

Dialoguri despre SOAP

Accesul la servicii Web prin SOAP, un protocol de transfer de date XML

(o variantă a acestui articol a apărut pe situl NET Report și pe CD-ul revistei în feb. 2003)

– Lenuța Alboaie, Sabin Corneliu Buraga

În care vom vedea ce poate să însemne SOAP...

Sabin: Știu că în cele mai multe cazuri, creatorii de software cad cu greu de acord asupra unui lucru.

Lenuța: Da, ei sunt loiali tehnologiilor pe care le utilizează și nu agreează ideea de a oferi sprijin pentru alte tehnologii. În mare parte, dezvoltatorii aflați pe poziții opuse, stau pe cât posibil cât mai departe unul de tehnologia altuia.

Sabin: Să ne gândim, de exemplu, la programatorii proveniți din lumi diferite... Și nu pot să nu mă refer aici la... hm!... cele două comunități: Windows și... restul lumii. Unii discută de COM/DCOM/COM+, alții de RPC ori de CORBA sau de J2EE sau de aplicații distribuite folosind soluții particulare, mai mult ori mai puțin obscure...

Lenuța: Această polaritate face dificilă realizarea unei interoperabilități!

Sabin: Exact. Ar fi grozav să se găsească o tehnologie standard asupra căreia toți să fie de acord.

Lenuța: Și să asigure și independența?

Sabin: Desigur. Acesta ar putea fi un răspuns la dificultatea actuală.

Lenuța: Păi meta-lingajul XML (*Extensible Markup Language*) și HTTP (*HyperText Markup Language*) pot fi considerate două astfel de tehnologii. Lor li s-ar putea alătura **SOAP (Simple Object Acces Protocol)**...

Sabin: Iată chiar subiectul acestui articol!

Lenuța: ...definit să utilizeze XML și HTTP (nu mă întrerupe!). SOAP poate fi utilizat pentru a accesa servicii, obiecte și servere într-o manieră independentă de platformă.

Sabin: SOAP este un protocol?

Lenuța: Da, SOAP reprezintă un protocol care se comportă ca un „lipici” între componentele software eterogene, aflate pe mașini diferite.

Sabin: Un protocol de genul TCP sau, pentru a merge mai sus, HTTP?

Lenuța: Nu, pot afirma că SOAP este un protocol la nivelul aplicație, peste HTTP. În fapt, SOAP se bazează pe două din cărămizile Web-ului actual: HTTP și XML, care deja sunt standardizate și au specificații și implementări deschise.

Sabin: Înseamnă că dacă dezvoltatorii de soft...

Lenuța: Plus consorțiile și întreaga comunitate Web!

Sabin: (Hm... cine zicea de întreruperi?) ...au căzut de acord asupra HTTP-ului și XML-ului, atunci SOAP-ul poate fi oferta „extraordinară” în adoptarea unui mecanism de construire a unei tehnologii competitive într-un mod standard.

Lenuța: Da, fără slogani de genul „numai noi vă colorăm viața în roz!”...

Sabin: Sau în biți!?

Lenuța: Revenind, principalul scop al protocolului SOAP este tocmai *facilitarea interoperabilității*.

Sabin: Deci dacă voi folosi SOAP...

Lenuța: Cu aromă universală! ☺

Sabin: ...clientii scriși în Visual Basic de unul dintre studenții mei vor putea cu ușurință invoca serviciile CORBA rulând pe mașini Unix?

Lenuța: Desigur! Sau vei putea avea o altă situație, când diverși clienți Macintosh vor invoca obiecte Perl rulând în Linux.

Sabin: Spre bucuria mea...

Lenuța: Lista exemplelor ar putea continua, dar cred că te-am convins...

Sabin: La nivelul superficial al pielii, SOAP pare interesant! ☺

Lenuța: După cum ai putut vedea, conceptul de SOAP este simplu și nu este ceva cu totul nou.

Sabin: Da, amintindu-mi de *RPC (Remote Procedure Call)*...

Lenuța: Vezi articolul lui Sabin de acum câteva numere, când NET Report avea un alt gen de format și conținut...

Sabin: Fără reclame! ☺ SOAP codifică în mod simplu proceduri deja existente într-un standard din care toata lumea poate beneficia (și nu mă mai întrerupe!)...

Mai multe detalii!

Sabin: Bun, aș dori să aflu mai multe despre SOAP... Cititorii sunt deja nerăbdători, cred.

Lenuța: Păi, ca să dau un fel de definiție: *SOAP este un protocol simplu, utilizat pentru schimbul de informații într-un mediu distribuit, descentralizat*.

Sabin: Un mediu ca Web-ul...

Lenuța: Exact. Fiind vorba de un protocol, constă din reguli stricte de comunicare între părțile dialogului. Dacă până acum, de cele mai multe ori schimbul de mesaje era binar...

Sabin: Ca la TCP/IP...

Lenuța: sau era bazat pe mesaje text, având o structură dependentă de protocol...

Sabin: Ca la HTTP, de exemplu... Mai ții minte? Puteam trimite serverului un mesaj de genul GET `http://www.infoiasi.ro/~busaco/ HTTP/1.0...`

Lenuța: Iar mă întrerupi! Rezumând, schimbul de mesaje la SOAP constă din date marcate în XML.

Sabin: Extra! Voi putea scrie mult mai ușor un server sau un client care să proceseze, via *SAX (Simple API for XML)* sau *DOM (Document Object Model)*, mesaje XML.

Lenuța: Exact. În orice limbaj dorit...

Sabin: Perfect. Deci deja pot folosi C, C++, Perl, PHP, Java...

