In diesem Kapitel

Apache SOAP

1.1. Um was geht's?

Sie wollen sich Apache SOAP kennen lernen, ohne über alle möglichen Probleme zu stolpern und ohne alle Details schon zu Beginn kennen lernen zu wollen.

1.1.1. Kurze Geschichte von Apache SOAP

Apache SOAP stammt wie so viele Apache Technologien von IBM. Ursprünglich hiess Apache SOAP IBM-SOAP. Apache hat die Software überarbeitet, mithilfe von IBM.

1.1.2. Zukunft von Apache SOAP

Das neuste SOAP Release von Apache heisst Axis, Es unterscheidet sich grundlegend von IBM-SOAP. Die Architektur von Axis ist vollständig neu. Axis verwendet den SAX Parser; Apache SOAP verwendet den DOM Parser. Axis ist dynamisch erweiterbar und kann Java Dateien selbständig übersetzen und einbinden. Axis ist modularer und schneller. Axis hat aber auch einige Nachteile: Release 1.0 war schlicht nicht funktionsfähig. Jetzt sieht's besser aus.

Apache SOAP können aber für viele verschiedene SOAP Server verwendet werden. Apache SOAP kennt zwei Patterns:

- RPC basiert:
  der Client sendet den Prozedurnamen und die Parameter und erwartet als Antwort ein oder mehrere Variablen.

- Message basiert:
  diese Version ist detaillierter; SOAP RPC basiert darauf. In diesem Modell müssen Sie sich selber um (fast) alles kümmern: Umschlag, Kopf, Rumpf, einpacken, auspacken.

Hier ein RPC Beispiel (als SOAP Message: diese sehen Sie aber kaum):

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:wiegehtsResponse xmlns:ns1="urn:wiegehts" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:string">Hans - Wie gehts Dir?</return>
</ns1:wiegehtsResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Der Datentyp der Rückgabe wird in XML Schema Schreibweise angegeben (`xsi:type="xsd:string"`).

Andere SOAP Implementierungen (Microsoft zum Beispiel) liefern eine Antwort, die anschliessend interpretiert, gecastet und analysiert werden muss.

In Axis können Sie einen Standard-Datentyp für Rückgabewerte angeben.

Axis unterstützt auch die aktuelle Version von WSDL (Web Service Description Language): falls Sie die WSDL Beschreibung eines Dienstes benötigen, geben Sie einfach "`<Dienst>?WSDL`" ein. Dann wird die WSDL Beschreibung generiert.

Beispiel:
`http://localhost:8080/axis/services/HelloWorld?wsdl.`

Zudem enthält Axis ein WSDL –to- Java Tool: damit können Stubs generiert werden, so dass der Methodenaufruf wie ein lokaler Aufruf aussieht (...CORBA / RMI Style...).

Sie finden mehr Details über Axis in der Starthilfe zu Axis und den Links dazu.

1.2. Installieren von Apache SOAP

Apache SOAP ist Open Source, kann also problemlos herunter geladen und genutzt werden. Bevor Sie SOAP nutzen können, müssen Sie einen Web Server, idealerweise mit einem Servlet Container, beispielsweise Apache Tomcat aus dem Apache Jakarta Projekt installieren.

Wie dies geschieht, steht in der "Starthilfe zu Tomcat". Eigentlich gibt's da nicht viel zu tun:

- herunterladen
- entzippen
- starten
- stoppen evtl. Konfiguration anpassen und neu starten

Nun können Sie SOAP herunterladen (http://xml.apache.org/soap). Entzppen Sie das ZIP Archiv, einfach in `C:` (`C:\soap-2_2` wird angelegt). Zudem sollten Sie die Umgebungsvariable `SOAP_HOME` setzen:

```
SET SOAP_HOME=C:\soap-2_2
```

Beachten Sie: Apache SOAP benötigt mail.jar aus JavaMail, jaf.jar aus Java Activation Framework und xerces,jar aus dem Xerces Apache Projekt, neben Apache SOAP soap.jar.

