SOAP in de praktijk

Het ontwikkelen van webdiensten met SOAP

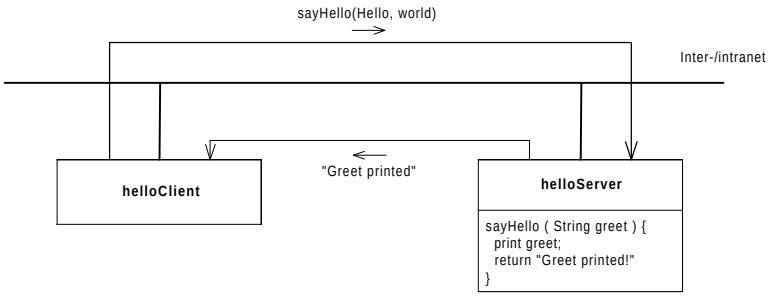
Danny Greefhorst en Matthijs Maat

Bij het ontwikkelen van applicaties die diensten aanbieden voor en via het internet speelt de nieuwe SOAP-standaard (Simple Object Access Protocol) een belangrijke rol. Bestaande technologieën voor het ontwikkelen van gedistribueerde applicaties, zoals bijvoorbeeld DCOM en CORBA, maakten het al eerder mogelijk via het net aangeboden diensten aan te roepen. Waarom dan nog SOAP? Wat biedt SOAP dat CORBA en DCOM niet hebben?

In dit artikel zullen de auteurs deze vragen beantwoorden door in te gaan op wat SOAP precies is en te laten zien hoe ontwikkelaars met SOAP webdiensten kunnen ontwikkelen. Voorbeelden van applicaties ontwikkeld met een aantal beschikbare SOAP-implementaties illustreren het gebruik van SOAP voor het ontwikkelen van webdiensten.

Webdiensten

Na component-gebaseerde ontwikkeling, distributie van softwarecomponenten, e-business en application service providers lijkt de gehele IT-wereld zich nu te richten op *webdiensten*. Webdiensten zijn op het Internet te huur of te koop aangeboden softwarecomponenten die door anderen gebruikt kunnen worden om het eigen productaanbod te verbreden. Een belangrijk aspect van webdiensten is het gebruik van algemeen geaccepteerde Internetstandaarden zoals TCP/IP, HTTP en XML.



Figuur 1 Aanroep van een webdienst

Hoewel het concept webdienst misschien nieuw is verschillen de technologische uitdagingen weinig van die van 'gewone' gedistribueerde applicaties. Uiteindelijk gaat het erom dat een applicatie diensten van een andere applicatie op het netwerk kan gebruiken, zoals schematisch weergegeven in figuur 1. Een in de context van webdiensten belangrijke standaard is SOAP. SOAP vervult de behoefte aan afspraken voor het gebruik van Internetstandaarden als TCP/IP, HTTP en XML voor het implementeren van webdiensten.

Distributie

Voor het ontwikkelen van applicaties bestaande uit gedistribueerde componenten bestaat al langer technologie, in te delen in twee categorieën. De eerste categorie bestaat uit technologie die zich richt op het *verbinden* van gedistribueerde componenten. DCOM, CORBA en RMI bieden de ontwikkelaar een standaard programmeermodel waarmee zowel client- als servercomponenten worden gemaakt. Er zijn generatoren voor zogenaamde *proxies* en *skeletons* die tussen de componenten inzitten en die zorg dragen voor het verpakken, transporteren en uitpakken van verzoeken. Het gegevensformaat dat hier voor wordt gebruikt is binair en technologiespecifiek.

De tweede categorie technologieën richt zich op het transparant aanbieden van allerlei *diensten* aan componenten. Voorbeelden van diensten zijn ondersteuning voor transacties, veiligheid, events, persistentie en procesbeheer. In deze categorie vallen recente ontwikkelingen als Enterprise JavaBeans, COM+ en de CORBA Component standaard.

Bovengenoemde technologieën worden in de praktijk gebruikt voor het ontwikkelen van gedistribueerde applicaties, maar hebben, vooral in een Internetcontext, een aantal nadelen. De belangrijkste zijn:

- Gebrek aan voldoende integratie tussen de technologieën onderling en tussen implementaties van leveranciers, waardoor niet altijd alle hoog-niveau diensten en optimalisaties gebruikt kunnen worden.
- De complexiteit van de technologie maakt het installeren en configureren niet eenvoudig, terwijl ook de aangeboden diensten niet altijd makkelijk te leren of te gebruiken zijn.