Lenuța: Și C#... dar să nu anticipez. Dar puteți să trageți cu ochiul la articolele lui Todi (Alexandru Pruteanu), de acum câteva numere în urmă, cât facem o pauză... ☺

Sabin: Lasă pauzele, chiar dacă sunt lungi și dese... Cum arată mesajele XML vehiculate?

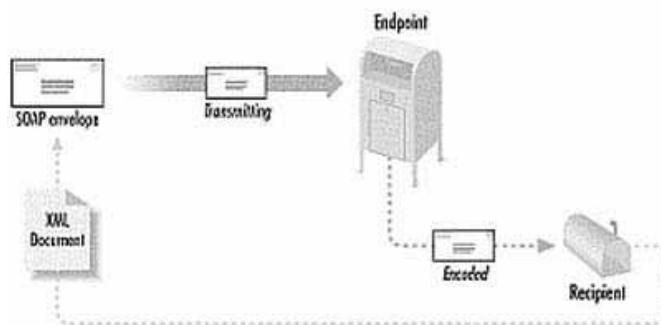
Lenuța: Există trei părți implicate: un plic (*SOAP envelope*), regulile de codificare asociate (*SOAP encoding rules*) și un mod de reprezentare a apelului unei metode la distanță (*SOAP RPC representation*).

Sabin: După cum îmi dau seama, putem gândi comunicarea între partenerii SOAP ca un schimb de scrisori?

Lenuța: Da, un mesaj SOAP poate fi privit ca o scrisoare (formată dintr-un plic cu un timbru și cu adresa scrisă pe el).

Sabin: Într-adevăr, această analogie ne ajută să vedem sensul acestui plic SOAP.

Lenuța: Exact, vezi și figura *Procesul de comunicare a mesajelor SOAP*.



Procesul de comunicare a mesajelor SOAP

Sabin: După câte înțeleg, documentul XML (reprezentând codificarea mesajelor dintre client și server) se ambalează „frumos” într-un plic, se transmite acest plic la punctul de destinație, acesta se uită care proces are nevoie de plicul în discuție, se decodifică informațiile și procesul SOAP destinat primește mesajul. Ca în realitate!

Lenuța: Da, mesajele sunt vehiculate exclusiv prin intermediul XML. De altfel, specificațiile SOAP introduc termenul de *XML Messaging*.

Sabin: Așadar, ideea de bază este că două aplicații, în mod independent de sistemul de operare, de limbajul de programare sau de alte detalii tehnice de implementare, pot partaja sau interchimba...

Lenuța: Deschis... este foarte important acest aspect!

Sabin: ...informații folosind nimic mai mult decât un mesaj simplu codificat într-un mod înțeles de ambele aplicații.

Lenuța: Da, iar acest mesaj este codificat în XML, deci va putea fi ușor procesat.

Sabin: Bine, dar totuși în Internet modul de reprezentare a datelor nu este unic. Să ne gândim doar la codificările seturilor de caractere sau la ordinea octeților folosită de procesoare...

Lenuța: Știm că vrei să mă încui cu observația aceasta. Dar soluții există. Înainte de asta, să-ți dau un exemplu concret. Presupunem că avem de realizat o aplicație, în care trebuie să codificăm numere de telefon.

Sabin: E o aplicație destinată SRI-ului?

Lenuța: De ce nu, dacă primim bani de la ei. Deci, eu – să zicem serverul – codific numărul de telefon astfel: `<numarTel>0211010101</numarTel>`.

Sabin: Bun... Și?

Lenuța: Tu – care ești clientul – vei schimba mesaje XML cu mine, marcând numărul de telefon de forma `<numarTel prefix="021" numar="1010101" />`.

Sabin: Da, înțeleg, chiar dacă XML este universal (deși nu e detergent), modalitatea de marcare a datelor este diferită. Care este codificarea corectă atunci?

Lenuța: Răspunsul dat de creatorii SOAP este acela că este corectă cea pe care aplicația o așteaptă.

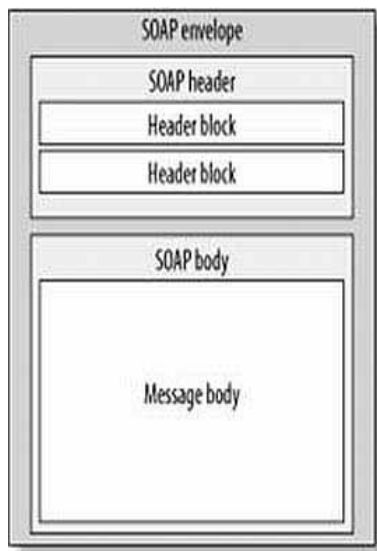
Sabin: Aha... Și cum rezolv totuși dilema aceasta?

Lenuța: Păi se vor defini niște convenții astfel încât programele noastre să poată (de)codifica informațiile primite, chiar dacă ele sunt codificate diferit în XML.

Sabin: Într-un fel similar cu *XDR (External Data Representation)* de la RPC...

Lenuța: Da. Totul e rezolvat de plicul SOAP.

Sabin: Adică mă trimiți la figura *Forma mesajelor SOAP...*



Forma mesajelor SOAP

Lenuța: După cum poți vedea, un mesaj SOAP constă într-un plic format dintr-un un antet (*header*) optional, plus un corp (*body*) obligatoriu.

Sabin: Întotdeauna oricine trebuie să aibă un corp... Iar unele au unul...☺

Lenuța: Hai să trecem la lucruri serioase. NET Report e revistă cu greutate, nu...

Sabin: Bine, dă-mi amănunte!

Lenuța: Antetul conține blocuri de informații relevante pentru modul cum va fi procesat mesajul (date despre expeditor, despre destinatar, date de autentificare și autorizare a transmisiei etc.).

Sabin: Ca la orice alt protocol... Iar corpul?

