Fachhochschule Kempten
Fakultät für Informatik

SEMINARARBEIT

SOAP
(Simple Object Access Protocol)

vorgelegt von:	Per Lasse Baasch
geboren am:	10.10.1980 in Hildesheim
Matrikelnummer:	176119
Studiengang:	Informatik

Verantwortlicher Professor:	Prof. Dipl.-Inform. Nikolaus Steger
Beginn der Arbeit:	10.04.2007
Abgabe der Arbeit:	02.07.2007

Inhaltsverzeichnis

Abkürzungsverzeichnis

.NET	Dot Net, Plattform für Programme von Microsoft
Broadcast	Rundsendung
bzw.	beziehungsweise
COM	Component Object Model von Microsoft
DCOM	Distributed Component Object Model von Microsoft
ebXML	auf XML basierte Alternative zu SOAP
etc.	et cetera
Firewall	Schutzmechanismus für Netzwerke
FTP	File Transfer Protocol
GXA	Initiative von Microsoft zur Gestalltung des Webs mit XML
HTTP	Hyper Text Transfer Protocol
JRE	Java Runtime Environment, Sun Microsystems
opensource	offengelegter, freiverfügbarer Quellcode
P2P	Peer too Peer Netzwerk
Proxy	vorratsspeichernder Server fürs Internet
s.o.	siehe oben
sog.	sogenannte(r)
RSS-Feed	abrufbare Nachrichten, wie ein Zeitungsabonnement
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
usw.	und so weiter
UTF-8	global gültiger Zeichensatz
vgl.	vergleiche
VPN-Tunnel	sichere und verschlüsselte Verbindung zwischen 2 Rechnern
W3C	World Wide Web Consortium
Weblog	Internettagebuch oder auch Internetzeitschrift
XML	Extensible Markup Language (erweiterbare Auszeichnungssprache)
z.B.	zum Beispiel

Listings

Kapitel 1

Einleitung

Der Begriff SOAP steht in vielen Veröffentlichungen als Synonym für Webservices und in letzter Zeit sogar für Web 2.0. In vielen Fachzeitschriften und auch Internetveröffentlichungen ist daher auch von SOAP Web Services und SOAP Security zu lesen. Was SOAP Security für eine Bedeutung haben soll lässt sich nicht klären, auch ist unklar wer auf die Idee gekommen ist, die Begriffe SOAP und Security zu vermischen, da sie im Widerspruch zueinander stehen[1].

Sicherlich lässt sich viel über abgeleitete und auch vermutlich falsche und irrtümliche Begriffe zum Thema SOAP sagen, jedoch ist dies nicht Bestandteil dieser Ausarbeitung.

1.1 SOAP als Begriff

SOAP ist zweideutig. SOAP stellt zum einem ein Akronym dar, zum anderen aber auch einen eigenständigen Namen. Ursprünglich stand SOAP für Simple Object Access Protocol. Auf deutsch: Einfaches Objektorientiertes Zugriffs Protokoll. Es stellte sich jedoch bald nach ersten Bekannt werden von SOAP heraus, dass die Namensfindung recht irrsinnig war und ist. SOAP ist weder einfach noch rein objektorientiert. Daher versuchte man erst den Namen auf Service Oriented Architecture Protocol umzudeuten. Auf deutsch: Dienst Orientiertes Architektur Protokoll. Das Problem an dieser neuen Deutung war aber jetzt, dass es auch auf den Zweck und die Arbeitsweise von SOAP nicht voll zutreffend war. Schließlich entschloß man sich auch aus schutzrechtlichen Gründen SOAP als Begriff stehen zulassen und nicht mehr als Akronym auszuweisen[2].

[1] Vgl. [Web-Std] Seite 39
[2] Vgl. [Web-Std] Seite 39f

1.2 Geschichte

SOAP ist auf Initiative von Microsoft aus den Ideen zur Weiterentwicklung von XML-RPC endstanden. Microsoft hatte seiner Zeit bereits großes Interesse an Dave Winer geweckt mit seiner Erfindung XML-RPC, so dass Redmond ihn schließlich für die Entwicklung von SOAP gewinnen konnte. Dave Winer war jedoch nur inoffiziell bei Microsoft angestellt, wie er es in seinem Weblog veröffentlichte. So ging es vorerst darum, XML-RPC um das Know-how von Microsoft zu erweitern. Aus dieser Zusammenarbeit zwischen Microsoft und Dave Winer endstand schließlich SOAP, was am Anfang zunächst XML-RPC weitesgehend identisch war. Microsoft selbst hatte seiner Zeit keinen Erfolg mit COM und DCOM. Darüber hinaus stand den Entwicklern die .NET-Plattform schon bevor, so dass Microsoft eine XML-Variante mit breiter Unterstützung suchte. Zu den Partnern zählten und zählen u.a. Compaq Computer Corporation, Hewlett Packard Company, SAP AG, Lotus Development Corporation. Als Ergebnis dieser Zusammenarbeit reichte Microsoft SOAP am 18.April 2000 beim W3C ein und so wurde die erste Version als „W3C Note 08 May 2000“ veröffentlicht[1].

