Web Services - Tutorial

RPG als Web Service Client

Mihael Schmidt

0.1

Juli 2007

Table of Contents

1. Allgemeines

In diesem Tutorial werden mehrere Softwarebausteine verwendet, um den gesamten Ablauf einer Web Service Anfrage von Client zu Anwendungsserver zu Client zu realisieren.

1.1 Client

Hier wird ein RPG Programm zum Webservice-konsumierenden Client mittels der HTTP API von Scott Klement. Die Serviceprogramme, die dem Softwarepaket enthalten sind, vereinfachen einem die Arbeit, wenn ein RPG Programm einen HTTP Client darstellen soll. Es ist nicht speziell auf Web Services ausgelegt, bietet aber die Möglichkeit mit Web Services zu kommunizieren.

Wahlweise kann man das XML-Dokument, welches für die Anfrage benötigt wird, im Quelltext oder in einer separaten Textdatei hinterlegen. Letzteres macht mehr Sinn und kann sehr einfach mittels MAP und TEMPLAT Serviceprogramm umgesetzt werden.

1.2 Anwendungsserver

Der Anwendungsserver ist in diesem Fall ein Apache Tomcat. Dieser nimmt die Anfrage des Clients entgegen.

Die Software Apache Axis läuft als Webanwendung auf dem Anwendungsserver, verarbeitet Web Service Anfragen und sendet ein entsprechendes XML-Dokument an den Client zurück.

2. Client

2.1 WSDL

Um mit einem Web Service zu kommunizieren, braucht man Informationen über den Aufbau von dem Web Service. Dieser wird meist in Form von WSDL dargestellt und festgehalten. WSDL ist eine Darstellung von Web Service Informationen in XML. XML ist in erster Linie nicht dafür geschaffen worden vom Menschen interpretiert zu werden, sondern von Maschinen bzw. Programmen (kann allerdings auch lesefreundlich gestaltet werden).

Der Web-Services-Explorer von WDSCi bietet die Möglichkeit mit WDSL zu arbeiten. Über diesen Explorer kann man schnell und einfach Web Services testen und SOAP Anfragen und Antworten in XML anzeigen lassen.

Gestartet wird der Web-Services-Explorer entweder über das Kontextmenü einer WSDL-Datei (Web-Services -> Mit Web-Services-Explorer testen) oder über das Menü Ausführen -> Web-Services-Explorer starten.

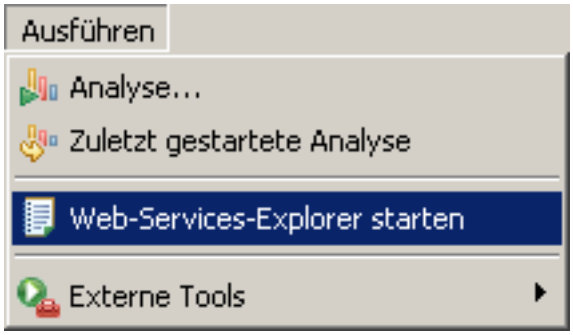

Bild 1: Web-Services-Explorer starten

Da keine WSDL-Datei vorhanden ist, wird der Web-Services-Explorer über das Menü gestartet. Der Web-Services-Explorer hat mehrere Sichten (UDDI, WSIL, WSDL). Um mit WSDL zu arbeiten, muss man einfach auf den WSDL-Knopf drücken.

Bild 2: WSDL Sicht auswählen

Nun kann man die entsprechende URL eintragen zu der gewünschten WSDL-Datei und der Explorer zeigt die in der WSDL-Datei beschriebenen Web Services an. Apache Axis bietet eine sehr komfortable Lösung, die WSDL eines Web Service zu entnehmen. An die entsprechende URL einfach der Parameter **wsdl** angehangen. Beispiel: http://localhost:8080/axis/services/Version?wsdl