Lenuța: Corpul va conține mesajul care va fi efectiv trimis și procesat. Poți face analogia cu sistemul de poștă electronică folosit pe Internet, dar aici lucrurile capătă un aspect obiectual, abstract.

Sabin: Perfect, însă totuși numai teoria asta de până acum... nu se face primăvară!

Lenuța: Păi e normal, iarna nu-i ca... ☺

Sabin: Gata! Hai să revenim la aplicațiile noastre!

În care un prim exemplu poate clarifica lucrurile?

Lenuța: Să-ți expun o situație simplă, în care vor interveni o cerere și un răspuns SOAP. Am un client care dorește să salute un server.

Sabin: E politicos, plus probabil îl cunoaște pe server de mai demult...

Lenuța: Tot ce se poate. Dacă vedeai prima dată această problemă, te apucai și meșterei un client și un server TCP, în C sau în orice alt limbaj favorit...

Sabin: Da, și foloseam o convenție de transmitere a datelor. De exemplu, șirul de salut precedat de lungimea lui.

Lenuța: Perfect. Dar aici vom folosi chiar HTTP-ul drept protocol de transport, iar datele vehiculate vor fi scrise în XML.

Sabin: Bun, aș putea folosi RPC și de fapt aș face apel la o procedură (metodă) de salut care va fi executată de server. Ca să anticipez obiecția ta că aplicația mea ar fi prea limitată dacă aș scrie clienți și servere TCP, dependente de mașină. În cazul aceasta desigur aș fi adoptat Linux (mai general Unix)...

Lenuța: SOAP îmi poate soluționa problema, fără să-mi bat capul cu dependența de platformă. Îți voi spune mai încolo cum, până atunci te invit să vezi care este schimbul de mesaje dintre clientul și serverul SOAP.

Sabin: Ca să ilustrezi concepul de *XML Messaging*, nu?

Lenuța: Exact. De exemplu, cererea ar fi de forma:

```
POST /workme/Buna/buna.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: lungime
SOAPAction: "urn:Hello/sayHello"
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
    <soap:Body>
      <sayHello xmlns="urn:Hello">
        <name>Un șir</name>
```

```
</sayHello>
</soap:Body>
</soap:Envelope>
```

Sabin: Da, observ metoda POST și câmpurile de antet specifice HTTP, urmate de plicul SOAP. Corpul e clar: salutăm lumea din XML via SOAP...

Lenuța: Iar răspunsul ar putea fi următorul:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: lungime
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <sayHelloResponse xmlns="urn:Hello">
      <sayHelloResult>Un șir</sayHelloResult>
    </sayHelloResponse>
  </soap:Body>
</soap:Envelope>
```

Sabin: Se folosește în continuare HTTP, iar corpul plicului conține răspunsul.

Lenuța: Da. Desigur, în acest exemplu lungime și Un șir vor fi înlocuite cu valorile corespunzătoare situației reale.

În care mai multe amănunte ar putea fi de prisos?

Sabin: Îmi vine în minte tot modelul agenților de transport a poștei electronice și te întreb: schimbul acesta de mesaje SOAP nu s-ar putea realiza și prin intermediul unor intermediari? Cum de exemplu se întâmplă și pe Web, folosind servere *proxy*...

Lenuța: Sigur că da. Un intermediar SOAP poate adăuga diferite funcționalități tranzacției de date dintre expeditor și destinatar.

Sabin: Deci vom avea în fapt o *cale de mesaje SOAP* între punctele finale ale conversației.

Lenuța: Exact, te-ai prins. Intermediarul de-a lungul acestei căi de mesaje se numește și *actor*.

Sabin: Bun, înseamnă că ar trebui furnizate anumite „indicații regizorale” destinate fiecărui astfel de actor (intermediar) ca să-și facă treaba cum trebuie.

Lenuța: Da, SOAP pune la dispoziție un mecanism, numit și *țintire (targeting)*, de specificare a părților din antetul unui mesaj destinate exclusiv unui actor. Ținta unui actor va fi specificată chiar printr-un atribut purtând numele actor.

Sabin: Și ce va conține acest atribut?

Lenuța: Un URI unde poate fi găsit intermediarul.

Sabin: O altă întrebare, pe care sigur și-a pus-o și cititorul nostru...

Lenuța: Dacă nu a adormit între timp sau a schimbat pe alt canal... pardon, articol.

Sabin: (Iar mă întrerupi!) SOAP modifică într-un anume fel semantica HTTP?

Lenuța: Nu, în nici un caz. SOAP doar se integrează foarte bine în HTTP, utilizând filosofia protocolului de transport. Astfel, întreg schimbul de mesaje dintre clientul și serverul SOAP va fi realizat via HTTP, nu prin intermediul unui protocol propriu.

Sabin: Da, aceasta este o idee strălucită. Pot folosi un simplu navigator pentru a vehicula mesaje cu serverul dorit, fără a trebui să apelez la cine știe ce client sofisticat. Care să mă mai și coste!...

Lenuța: La fel, orice server Web poate ușor oferi suport pentru SOAP!

Sabin: Perfect, de aceea observ că SOAP merge mână în mână cu serviciile Web...

Lenuța: Să nu anticipăm totuși.

Sabin: Bun, ziceai însă că SOAP este înrudit și cu mecanismul RPC. De ce?

Lenuța: Hai să presupunem că nu vrei doar să saluți servere (fie ele cunoscute sau nu ☺), ci dorești mai mult...

Sabin: Uneori aș dori, însă parcă tu ziceai că NET Report-ul este o revistă de altă factură, ha, ha.

Lenuța: Ha!... Mă refeream la interesul pur științific de a executa ceva la distanță, pe serverul SOAP. Adică în mesajul XML am posibilitatea să încapsulez nu atât doar date, ci și numele unei proceduri care să fie recunoscute de server, să fie apelată și apoi prin intermediul unui alt mesaj SOAP clientul să primească rezultatul, codificat tot în XML.

Sabin: Perfect! Atunci mie, ca și client, nu-mi rămâne decât să definesc în XML tipurile și lista argumentelor de intrare și tipul rezultatului așteptat, să pun totul într-un plic SOAP și să-l trimit serverului spre procesare.

Lenuța: Exact, de cele mai multe ori schimbul de mesaje SOAP este o pereche formată din cerere (*request*) – clientul trimite numele și argumentele unei metode pentru a fi apelată de pe server – și răspuns (*response*) – serverul expediază valoarea răspunsului înapoi la client. Vezi și figura *Cererea și răspunsul SOAP*.



Cererea și răspunsul SOAP

Sabin: În cel mai tipic mod RPC...

Lenuța: Dacă tot îți place această abordare, ai putea să dai și un exemplu, ce zici?

Sabin: Bine. Cum tot e moda acum să se vorbească de afaceri pe Web, hai să ne închipuim că ne putem conecta la serverul Web al companiei *Tiffany's* pentru a vedea care este prețul de pornire al licitației rămășițelor plăcintei care l-a bucurat mult pe domnul Bill Gates, acum câtva timp. ☺

Lenuța: He, he!

Sabin: Adică pur și simplu, aplicația mea SOAP va apela metoda `getPrice` pusă la dispoziție de dezvoltatorii de la *Tiffany's*, pentru a afla răspunsul dorit.

Lenuța: Eu aș fi numit metoda `getPrize`... ☺

Sabin: Plicul SOAP în acest exemplu ar putea conține metoda și argumentele sale:

```

<s:Envelope xmlns:s="http://www.w3.org/2001/06/soap-envelope">
  <s:Header>
    <m:transaction xmlns:m="soap-transaction" s:mustUnderstand="true">
      <transactionID>1234</transactionID>
    </m:transaction>
  </s:Header>
  <s:Body>
    <t:getPrice xmlns:t="urn:Tiffany.com:PriceService">
      <symbol xsi:type="xsd:string">Gates' Creamy Pie</symbol>
    </t:getPrice>
  </s:Body>
</s:Envelope>
  
```

Lenuța: Trebuie să mai precizez că `s` din exemplu este spațiul de nume SOAP, `m` – un spațiu de nume desemnând o tranzacție de date SOAP, iar `t` este spațiul de nume care referă tipurile de documente XML specificate de *Tiffany's*.

Sabin: Exact... Iar dacă toate aceste lucruri au loc exclusiv în lumea Web-ului, atunci...

Lenuța: Dăm peste *serviciile Web*!

În care se va scrie un serviciu Web folosind .NET

Sabin: După cum ne putem da deja seama, un serviciu Web reprezintă o extindere a funcționalităților oferite de un sit Web prin punerea la dispoziție la același port (de obicei cel standard 80, specificat de HTTP) a unei liste de metode (servicii) implementate pe acea mașină.

Lenuța: E la fel ca la RPC? Și acolo aveam un dispecer care asculta un anumit port și în funcție de ce primea de la un client trimitea mesajul procedurii în cauză spre a fi executată, iar răspunsul pe urmă era expediat clientului RPC.

Sabin: Da, conceptul este similar, dar totul acum se rezolvă pe Web, grație HTTP, XML și... SOAP.

Lenuța: Și de unde știu ce servicii Web îmi oferă un anumit sit? De exemplu, Google?

Sabin: Păi există un registru universal, ce-i drept nu prea evoluat încă, la care pot întreba despre un anumit serviciu dorit. Un fel de *Paginile Aurii*.

Lenuța: Da, îmi amintesc că am citit despre *UDDI (Universal Description, Discovery, and Integration)*.

Sabin: În plus, s-a găsit o modalitate flexibilă de descriere a sintaxei și semanticii unui serviciu Web prin...

Lenuța: Știu! Prin *WSDL (Web Service Description Language)*, care este o specificație bazată tot pe XML.

Sabin: Desigur. Folosind WSDL, se pot vedea ce mesaje (tip, structură, parametri) pot fi trimise unui serviciu Web.

Lenuța: Perfect, iar aceste mesaje vor fi transportate de la clientul nostru care dorește să apele un serviciu Web până la situl care oferă acel serviciu prin SOAP.

Sabin: Exact... Ce-ar fi deja să te apuci de implementat?

Lenuța: Înainte de asta, cred că ar fi bine să-ți spun că există mai multe instrumente de lucru dedicate SOAP-ului. Cele mai populare sunt *Apache SOAP* pentru Java, *SOAP:Lite* pentru Perl și *Microsoft .NET* pentru limbajele suportate de platforma .NET. Vezi și <http://xml.apache.org/soap/>, <http://search.cpan.org/search?module=SOAP> și, respectiv, <http://msdn.microsoft.com>.

Sabin: Aha, deci nu e un apanaj exclusiv Microsoft...

Lenuța: Nici pe departe. Poți găsi și alte instrumente la adresele <http://www.soaplite.com> și <http://www.soapware.org>. De aici se pot prelua și testa instrumente SOAP pentru cele mai populare medii și limbaje de programare: Java, C#, C/C++, Perl, PHP, Python etc. De asemenea, se pot vizita și adresele <http://www.codeproject.com/soap/>, <http://www.soapuser.com/client3.html> sau <http://www.alphaworks.ibm.com/tech/soap4j>.

Sabin: Presupun că indiferent care instrument e folosit, utilizarea serviciilor web SOAP este aceeași.

Lenuța: Desigur, în esență. Detaliile însă...

Sabin: Bun, să lăsăm asta în grija cititorilor. Trebuie și ei să muncească la ceva... ☺ Ce propui în continuare?

Lenuța: Păi, voi prezenta cum poate fi conceput un serviciu Web folosind drept instrument de lucru Microsoft .NET. Nu din masochism cum ar putea crede unii, ci chiar din dorința de a arăta că întreg procesul este foarte comod și poate fi realizat relativ simplu.

Sabin: Am auzit că o parte dificilă este chiar cea a instalării mediului .NET.

Lenuța: Trebuie să-i avizez pe programatori că serviciile Web bazate pe .NET funcționează în strânsă legătură cu mediul oferit de serverul IIS. Așadar, înainte de instalarea .NET-ului, trebuie să aveți instalat *Microsoft Internet Information Server (IIS)*, iar acest serviciu să fie și în starea de rulare (*running*).

Sabin: Vezi *Services* din meniul *Administrative Tools* din Windows XP, de exemplu...

Lenuța: Recomand ca navigator Internet Explorer 5 sau o versiune superioară.

Sabin: Bine, după ce lucrurile s-au instalat și ești gata de muncă...

Lenuța: Programator plin de voieșie și elan informaticesc...

Sabin: ...îți propun să scrii un serviciu Web .NET foarte simplu și anume unul care să returneze salutari. Deci, dacă eu mă voi recomanda lui „Sabin”, el să-mi returneze mesajul „Bună Sabin”. Desigur, pot spune „ziua” și va returna „Buna ziua”...

Lenuța: Perfect. Primul pas în programarea serviciului este să creăm un fișier .asmx (putem folosi orice editor, dar cred că *Visual .NET* oferă unul foarte comod). Vom numi acest fișier salutari.asmx în care voi scrie:

```
<%@ WebService Language="C#" Class="Hello" %>
using System.Web.Services;
[WebService(Namespace="urn:Hello")]
public class Hello
{
    [WebMethod] public string sayHello(string nume)
    {
        return "Buna " + nume;
    }
}
```

Sabin: Remarc faptul că serviciul Web este scris în C# și se numește Hello.

Lenuța: Da, un cod similar se poate scrie și pentru Java. În acest exemplu, se definește o singură metodă apelabilă la distanță – sayHello.

Sabin: Ce este cu prima linie? Pare o directivă ASP...

Lenuța: Linia <%@ WebService Language="C#" Class="Hello" %> este cea care indică mediului .NET faptul că se exportă un serviciu Web, scris în C# și implementat în clasa Hello. Linia using realizează includerea modulelor necesare (în cazul nostru a claselor legate de serviciile Web). Linia [WebService(Namespace="urn:Hello")] este opțională, dar ne permite să declarăm diferite atribute serviciului Web pe care-l dezvoltăm. În acest caz am setat un spațiu de nume explicit, altfel .NET ar fi ales unul implicit. Alte atribute care ar mai putea fi atașate sunt numele și descrierea acestui serviciu. Linia [WebMethod] setează un atribut pentru metodele din clasa Hello prin care acestea vor face parte din serviciul Web. Ca și pentru atributul de mai sus, în această linie putem defini anumite proprietăți legate de operațiile pe care le realizează serviciul Web.

Sabin: Bun, ce fac după ce am scris un astfel de fișier?

Lenuța: Voi salva fișierul în rădăcina sitului Web deservit de serverul IIS. Atenție însă la URL-ul fișierului! De exemplu, dacă serverul IIS este instalat la c:\inetpub (aici se instalează implicit), atunci rădăcina sitului va fi c:\inetpub\wwwroot.

Sabin: Deci, acum fișierul salutari.asmx va fi localizat aici și URL-ul lui va fi http://localhost/salutari.asmx, desigur localhost fiind domeniul mașinii pe care rulează IIS.

Lenuța: Exact, pentru a folosi serviciul Web scris de noi...

Sabin: De tine. Să nu mai zică lumea că fetele nu pot fi programatoare... ☺

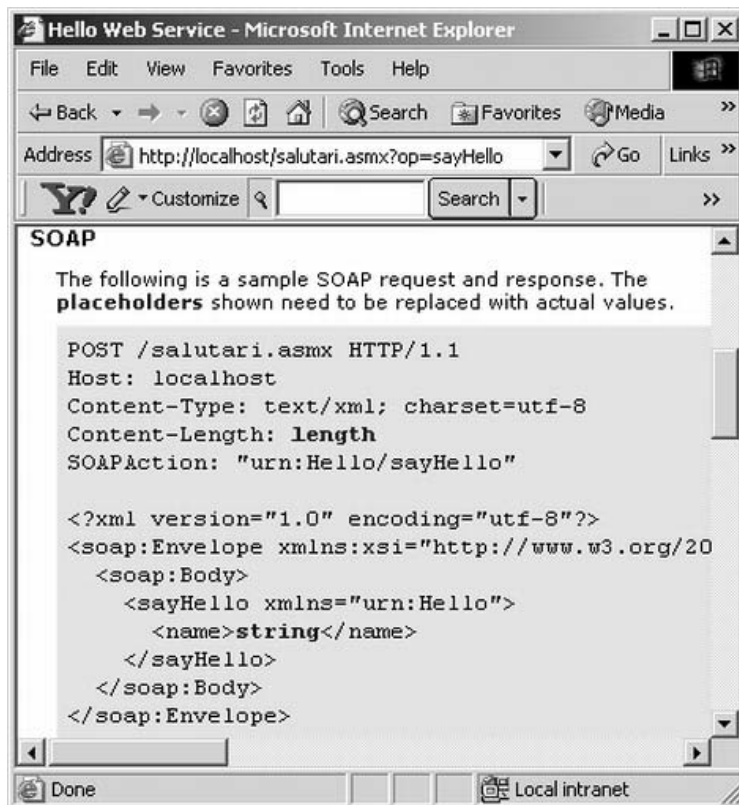
Lenuța: ...deschidem navigatorul și introducem http://localhost/salutari.asmx. Ar trebui să apară ceva asemănător cu imaginea din figura *Apelarea serviciului Web*:



Apelarea serviciului Web

Sabin: Perfect. Iată că a mers într-adevăr!

Lenuța: Dând un clic pe legătura `sayHello` vei putea vedea o descriere detaliată despre cum are loc invocarea operației implementate. Vezi și captura-ecran *Descrierea serviciului Web*.

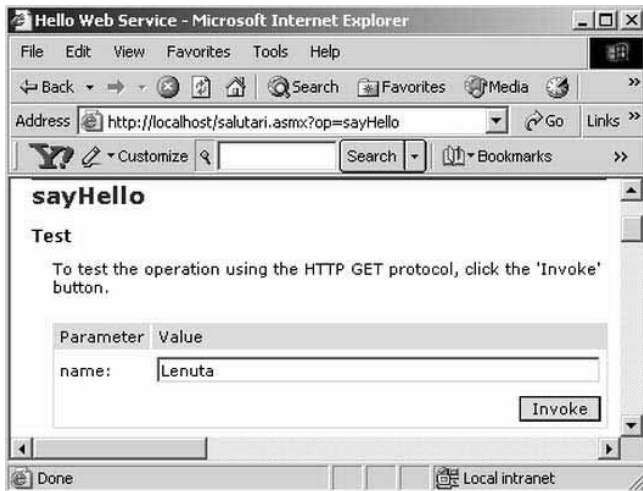


Descrierea serviciului Web

Sabin: Observ că răspunsul este oferit în formatul SOAP.

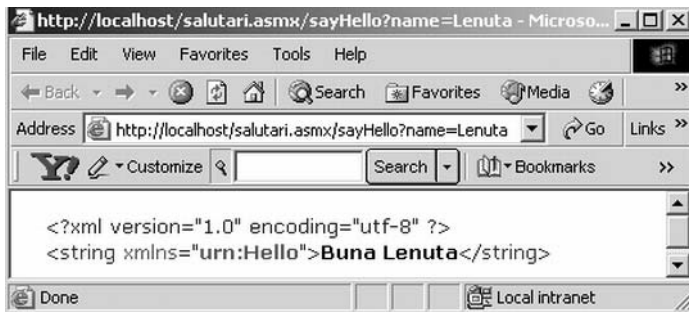
Lenuța: Sigur că da, nativ se folosește SOAP. Despre asta vom discuta puțin mai încolo însă... Ai putea acum să testezi funcționalitatea „minunatului” serviciu Web implementat.