Ein Blick auf die Autoren der Liste lohnt sich, da man das Interesse der größten Unternehmen auf dem Gebiet der Kommunikation mittels XML klar erkennen kann:

- Don Box, DevelopMentor
- David Ehnebuske, IBM
- Gopal Kakivaya, Microsoft
- Andrew Layman, Microsoft
- Noah Mendelsohn, Lotus Development Corp.
- Henrik Frystyk Nielsen, Microsoft
- Satish Thatte, Microsoft
- Dave Winer, UserLand Software, Inc.

Auffällig an dieser Liste ist, dass eines der größten Unternehmen fehlt. Sun Microsystems hatte seiner Zeit eigene Interessen, Vorstellungen und auch Entwicklungen, um die Anforderungen von und an SOAP abzudecken. Mit Java und

[1] Siehe auch http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

den anhängigen Applikationen und Protokollen hatte Sun Microsystems bereits eine funktionsfähige und plattformübergreifende Technologie zur Verfügung. Verständlicherweise wollte Sun Microsystems auch keine Allianz mit seinem Erzrivalen Microsoft eingehen wo auch bis heute die Fronten verhärtet sind, wie beispielweise beim Einsatz der JRE in Microsoft Windows Produkten zeigt.

1.3 Versionen

Die erste offizielle Version von SOAP trägt die Versionsnummer 1.1 und wurde wie im vorherigen Abschnitt bereits erwähnt am 8. Mai 2000 veröffentlicht. Diese Version besteht aus einem einzigen Modul, das heißt, dass im Grunde keine Aufspaltung und auch keine Differenzierung in den einzelnen Arbeitsbereichen von SOAP vorgenommen wurde. SOAP 1.1 wird beim W3C nur als so genannte Note geführt. Bei einer Note handelt es sich nicht wie gewohnt beim W3C um einen verabschiedeten einheitlichen Standard, sondern um eine meist von Externen eingebrachte Veröffentlichung, die zwar opensource gesetzt wird, aber vorzugsweise zur Weiterentwicklung bereitgestellt wird[1].

Kurz nach Veröffentlichung von Version 1.1 hat sich das W3C entschlossen eine neue Version zu entwickeln, welche unter dem Namen SOAP Version 1.2 bekannt ist[2]. Die neue Version wurde offiziell am 24. Juni 2004 zu einem Standard (W3C Recommendation) und ist im Gegensatz zur vorherigen Version in die folgenden drei Teilbereiche aufgeteilt:

- SOAP Version 1.2 Part0: Primer[3]
- SOAP Version 1.2 Part1: Messaging Framework[4]
- SOAP Version 1.2 Part2: Adjuncts[5]

Interessanterweise haben die Entwickler der Version gewechselt, was an einer allgemeinen Unzufriedenheit lag, die bei der Entwicklung von SOAP 1.1 begann. So schreibt Dave Winer in seinem Weblog, dass er SOAP als viel zu kompliziert im Gegensatz zu XML-RPC hält. Des Weiteren lässt sich auch feststellen, dass die meisten heute zur Verfügung gestellten Webservices, die SOAP Technologie

[1] Vgl. [Web-Std] Seite 42
[2] Siehe auch http://www.w3.org/TR/soap12/
[3] Siehe auch http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
[4] Siehe auch http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
[5] Siehe auch http://www.w3.org/TR/2003/REC-soap12-part2-20030624/

nutzen, immer noch auf Version 1.1 basieren. Es gibt derzeit aber starke Tendenzen auf die neuere Version umzusteigen. In diversen Weblogs und Foren wie z.B. bei xml-coverpages [1] wird berichtet, dass es in allen Bereichen Bestrebungen gibt, auf die neue Version umzusteigen und dass dieser Standard auch den Markt der Webservices Protokolle in naher Zukunft beherrschen wird.