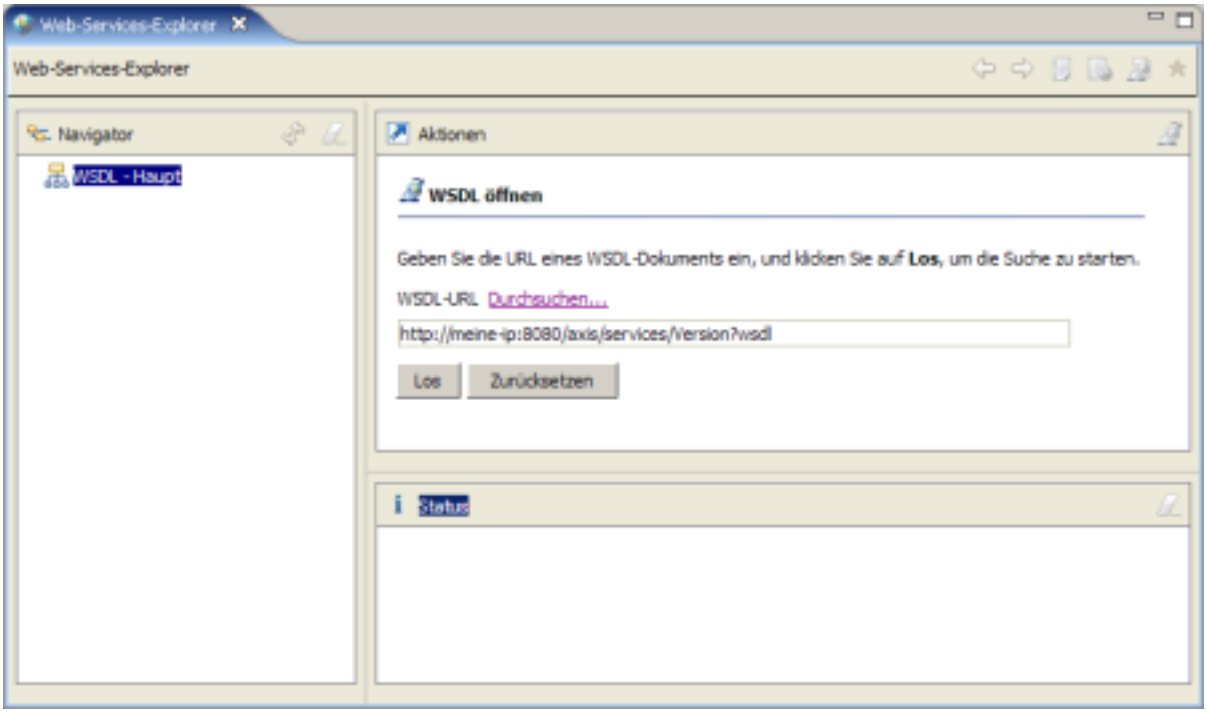

Bild 3: WSDL URL öffnen

2.2 SOAP

Nachdem man den Web Service getestet hat, kann man sich die SOAP Anfrage und Antwort Dokumente anzeigen lassen, indem man im Statusfenster auf **Quelle** klickt.

Und jetzt kommt der Teil für das RPG Programm: Die Anfragenachricht kann entweder ins RPG Programm selber oder besser in eine Textdatei kopiert werden, die dann im RPG Programm mittels TEMPLAT_SV Serviceprogramm ausgelesen werden kann. Somit bleibt der RPG Quelltext übersichtlich. In diesem Tutorial wird die SOAP Anfrage in eine Textdatei kopiert und auf die System i exportiert (per WDSC oder FTP). Abgelegt wird die Datei im IFS unter folgenden Pfad:

/QOpenSys/usr/share/docs/ws-test/version-soap.tmpl

Diese Datei ist nur notwending, wenn die SOAP Anfrage nicht in den Quelltext des Programmes geschrieben wird, sondern ausgelagert wird.

Aus der SOAP Antwort kann man erkennen, wie der XML-Knoten heisst, der den Antwortwert enthält. In dem Fall der Verions-Web Service ist das der Knoten **getVersionReturn** .

2.3 RPG

Web Services werden über das HTTP Protokoll angesprochen. Um nicht selber das HTTP Protokoll neu implementieren zu müssen, kann man auf die Serviceprogramme von Scott Klement zurückgreifen. Standardmässig wird das HTTPAPI Serviceprogramm in der Bibliothek LIBHTTP installiert. Die API wird über folgenden Code eingebunden:

```
H BNDDIR('LIBHTTP/HTTPAPI')
```

Die Prototypen werden über folgenden Code aufgenommen:

```
/copy LIBHTTP/QRPGLESRC,HTTPAPI_H
```

Eine Callback-Prozedur muss noch definiert werden, die für jeden Knotenpunkt im empfangenen XML-Dokument aufgerufen wird.

```
D Incoming        PR
D   version                50A
D   depth                  10I 0 value
D   name                 1024A   varying const
D   path                24576A   varying const
D   value               32767A   varying const
D   attrs                    *   dim(32767)
D                                const options(*varsize)
```

Die SOAP Anfragenachricht wurde zuvor in einer Textdatei gespeichert und auf die System i kopiert. Mittels der Template Engine API kann man den Inhalt der Textdatei einfach in eine Variable füllen.

```
rc = tmpl_readFile('/QOpenSys/usr/share/docs/ws-test/version-soap.tmpl' :
                   %addr(soap_msg) :
                   32767);
```