1.3. Tomcat Konfiguration für Apache SOAP

Sie haben zwei Möglichkeiten, Apache Tomcat für SOAP zu konfigurieren, SOAP fähig zu machen (Achtung: die Installationsanleitung in der Doc zu Apache SOAP ist fehlerhaft. Eine korrigierte Version finden Sie unter [alles auf einer Zeile]
`http://cvs.apache.org/viewcvs.cgi/*checkout*/xml-soap/java/docs/install/tomcat.html?rev=HEAD&content-type=text/html`):

`CATALINA_HOME` ist bei mir `CATALINA_HOME=C:\jakarta-tomcat-4.1.27`

Variante 1: mithilfe des SOAP Web Archives

1. Im Archiv SOAP_HOME finden Sie im Verzeichnis `%SOAP_HOME%\webapps` ein sogenanntes Web Archiv: `soap.war`.
2. kopieren Sie dieses Archiv ins Verzeichnis `%CATALINA_HOME%\webapps`
3. kopieren Sie die folgenden Archive ins `%CATALINA_HOME%\common\lib` Verzeichnis:
   aus JavaMail: `mail.jar`
   aus Java Activation Framework (JAF) : `activation.jar`
   aus Apache Xerces : `Xerces.jar` oder (je nach Version) die `jar`'s den Donwloads.
4. starten Sie Tomcat neu. Sie haben SOAP installiert!
5. testen Sie SOAP: `http://localhost:8080/soap`

Variante 2: mithilfe eines Verweise aus dem `server.xml` auf SOAP:

1. erweitern Sie `Startup.bat` im Verzeichnis `%CATALINE_HOME%\bin` (Tomcat).
   Ich habe folgende Zeile eingefügt (zeimlich am Anfang):
   `call %SOAP_HOME%\soap_path.bat`
2. `server.xml` auf SOAP verweisen lassen.
   Kreieren Sie einen neuen Context :
   ```
   <Context path="/soap" docBase="path-to-apache-soap/webapps/soap.war"
   debug="1" reloadable="true" />
   ```
3. SOAP Archive:
   Hier haben Sie zwei Möglichkeiten
   1) kopieren Sie das `soap.jar` Archiv ins Verzeichnis `%CATALINA_HOME%/classes` oder `%CATALINA_HOME%/lib`,
   ODER
   2) a) entzippen Sie das WAR Archiv im `SOAP_HOME\webapps` Verzeichnis ins `SOAP_HOME\webapps\soap.`
   2 b) Kreieren Sie den Context im server.xml im `%CATALINE_HOOME%\conf` Verzeichnis:
   ```
   <Context path="/soap" docBase="path-to-apache-soap/webapps/soap"
   debug="1" reloadable="true" />
   ```
   2 c) kopieren Sie die Java Archive in `%CATALINA_HOME%/classes` oder `%CATALINA_HOME%/lib`,
   oder
   `%SOAP_HOME%/webapps/soap/WEB-INF`

Die zweite Variante klingt komplexer. Sie wird von Eclipse, dem Tomcat Plugin, eingesetzt. Für erste Versuche können Sie ja mit der Variante 1 arbeiten. Im ZIP finden Sie alles für Variante 1 und 2.
Testen Sie Ihre Installation:

1. http://localhost:8080/soap
2. http://hostname:port/soap/servlet/rpcrouter

1.4. CLASSPATH Definition

Sie wollen sicher nicht nur die Standardseite von SOAP auf Ihrem Bildschirm sehen sondern *eigene* Applikationen mit SOAP entwickeln. Damit dies problemlos funktioniert, müssen Sie den CLASSPATH für `javac` bzw. `java` setzen.

Falls Sie wie oben ein `soap_path.bat` definieren wollen, hier ist meines:

```
@echo off
Rem ------------------------------------------------------------
Rem SOAP Pfad : SOAP_HOME ist bereits als Umgebungsvariable definiert
Rem ------------------------------------------------------------
set SOAP_HOME=C:\soap-2_3_1
set JAVA_MAIL=C:\javamail-1.3.1
set JAVA_ACTIVATION=C:\jaf-1.0.2
set XERCES_HOME=C:\xerces-2_5_0
set CLASSPATH=""
Rem ------------------------------------------------------------
Rem SOAP
Rem ------------------------------------------------------------
set CLASSPATH=%CLASSPATH%;%SOAP_HOME%\lib\;%SOAP_HOME%\lib\soap.jar
Rem ------------------------------------------------------------
Rem JAVA Mail : nur mail.jar ist nötig, ausser Sie wollen SOAP über SMTP
nutzen
Rem ------------------------------------------------------------
set
CLASSPATH=%CLASSPATH%;%JAVA_MAIL%\;%JAVA_MAIL%\mail.jar;%JAVA_MAIL%\lib\ima
p.jar;%JAVA_MAIL%\lib\mailapi.jar;%JAVA_MAIL%\lib\pop3.jar;%JAVA_MAIL%\lib\
smtp.jar
Rem ------------------------------------------------------------
Rem JAVA Activation
Rem ------------------------------------------------------------
set CLASSPATH=%CLASSPATH%;%JAVA_ACTIVATION%\activation.jar
Rem ------------------------------------------------------------
Rem Apache Xerces : nur Xerces.jar ist nötig, ausser Sie wollen die
Beispiele ... nutzen
Rem ------------------------------------------------------------
set
CLASSPATH=%CLASSPATH%;%XERCES_HOME%\;%XERCES_HOME%\xercesImpl.jar;%XERCES_H
OME%\xercesSamples.jar;%XERCES_HOME%\xml-
apis.jar;%XERCES_HOME%\xmlParserAPIs.jar
```

