces.snowboard-info.com"
  xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
  xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <!-- wsdl:types encapsulates schema definitions of communication types; here
using xsd -->
  <wsdl:types>
    <!-- all type declarations are in a chunk of xsd -->
    <xsd:schema targetNamespace="http://namespaces.snowboard-info.com"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema">
```

```
      <!-- xsd definition: GetEndorsingBoarder [manufacturer string, model
string] -->
      <xsd:element name="GetEndorsingBoarder">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="manufacturer" type="string"/>
            <xsd:element name="model" type="string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <!-- xsd definition: GetEndorsingBoarderResponse [... endorsingBoarder
string ...] -->
      <xsd:element name="GetEndorsingBoarderResponse">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="endorsingBoarder" type="string"/>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
      <!-- xsd definition: GetEndorsingBoarderFault [... errorMessage
string ...] -->
      <xsd:element name="GetEndorsingBoarderFault">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="errorMessage" type="string"/>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <!-- wsdl:message elements describe potential transactions -->
  <!-- request GetEndorsingBoarderRequest is of type GetEndorsingBoarder -->
  <wsdl:message name="GetEndorsingBoarderRequest">
    <wsdl:part name="body" element="esxsd:GetEndorsingBoarder"/>
  </wsdl:message>
  <!-- response GetEndorsingBoarderResponse is of type
GetEndorsingBoarderResponse -->
  <wsdl:message name="GetEndorsingBoarderResponse">
    <wsdl:part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
  </wsdl:message>
  <!-- wsdl:portType describes messages in an operation -->
  <wsdl:portType name="GetEndorsingBoarderPortType">
    <!-- the value of wsdl:operation eludes me -->
    <wsdl:operation name="GetEndorsingBoarder">
      <wsdl:input message="es:GetEndorsingBoarderRequest"/>
      <wsdl:output message="es:GetEndorsingBoarderResponse"/>
      <wsdl:fault message="es:GetEndorsingBoarderFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- wsdl:binding states a serialization protocol for this service -->
  <wsdl:binding name="EndorsementSearchSoapBinding"
                type="es:GetEndorsingBoarderPortType">
    <!-- leverage off soap:binding document style @@@(no wsdl:foo pointing at
the soap binding) -->
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <!-- semi-opaque container of network transport details classed by
soap:binding above @@@ -->
    <wsdl:operation name="GetEndorsingBoarder">

      <!-- again bind to SOAP? @@@ -->
      <soap:operation soapAction="http://www.snowboard-
info.com/EndorsementSearch"/>
```

```
      <!-- furthur specify that the messages in the wsdl:operation
"GetEndorsingBoarder" use SOAP? @@@ -->
      <wsdl:input>
        <soap:body use="literal"
                   namespace="http://schemas.snowboard-
info.com/EndorsementSearch.xsd"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"
                   namespace="http://schemas.snowboard-
info.com/EndorsementSearch.xsd"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
    <!-- connect it to the binding "EndorsementSearchSoapBinding" above -->
    <wsdl:port name="GetEndorsingBoarderPort"
               binding="es:EndorsementSearchSoapBinding">
      <!-- give the binding an network address -->
      <soap:address location="http://www.snowboard-info.com/EndorsementSearch"/>
    </wsdl:port>
  </wsdl:service>
 </wsdl:definitions>
```

4.2 SOAP (Simple Object Access Protocol)

SOAP a fost primul standard definit pentru serviciile Web, el fiind si în prezent cel mai important standard si totodata cel mai raspandit. Acest standard specifica un protocol de comunicare bazat pe XML simplu si extensibil.

Este echivalentul Java RMI, dar mult mai simplu si de departe mult mai usor de implementat. SOAP este un standard suficient de simplu astfel incat el poate fi cu usurinta implementat de programatori.

Această simplitate rezulta din faptul ca sunt evitate probleme complexe cum ar fi, un Colector de Memorie distribuit sau trimiterea obiectelor prin referința. Tot ceea ce standardul SOAP este sa defineasca un protocol de comunicare bazat pe mesaje simplu si extensibil, care ss permita invocarea serviciilor aflate la distanța utilizsnd protocoale precum HTML, SMTP, UDP sau orice alt protocol.

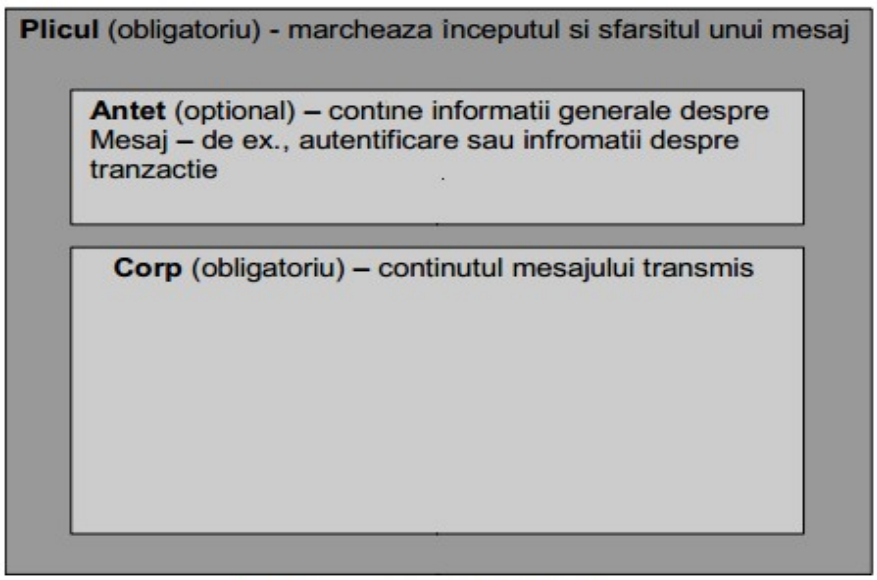

Fig. Structura unui mesaj SOAP.

In figura de mai sus este prezentata structura unui mesaj SOAP. Antetul mesajului contine informatie despre incarcarea mesajului, poate sa includa informații de securitate sau informații referitoare la o tranzactie.

Corpul mesajului reprezinta de fapt continutul mesajului. Standardul SOAP nu impune ce informatie sa fie inclusa in antet, astfel ca el poate fi extins de noi standarde, ca de exemplu, WS-Security, care pot sa defineasca noi elemente pentru antet fara sa fie nevoie sa modifice standardul SOAP.

Exista o serie de standarde care intra în categoria de standarde Mesaje, de exemplu, WS-Addressing și WS-Eventing. WS-Addressing oferă un mecanism de adresare independent de canalul de transport.

WSEventing ofera suport pentru modelul public-subscrib, el defineste formatul de mesaj care reprezinta o cerere de subscriere si pe care clientii il pot trimite pentru a subscrie la un anumit topic. Mesajele publicate care respecta un anumit filtru sunt trimise catre clientii subscrisi utilizand mesaje SOAP simple.

Mesajele SOAP Request si SOAP Response peste protocolul HTTP sunt sincrone.

SOAP Request

Primele linii sunt parte din cererea HTTP. Acestea indica ca s-a facut o cerere HTTP POST la URL de mai jos, apoi se specifica continutul si tipul continutului. Valoarea “ ” a parametrului SOAPAction indica ca cererea este specificata de URL-ul insusi – /axis/HelloWorld.jws. In final este afisat continul XML al mesajului SOAP.