- Het gebruik van eigen communicatieprotocollen gaat niet goed samen met firewalls die communicatie via deze protocollen vaak niet zonder aanpassing van de beveiliging zullen toestaan.

In sommige gevallen zijn deze bezwaren nauwelijks relevant, bijvoorbeeld wanneer communicatie binnen een gecontroleerde omgeving plaats moet vinden. Een webfarm (communicerende, nauwverbonden servers) of integratie van applicaties binnen een organisatie zijn voorbeelden van situaties wanneer CORBA- of DCOM-oplossingen goed denkbaar zijn. In andere gevallen, waar grote aantallen en distributie over internet juist van groot belang zijn, wegen deze bezwaren zwaarder. Webdiensten zijn voorbeelden van deze gevallen.

HTTP als communicatieprotocol

In een webomgeving communiceren clients (webbrowsers) met servers (webservers) via *HTTP* (HyperText Transfer Protocol). HTTP is een simpel protocol dat met de meeste firewalls overweg kan; daarnaast is het HTTP-protocol zeer wijdverspreid. Zou het niet mogelijk zijn om de beschikbare HTTP-infrastructuur te gebruiken voor het implementeren van webdiensten?

Maak kennis met SOAP (Simple Object Access Protocol); een eenvoudige standaard voor het versturen van XML-gebaseerde verzoeken tussen componenten of webdiensten. SOAP maakt het mogelijk dat applicaties, ongeacht technologische afkomst, elkaar berichten sturen en zo van elkaars diensten gebruik maken. SOAP gebruikt het algemeen geaccepteerde en eenvoudig te genereren en lezen XML als berichtformaat. Als transportlaag kan SOAP het populaire HTTP-protocol gebruiken, waardoor componenten kunnen integreren zonder last te hebben van firewalls.

SOAP is ook een heleboel niét. SOAP-componenten moeten het stellen zonder een standaard programmeermodel, waardoor ze niet portabel zijn tussen verschillende SOAP-implementaties. Daarnaast biedt SOAP een magere infrastructuur, waarbij diensten voor bijvoorbeeld transacties of persistentie ontbreken. Het tekst-gebaseerde gegevensformaat kan tenslotte wat snelheid betreft niet op tegen de binaire berichtformaten van CORBA of DCOM.

De tekortkomingen van SOAP zijn niet onoverkomelijk en hangen, net als de tekortkomingen van andere technologieën, voornamelijk samen met de wijze van gebruik. SOAP heeft vooral waarde in de context van webdiensten, met grote aantallen door Internet van de server gescheiden clients en grove componenten. In zo'n situatie is het gebruikte gegevensformaat (en de verwerkingssnelheid daarvan) minder van belang gezien de vertraging door lange responsetijd van het netwerk. Overigens zou SOAP ook als transportlaag voor bijvoorbeeld CORBA kunnen dienen en daarmee de rol van IIOP overnemen.

SOAP in meer detail

SOAP is een initiatief van een aantal bedrijven, waaronder IBM, Ariba en Microsoft. SOAP zelf is alleen een specificatie; implementaties van de standaard komen van verschillende partijen. De huidige versie van de standaard, SOAP 1.1, is gedeponeerd bij het W3C (World Wide Web Consortium).