[1] Siehe auch http://xml.coverpages.org/soap.html

Kapitel 2

Funktionsweise

In diesem Kapitel möchte ich etwas näher auf die Funktionsweise von SOAP und den Funktionsumfang eingehen. Ich werde von XML-RPC ausgehen, welches die Grundlage von SOAP darstellt. Daher ist es sinnvoll die Entwicklung von SOAP in Bezug auf XML-RPC zu betrachten, da aus dessen Problemen und Funktionsmängeln es erst zur Entwicklung von SOAP gekommen ist (siehe Geschichte von SOAP).

2.1 XML-RPC

XML-RPC steht für „Extensible Markup Language Remote Procedure Call“ und stellt ein einfaches Protokoll bzw. Definition für eine Kommunikation auf Basis von XML dar. XML-RPC arbeitet immer synchron, was bedeutet, dass eine Anfrage immer eine Antwort erwartet[1].

Listing 2.1: XML-RPC synchrone Kommunikation, Beispiel 1

```
1 Client stellt/sendet Anfrage:    Ich gehe heute ins Kino
2 Server gibt eine Antwort:        OK
```

In diesem obigen einfachen Beispiel wird deutlich, dass zum einem auf eine Anfrage immer eine Antwort folgt und zum anderen auch direkt, wie sinnlos diese auch sein mag. Dies erzeug einen großen Umfang an Kommunikation, wenn auch gar keine benötigt wird oder auch erst später stattfinden kann. Eine Antwort kann nicht einmal explizit ausgeschlossen werden.

XML-RPC arbeitet ausschließlich mit HTTP und XML. Dazu wird HTTP ab der Version 1.0 unterstützt. Meist wird jedoch aufgrund eines größeren Funktionsum-

[1] Vgl. [Web-Tech] Seite 312ff

fangs auf HTTP 1.1 zurückgegriffen. Client und Server senden bei einer Kommunikation jeweils einen HTTP-Header und ein XML-Dokument. Im HTTP-Header selbst wird die Art der Kommunikation festgelegt, welche in diesem Fall „POST /xmlrpc HTTP/1.0“ ist und den Inhalt der folgenden Schritte auf „Content-Type: text/xml“ definiert. Weitere Parameter des XML-Headers sind an dieser Stelle erstmal irrelevant und werden daher nicht näher behandelt. Im Anschluß an den HTTP-Header folgt das eigentliche XML-Dokument, welches einen Aufbau wie jedes andere XML-Dokument hat. Sämtliche Tags sind durch den Server/Client und derer benutzter XML-Schemata vordefiniert. Es gibt hierzu einige Standards, siehe die Spezifikation von Dave Winer[1], jedoch sind diese nicht bindend und können je nach Belieben verändert werden.

Der folgende Abschnitt verdeutlicht noch einmal den Ablauf einer XML-RPC Kommunikation.

Listing 2.2: XML-RPC Kommunikationsablauf, Beispiel 2

```
1 Client sendet HTTP-Header an Server
2 Client sendet XML-Nachricht an Server
3 Server sendet HTTP-Header an Client zurück
4 Server sendet XML-Nachricht an Server zurück
```

2.2 SOAP

SOAP arbeitet wie sein sog. Vorgänger XML-RPC auch mit HTTP-Headern und XML-Dokumenten. Jedoch ist SOAP selbst nicht auf HTTP-Header beschränkt und kann daher auch in anderen Umgebungen wie z.B. FTP benutzt werden. Im Folgenden bleibe ich jedoch bei der Darstellung mit HTTP, da dies einfacher und verständlicher ist. SOAP arbeitet synchron wie XML-RPC, aber auch asynchron, welches eines der größten Probleme von XML-RPC darstellt. Das bedeutet im Klartext, dass eine einseitige Kommunikation möglich ist.
SOAP ist jedoch entgegen erster Vermutungen keineswegs auf den reinen Nachrichtenaustausch via RPC's oder ähnlichen beschränkt. Ferner sind laut der Spezifikation auch dezentrale Kommunikationen möglich, wie man sie aus dem Bereich des File-Sharings kennt (P2P). Das bedeutet folglich, dass die Kommunikation nicht 1:1 erfolgen muss, sondern auch Broadcasts möglich sind, was eine 1:n Beziehung darstellt. Diese Art ist jedoch in der Praxis noch nicht in Gebrauch, aber

[1] Siehe auch http://www.xmlrpc.com/spec

man könnte sich vorstellen, dass eine P2P-Client/Server Applikation wie z.b. E-Donkey mit so etwas arbeiten könnte. Wenn sich ein neuer Client/Server bekannt machen will, so schickt dieser einen Broadcast an alle Teilnehmer um darüber zu informieren, dass er da ist. Als zweites könnte er im Broadcast eine Liste von Dateien/Diensten bereitstellen, welche er den Broadcast-Teilnehmern zur Verfügung stellt.