```
POST/axis/HelloWorld.jws HTTP/1.0
Content-Length: 492
Host: localhost
Content-Type: text/xml; charset=utf-8
SOAPAction: " "
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 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:Body>
  <getHelloWorldMessage>
   <name xsi:type="xsd:string">AXIS</name>
  </getHelloWorldMessage>
 <SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

Raspunsul SOAP legat (binded) de HTTP este urmatorul.

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

```
Content-Length: 508
Date: Sat, 16 Mar 2002 04:01:50 GMT
Server: Apache Tomcat/4.0.3 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
  <getHelloWorldMessageResponse SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">
  <getHelloWorldMessageResult xsi:type="xsd:string">
Hello World to AXIS</getHelloWorldMessageResult>
    </getHelloWorldMessageResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.3 REST

Representational State Transfer (REST) este un stil arhitectural software pentru sisteme hipermedia distribuite, cum ar fi world wide web. Termenul a fost folosit pentru prima dată în lucrarea de doctorat "Architectural Styles and the Design of Network- based Software Architectures" scrisă de Roy Fielding, unul din principalii autori a specificatiilor protocolului HTTP.

REST se refera strict la o colectie de principii arhitecturale intr-o retea care subliniază felul in care resursele sunt definite si adresate. Intr-un context mai putin riguros, REST descrie orice interfata care transmite date prin HTTP fara un nivel aditional de mesagerie cum ar fi SOAP sau folosirea sesiunilor prin cookie-uri HTTP. Este posibil ss dezvoltam sisteme software in concordata cu stilul arhitectural REST fara sa folosim HTTP si fara sa interactionam pe World Wide Web. De asemenea, este posibil sa dezvoltam un sistem software care sa foloseasca HTTP si XML care sa nu respecte principiile REST, urmand în schimb modelul arhitectural RPC (remote procedure call).

Principii REST

I.Resurse

Un concept important care tine de arhitectura REST este existenta resurselor. In mod uzual o resursa este o entitate care poate fi stocata intr-un computer si poate fi reprezentata printr-un sir de biti: un document, un rand dintr-o baza de date, sau rezultatul rularii unui algoritm. Fiecare resursa are atasat un identificator global (un URI). Pentru a manipula resursele, componentele retelei (clientii si serverele) comunica printr-o interfata standardizata (de exemplu HTTP) si schimba reprezentari ale resurselor (documentele care transporta informatiile). De exemplu, o resursa care este un cerc ar putea accepta şi returna o reprezentare care specifică un centru si o rază, formatată în SVG, dar poate

accepta si returna o reprezentare care specifica trei puncte diferite situate pe cerc, separate prin virgula.

Iată câteva posibile resurse:

- Versiunea 3.2 a unui release software
- Ultima versiune a unui release software
- O hartă rutiera a Bucurestiului
- Cateva informatii generale despre filozofie
- Următorul număr prim după 13

II. URI

Orice resursa trebuie sa aiba cel puţin un URI. URI-ul este numele si adresa resursei. Dacă o informatie nu are un URI, nu este o resursa si nu putem spune ca se afla pe Web.

URI-urile trebuie sa fie descriptive. Trebuie sa existe o corespondența intuitiva între un URI și o resursa. Iata cateva exemple bune de URI-uri pentru resursele introduse mai devreme:

- http://www.xwiki.com/enterprise/releases/2.0.1.tar.gz
- http://www.xwiki.com/enterprise/releases/latest.tar.gz
- http://www.harti.ro/harta/rutiera/Romania/Bucuresti

URI-urile trebuie sa aiba o structura, adica sa varieze într-un mod predictibil. Daca un client cunoaste structura URI-urilor unui serviciu, isi poate crea propriile puncte de intrare in acel serviciu si obtine informatii intr-un mod eficient.

III. Adresabilitate

O aplicatie este adresabila daca expune aspecte interesante legate de setul sau de date ca si resurse. Cum resursele sunt expuse prin URI-uri, o aplicaţie adresabilă expune un URI pentru orice informaţie care s-ar putea dovedi de folos. De obicei, se ajunge la un număr foarte mare de URI-uri.

Adresabilitatea este una din cele mai bune caracteristici ale unei aplicaţii web. Permite clienţilor să folosească aplicaţii web in feluri la care dezvoltatorii nu s-au gandit. Urmând acest principiu în dezvoltarea unui serviciu web, utilizatorii serviciului vor beneficia de mai multe avantaje ale REST. Din acest motiv serviciile REST-RPC sunt atât de comune: combină adresabilitatea cu modelul programării bazat pe apeluri de funcţii.