De SOAP-specificatie bestaat uit een aantal delen. In de eerste plaats definieert SOAP een standaard XML-berichtstructuur. Deze structuur bestaat uit een overkoepelende *envelope*, een optionele *header* voor extra informatie en de *body* waarin de daadwerkelijk uit te wisselen gegevens staan. Een voorbeeld van deze structuur is te zien in het bericht in figuur 2, bestaande uit een HTTP-header en het daadwerkelijke SOAP-bericht. SOAP maakt gebruik van XML *namespaces* die *tags* voor de structuur van de envelope en voor de codering van de gegevens definiëren.

```
 POST /Hello HTTP/1.1
Host: www.myserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
      5
    </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:sayHello xmlns:ns1="Some-URI">
      <text xsi:type="xsd:string">Hello World</text>
    </ns1:sayHello>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figuur 2 Voorbeeld SOAP-bericht

De codering is het tweede belangrijke onderdeel van de SOAP-standaard. Deze codering beschrijft hoe bepaalde soorten gegevens zoals strings, getallen en arrays worden vertaald in het SOAP-bericht. Het gebruik van de codering, die sterk is gebaseerd op de XML-schema standaard, is aanbevolen. Er kan eventueel ook een andere codering worden gekozen.

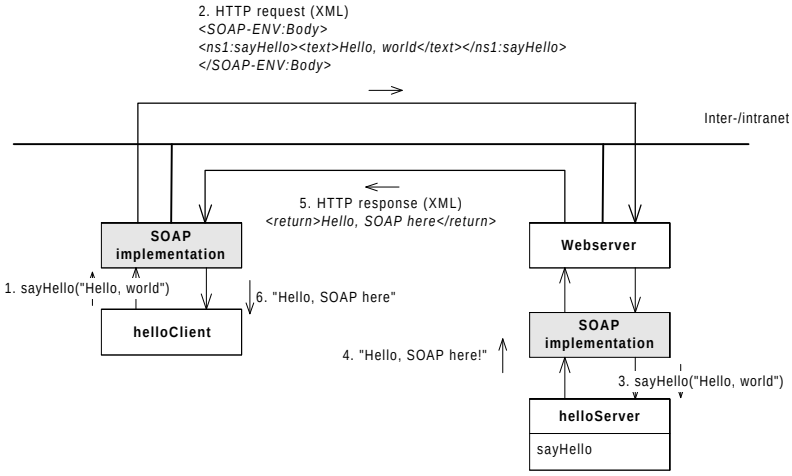
Een derde belangrijk onderdeel van de SOAP-standaard is een conventie voor het gebruik van SOAP als mechanisme voor *remote procedure calls* (RPC). De body van een verzoek bevat een XML-tag voor de aan te roepen operatie (<`ns1:sayHello`> in figuur 2), daarbinnen staan tags voor de verwachte invoerparameters (<`text`>). Het antwoord op een verzoek ziet er soortgelijk uit, maar bevat in plaats van de invoerparameters de uitvoerparameters van de aangeroepen operatie.

Het laatste deel van de SOAP-specificatie bestaat uit een standaard voor het gebruik van het HTTP-protocol voor het versturen van SOAP-berichten. Belangrijk hierbij is het

gebruik van het juiste content-type (text/xml) en de *SOAPAction* header die aangeeft wat de intentie van het SOAP-bericht is.

Ontwikkelen van SOAP-applicaties

Het ontwikkelen van SOAP-clients of -servers komt neer op het ontwikkelen van componenten die respectievelijk SOAP-berichten kunnen opstellen of juist SOAP-verzoeken kunnen begrijpen en uitvoeren. Voor de simpele webdienst uit figuur 1 komt dit neer op het ontwikkelen van een client die de methode `sayHello` (met als argument "Hello, world") aanroept op de `helloServer` en een server die op `sayHello` verzoeken kan reageren door het argument uit te printen en een antwoord terug te sturen. Het is de rol van een *SOAP-implementatie* om de tussenliggende stappen (van aanroep tot SOAP-request en/of van SOAP-request tot aan implementatie) voor de ontwikkelaar te verbergen. Figuur 3 geeft dit weer.



Figuur 3 Implementatie van een SOAP-client en -server

In de open source community zijn SOAP-implementaties beschikbaar voor de meest uiteenlopende talen en omgevingen. Voorbeelden van dergelijke implementaties Apache SOAP voor Java en SOAP::Lite voor Perl. Een andere categorie SOAP-implementaties ontstaat als aanvulling op bestaande middleware, otwikkelomgevingen en besturings-systemen. Hier spelen partijen als Iona, Roguewave en Microsoft een rol.

De verschillende SOAP-implementaties zorgen ervoor dat de ontwikkelaars zich geen zorgen meer hoeven te maken over het formaat van SOAP-berichten, over hoe berichten verstuurt worden en over hoe moet worden omgegaan met fouten. Veel producten maken het daarnaast mogelijk om bestaande Java-, COM- of CORBA-componenten om te vormen tot SOAP-webdienst. De meeste implementaties zijn gebaseerd op een RPC-

achtige manier van communicatie en bieden afbeeldingen op zowel HTTP als SMTP (Simple Mail Transfer Protocol).

De SOAP-implementaties verschillen onder andere in de mate van ondersteuning voor de SOAP-standaard, het gebruik van WSDL (Web Service Description Language), berichtcoderingen, transportprotocollen, programmeertalen en de mate van integratie met de programmeertaal.

Microsoft SOAP toolkit voor Visual Studio 6.0

Microsoft SOAP toolkit voor Visual Studio 6.0 (gratis te downloaden) is een uitbreiding op de ontwikkelomgeving van Microsoft. De toolkit biedt een COM-gebaseerde interface voor het maken van SOAP-clients en -servers. Deze kunnen daardoor in alle talen worden geschreven die COM ondersteunen, zoals Visual Basic, Visual C++ of zelfs Borland Delphi.

De SOAP-toolkit biedt keuze tussen een hoog-niveau of een laag-niveau programmeer-interface; de laatste geeft meer controle over de SOAP-communicatie. SOAP-servers kunnen gebruik maken van de bestaande Microsoft webinfrastructuur in de vorm van IIS en ASP. Om een SOAP-server te maken moeten er vier bestanden worden aangemaakt; het COM-object dat de dienst implementeert, een ASP-pagina die wordt aangeroepen door een SOAP-client, een WSDL-bestand dat de interface van de dienst beschrijft en een Microsoft-specifiek WSML bestand dat de relatie naar het COM-object legt. De laatste twee bestanden kunnen worden gegenereerd door een bijgeleverd hulpprogramma.

Een SOAP-client die gebruik maakt van de hoog-niveau programmeerinterface is eenvoudig, zie figuur 4. In de eerste regel wordt het algemene SOAP-client COM-object aangemaakt. Dit object wordt in de tweede regel geparametriseerd met een WSDL beschrijving waarin onder meer de exacte locatie en de interface van de dienst is beschreven. Het daadwerkelijk aanroepen van een operatie ziet er hetzelfde uit als een lokale aanroep, het is bijvoorbeeld niet nodig informatie over de types van de argumenten mee te geven. Het resultaat van het uitvoeren van deze code zou het voorbeeldbericht in figuur 2 kunnen zijn.