2.2.1 Kommunikation

SOAP kennt, wie bereits erwähnt, verschiedenste Arten der Kommunikation. Nachfolgend werden die wichtigsten und auch die meist genutzten Formen vorgestellt. Hierbei sollte das Augenmerk auf dem liegen, wie übermittelt wird. Grundsätzlich unterscheidet man dabei zwischen einer Nachricht und einer Anfrage/Anwort. Beide Formen können sowohl einseitig als auch bidirektional erfolgen, wie die Varianten 1-3 verdeutlicht. Die Variante 4 ist als Ausblick zu sehen, was laut der Spezifikation von SOAP noch möglich ist.

Variante 1 der Kommunikation (einseitig)[1]:
(Sender) → Nachricht → (Empfänger)
Beispiel: Der Postbote steckt jeden Tag die Post in ihren Briefkasten.

Variante 2 der Kommunikation (bidirektional)[2]:
(Sender) → Anfrage → (Empfänger)
(Sender) ← Antwort ← (Empfänger)
Beispiel: Sie öffnen dem Postboten die Tür, damit er ihnen die Post geben kann.

Variante 3 der Kommunikation[3]:
(Sender) → Nachricht → (Empfänger1..N)
Beispiel: Sie öffnen dem Postboten die Tür, damit er allen Bewohnern die Post vor die Tür legen kann.

Variante 4 der Kommunikation (einseitig Vermittlung):
(Sender) → Nachricht → (Intermediär1..N) → Weiterleiten → (Empfänger1..N)
Beispiel: Der Postbote gibt ihre Post dem Nachbarn, welcher ihnen diese später

[1] Vgl. [Web-Std] Seite 44, Abb. 2.1b
[2] Vgl. [Web-Std] Seite 44, Abb. 2.1a
[3] Vgl. [Web-Std] Seite 44, Abb. 2.1c

gibt.

Grundsätzlich sind für das Verständnis der Kommunikation von SOAP drei Kernelemente immer zu berücksichtigen bzw. zu unterscheiden. Die Spezifikation ist noch weitaus komplexer als die dargestellten Beispiele es zeigen, jedoch reicht dieses Verständnis für die Durchführung selbst größerer Projekte aus.

- **Nachrichten-Art**
  RemoteProcedueCall und nachrichtenorientiert
- **Nachrichten-Pfad**
  Direkt (Client-Server) oder indirekt über Intermediär (Vermittler)
- **Transportprotokoll**
  HTTP, SMTP, FTP etc.

2.2.2 Nachrichten-Aufbau

Eine SOAP-Nachricht besteht aus drei Teilen[1]:

- **SOAP-Envelope** (=Umschlag)
  Der SOAP-Envelope ist der Rahmen bzw. Umschlag, der die eigentliche SOAP-Kommunikation ausmacht. In ihm sind zum einem der SOAP-Header also der Kopf, als auch der eigentliche Hauptteil, der SOAP-Body enthalten.
- **SOAP-Header** (=Kopf)
  Der SOAP-Header kann z.B. Authentifizierungs- , Autorisierungs- und Transaktionsinformationen enthalten.
- **SOAP-Body** (=Hauptteil)
  Im SOAP-Body steht die eigentliche Nachricht, die Variablen, die Funktionsaufrufe, die Deklarationen usw., welche im Folgenden behandelt werden.

2.2.3 SOAP-Nachricht

Das HTTP-Protokoll ist derzeit das am meist genutzte Protokoll zum Austausch von Nachrichten mittels SOAP. Des Weiteren ist ein HTTP-Header sehr leicht verständlich und das Protokoll an sich auch jedem bekannt, was eine Erklärung leichter macht.

[1] Vgl. [Web-Std] Seite 47, Abb. 2.3

Im Normalfall läuft eine Kommunikation mittels HTTP über den Port 80, dies soll auch dem Folgenden als Grundlage dienen.