IV. Fără stare

Lipsa stării se traduce prin faptul că orice cerere HTTP se întâmplă într-o izolare completă. Atunci când un client face o cerere HTTP, include informaţiile necesare pentru ca serverul să îndeplinească acea cerere. Serverul nu se bazează niciodată pe cereri precedente.

Mai practic, putem considera lipsa stării în termeni de adresabilitate. Adresabilitatea ne obligă să expunem orice informaţie utilă ca şi resursă cu un URI propriu. Pentru a respecta principiul lipsei stării trebuie să definim ca resurse şi stările posibile pe care le poate avea serverul. Clientul nu trebuie să aducă serverul într-o anumită stare pentru a-l face mai receptiv unei anumite cereri.

V. Reprezentări

Atunci când împărţim aplicaţia în resurse, mărim zona de interacţiune cu clienţii. Utilizatorii îşi pot construi un URI potrivit pentru a accesa exact datele de care au nevoie. Dar resursele nu sunt datele, ci doar ideea arhitectului despre cum să împartă datele. Un

server web nu poate transmite o idee, trebuie să trimită octeţi, într-un anumit format de fişier, care este reprezentarea unei resurse.

O resursă este o sursă de reprezentări, iar o reprezentare este formată din date despre starea curentă a unei resurse. Multe resurse sunt ele însele date (cum ar fi o listă de buguri deschise), deci o reprezentare evidentă a unor astfel de resurse sunt chiar datele. Serverul ar putea prezenta lista bugurilor deschise printr-un document XML, o pagină web sau text simplu. Valoarea vânzărilor din trimestrul al patru-lea poate fi reprezentat numeric sau printr-un grafic. Pentru orice resursă putem alege între mai multe tipuri de reprezentări.

VI. **Interfaţa uniformă**

În spaţiul World Wide Web, sunt doar câteva acţiuni pe care le poţi face cu o resursă. HTTP oferă patru metode de bază pentru cele mai comune operaţii:

- Returnarea unei reprezentări a unei resurse: HTTP GET
- Crearea unei noi resurse: HTTP PUT unui nou URI, or HTTP POST unui URI existent
- Modificarea unei resurse: HTTP PUT unui URI existent
- Ştergerea unei resurse: HTTP DELETE

Pentru a obţine sau a şterge o resursă, clientul trimite o cerere GET sau DELETE asupra URI+ului resursei. În cazul unei cereri GET, serverul returnează o reprezentare în corpul răspunsului. Pentru o cerere DELETE, corpul răspunsului poate conţine un mesaj de stare, sau nimic.

VII. Siguranţa şi idempotenţa

Dacă în dezvoltarea unui serviciu web folosim ca interfaţă uniformă HTTP, obţinem două proprietăţi folositoare. Când sunt folosite corect, cererile GET şi HEAD sunt sigure. Iar cererile GET, HEAD, PUT şi DELETE sunt idempotente.

O cerere GET sau HEAD se face pentru a citi anumite date, nu pentru a schimba ceva pe server. Clientul poate să facă o cerere GET sau HEAD de zece ori, iar resursa se comportă de parcă nu s-ar fi făcut nici o cerere. Faptul că o cerere este sigură se traduce prin faptul că doar se returnează o reprezentare a resursei. Un client ar trebui să aibe posibilitatea de a trimite cereri GET şi HEAD asupra unui URI necunoscut, fără să se întâmple ceva nedorit.

Idempotenţa este o idee un pic mai greu de perceput decât siguranţa. Ideea vine din matematică, unde o operaţie este idempotentă dacă are acelaşi efect chiar dacă o aplici de mai multe ori. Multiplicarea unui număr cu zero este idempotentă: 4 ×0 ×0 ×0 este acela;i lucru cu 4 ×0. Prin analogie, o operaţie asupra unei resurse este idempotentă dacă a face o singură cerere este similar cu a face o serie de cereri identice. A doua cerere şi următoarele lasă resursa în aceeaşi stare pe care o avea după prima cerere.

Siguranţa şi idempotenţa oferă posibilitatea unui client să facă cereri HTTP sigure pe o reţea nesigură. Dacă facem o cerere GET şi nu primim nici un răspuns, putem să facem o nouă cerere fără nici o problemă. La fel se întâmplă si cu o cerere PUT. Metoda POST, în schimb, nu este nici sigură şi nici idempotentă. Dacă facem două cereri POST asupra unei resurse, cel mai probabil, vom obţine două resurse subordonate, conţinând aceeaşi informaţie.

Beneficiile utilizării arhtecturii REST în dezvoltarea serviciilor web sunt multiple. Datorită faptului că reprezentările pot fi cache-uite timpul de răspuns al serverului şi încărcarea acestuia sunt reduse. Scalabilitatea serverului este îmbunătăţită reducându-se

nevoia ca serverul să menţină anumite stări care ţin de o sesiune. Astfel servere diferite pot fi folosite pentru a trata diferite cereri dintr-o sesiune.