Wie Sie sehen, und weiter oben bereits lesen konnten, benötigt SOAP:

- `%SOAP_HOME%\lib\soap.jar` :
  aus http://ws.apache.org/soap/
- `%TOMCAT_HOME%\common\lib\xerces.jar` :
  aus http://xml.apache.org/xerces2-j/
- `%TOMCAT_HOME%\common\lib\mail.jar`
  aus http://java.sun.com/products/javamail/
- `%TOMCAT_HOME%\common\lib\activation.jar`
  aus http://java.sun.com/products/javabeans/glasgow/jaf.html

Da ich die einzelnen Komponenten bereits sonst benötigt habe, ist mein Classpath etwas umfangreicher (ich benutze JavaMail, das Java Activation Framework und Xerces unabhängig von Tomcat).

1.5. Deploying und Nutzen eines "Hello World" Service

Nun sollte Ihre SOAP Installation funktionsfähig sein. Falls Sie einen Fehler festgestellt haben:

- falls eine Klasse nicht gefunden wurde, liegt's am Pfad
- falls Tomcat nichts findet, haben Sie evtl. die soap, ... Archive nicht ins richtige Verzeichnis kopiert.

Sie müssen in der Lage sein, über http://localhost:8080/soap auf die SOAP Seite zuzugreifen:

Hello! Welcome *to Apache-SOAP.*

What do you want to do today?

- *Run the admin client*
- *Visit the SOAP RPC router URL for this SOAP server*

Testen Sie beide (Run ->List und Visit). Falls ein Stack Dump erscheint, dann kann evtl eines der Archive nicht gefunden werden, oder diese passen nicht zusammen (laden Sie die neusten Jar's runter, verwenden Sie nicht irgend ein `mail.jar` oder `xerces.jar` aus dem Jakarta Verzeichnis oder von sonstwo. Xerces.jar finden Sie in der neusten xerces Version nicht mehr! J2SDK verwendet eine alte, fehlerhafte Version von Xerces. Wo Sie die Archive hinkopieren (und Tomcat neu starten) sollten, steht im vorherigen Abschnitt.

Falls alles klappt, wollen wir nun SOAP endlich nutzen.
Web Services / SOAP Services können Sie auf zwei Arten allgemein bekannt machen:

1) mit einem Apache SOAP Kommandozeilen Hilfsprogramm
2) über das Admin Tool.(Web basiert)

Zuerst brauchen wir aber ein HelloWorld.java.

1.5.1. WieGehts.java – Ein einfacher Text Service

Das ist nichts neues:

```
package apachesoap;

public class WieGehtsService {
  public String wiegehts(java.lang.String name) {

    return name+" - Wie gehts Dir?";
  }
}
```

Die Methode der Klasse `WieGehtsService` heisst also `wieGehts` .

1.5.2. Klassen ind WEB-INF Verzeichnis kopieren

Nun wollen wir sehen, wie diese Methode über Apache SOAP genutzt werden kann. Dazu muss diese Klasse dem Apache SOAP Laufzeitsystem innerhalb Tomcat bekannt sein.

Auch hier haben Sie mehrere Möglichkeiten:

1. a) kreieren Sie ein Java Archiv `HelloWorld.jar` mit der Class Datei zu obigem Programm.
   b) kopieren Sie dieses Jar in das Verzeichnis `%CATALINA_HOME%\common\lib`

2. a) Ohne Jar: kopieren Sie einfach die Class Dateien ins (neu zu erstellende) Verzeichnis `%CATALINA_HOME%\webapps\soap\WEB-INF\classes\apachesoap`

1.5.3. Deployen via Kommandozeile

Apache SOAP benötigt einen "Deployment Descriptor", um auf den Dienst zugreifen zu können. Deployen kann man entweder mithilfe eines XML Deployment Decriptors, oder mithilfe des SOAP Admin Tools über den Browser. Aus das Web Tool kommen wir noch zurück.

1.5.3.1. Deployment Descriptoren

Im Deployment Descriptor wird der Dienst genauer beschrieben. Es handelt sich um eine XML Datei. Sie wird bei EJB's, ... eingesetzt. Es lohnt sich also etwas genauer hinzusehen (siehe weiter hinten). Im Moment wollen Sie sicher einfach endlich das Hello World Programm starten und nutzen. Deswegen kommen wir erst weiter hinten auf die Details zurück.