Listing 2.3: SOAP-Nachricht, Beispiel 1

```
1  POST /ausgabe.php HTTP/1.1
2  HOST: localhost
3  Content-Type: text/xml; charset=utf-8
4  Content-Length: 350
5  SOAPAction: "http://somesite.you/ausgeben"
6  <?xml version="1.0" encoding="utf-8"?>
7  <soap:Evelope xmlns:xsi="http://www.w3.org/2001/XMLSchema
8  -instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
10  <soap:Header>
11   <m:RequestID xmlns:m="http://somesite.you/hallowelt/">
12    default
13   </m:RequestID>
14  </soap:Header>
15  <soap:Body>
16   <ausgeben xmlns="http://somesite.you/hallowelt">
17    <text>Hallo Welt!</text>
18   </ausgeben>
19  </soap:Body>
20 </soap:Envelope>
```

Zeilen Erläuterung nach Zeilennummern[1]:

1. zeigt, welche lokale Datei gerade abgesendet wird sowie die HTTP-Protokoll-Version.
2. sagt aus, von welcher Quelle (Rechner) die Anfrage abgesetzt wird.
3. setzt den Inhalt der späteren Nachricht auf Plaintext-XML mit dem Zeichensatz UTF-8.
4. zeigt an, wie groß der Inhalt der folgenden Nachricht ist, 350 Byte.
5. zeigt an, welche SOAP Aktion ausgeführt werden soll.

[1] Vgl. [Web-PHP] Seite 73ff

6. Beginn der eigentlichen Nachricht und Deklaration als ein XML-Nachricht mit dem Zeichensatz UTF-8
7. Deklaration der SOAP-Nachricht mit Definition des Envelope, benutzte Formate und Standards
8. s.o.
9. s.o.
10. Beginn des SOAP-Headers // der gesamte SOAP-Header ist optional!
11. Beginn einer Subroutine zur Authentifizierung
12. Argument „default“ wird übergeben
13. Ende der Subroutine
14. Ende des SOAP-Headers
15. Beginn des SOAP-Bodys
16. Aufruf der Routine „ausgeben“ auf „http://soap.skycube.net/hallowelt“
17. Übergabe einer Variable vom Typ Text mit Inhalt „Hallo Welt!“
18. Ende des Routinenaufrufs „ausgeben“
19. Ende des SOAP-Bodys
20. Ende des SOAP-Envelope

Kapitel 3

SOAP Details

Im Folgenden beziehe ich mich ausschließlich auf die Spezifikationen von SOAP 1.2. Dies hat zur Folge, dass einige bekannte Attribute nicht genannt bzw. ausgelassen werden.

3.1 SOAP-ENVELOPE

Im SOAP-Header werden u.a. die zu verwendenden Namensräume angegeben. Ein Beispiel dazu finden Sie im Kapitel 2. Hierbei gibt es keine Begrenzung auf die Anzahl der Namensräume. Theoretisch könnte man daher Hunderte angeben. Grundsätzlich fängt man jedoch immer mit den Standardnamensräumen der XML- und SOAP-Spezifikationen an. Der Namespace gibt dabei an, welche Tags möglich sind und welche Schemata zur Verfügung stehen. Man kann natürlich neben den offiziellen von W3C veröffentlichten Namensräumen auch eigene schreiben. Dies kann aber bei der Kompatibilität von Client-Server Anwendungen zu Problemen führen, wenn der eine Partner die Namensräume nicht kennt, angibt oder erreichen kann[1]. Als weiteres kann man noch einen encodingStyle einbinden, dieser wird im Bereich SOAP-Header näher behandelt.

Einen Namensraum gibt man wie folgt an:
xmlns:xxx=“http://domain.com/meinnamespace/„
„xxx“ ist die später zu verwendende Kurzbezeichnung mit der im Späteren darauf referenziert werden kann.

[1] Vgl. [Web-Std] Seite 51f

Bei der Kurzbezeichnung gibt es reservierte Namen:

xsi	steht für eine XML-Schema-Instanz
xsd	bindet das XML-Schemata mit ein
soapenc	legt den encodingStyle fest
soap	legt die zu verwendende SOAP-Spezifikation fest
type	legt die zur Verfügung stehenden Typen (Variablen) fest

3.2 SOAP-HEADER

Zu dem SOAP-Header muss man als erstes anmerken, dass er laut der Spezifikation optional ist. Das bedeutet, dass man ihn in der eigenen Implementierung einfach weglassen kann. In der Praxis geschieht dies auch meist, da für den überwiegenden Nachrichtenaustausch in aller Regel keine Authentifizierungsmethoden oder Ähnliches benötigt werden[1].