Sabin: În regulă. Voi introduce un nume și voi apăsa butonul *Invoke...* Poți să urmărești aceasta în figura *Invocarea metodei serviciului Web*.



Invocarea metodei serviciului Web

Lenuța: Desigur, dacă serviciul funcționează perfect, vei primi răspunsul „Bună Lenuța” sub formă XML, ca în captura *Returnarea rezultatului*.



Returnarea rezultatului

Sabin: Am văzut cum funcționează. Totuși mă întreb care sunt modalitățile de apelare a unui serviciu Web. Pot folosi doar metoda GET sau și metoda POST, așa cum puteam face în cazul scripturilor CGI?

Lenuța: Înainte de a-ți da răspunsul, trebuie să precizez că partea *.NET runtime* la prima invocare a serviciului va compila codul acestuia.

Sabin: O idee asemănătoare la servlet-uri...

Lenuța: Apoi *.NET runtime* va determina ce fel de cerere a fost făcută. Și sunt câteva posibilități:

- o cerere care se interesează despre ce funcționalități oferă serviciul Web;
- o cerere care se interesează în mod explicit de o metodă furnizată de acest serviciu Web;
- sau o cerere care invocă o operație furnizată de serviciul Web.

Sabin: Totuși care e metoda de invocare?

Lenuța: Ce nerăbdător ești! Mediul *.NET* oferă trei maniere de invocare: primele două sunt cele clasice, via metodele GET și POST puse la dispoziție de HTTP, iar ultima folosește SOAP.

Sabin: Interesant, atunci pot scrie chiar un client (diferit de navigator) care să invoce o metodă a unui serviciu Web via SOAP, așa cum am anticipat din scenariul cu prețul unei licitații...

Lenuța: Exact. În plus, trebuie să-ți spun că *.NET* este printre puținele platforme permițând ca serviciile Web să fie invocate prin utilizarea unor protocoale multiple.

Sabin: Dacă tot am amintit de scrierea unui client care să apeleze serviciul Web conceput de tine, ce zici să te apuci efectiv de treabă?

Lenuța: ...Hm! După pauză... Cât sunt reclame, am putea aviza autorul că mai multe detalii despre SOAP și conceperea serviciilor Web vor fi disponibile în cartea noastră dedicată acestui subiect... Fiți pe fază! ☺

În care se prezintă un client SOAP pentru invocarea serviciului Web

Sabin: Am revenit!

Lenuța: Bine... Din perspectiva *.NET*, un serviciu Web constă din trei elemente. Primul dintre ele este receptorul care primește mesajul (cererea), al doilea este un *proxy* care preia mesajul și îl transmite într-o acțiune care trebuie îndeplinită (situație asemănătoare cu apelul unui metode a unui obiect din Java). Ultimul element o aplicație care implementează acea acțiune.

Sabin: Va trebui să invoci serviciul Web Hello prin intermediul SOAP. Trebuie așadar scris și *proxy*-ul?

Lenuța: Da, îl voi concepe în C#. Urmărește *listing*-ul `SalutariClient.cs`.