```
hello := CreateOleObject('MSSOAP.SoapClient');
hello.mssoapinit('http://www.server.com/hello.wsdl','','','');
result := hello.sayHello('Hello World');
```

Figuur 4 Voorbeeld SOAP-client, geschreven in Delphi met Microsoft SOAP-toolkit

Microsoft .NET

In het nieuwe *Microsoft .NET* platform heeft SOAP een belangrijke rol gekregen. Niet alleen dient SOAP in het platform als technologie voor het realiseren van webdiensten,

ook speelt SOAP een belangrijke rol in *.NET Remoting*; het equivalent van DCOM in .NET.

Voor het maken van webdiensten wordt in principe gebruik gemaakt van het nieuwe raamwerk voor webapplicaties, *ASP.NET*. ASP.NET webpagina's kunnen programmacode bevatten die is geschreven in een willekeurige .NET-taal en die mogelijk een webdienst implementeert. Een voorbeeld van zo'n webpagina is weergegeven in figuur 5. Merk op dat het enige SOAP-specifieke aan de gedefinieerde klasse is dat hij erft van een standaard klasse en dat alle methoden die beschikbaar moeten zijn via SOAP het attribuut `WebMethod` krijgen. Deze klasse zou overigens ook los van de pagina kunnen worden gemaakt en gecompileerd.

```
<%@ WebService Language="C#" Class="HelloWorld" %>
using System.Web.Services;
public class Hello : WebService {
     [ WebMethod ]
     public string sayHello(string text) {
       System.Console.WriteLine(text);
       return "Hello, SOAP here!";
     }
}
```

Figuur 5 Voorbeeld SOAP-server, geschreven in C# met ASP.NET

Apache SOAP

Apache SOAP is een open source SOAP-implementatie voor Java en is gebaseerd op de eerdere SOAP-implementatie van IBM. Apache SOAP integreert makkelijk met bestaande webinfrastructuur: het beschikbaar maken van een SOAP-server vraagt naast Java niet meer dan een webserver en een XML-parser, zoals de vrij beschikbare Apache Tomcat en Apache Xerces.

Het ontwikkelen van een Java SOAP-server met Apache SOAP is simpel. Er is alleen een implementatie nodig van een klasse die een via het web aan te bieden methode implementeert. De Java-variant van de `HelloWorld` SOAP-server is te zien in figuur 6.