Codul de pe client este redus deoarece browserul poate fi folosit pentru a accesa orice resursă. De asemenea, un serviciu web RESTful depinde mai puţin de formate proprietare şi de frameworkuri de mesagerie deasupra HTTP.

4.4 UDDI(Universal Description, Discovery, and Integration)

Este un catalog distribuit, universal, al listei de servicii Web disponibile (inregistrate).UDDI este in fapt serviciu Web, invocabil prin SOAP inregistrarile sunt replicate actualmente, disponibil la nivel privat (enterprise).

UDDI furnizeaza un standard bazat pe gramatica XML, standard ce descrie serviciile generale si locatia serviciului Web.

UDDI este un serviciu Web si este in continua dezvoltare. Microsoft a inclus un serviciu de descoperire a serviciilor Web, mecanism numit disco, ce este bazat pe XML. Intentia mecanismului disco este de a descoperi serviciile Web gazduite de un anumit site Web.

UDDI-ul este cel mai puțin folosit dintre cele trei standarde. Majoritatea serviciilor existente în momentul de față nu folosesc UDDI pentru a permite descoperirea, ci folosesc alte metode de localizare, de exemplu liste de servicii publicate pe site-uri Web. În viitor însă acest lucru s-ar putea să se schimbe.

Proiectele Web service din Visual Studio .NET contin fisiere cu extensia .vsdisco, fisiere ce sunt numite dynamic discovery files, iar fisierele cu extensia .disco sunt numite static discovery files.

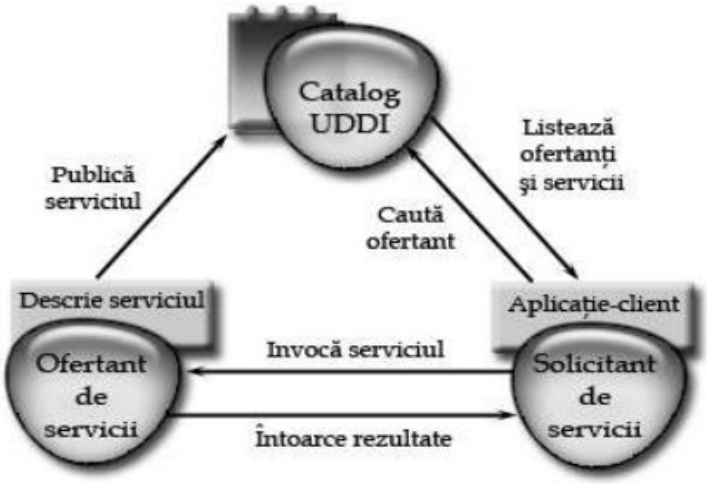

Fig5.

5. Platforme utilizate pentru web services

5.1 .NET Framework

Componente principale pentru .NET sunt urmatoarele:

CLR – Common Language Runtime;
CTS – Common Type System;
CLS – Common Language Specification;
BCL / FCL – Base Class Library / Framework Class Library.

Trasaturi ale frameworkului .NET:

- Interoperabilitate cu codul existent (COM poate interopera cu .NET si invers, apel functii din C/C++);
- Integrare completa si totala a limbajului (mostenire intre tipuri create in limbaje diferite, manipularea exceptiilor, depanare) ;
- Motor de runtime comun tuturor limbajelor (CLR) ;
- Biblioteca unica de clase (FCL/BCL) ;
- Constructia componentelor COM mult mai usoara (nu e nevoie de IClassFactory, IUnknown, IDispatch, cod IDL, etc.) ;
- Model de distribuire a aplicatiilor simplificat (nu mai e nevoie de inregistrare in registri, se permit multiple versiuni ale aceleasi biblioteci *.dll).

COM / DCOM, .NET remoting presupun existenta unui mediu omogen, adica informatiile sunt structurate si fiecare componenta stie sa lucreze cu acestea. Mai mult, .NET remoting presupune ca aplicatii .NET exista la ambele capete ale unui canal de comunicatie.

Spatiul de nume System.Web.Services - Clasa WebService - Clasa de baza pentru servicii Web XML. Furnizeaza acces la obiecte comune din ASP.NET, cum ar fi obiectul Application si Session.

Proprietati uzuale (MSDN):

Name	Description
Application	Gets the application object for the current HTTP request.
Context	Gets the ASP.NET HttpContext for the current request, which encapsulates all HTTP-specific context used by the HTTP server to process Web requests.
Events	Gets the list of event handlers that are attached to this component.
Server	Gets the HttpServerUtility for the current request.
Session	Gets the HttpSessionState instance for the current request.
SoapVersion	Gets the version of the SOAP protocol used to make the SOAP request to the XML Web service.
User	Gets the ASP.NET server User object. Can be used to authenticate whether a user is authorized to execute the request.

Serviciile Web ofera posibilitatea ca site-urile Web si aplicatiile sa interactioneze precum componentele.

Construirea si consumarea unui serviciu Web in .NET.Relatia dintre IIS, ASP.NET si serviciile Web

In momentul de fata, se foloseste ASP.NET pentru a construi servicii Web in .NET. Serviciile Web folosesc o extensie pentru ISAPI (aspnet_isapi.dll) si un proces separat ASP.NET (aspnet_wp.exe). IIS ruteaza cererile pentru fisiere cu anumite extensii (.aspx, .asmx, .asax, etc.) la aspnet_isapi.dll si apoi ajung la procesul ASP.NET (aspnet_wp.exe).

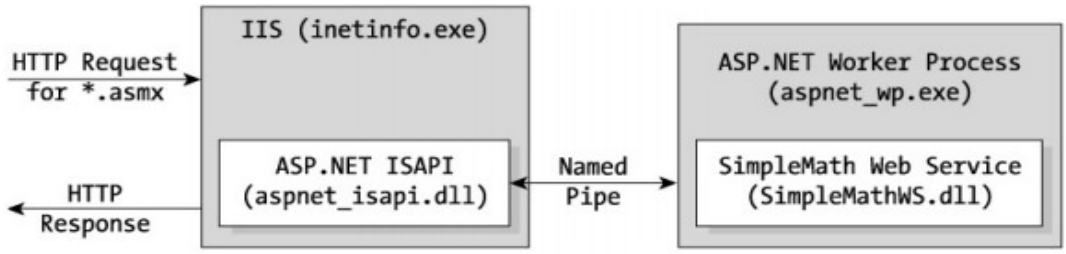

Fig6

XML si schema XML sunt folosite in aplicatiile bazate pe servere Web deoarece returneza un document HTML ce contine partea de prezentare cat si datele necesare, dar nu intr-o forma structurata, usor de parsat de catre un client. Partea dificila o constituie si faptul ca aceste pagini pot fi schimbate in timp si atunci este necesar sa se faca modificari si in aplicatia client ce foloseste aceasta pagina.

