WebService mit Java 6 Proof of Concept / Mini_HowTo

Die Bereitstellung von WebServices mit Java ist natuerlich schon laenger moeglich, z.B. mittels JAX-WS, XFire oder Apache Soap bzw. Axis. Das Spannende ist nun (fuer mich zumindest), dass JAX-WS (SUN-Standard) in Java 6 in den Bordmitteln enthalten ist (javax.jws.* bzw javax.jws.soap.*). Es hat das komplette Java Web Services Developer Pack (JWSDPD) in Java 6 Einzug gehalten. Die Nutzung von SOAP ist Standard. (Es existieren natuerlich auch APIs fuer weitere Programmiersprachen.)

Es wird ein WebService mit zwei Methoden auf (dem Standard-) Port 8080 in Java implementiert. Nach Veroeffentlichung des WebService wird mittels des in Java 6 enthaltenen Tools "*wsimport*" (hier: JAX-WS RI 2.0_02-b08-fcs) die WSDL/XML des WebService abgefragt. Dabei werden die Stubs fuer den Client automatisch erzeugt. Der Client greift dann auf den WebService zu und nutzt jede der Methoden mindestens einmal.

Die Ausarbeitung ist HowTo-alike angelegt, da es sich um ein Vorgehen in mehreren Schritten handelt. Es kann auch problemlos (so hoffe ich) nachvollzogen werden. Das erste Hurra/Aha-Erlebnis ist nach Minuten garantiert.

Als "IDE" benutze/empfehle ich Notepad, es klappt aber auch mit Notepad++, Kate, BBEdit, TextMate, vi, Nano oder joe. Eine Kommandozeile/MS-Dos Eingabeaufforderung ist auch notwendig (eigentlich zwei davon wenn der Server mal laeuft und die ersten „DosBox“ belegt).

Serverseitig gibt es **zwei** Dateien:

- WSJavaSixProofOfConcept.java
  Der eigentliche WebService samt zugehoeriger Methoden und notwendiger Annotationen
- WsServer.java
  Der Server, der den WebService instantiiert und den Server als solches stellt

Clientseitig wird es **drei** Dateien geben:
Von diesen drei Dateien wird aber nur der eigentliche Client "selbst" geschrieben (WsClient.java). Die anderen beiden Klassen sind die notwendigen und von "wsimport" automatisch generierten Stubs (hier: WSJavaSixProofOfConceptService.java und DegiorgiWs.java).

71desa1bif@hft-stuttgart.de

Verzeichnisstruktur:

Die Verzeichnis Struktur ist wichtig, da der Einfachheit wegen auf eine Paketangabe (import package.some.more.detail) verzichtet wurde. Stattdessen enthaelt WSJavaSixProofOfConcept.java in der Annotation @WebService(...) neben der Angabe des Namens des WebService ebenso die Angabe eines (target)Namespace.

- / ("root" Verzeichnis)
- /WSJavaSixProofOfConcept.java
- WsServer.java
- /src/
  Hier liegt lediglich der (selbstgeschriebene) Client
- /src/WsClient.java
- /src/foo/
  Hier werden die von wsimport erzeugten Client-Stubs liegen (automatisch erzeugt). Je nach Paketangabe bzw. targetNamespace ist diese Struktur natuerlich anders.
- /src/foo/ WSJavaSixProofOfConceptService.java (automatisch erzeugt)
- /src/foo/ DegiorgiWs.java (automatisch erzeugt)

Server:

- *WSJavaSixProofOfConcept.java*
  Der WebService wird ueber die Annotation @WebService(..) definiert. In diesem Falle erfolgt noch die Vergabe eines Namens und die Benennung eines targetNamespaces. Letzteres kann entfallen wenn mit packages gearbeitet wird.

  Hier also:
  @WebService(name=“degiorgi_ws“, targetNamespace=“foo“)

Der Webservice soll ueber zwei Methoden verfuegen. Eine mit (benanntem/mittels der Annotation @WebParam(name="yourname")) Parameter sayHelloWorld(name), die lediglich das uebliche HelloWorld Spiel mit einem uebergebenen Namen spielt.

Die zweite Methode ist parameterlos und gibt per Zufallsgenerator einen String zurueck der in einer von 29 verfuegbaren Sprachen "PROST" bedeutet.

Die Methoden muessen mit der Annotation @WebMethod versehen sein, damit sie erreichbar/veroeffentlich werden.