```
public class Hello {
    public String sayHello(String text) {
      System.out.println(text);
      return "Hello, SOAP here!";
    }
}
```

Figuur 6 Voorbeeld Java SOAP-server

Om de SOAP-server voor clients beschikbaar te stellen moet de server gekoppeld worden aan een webserver. Apache SOAP biedt daarvoor een speciaal tool. Figuur 7 laat zien hoe met dit tool een ontwikkelaar de `Hello` SOAP-server die de webdienst `sayHello` aanbiedt kan koppelen aan de implementerende `Hello` javaklasse.

Figuur 7 Deployment Java SOAP-server

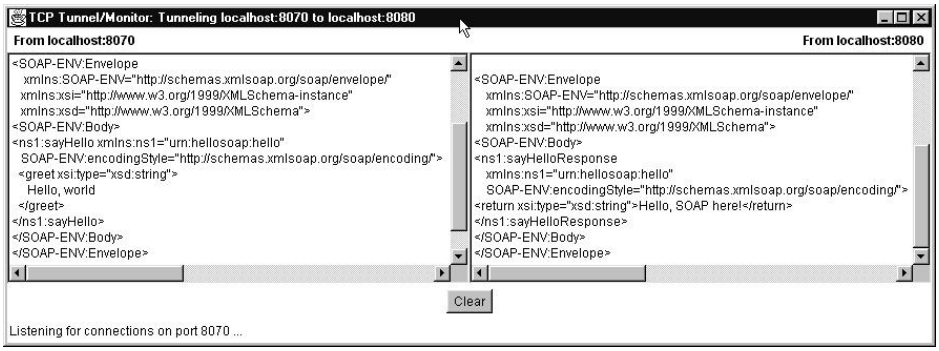
Na het instellen van deze informatie kan de webdienst direct gebruikt worden. De code in figuur 8 laat zien hoe een Java SOAP-client eruit zou kunnen zien die de eerder gedefinieerde server gebruikt. Voor het aanroepen van een webdienst moet een `Call` object worden aangemaakt waarin de locatie en naam van de SOAP-server worden aangegeven; ook de naam van de aan te roepen methode en eventuele argumenten moeten worden ingevuld. Het daadwerkelijke aanroepen van de operatie (`invoke`) levert een `Response` object op waarin het eventuele resultaat van de aanroep en mogelijke fouten zijn opgeslagen.

```
URL url = new URL(http://localhost:8080/soap/servlet/rpcrouter");
String urn = "urn:hellosoap:hello";

Call call = new Call();
call.setTargetObjectURI(urn);
call.setMethodName("sayHello");
call.setEncodingStyleURI( Constants.NS_URI_SOAP_ENC );
Vector params = new Vector();
params.addElement(new Parameter(“greet”, String.class, “Hello, world”,
    null));
call.setParams(params);

Response response = call.invoke(url, "");
Parameter result = response.getReturnValue();
System.out.println("Result= " + result.getValue());
```

Figuur 8 Voorbeeld Java SOAP-client

Bij de Apache SOAP-distributie zit ook een tool waarmee inzichtelijk kan worden gemaakt wat uiteindelijk tussen client en server heen en weer wordt gestuurd: het echte

SOAP-verzoek en antwoord dus. Figuur 9 laat zien hoe de SOAP-berichten er voor de beschreven Java server en- client uit zien.



Figuur 9 SOAP-request en –response

Er bestaan meer SOAP-implementaties in de open source community voor vele verschillende programmeeromgevingen. *SOAP::Lite* is een implementatie van SOAP voor Perl: de voorbeeldcode uit figuur 10 is de exacte perl-equivalent van de Java SOAP-client uit figuur 8. Zoals al eerder geschetst hoeven client en server dus niet met behulp van dezelfde SOAP-implementatie geschreven te zijn; de Perl client werkt met dezelfde Java SOAP-server.

```
#!perl -w

use SOAP::Lite;