```
<TR name ='only row for data'>
  <TD WIDTH='40%' VALIGN='top'>
    <A href='/courses/CourseDetails.asp?LOC=details&ID=99115'>
    <P ALIGN='left'>Extreme .NET</p></a></TD>
  <TD align='Center' WIDTH='10%' VALIGN='TOP'>
    <Font class='whitefont'> __ </font> </TD>
  <TD align='Center' WIDTH='10%' VALIGN='TOP'>
    <Font class='whitefont'>
    <A href='https://www.intertech-inc.com/enroll.asp?CourseDateID=889'>
    <font class='Bold75'>2010-6-10</font></A>
    </font> </TD>
  <TD align='Center' WIDTH='10%' VALIGN='TOP'>
    <Font class='whitefont'> __ </font></TD>
  <TD align='Center' WIDTH='10%' VALIGN='TOP'>
    <Font class='whitefont'> __ </font></TD>
</TR>
```

Ceea ce este in bold si rosu constituie date de interes, restul reprezinta detalii de prezentare.

Serviciile Web folosesc XML pentru a transmite date si metadata.

Exemplu anterior poate deveni:

```
<courseSchedule>
  <course id="99115">
    <name>Extreme .NET</name>
    <date id="889">2010-6-10</date>
  </course>
</courseSchedule>
```

- .NET poate converti XML in data ce poate fi inteleasa de client.
- Serviciile Web nu au informatii despre CTS.

• .NET translateaza un tip din schema XML la un tip corespondent din CTS.

Daca serverul este construit folosind servicii Web sub .NET, atunci CLR converteste in mod automat tipurile din schema XML.

Descoperirea dinamica permite ASP.NET sa investigheze fiecare director virtual al unui site Web pentru a gasi si raporta serviciile Web disponibile. Din motive de securitate, versiunea release a .NET-ului dezactiveaza aceasta optiune.

Odata ce am gasit un serviciu Web, modul de folosire este descris de WSDL.

Un document WSDL contine metodele unei clase, parametrii metodei si valoarea returnata (acelasi lucru pe care-l face CLR cand foloseste o anumita clasa).

Exemplu. Consideram urmatorul serviciu Web:

```
public class ClassSchedule
{
  [WebMethod] // Metoda folosita in cadrul serviciului
  public DateTime GetNextClassDate(string className)
  { ... }
}
```

O parte a documentului furnizat de WSDL poate arata ca in exemplu urmator (este descrisa metoda GetNextClassDate):

```
<s:element name="GetNextClassDate">
  <s:complexType>
    <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="className" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetNextClassDateResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="GetNextClassDateResult"
              type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

WCF este platforma Microsoft pentru construirea aplicatiilor distribuite service-oriented. WCF inlocuieste tehnologii ca .NET Remoting, Enterprise Services si ASP.NET Web Services (ASMX).

WCF ofera un model pentru construirea aplicatiilor care suporta aceleasi scenarii in cadrul sistemelor distribuite. De exemplu, in cadrul programarii web, avem scenarii care implica WCF pentru JSON (Javascript Object Notation) sau REST (Representatioal State Transfer).

Windows Communication Foundation in .Net Framework apare in versiunea 3.0, odata cu lansarea Windows Vista (2007). Alaturi de WCF, sunt introduse si Windows Workflow Foundation (WF) si Windows Presentation Foundation (WPF).

WCF a fost proiectat pentru interoperabilitatea cu alte tehnologii. Un client WCF se poate conecta prin proceduri remote la COM (Component Object Model), la aplicatii distribuite construite pe .Net Remoting sau cu servicii ASP.NET , REST.

Pentru a putea folosi WCF, mai intai trebuie sa intelegem notiunile pe care acesta le foloseste.

ENDPOINT

Un endpoint reprezinta o destinatie pentru mesaj. Facand analogie, acesta este similar unui receptor in cazul unei comunicatii telefonice. Cand un client comunica cu un serviciu, ambii au un endpoint.

Pentru a crea un endpoint, avem nevoie de Adress, Binding, Contract.

Address este definit printr-un URI. Exista mai multe tipuri de URI(HTTP, HTTPS – suportate in Compact Framework, TCP, Named Pipes, MSMQ). Fiecare este specific protocolului de transport compatibil WCF. De la URI, clientul poate localiza serverul in retea si apoi serviciul din cadrul serverului.

BINDING

Binding va stabili modul in care clientul va comunica cu serverul. Binding va descrie tot ce va contine mesajul transmis intre client si server. Binding-ul contine mai multe elemente, unele pentru securiate, codarea mesajului, transport, tranzactii. Unele sunt optionale, insa cele pentru codarea mesajului si transport sunt obligatorii.

Desi exista 13 binding-uri predefinite in Compact Framework este suportat doar unul: *basicHttpBinding*. Acesta va permite servicii de Web si transport.

CONTRACT

Contractul va furniza detaliile modului in care clientul va interactiona cu serviciul specific. Prin contract intelegem apelurile metodelor disponibile (*service contracts*), layout-ul structurilor de date (*data contracts*) si modul in care exceptiile sunt tratate (*fault contracts*– nu sunt suportate de Compact Framework).

WCF este o buna alegerea in crearea de servicii care necesita o interoperabilitate mare. WCF ofera suport si pentru tehnologii mai vechi, de exemplu, Win32NamedPipes.

In .Net Compact Framework, se pot crea doar clientii WCF si nu servicii WCF. Serviciile WCF trebuie gazduite pe sistemul desktop sau pe un server si pot fi apelate din dispozitivul Windows Mobile.

De regula, pasii pentru construirea unui serviciu WCF incep de la proiectarea lui (definirea si implementare service contract), implemetare (endpoint, configurare binding, configurare comportament runtime), gazduire (initializare ServiceHost).

5.2 WSO2 WSF/PHP

WSO2 companie open-source middleware a lansat Web Services-cadru pentru PHP (WSF / PHP), este WSF / C şi PHP ca un caracter obligatoriu.WSO2, de asemenea, a lansat pentru C a WSO2 Web Services-cadru, care ofera un cadru de baza, in acest cadru de baza poate fi stabilita pe set de alte limbaje de scripting pentru a folosi aceste limbaje. In primul rand, PHP poate fi folosit pentru WSO2 Web Services-cadru, WSF a / C si legaturi PHP.