```

using System;
using System.Diagnostics;
using System.Xml.Serialization;
using System.Web.Services;
using System.Web.Services.Protocols;
namespace salut
{
    [System.Web.Services.WebServiceBindingAttribute(
        Name="HelloSoap",
        Namespace="urn:Hello")]
    public class ClientSalut: System.Web.Services.Protocols.SoapHttpClientProtocol
    {
        public ClientSalut()
        {
            this.Url="http://localhost/salutari.asmx";
        }

        [System.Web.Services.Protocols.SoapDocumentMethodAttribute(
            "urn:Hello/sayHello",
            RequestNamespace="urn:Hello",
            ResponseNamespace="urn:Hello",
            Use=System.Web.Services.Description.SoapBindingUse.Literal,
            ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
        public string sayHello(string name)
        {
            object[] results= this.Invoke("sayHello",new object[] {name});
            return ((string)(results [0]));
        }
    }
}

```

Listing-ul SalutariClient.cs

Sabin: Cititorul atent al seriei de articole despre .NET, serie semnată în principal de colegul nostru Alexandru Pruteanu, ar putea să-și dea seama că aici este vorba de un *assembly*...

Lenuța: Chiar așa. Linia [System.Web.Services.WebServiceBindingAttribute...] este cea care indică la compilare că respectivul *assembly* va fi folosit pentru a invoca un serviciu Web. Când acest *assembly* este compilat, platforma NET generează structura astfel încât să poată funcționa cererile SOAP. Linia ClientSalut: System.Web.Services.Protocols.SoapHttpClientProtocol specifică protocolul care se dorește a fi utilizat (în acest caz, SOAP peste HTTP).

Sabin: Observ că în interiorul constructorului clasei ClientSalut se setează URL-ul la care se găsește disponibil serviciul Web.

Lenuța: Da, aici este codat chiar în cadrul programului, dar cititorul poate încerca și alte modalități de a-l furniza.

Sabin: Ce s-a mai scris în interiorul clasei?

Lenuța: S-a mai creat un *proxy* pentru operația sayHello. S-au specificat diferite atribute ale serviciilor Web invocate. De asemenea, se apelează metoda Invoke pusă la dispoziție de mediu și se așteaptă întoarcerea rezultatului.

Sabin: După conceperea acestui *proxy*, mai urmează să scriem clientul propriu-zis...

Lenuța: Adică să-l scriu eu... Voi folosi în continuare limbajul C# pentru această implementare.

Sabin: Deci trebuie să consult *listing*-ul FormaBuna.cs...

```

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace salut
{
    public class FormBuna : System.Windows.Forms.Form
    {
        // cod generat automat de mediul Visual .NET
        // (omitem aceasta parte)
        private void bInvoke_Click(object sender, System.EventArgs e)

```