print SOAP::Lite
  -> uri('urn:hellosoap:hello')
  -> proxy('http://localhost:8080/soap/servlet/rpcrouter')
  -> sayHello('Hello, world')
  -> result;
```

Figuur 10 Voorbeeld Perl SOAP-client

Gerelateerde standaarden

De voorbeelden van SOAP-clients en –servers gemaakt met verschillende implementaties illustreren hoe met SOAP webdiensten ontwikkeld kunnen worden. SOAP is echter maar één van de standaarden die van belang zijn voor webdiensten. SOAP zelf biedt niet meer dan een aantal afspraken over de structuur van XML-gebaseerde berichten. Standaarden voor het transporteren van deze berichten, zoals TCP/IP, HTTP en SMTP zijn dus onontbeerlijk. XML en bijbehorende standaarden spelen een belangrijke rol omdat zij een standaard syntax bieden voor SOAP-berichten. Er is een aanvullende specificatie van SOAP 1.1 die beschrijft hoe ook attachments met een SOAP-bericht mee kunnen worden verstuurd, gebaseerd op de MIME-standaard.

Op hoger niveau is er behoefte aan manieren voor het beschrijven en kunnen vinden van webdiensten. Aangewezen standaarden hiervoor zijn respectievelijk *WSDL* (Web Service Description Language) en *UDDI* (Universal Description Discovery en Integration). WSDL is een XML-gebaseerde taal voor het beschrijven van de berichten die een webdienst kan ontvangen en versturen en de wijze waarop deze berichten worden gebruikt in de aangeboden operaties. UDDI is een standaard voor repositories van webdiensten en geeft een indeling in verschillende soorten gegevens over bijvoorbeeld de aanbiedende partij en de aangeboden webdiensten. UDDI biedt ook een aantal SOAP-gebaseerde interfaces voor het registreren en zoeken van aanbieders en webdiensten.

SOAP in de praktijk

Er zijn op dit moment veel SOAP-implementaties beschikbaar. De verschillende voorbeelden laten zien dat het mogelijk is om op een relatief eenvoudige manier gebruik te maken van SOAP, waarbij wel moet worden opgemerkt dat in de voorbeelden bij dit artikel er geen gebruik is gemaakt van geavanceerdere mechanismen, bijvoorbeeld voor het doorgeven van objecten als parameter aan een operatie.

SOAP is geen directe plaatsvervanger voor de al langer bestaande technologie voor het ontwikkelen van gedistribueerde applicaties, maar wel een zeer bruikbaar nieuw alternatief voor situaties waarin technologieën als CORBA en DCOM minder goed toepasbaar zijn. Als het gaat om grote aantallen lichtgewicht clients en distributie over Internet, typisch het geval bij de ontwikkeling van webdiensten, bieden de eenvoud van SOAP en de mogelijkheid tot hergebruik van de wijdverspreide webinfrastructuur zeker toegevoegde waarde.

Referenties

Ballinger 2000 — K. Ballinger, J. Hawkins, P. Kumar, SOAP in the Microsoft .NET framework and Visual Studio.NET, Microsoft MSDN Library, http://msdn.microsoft.com/library/techart/Hawksoap.htm, 2000.

Barton 2000 — J.J. Barton, S. Thatte, H.F. Nielsen, SOAP Messages with Attachments, W3C Note, http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211, 2000.

Box 2000 (a) — D. Box, A young person's guide to the simple object access protocol, MSDN Magazine 3, jaargang 1, 2000.

Box 2000 (b) — D. Box et. al., Simple Object Access Protocol (SOAP) 1.1, W3C Note, http://www.w3.org/TR/SOAP/, 2000.

Box 2000 (c) — D. Box, A. Skonnard, J. Lam, Essential XML – Beyond markup, Addison-Wesley, New York, 2000.

Duivestein 2000 — S. Duivestein, A. Harteveld, SOAP = $(CBD)^2$ + XML, Software Release Magazine 3, jaargang 5, 2000.

Glass 2000 — Graham Glass, The web services (r)evolution, IBM DeveloperWorks Library, http://www.ibm.com/developerworks, 2000.

Kulchenko 2001 — P. Kulchenko, Quick start with SOAP, O'Reilly's PERL.com, http://www.perl.com/pub/2001/01/soap.html, 2001.

Loshin 1999 — P. Loshin, Web services and the simple object access protocol, Microsoft MSDN Library, http://msdn.microsoft.com/xml/general/soap_webserv.asp, 1999.

Marchal 2001 — B. Marchal, Soapbox: Why I'm using SOAP, IBM DeveloperWorks Library, http://www.ibm.com/developerworks, 2001.

Schonfeld 2001 — P. Schonfeld, Making SOAP out of Java, Java Pro 4, jaargang 5, 2001.

Skonnaard 2000 — A. Skonnaard, SOAP: The Simple Object Access Protocol, Microsoft Internet Developer 1, jaargang 5, 2000.

Resources

Apache SOAP, Xerces en Tomcat — http://www.apache.org/
SOAP::Lite — http://www.soaplite.com
Microsoft SOAP — http://msdn.microsoft.com/soap/
Microsoft .NET — http://msdn.microsoft.com/net/

Danny Greefhorst en Matthijs Maat zijn werkzaam bij het Software Engineering Research Centre te Utrecht.