Laut SOAP-Spezifikation (kann) ein SOAP-Header vier Attribute enthalten:

mustunderstand: Dieses Attribut weißt den Empfänger einer SOAP-Nachricht an, dass er den Header unbedingt kennen muss. Das Value kann TRUE oder FALSE einnehmen.

encodingStyle: Legt fest, in welcher Art und Form der Header und der Body geschrieben und gelesen werden muss. Der encodingStyle kann, muss aber nicht festgelegt werden. Denkbar ist es für lokale Lösungen, dass man den encodingStyle jeweils im zu verwendenden TAG im SOAP-Body festlegt. Die am meisten verwendeten Methoden sind hierbei „rpc/encoded“ und „document/literal“.

document/encoded = ist eine Nachricht, die nach den SOAP-Regeln kodierte Parameter besitzt
rpc/encoded = arbeitet mit zwei Teilen der SOAP-Spezifikationen, RPC's und Encoding, kann nicht asynchron verwendet werden
document/literal = verwendet keinen Teil der SOAP-Specifikationen, sondern wird meist synchron verwendet

[1] Vgl. [Web-Std] Seite 52f

rpc/literal = kommt in der Kombination nicht in der Praxis vor

relay: Das Relay (zu deutsch: weitergeben) Attribut weist einen Intermediär dazu an, die Nachricht an den nächsten Intermediär oder den endgültigen Empfänger weiterzugeben. Das Attribut kann TRUE oder FALSE einnehmen.

role: Bestimmt, welche Rolle der aktuelle Empfänger (Intermediär oder End-Empfänger) einnimmt. [1]

3.3 SOAP-BODY

Der SOAP-Body liegt, wie der SOAP-Header innerhalb des SOAP-Envelope. Sollte der optionale SOAP-Header vorhanden sein, so folgt der SOAP-Body nach dem SOAP-Header.
Der SOAP-BODY enthält die eigentlichen Variablen und damit den Inhalt der SOAP-Nachricht. Das Aussehen des SOAP-Body wird wiederum durch den Header und den Envelope festgelegt. Der SOAP-Body beginnt immer mit dem TAG „<SOAP-ENV:Body>“ und endet mit dem TAG „</SOAP-ENV:Body>“. Die hier zu Verfügung stehenden Variablen und Funktionen werden im Header, Envelope oder direkt im TAG eingebunden[2].

3.4 Datentypen

Wie in jeder Programmiersprache kann und muss man Variablen deklarieren, die nach gewissen Standards genormt interpretiert werden. Eine komplette Liste der möglichen Variablen steht unter http://www.w3.org/2003/05/soap-encoding zur Verfügung. Diese Liste entspricht fast komplett den XML-Schemata der Version 2, welche unter http://www.w3.org/TR/xmlschema-2/ einsehbar sind.

[1] http://www.w3.org/TR/2003/REC-soap12-part1-20030624/#soaproles
[2] Vgl. [Web-Std] Seite 54ff

Eine Liste der wichtigsten Datentypen:

Name:	**Wert/Beispiel:**	**Beschreibung:**
string	„hallo welt“	Zeichenketten in Unicode
text	„hallo welt“	wie String
boolean	true/false	Wahrheitswert
int	+-2.147.483.64(7)(8)	Ganzzahl
float	4 Byte	Fließkommazahl
double	8 Byte	Fließkommazahl
dateTime	2004-04-18T12:00:00	Datum im ISO-Format
language	de	Sprachstandard gemäß RFC 1766

3.5 SOAP-Fehler

SOAP benutzt zum einem das Fehlermanagement des Trägerprotokolls, besitzt aber auch ein eigenes. Hierzu ist grundsätzlich zu beachten, dass die Versionen 1.1 und 1.2 verschieden und nicht kompatibel zueinander sind. Sie unterscheiden sich grundsätzlich voneinander. Im Folgenden wird daher nur auf die eine, die neuere Variante der Version 1.2 eingegangen[1].

Sollte ein Fehler bei der SOAP-Kommunikation auftreten und dieser durch das SOAP-Fehlermanagement behandelt werden, so muss im SOAP-Envelope-Body und SOAP-Body ein *Fault*-Tag stehen.