Pentru C, WSO2 Servicii Web-cadru este o bibliotecă complex, clasa enterprise open source care permite C de a stabili un robust, servicii sigure, de incredere web. Acesta sustine serviciile cele mai utilizate pe scara larga Web (WS *-) punerea in aplicare a caietul de sarcini, are o "wsclient" instrument de linie de comanda pentru a permite dezvoltatorilor sa utilizeze WSF / C pentru a folosi serviciile Web. WSO2 WSF / C 1.0 este lansat sub ApacheLicense 2.0, ce este o licenta bazată pe proiecte open source Apache, inclusiv Apache Axis2 / C, metereze Apache / C, Sandesha2 Apache / C, şi Savan Apache / C.

WSF / PHP 1.0 este popular limbaj de scripting PHP, extensia ce sustine un set complet de servicii Web ce ajuta dezvoltatorii PHP folosind enterprise-class SOAP bazate pe servicii Web.

Serviciile WSO2 web pentru PHP accepta de baza Web cadru de servicii pentru standardele, inclusiv SOAP 1.1, SOAP 1.2, 1.1 şi WSDL WSDL 2.0. Caracteristicile principale sunt:

-- Sprijină în totalitate WS *-Packet , inclusiv WS-Addresing, WS-Security, WS-PoliticalSecurity, WS-Reliable Messaging si SOAP Mecanism for Optimasing Messaing (MMD).

-- Securitatea serviilor este furnizata prin: codare si semnatura mesaj SOAP. Utilizatorii pot folosi, de asemenea, UsernameToken pentru a trimite un mesaj.

-- Compatibil cu PHP5, permite dezvoltatorilor WSF / PHP sa poata fi folosit cu codul actual, fara a trece prin nici o modificare.

6. Exemple de utilizare

Crearea şi consumarea de servicii Web pe platforma .NET

Exemplu: un serviciu care oferă lista furnizorilor produselor vândute de un magazin virtual

1. Crearea serviciului Web:

Fisierul serviciu Web ProviderInfo.asmx

```
<%@ WebService Language="VB" Class="ProviderInfo" %>

Imports System
Imports System.Web.Services
Imports System.Data
Imports System.Data.OleDb

Public Class ProviderInfo: Inherits WebService
   <WebMethod()> Public Function ShowSuppliers () As DataSet
          Dim DS As DataSet
          Dim MyConnection As Oledb.OleDbConnection
          Dim MyAdapter As Oledb.OleDbDataAdapter

                                    MyConnection    =    New    OleDb.OleDbConnection
("Provider=Microsoft.Jet.OLEDB.4.0;
                                                  Data Source=…\EComm.mdb")
          MyAdapter = New OleDbDataAdapter("SELECT denumire from Producatori;",
                                           myConnection)
          DS = new DataSet()
          MyAdapter.Fill(DS, "Producatori")
         return DS
         MyConnection.close()
   End Function
End Class
```

http://oraserv/ECommStorefront/ProviderInfo.asmx - serverul Web va furniza descrierea serviciului, numele serviciului Web, operaţiile suportate. Selectând o operaţie furnizată, se poate vedea cum poate fi apelată, peste ce protocoale.

http://oraserv/ECommStorefront/ProviderInfo.asmx?WSDL - descrierea formală a serviciului, în WSDL.

2. Consumare serviciului Web:

Crearea unei clase proxy ProviderInfo.vb folosind unealta wsdl furnizata de platforma .NET

```
wsdl.exe /l:VB /n:WService /out:bin/ProviderInfo.vb http://oraserv/ECommStrorefront/
ProviderInfo.asmx?WSDL
```

Compilarea fişierului sursă creat într-un DLL

```
vbc /t:library /out:bin\ProviderInfo.dll bin\ProviderInfo.vb
/
reference:System.dll,System.Data.dll,System.Web.dll,System.Web.Services.dll,System.XML.dl
l
/optimize
```

Crearea unei pagini aspx consumator pentru serviciu – regăseşte şi afişează lista furnizorilor http://oraserv/ECommStorefront/ProviderInfo.aspx

```
<%@ Page Language="VB" %>
<%@ Import Namespace="WService" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
 <html>
 <script runat="server">
  Sub SubmitBtn_Click(sender As Object, e As EventArgs)
       Dim dt As New DataTable

    'Instantiate DLL
    dim pi as new ProviderInfo

    'No parameter to pas to DLL function
    dim MyData as DataSet
    MyData = pi.ShowSuppliers()

    MyDataGrid.DataSource = MyData.Tables(0).DefaultView
    MyDataGrid.DataBind()
  end sub

  Sub Page_Load(sender As Object, e As EventArgs)
  End Sub
</script>

<body style="font: 10pt verdana">

<h4>Accessing Data with Web Services</h4>

<form runat="server">
<asp:button text="Submit" OnClick="SubmitBtn_Click" runat=server/>
<p>
<span id="Message" runat="server"/>
<p>

<ASP:DataGrid id="MyDataGrid" runat="server"
    AutoGenerateColumns="True"
    Width="100%"
    BackColor="White"
    Border="1"
    BorderWidth="1"
    CellPadding="1"
    CellSpacing="1"
    Font-Size="10pt"
    HeaderStyle-BackColor="White"
    HeaderStyle-ForeColor="Blue"
    AlternatingItemStyle-BackColor="White"
    AlternatingItemStyle-ForeColor="Black"
    ShowFooter="false"
/>
</form>

</body>
</html>
```