In unserem Beispiel sieht die Datei folgendermassen aus (`HelloWorldDD.xml`):

```
<?xml version="1.0"?>
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment" id="urn:wiegehts">
   <isd:provider type="java" scope="Application" methods="wiegehts">
      <isd:java class="apachesoap.WieGehtsService"/>
   </isd:provider>
   <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
</isd:service>
```

1.5.3.2. Registrieren des Service

Apache SOAP verfügt über ein Hilfsprogramm, um Dienste registrieren zu können:

```
org.apache.soap.server.ServiceManagerClient
```

Der Aufruf zum Registrieren des Hello World (als Batch im ZIP)

```
@echo off
if "%JAVA_HOME%" == "" goto homeless
call %SOAP_HOME%\soap_path.bat
%JAVA_HOME%\bin\java org.apache.soap.server.ServiceManagerClient
http://localhost:8080/soap/servlet/rpcrouter deploy WieGehts.xml
goto ende
:homeless
@echo Sie muessen JAVA_HOME als Umgebungsvariable definieren
@echo Zum Beispiel : Set JAVA_HOME=C:\J2SDK1.4.2
goto ende
:ende
pause
```

Der erste Parameter beschreibt die URL des Apache SOAP RPC Routers. Dieser muss Anfragen an den passenden Dienst weiterleiten:
http://localhost:8080/soap/servlet/rpcrouter

Der zweite Parameter gibt an, was zu passieren hat: **deploy.**

Der dritte Parameter ist unsere Deployment Beschreibung; **WieGehts.xml**

Das Ergebnis können Sie im Browser unter :
http://localhost:8080/soap/admin/index.html
anschauen. Der Service müsste nun eigenlich aufgelistet werden!

Das selbe können wir auch auf der Kommandozeile überprüfen:
```
%JAVA_HOME%\bin\java org.apache.soap.server.ServiceManagerClient
http://localhost:8080/soap/servlet/rpcrouter list
```

Sie sollten eine Ausgabe etwa wie die folgende sehen:
```
Deployed Services:
        urn:wiegehts
```

Nun löschen / undeployen wir den Dienst, zum Üben:
```
%JAVA_HOME%\bin\java org.apache.soap.server.ServiceManagerClient
http://localhost:8080/soap/servlet/rpcrouter undeploy "urn:wiegehts"
```

Falls Sie Attribute des Dienstes abfragen wollen:
```
java org.apache.soap.server.ServiceManagerClient
http://localhost:8080/soap/servlet/rpcrouter query " urn:wiegehts "
```

Wenn Sie den Dienst gerade gelöscht hatten, sollten Sie ihn zuerst wieder anmelden.
Die Ausgabe sollte etwa folgendermassen aussehen:
```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment" id="urn:wiegehts"
checkMustUnderstands="false">
  <isd:provider type="java" scope="Application" methods="wiegehts">
    <isd:java class="apachesoap.WieGehtsService" static="false"/>
  </isd:provider>
  <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
        </isd:service>
```

Wir haben unseren Deployment Descriptor wieder!

1.5.3.2.1. Authorisierung

Diese Anleitung folgt später, weil die ersten Tests schief gingen!

1.5.4. Der Client

Der Aufbau eines Clients ist ziemlich stur, gemäss Apache SOAP Dokumentation.

```
package apachesoap;

import java.io.*;
import java.net.*;
import java.util.*;
import org.apache.soap.*;
import org.apache.soap.rpc.*;

public class WieGehtsClient {

  public static void main(String[] args) throws Exception {

      String strURL=args[0];
      System.out.println("[WieGehtsClient]URL: "+strURL);
      URL url = new URL (strURL);
      //"http://localhost:8080/soap/servlet/rpcrouter");

      // Call Aufbau.
      Call call = new Call();
      call.setTargetObjectURI("urn:wiegehts");
      call.setMethodName("wiegehts");
      call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
      // Parameter
      Vector params = new Vector();
      String name="Hans";
      params.addElement(new Parameter("name", String.class, name, null));
      call.setParams (params);
      // Antwort
      Response resp = call.invoke(url, "" );
      // Antwort Analyse.
      if ( resp.generatedFault() ) {
            Fault fault = resp.getFault ();
            System.out.println("Fehlerhafter Call: ");
            System.out.println("Fault Code   = " + fault.getFaultCode());
            System.out.println("Fault String = " + fault.getFaultString());
      } else {
            Parameter result = resp.getReturnValue();
            System.out.println(result.getValue());
      }
  }
}
```