Für das Testen des Programmes ist es hilfreich das automatische Debugging der HTTP API einzuschalten und die Informationen in einer Datei (/tmp/httpdebug.txt) auszugeben.

```
http_debug(*on : '/tmp/httpdebug.txt');
```

Der eigentliche Web Service Aufruf wird über die Prozedur **http_url_post_xml** realisiert. Der letzte Parameter der Prozedur gibt den HTTP Header Eintrag SOAPAction an. Dieser darf für einen Web Service mittels Apache Axis nicht Null oder leer sein. Am besten gibt man dort die URL des Web Service an, den man gerade aufruft.

```
rc = http_url_post_xml(
              'http://' + %trim(ip) + ':8080/axis/services/Version'
```

```
                : %addr(soap_msg)
                : %len(%trimr(soap_msg))
                : *NULL
                : %paddr(Incoming)
                : %addr(version)
                : HTTP_TIMEOUT
                : HTTP_USERAGENT
                : 'text/xml'
                : 'http://' + %trim(ip) + ':8080/axis/services/Version');
```

In der Callback-Prozedur **Incoming** wird das zurückkommende XML-Dokument geparst. Anhand des SOAP Antwort Dokumentes kann man sehen, dass der Rückgabewert des Web Services in dem XML-Knoten **getVersionReturn** enthalten ist.

```
<getVersionReturn xsi:type="xsd:string">
 Apache Axis version: 1.4 Built on Apr 22, 2006 (06:55:48 PDT)
</getVersionReturn>
```

Durch eine einfache Prüfung auf den Knotennamen kann man feststellen, ob die Callback-Prozedur für den richtigen Knoten (getVersionReturn) aufgerufen wurden und kann die Version der Variable **value** entnehmen.

```
if (name = 'getVersionReturn');
 version = value;
endif;
```

3. Server

3.1 Apache Tomcat Installation

Die Installation des Apache Tomcat ist recht unproblematisch. Die Standardeinstellungen sind für dieses Beispiel vollkommen ausreichend. Also einfach eine Tomcat Version herunterladen. Dieses Tutorial wurde mit einer Version 5.x getestet. Bei den **Binary Distributions** sollte man die **Core** Distribution auswählen.

Nach der Installation kann man noch ein paar Konfigurationen vornehmen. Zum einen sollte man sich einen Management bzw. Admin Benutzer anlegen in der Datei $TOMCAT_HOME/conf/tomcat-users.xml:

```
<user username="mein_admin" password="mein_passwort" roles="admin,manager"/>
```

Nachdem man Tomcat gestartet hat, vorzugsweise über Systemsteuerung -> Verwaltung -> Dienste, kann man die Manager Webanwendung über die URL http://localhost:8080/manager/html/ aufrufen. Mit diesem Benutzer kann man sich an der Tomcat Manager Webanwendung authentifizieren.

3.2 Apache Axis Installation

In dem Softwaredownload von Apache Axis befinden sich mehrere Ordner. Einer davon hat den Namen **webapps** . In diesem Ordner befindet sich die Webanwendung Axis, welche die Web Service Anfrage des Client verarbeiten soll. Der Unterordner **axis** wird einfach in das **$TOMCAT_HOME/webapps**-Verzeichnis kopiert. Die Webanwendung benötigt noch ein paar weitere Java Bibliotheken, u. a. JavaBeans Activation Framework. JAF ist auf den Webseiten von Sun zu finden. Ist die Zusatzbibliothek installiert, kann man nun die Anwendung unter der URL http://localhost:8080/axis/ aufrufen.

Unter dem Punkt **Validation** kann man nochmals überprüfen, ob alle notwendigen Bibliotheken installiert sind. Optionale Bibliotheken werden für dieses Tutorial nicht benötigt.

Da in diesem Tutorial auf den von Apache Axis mitgelieferten Web Service **Version** zugegriffen wird, ist die Installation des Servers und Web Services hiermit abgeschlossen.