Listing 3.1: SOAP-Fehlermanagement, Beispiel 1

```
1  <env:Envelope
2  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
3  xmlns:m="http://www.example.org/timeouts"
4  xmlns:xml="http://www.w3.org/XML/1998/namespace">
5   <env:Body>
6    <env:Fault>
7     <env:Code>
8       <env:Value>env:Sender</env:Value>
9       <env:Subcode>
10       <env:Value>m:MessageTimeout</env:Value>
11      </env:Subcode>
12    </env:Code>
```

[1] Vgl. [Web-Std] Seite 62ff

```
13      <env:Reason>
14        <env:Text xml:lang="en">Sender Timeout</env:Text>
15      </env:Reason>
16      <env:Detail>
17        <m:MaxTime>P5M</m:MaxTime>
18      </env:Detail>
19    </env:Fault>
20  </env:Body>
21 </env:Envelope>
```

Quelle: SOAP Spezifikation[1]

Zeilen Erläuterung:

1. Beginn des SOAP-Envelope (Umschlag)
2. Namespace Deklaration
3. Namespace Deklaration
4. Namespace Deklaration
5. Beginn des SOAP-Envelope-Body
6. Beginn des Fehlermanagements
7. Der Fehlercode, hat einen oder mehrere untergeordnete Fehlercodes
8. Erster Fehlercode, mit dem Namen Sender, enthält einen Fehler
9. Beginn der untergeordneten Fehlercodes (in diesem Fall nur einer)
10. Deklaration des Fehlercodes m:MessageTimeout
11. Ende des untergeordneten Fehlercodes
12. Ende der Fehlercode Deklaration
13. Beginn der Fehlercode Kurzbeschreibung
14. Kurzbeschreibung des Fehlers/ Begründung
15. Ende der Fehlercode Kurzbeschreibung

[1] Siehe auch http://www.w3.org/TR/2003/REC-soap12-part1-20030624/#faultcodes

16. Beginn der ausführlichen Fehlerbeschreibung
17. Ausführliche Beschreibung des Fehlers (optional)
18. Ende der ausführlichen Fehlerbeschreibung
19. Ende des Fehlermanagements
20. Ende des SOAP-Envelope-Body
21. Ende des SOAP-Envelope(Umschlag)

Das SOAP-Fehlermanagement ist recht komplex und lässt sehr viele Möglichkeiten zu. Eine komplette Kommentierung und Nennung aller Möglichkeiten laut der Spezifikation von SOAP Version 1.2 würde diesen Rahmen sprengen. Wer sich dennoch tiefer mit SOAP auseinandersetzen und SOAP einsetzen möchte, sollte das Fehlermanagement vor einer Implementierung genau betrachten, da es in der Regel eine große Arbeitsersparnis nach sich zieht.

Kapitel 4

Ausblick

4.1 Transportprotokolle

Wie bereits in den vorherigen Abschnitten erwähnt wurde, handelt es sich bei SOAP um ein Protokoll, genauer gesagt um ein Nachrichtenübertragungsprotokoll. SOAP ist aber alleine nicht nutzbar, es bedarf eines Übertragungsprotokolls. Diese Aufgabe übernimmt in den vorherigen Abschnitten das bereits erwähnte HTTP-Protokoll. Der Vorteil des HTTP-Protokolls liegt auf der Hand. Jeder Host und Client versteht und kennt das Protokoll. Firewalls und Proxy-Server lassen dieses Protokoll normalerweise passieren. Die Funktionen des Protokolls, wie Security, Error-Handling und Intigritätsprüfung können dabei ohne Einschränkungen verwendet werden. Im HTTP-Protokoll sind diese Funktionen ausgereift und bewährt. Darüber hinaus steht mit HTTPS sogar eine verschlüsselte Variante des HTTP-Protokolls zur Verfügung[1].

Das HTTP-Protokoll ist aber nicht das einzige Protokoll, auf dem SOAP- Nachrichten quasi huckepack versendet werden können. Zwar werden derzeit mehr als 95% aller SOAP-Nachrichten darüber versand, jedoch gibt es auch Alternativen.

Hier steht an erster Stelle das SMTP-Protokoll (Simple Mail Transport Protocol), welches für den Versand von E-Mails normalerweise zuständig ist. Ein Problem stellen die möglichen Kommunikationsformen dar, weil SMTP nur eine Anfrage (Request) zulässt und meist nur mit einem „OK“ antwortet. Bei einer komplexeren Kommunikation würde das bedeuten, dass man die gesamte Kommunikation via SMTP neu implementieren müsste[2].