1.5.5. Starten des Clients

Damit ich mir die Ausgabe auch im Tomcat Tunnel ansehen kann, übergebe ich die URL als Parameter :

Ohne Tunnel

```
%JAVA_HOME%\bin\java -classpath .;..;%CLASSPATH%;
apachesoap.WieGehtsClient http://localhost:8080/soap/servlet/rpcrouter
```

Mit Tunnel

```
%JAVA_HOME%\bin\java -classpath .;..;%CLASSPATH%;
apachesoap.WieGehtsClient http://localhost:5555/soap/servlet/rpcrouter
```

1.5.5.1. Ausgaben

Ohne Tunnel:

```
[WieGehtsClient]URL: http://localhost:8080/soap/servlet/rpcrouter
Hans - Wie gehts Dir?
```

Mit Tunnel

Ausgabe im JVM Fenster:

```
Sie muessen zuerst den Tomcat_Tunnel starten (Tomcat_Tunnel.bat)
Press any key to continue . . .
[WieGehtsClient]URL: http://localhost:5555/soap/servlet/rpcrouter
Hans - Wie gehts Dir?
```

Ausgabe im linken Tunnel Fenster:

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: localhost:5555
Content-Type: text/xml; charset=utf-8
Content-Length: 442
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:wiegehts xmlns:ns1="urn:wiegehts" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<name xsi:type="xsd:string">Hans</name>
</ns1:wiegehts>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ausgabe im rechten Tunnel Fenster:

```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=5BB1D7C0FE51C0A95A40A7CD0D8A5E46; Path=/soap
Content-Type: text/xml; charset=utf-8
Content-Length: 481
Date: Sat, 16 Aug 2003 17:28:10 GMT
Server: Apache Coyote/1.0
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:wiegehtsResponse xmlns:ns1="urn:wiegehts" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:string">Hans - Wie gehts Dir?</return>
</ns1:wiegehtsResponse>

</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

1.6. Deployen via Web Administration Tool

Anstatt auf der Kommandozeile kann man auch mit einem Web basierten Web Tool

- neue Dienste zu definieren,
- existierende Dienste aufzulisten
- Dienste zu löschen

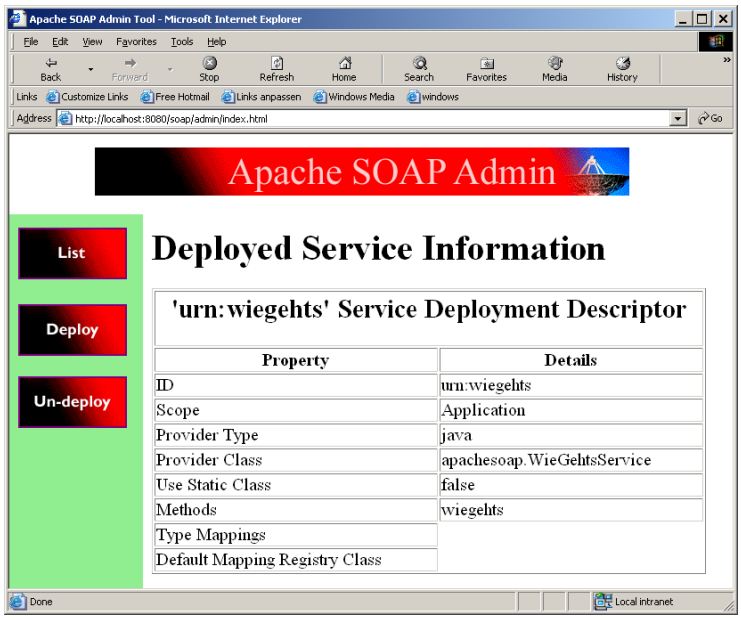
Anstelle des Deployment Descriptors werden die Daten einfach in eine Maske eingegeben.

1.6.1.1. Das SOAP Admin Tool

Unter `http://localhost:8080/soap/admin/index.html` sehen Sie folgende Seite::

Unter "List" sehen wir unseren "WieGehtsService" :

Falls wir auf den Dienst klicken, sehen wir die Details unseres Services:



Wenn Sie mutig sind (das Risiko ist gering), können Sie sich die Angaben notieren, dann den Dienst löschen und anschliessend selber deployen.

Eigentlich könnten Sie jetzt selber loslassen, hoffentlich erfolgreich.

Die nächsten Abschnitte gehen auf Themen wie:

- Deployment Descriptor
- Services mit mehreren Methoden
- Gibt es Überladen einer Methode (selber Name, unterschiedliche Parameter)
- Einbinden von Servlets
- Datentypen und Datenkonversion
- …