```

    {
        string str;
        str=textBoxNume.Text;
        ClientSalut cs = new ClientSalut();
        MessageBox.Show(this,cs.sayHello(str));
    }
}

```

Listing-ul FormaBuna.cs

Lenuța: Am folosit facilitățile oferite de *Visual .NET* și *Windows.Forms* pentru a genera interfața cu utilizatorul.

Sabin: Am remarcat că în client trebuie doar să creăm o instanță a *proxy*-ului creat mai sus și să apelăm apoi metoda pe care o implementează.

Lenuța: Exact, *proxy*-ul este cel care trimite cererea la server și întoarce rezultatul furnizat, utilizând schimburi de mesaje SOAP.

Sabin: Am compilat programul și am introdus un nume.

Lenuța: Vei obține, într-o boxă de dialog modală, răspunsul furnizat de serviciul Web. Vezi captura-ecran din figura *Invocarea serviciului Web via un client SOAP*.



Invocarea serviciului Web via un client SOAP

Sabin: Când am apăsat butonul *Invoke* s-a produs un eveniment tratat într-o metoda aflată pe partea de client. În cadrul corpului acestei metode se apelează *sayHello* din clasa implementată de *proxy*...

Lenuța: ...Iar aceasta invocă via SOAP respectiva operație de pe server oferind serviciul Web.

Sabin: Se întoarce apoi (aici în manieră sincronă) rezultatul întors de metoda implementată de serviciul Web scris.

În care concluziile înseamnă un fel de sfârșit?

Lenuța: Ce a mai rămas de spus?

Sabin: Păi pot concluziona cu următoarele: SOAP oferă o manieră independentă de platformă pentru vehicularea datelor marcate în XML, printr-o manieră orientată-mesaj, peste protocolul HTTP. SOAP reprezintă deci un *summum* între HTTP, XML și RPC, putând fi privit în acest sens ca o abordare la nivel înalt a paradigmei RPC, mai general decât *RMI (Remote Method Invocation)* din Java.

Lenuța: Aș adăuga suportul foarte bun pentru SOAP și dezvoltarea de servicii Web oferit de mediul .NET, cu observația că există de asemenea și alternative *open-source* sau *freeware* pentru conceperea de aplicații orientate spre SOAP.

Sabin: În plus, nu doar protocolul SOAP poate fi considerat soluția unică pentru interschimbul de mesaje între aplicații Web, o altă soluție fiind, de exemplu, XML-RPC. Pe situl Consorțiul Web există multe informații referitoare la protocolele de comunicație bazate pe XML (vezi și <http://www.w3.org/TR/>).

Lenuța: Aș mai menționa drept surse de informare <http://soap.weblogs.com/> și <http://www.w3.org/2002/ws/>.

Sabin: Desigur, lucrurile nu se opresc aici. Deja avem posibilitatea de a invoca din cadrul aplicațiilor noastre, via SOAP, diverse servicii Web puse la dispoziție de unele organizații, ca exemple notabile putând fi date Google ori Amazon. Fiind încă o tehnologie aflată la începuturi, serviciile Web în conjuncție cu protocolele de comunicație bazate pe XML trebuie să găsească o cale de dezvoltare și de maturizare. Cert este că oricare dintre noi ar trebui să țină seamă de oportunitățile și provocările oferite de acestea.

Lenuța Alboaiă urmează studiile aprofundate în domeniul procesării distribuite la Facultatea de Informatică a Universității de Informatică, având ca principale teme de interes dezvoltarea de servicii Web comunicând via SOAP, prin prisma unor limbaje și medii de programare eterogene, putând fi contactată la [adria@infoiasi.ro](mailto:infoiasi.ro).

Sabin-Corneliu Buraga este doctorand în Computer Science la Facultatea de Informatică, Universitatea „A.I. Cuza” din Iași și poate fi contactat la busaco@infoiasi.ro.

Glosar de termeni

Assembly

Formula atomică prin care o aplicație .NET este reprezentată în momentul rulării (*runtime*) se numește *assembly*. Acest *assembly* poate fi comparabil cu un fișier `.class` din Java.

COM (Component Object Model)

Tehnologie Microsoft, independentă de platformă, reprezentând un sistem orientat-obiect utilizat pentru crearea de componente software binare care pot interacționa. Obiectele COM pot fi create în diferite limbaje de programare.

CORBA (Common Object Request Broker Architecture)

Cadru de dezvoltare a aplicațiilor distribuite în medii eterogene. CORBA se bazează pe *OMA (Object Management Architecture)*, model propus de *OMG (Object Management Group)*, pentru folosirea într-un mediu distribuit a programării orientate pe obiecte, ceea ce implică o dezvoltare rapidă a aplicațiilor, reutilizarea și protecția eficientă a datelor.

DCOM (Distributed Component Object Model)

Extindere, propusă de Microsoft, a modelului COM la medii distribuite, eterogene, disponibile pe calculatoare diferite.

DOM (Document Object Model)

Modalitate, independentă de limbaj și de platformă, de accesare a elementelor și atributelor dintr-un document adnotat în XML sau HTML. DOM oferă un set standard de obiecte și de interfețe predefinite pentru prelucrarea documentelor XML/HTML.

HTTP (HyperText Transfer Protocol)

Protocol la nivelul aplicație, bazat pe TCP/IP, utilizat pentru transferul informațiilor hipertext.

.NET

Reprezintă un set de tehnologii software Microsoft pentru interschimb de informații între oameni, sisteme, aplicații și alte dispozitive. Oferă un nivel ridicat de integrare prin intermediul serviciilor Web bazate pe XML.

Platformă

Inițial, termen referitor la o anumită configurație hardware specifică (de exemplu, tip de procesor, arhitectură de rețea). Ulterior s-a extins pentru a desemna o combinație specifică de hardware și software (tip sau versiune de sistem de operare – Linux, MacOS, Windows, OS/2 – și/sau compilator ori limbaj de programare – GNU C/C++, Visual C, Perl, Java etc.).

Protocol

Mulțime de reguli pentru transmiterea datelor în cadrul rețelelor de calculatoare. Protocoalele se structurează pe niveluri, aplicațiile folosindu-se de protocoale de pe nivelul superior. În Internet, se utilizează suita de protocoale *TCP/IP (Transmission Control Protocol/Internet Protocol)*.

SAX (Simple API for XML)

Interfață de programare permițând dezvoltătorilor de aplicații să analizeze documentele XML. Spre deosebire de *DOM*, SAX nu necesită încărcarea întregului fișier în memorie. Analiza XML este condusă de evenimente, în manieră incrementală.

Serviciu Web

Conceptul de serviciu Web reprezintă următorul pas în evoluția spațiului WWW și permite ca elemente programabile să fie plasate pe siturile Web. Alți programatori pot accesa funcționalitățile acestor programe, în manieră distribuită și directă, pentru orice număr potențial de sisteme independente, prin folosirea unor standarde precum XML și HTTP. Vezi *SOAP* și *WSDL*.

SOAP (Simple Object Access Protocol)

Protocol simplu utilizat pentru schimbul de informații într-un mediu distribuit, descentralizat. SOAP furnizează o cale de comunicare între aplicații prin intermediul mesajelor XML și permite schimbul de tipuri de informații structurate pe Web (poate fi privit ca o standardizare RPC pentru Web, folosind protocolul HTTP și meta-limbajul XML).

Spațiu de nume

Mecanism permițând dezvoltătorilor XML să califice în mod unic numele de elemente și relațiile dintre ele, evitându-se conflictele de elemente definite în vocabulare diferite. Un spațiu de nume identifică un vocabular XML prin intermediul unui identificator uniform de resursă – *URI (Uniform Resource Identifier)*.

UDDI (Universal Description, Discovery, and Integration)

Un set de protocoale și un catalog public pentru înregistrarea și regăsirea în timp real a serviciilor Web.

WSDL (Web Services Description Language)

Standard pentru descrierea în format XML a serviciilor Web. Servește drept convenție universală a descrierii funcționalității (semanticii) unui serviciu Web și instruește potențialii clienți care pot interacționa cu acesta.

XML (Extensible Markup Language)

Standard și meta-limbaj de marcare în continuă dezvoltare în cadrul Consorțiului Web, oferind un set mai restrâns, dar mai facil, de reguli de adnotare decât SGML din care provine. XML este compus în fapt dintr-o familie de limbaje de bază (*i.e.* XSL, XPath, XBase etc.), având sute de limbaje derivate (precum MathML, RDF, SMIL ori SOAP).