Consumarea serviciului de căutare Google

Crearea unei clase proxy

```
wsdl.exe /l:VB /n:WGoogleService /out:bin/Google.vb
http://api.google.com/GoogleSearch.wsdl
```

Compilarea fişierului sursă creat într-un DLL

```
vbc /t:library /out:bin\Google.dll bin\Google.vb /reference:System.dll,
System.Data.dll, System.Web.dll, System.Web.Services.dll, System.XML.dll /optimize
```

Crearea unei pagini aspx consumator pentru serviciul de cautare Google

```
<%@ Page Language="C#" Explicit="true" Strict="true" Buffer="true" Debug="true"%>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="WGoogleService" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Collections" %>
<%@ Import Namespace="System.ComponentModel" %>
<%@ Import Namespace="System.IO" %>

<html>
<script language="C#" runat="server">

void Page_Load(Object sender, EventArgs e) { }

void SubmitBtn_Click (object sender, System.EventArgs e) {
                // create Google Search object
                GoogleSearchService s = new GoogleSearchService();
                GoogleSearchResult r;

                // call search function
                r = s.doGoogleSearch(
                      "4B0KufpQFHJxhAxzua0tR11ElLNrHRJ6",
                      "cuvant",
                      0,
                      10,
                      false, "", false, "", "", "");

                // create HTML document to show result
                StreamWriter sw =  new StreamWriter ("D:\\...\\result.html");

                // Header inforamtion
                sw.WriteLine("<HTML><HEAD><style>      BODY { font-family : Verdana,
Geneva,
                Arial, Helvetica, sans-serif; font-size : 9pt; color : #000000;
            SCROLLBAR-
                FACE-COLOR: white; SCROLLBAR-HIGHLIGHT-COLOR: #003366; SCROLLBAR-
                SHADOW-COLOR: #003366; SCROLLBAR-3DLIGHT-COLOR: #f9f9f9; SCROLLBAR-
                ARROW-COLOR: #003366; SCROLLBAR-TRACK-COLOR: white; SCROLLBAR-
                DARKSHADOW-COLOR: #f9f9f9 }</style></HEAD><BODY>");

                // Category
                foreach(DirectoryCategory dc in r.directoryCategories)
                {
                      sw.Write("<b>Category</b> : ");
                      sw.WriteLine(dc.fullViewableName);
                      sw.WriteLine("<br><br><br>");
                }

                foreach(ResultElement re in r.resultElements)
                {
                      // Title
                      string strTitle="<a href=\""+re.URL+"\">" + re.title +
"</a><br>";
                      sw.WriteLine(strTitle);

                      // snippet
                      string strSnippet = re.snippet +"<br>";
                      sw.WriteLine(strSnippet);
```

```
                        // link and cache size
                        string strLink = "<a href=\"" + re.URL + "\">" + re.URL + "</a>
- "
                         + re.cachedSize + "<br><br>";
                        sw.WriteLine(strLink);

                        // 2 line
                        sw.WriteLine("<br><br>");
                    }

                    // file close
                    sw.Close();
    }
    </script>
    <body style="font: 10pt verdana">

    <h4>Accessing Google Search Web Service</h4>

    <form runat="server">
    <p>
    <asp:button text="Submit" OnClick="SubmitBtn_Click" runat=server/>
    </form>

    </body>
    </html>
```

http://oraserv/ECommStorefront/google.aspx
rezultatul in http://oraserv/ECommStorefront/result.html

Bibliografie:

1. Web Services Essentials, O'Reilly, Ethan Cerami, 2002
2. XML : http://www.w3schools.com/xml/default.asp
3. Google authorization key: http://www-128.ibm.com/developerworks/lotus/library/google-app/
4. http://www.codeproject.com/cs/webservices/mygoogle.asp
5. Ian Gorton. Essential Software Architecture. Editura Springer. 2006.
6. http://www.w3schools.com/soap/default.asp – tutorial SOAP
7. [http://www.codeproject.com/KB/XML/DotNetWebServiceConcepts.aspx – servicii web .NET
8. http://www.w3schools.com/webservices/ws_platform.asp

9.Figuri : http://www.w3.org/TR/ws-arch/#relwwwrest
http://www.w3.org/TR/wsdl
http://blogs.msdn.com/blogfiles/alikl/WindowsLiveWriter/ConnectionP_15058/image.png
fig1,fig2: http://upload.wikimedia.org/wikipedia/com/4/4a/Webservices
fig6:http://blogs.msdn.com/blogfilesWindowsLiveWriter/IdentityWhenASPPageCon.NETWebService_CB38/image.png