4. Anhang

4.1 SOAP Dokumente

4.1.1 SOAP Anfrage für Web Service Version

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns0:getVersion xmlns:ns0="http://axis.apache.org"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</soapenv:Body>
</soapenv:Envelope>
```

4.1.2 SOAP Antwort für Web Service Version

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:getVersionResponse xmlns:ns1="http://axis.apache.org"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<getVersionReturn xsi:type="xsd:string">Apache Axis version: 1.4 Built on
Apr 22, 2006 (06:55:48 PDT)</getVersionReturn>
</ns1:getVersionResponse>
</soapenv:Body>
</soapenv:Envelope>
```

4.2 RPG Quelltext

```
     /copy QCPYSRC,RPGHEADER
    H BNDDIR('LIBHTTP/HTTPAPI')

     *---------------------------------------------------------------------
     * PEP
     *---------------------------------------------------------------------
    D main            PR                  extpgm('WSBSP01')
    D   ip                           15A   const
    D main            PI
    D   ip                           15A   const

     *---------------------------------------------------------------------
```

```
     * Prototypen
     *---------------------------------------------------------------------
     /copy LIBHTTP/QRPGLESRC,HTTPAPI_H
     /copy QCPYSRC,TEMPLAT

    D Incoming        PR
    D   version                     50A
    D   depth                       10I 0 value
    D   name                      1024A   varying const
    D   path                     24576A   varying const
    D   value                    32767A   varying const
    D   attrs                         *   dim(32767)
    D                                     const options(*varsize)


     *---------------------------------------------------------------------
     * Variablen
     *---------------------------------------------------------------------
    D rc              s             10I 0
    D soap_msg        s          32767A
    D version         s             50A

     /free
      if (%parms <> 1);
        dsply 'Keine IP-Adresse angegeben.';
        *inlr = *on;
        return;
      endif;

      rc=tmpl_readFile('/QOpenSys/usr/share/docs/ws-test/version-soap.tmpl':
                       %addr(soap_msg) :
                       32767);
      if (rc = 0);
        http_debug(*on : '/tmp/httpdebug.txt');

        // hier muss die IP-Addresse des Servers für den Web Service
        // angepasst werden (und vielleicht auch entsprechend der
        // Rest der URL)
        rc = http_url_post_xml(
                    'http://' + %trim(ip) + ':8080/axis/services/Version'
                    : %addr(soap_msg)
                    : %len(%trimr(soap_msg))
                    : *NULL
                    : %paddr(Incoming)
                    : %addr(version)
                    : HTTP_TIMEOUT
                    : HTTP_USERAGENT
                    : 'text/xml'
                    : 'http://' + %trim(ip) + ':8080/axis/services/Version');

        if (rc = 1);
          dsply 'ok';
          dsply version;
        else;
          dsply 'nicht ok';
```

```
          http_crash();
        endif;

       else;
        dsply 'Datei konnte nicht gelesen werden.';
       endif;

       *inlr = *on;
       return;

      /end-free


      *---------------------------------------------------------------------
      * Prozeduren
      *---------------------------------------------------------------------

     /**
      * Incoming : Callback Prozedur
      */
     P Incoming        B
     D Incoming        PI
     D   version                     50A
     D   depth                       10I 0 value
     D   name                      1024A   varying const
     D   path                     24576A   varying const
     D   value                    32767A   varying const
     D   attrs                         *   dim(32767)
     D                                     const options(*varsize)
      /free
       if (name = 'getVersionReturn');
         version = value;
       endif;
      /end-free
     P                 E
```

5. Links

- Apache Software Foundation
- Apache Tomcat - Servlet Container
- Web Services - Axis
- Scott Klement's HTTP API
- Sun Javadocs 1.5 SE
- JavaBeans Activation Framework
- JavaMail
- W3C - World Wide Web Consortium
- W3 Schools

6. Glossar

Quelle für die Erklärungen ist Wikipedia.

XML

Die Extensible Markup Language, abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet.

SOAP

SOAP (ursprünglich für Simple Object Access Protocol) ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf die Dienste anderer Standards: XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht (vgl. TCP/IP-Referenzmodell) zur Übertragung der Nachrichten. Die gängigste Kombination ist SOAP über HTTP und TCP. Die Abkürzung SOAP wird jedoch offiziell seit Version 1.2 nicht mehr als Akronym gebraucht, da es erstens (subjektiv) keineswegs einfach (Simple) ist und da es zweitens nicht (nur) dem Zugriff auf Objekte (Object Access) dient.

WSDL

WSDL ist eine Metasprache, mit deren Hilfe die angebotenen Funktionen, Daten, Datentypen und Austauschprotokolle eines Web Service beschrieben werden können. Es werden im Wesentlichen die Operationen definiert, die von außen zugänglich sind, sowie die Parameter und Rückgabewerte dieser Operationen.

Web Service

Ein Web Service bzw. Webdienst ist eine Software-Anwendung, die mit einem Uniform Resource Identifier (URI) eindeutig identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können. Ein Web Service unterstützt die direkte Interaktion mit anderen Software-Agenten unter Verwendung XML-basierter Nachrichten durch den Austausch über internetbasierte Protokolle.