[1] Vgl. [Web-Std] Seite 65f
[2] Vgl. [Web-Std] Seite 67, Absatz 1

Als weitere Alternative wäre die in die Jahre gekommene Kommunikation mittels FTP (File Transfer Protocol) zu nennen. Eigentlich dient dieses Protokoll zur Übertragung von ASCII- und Binärdaten auf einen anderen Host. Denkbar wäre eine komplette Kommunikation eines FTP-Client/Servers mittels SOAP oder auch eine Nachrichten-Verteilungszentrale. Der Vorteil besteht darin, dass der FTP-Server bereits mit ASCII- und Binärdaten umgehen kann und diese sehr effektiv mittels FTP-Protokoll versenden kann. Das HTTP- und das SMTP-Protokoll wurden nicht entwickelt, um große Datenmengen auszutauschen, wie es beim FTP-Protokoll gedacht ist. Dies verursacht immer noch große Probleme, sowie unnötigen und überflüssigen Traffic. Als Negativ ist die Sicherheit des FTP-Protokolls hervorzuheben, es gibt praktisch keine gute Implementierung des FTP-Protokolls mit eigener Sicherheit. Es wird hierzu immer empfohlen einen VPN-Tunnel zu benutzen[1].

Zuletzt ist noch das SNMP-Protokoll zu nennen (Simple Network Management Protocol). Das SNMP-Protokoll dient eigentlich zur Verwaltung von Netzwerkkomponenten wie Router, Server, Switches, Drucker usw., ist aber aufgrund seiner Architektur auch für andere Aufgaben geeignet. Die Aufgabe des SNMP-Protokolls besteht im Regelfall darin Netzwerkkomponenten zu überwachen, fernzusteuern und eine Fehlererkennung/ Fehlerbenachrichtigung durchzuführen. Die Überwachungsfunktion könnte gut als Dienst wie beispielsweise das Lesen von RSS-Feeds interpretiert werden.

4.2 Zukunft

Was die Zukunft dem SOAP-Protokoll bringt, wenn man das Internet nach Übertragungsprotokollen mit dem Ziel einer XML Nachrichtenübermittlung durchsucht, scheint festgelegt zu sein. SOAP gehört inzwischen zu dem Standard schlechthin und hat seinen sog. Vorgänger XML-RPC so gut wie verdrängt. Alternativen zu SOAP gibt es kaum, bzw. finden kaum Verbreitung wie z.B. die Initiative von Microsoft mit GXA (Global XML Web Services Architecture). Als weitere Alternative ist ebXML zu nennen, welches zwar historisch gesehen schon länger im Internet herumgeistert, aber nie Begeisterung bei den Entwicklern und Softwareschmieden gefunden hat. Interessanterweise benutzt inzwischen auch ebXML SOAP-Fetaures

[1] Vgl. [Web-Std] Seite 67, Absatz 2

und ist angeblich kompatibel[1].

Wer also heute seine Anwendung auf SOAP-Nachrichten basierend aufbaut, macht daher keinen Fehler. Als Zukunfsaussicht ist ein Workdraft des W3C zu beobachten, was ein gängiges Problem von XML und SOAP zu lösen versucht, nämlich Austausch von Binärdaten. Das sog. Attachment-Feature ist das einzige, was derzeit SOAP noch fehlt, um eine komplette Kommunikation ohne Hilfsprotokolle zu gewährleisten[2].
Siehe dazu auch http://www.w3.org/TR/soap12-af/

[1] Vgl. [Web-Std] Seite 69f
[2] Vgl. [Web-Std] Seite 71f

Literaturverzeichnis

[Web-Std] *Tobias Hauser und Ulrich M. Löwer*(Hrsg.):
Web Services - Die Standards: Einstieg in alle Standards und Spezifikationen, 2004; 1. Aufl., Bonn: Galileo Press, 2004

[Web-PHP] *Christian Wenz und Tobias Hauser*(Hrsg.):
Web Services mit PHP: Technische Grundlagen und Praxisbeispiele, 2004; 1. Aufl., Bonn: Galileo Press, 2004

[Web-Tech] *Anatol Badach, Sebastian Rieger, Matthias Schmauch*(Hrsg.):
Web-Technologien: Architekturen, Konzepte, Tends, 2003; 1. Aufl., München: Carl Hanser Verlag, 2003

Erklärung

ERKLÄRUNG

Ich versichere, dass ich diese Seminararbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben, sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

Kempten, den _____ Unterschrift _____

ERMÄCHTIGUNG

Hiermit ermächtige ich die Fachhochschule Kempten zur Veröffentlichung von Teilen und/oder meiner gesamten Arbeit, z.B. auf gedruckten Medien oder auf einer Internetseite, sofern die GNU Free Documentation License v1.2 eingehalten wird.

Kempten, den _____ Unterschrift _____