Hier ist eine (Um-)Benennung der Methoden mittels @WebMethod(operationName="some_name") moeglich und auch angewandt.

Der Konstruktor der Klasse befuellt lediglich das String Array *intProsts* mit den "PROST"s.

71desa1bif@hft-stuttgart.de

- *WsServer.java*
  Diese Klasse enthaelt lediglich eine statische main-Methode, die die eigentliche WebService Klasse instantiiert und via javax.xml.ws.Endpoint als „Endpunkt“ zur Verfuegung stellt / „publish()’ed. So ist der WebService „via Browser“ verfuegbar, bis der „Endpoint.stop()“ gestoppt wird.

Einschub Annotationen -Start-

Liste der in/durch javax.jws nutzbaren Annotationen:

- **@WebService**
  Jede WebService Klasse benoetigt diese Annotation
- **@SOAPBinding**
  "Setzt den Stil der Nachrichten auf Dokument oder RPC"
  hier: @SOAPBinding(style = SOAPBinding.Style.RPC) / Bindung via SOAP Protokoll, prozedurorientiert
- **@WebMethod**
  siehe oben
- **@WebParam**
  Dient zur Beschreibung des (ggf.) Parameters
- **@WebResult**
  Beschreibt Rueckgabe(wert) einer WebService Methode
- **@OneWay**
  Kennzeichnung fuer einen asynchronen Methodenaufruf
- **@HandlerChain**
  Kann genutzt werden um Reihe(n) externer Handler festzulegen
- "viele" weitere unter *javax.xml.ws*, *javax.xml.bind* und *javax.annotation*, z.B. BindingType, RequestWrapper, ResponseWrapper, WebEndpoint, XmlRootElement, Resource, ...

fuer mehr siehe **https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html**

Einschub Annotationen -Ende-

Kompilierung des Servers:

*I***:>** *javac WsServer.java*
Kompilierung des Servers. Dies sollte eine automatische Mit-/Vorkompilierung der WebService Klasse mitbringen. Sollte es zu Problemen kommen eben haendisch vorher ein...

*I***:>** *javac WSJavaSixProofOfConcept.java*
(Kompilierung der WebService Klasse)

...absetzen.

71desa1bif@hft-stuttgart.de

Start des Servers:

:/> *java WsServer*

Server up-and-running. Zum Test der Funktionalitaet einen Webbrowser oeffnen. Je nachdem ob im WsServer.java der Endpoint angepasst wurde ist nun WSDL (XML) Beschreibung des WebService erreichbar.

"Standard" ist (Zeile 8 / WsServer.java):

Endpoint endpoint = Endpoint.publish("http://localhost:8080/services", new WSJavaSixProofOfConcept());

Somit ist der Abruf des WSDL moeglich unter:

http://localhost:8080/services?wsdl

Man erhaelt hier die vollstaendige Beschreibungsdatei.

Fuer eine "von aussen" erreichbare Bereitstellung muss statt "localhost" beispielsweise die IP-Adresse angegeben werden unter der der Rechner im Netz erreichbar ist. Also etwa:

Endpoint endpoint = Endpoint.publish("http://192.168.0.23:8080/services", new WSJavaSixProofOfConcept());

Dann kann auch von anderen Rechnern auf den WebService zugegriffen werden. (tested @ homeLan)

Erzeugung der Client-Stubs:

Auch hier gibt es wieder mehrere Moeglichkeiten, natuerlich bishin zur haendischen Erstellung. Jedoch rate selbst ich dringend von der "Handarbeits-Variante" ab und empfehle das Java 6 beiliegende "wsimport".

Die Kommandozeile fuer die automatische Generierung lautet:

:/> *wsimport -d src -keep http://localhost:8080/services?wsdl*

Nach dem Aufruf wurden automatische zwei Klassen in /src/foo erzeugt und auch schon kompiliert:

- /src/foo/DegiorgiWs.java
  Interface/Stub fuer den WebService. Die Methoden liegen sozusagen als Prototypen/Signaturen vor

- /src/foo/WSJavaSixProofOfConceptService.java
  Service Klasse fuer den Clienten um an den Endpoint/Port des Service "heranzukommen"

(Den **Hinweis** von Herrn Ullenboom / Java-Champion / Autor "Java-Insel" moechte ich nicht verschweigen. Es gibt wohl (noch) einige Probleme mit der automatischen Erstellung dieser Stub Klassen. Selbst mit google-Services. Jedoch dieses Beispiel sollte einwandfrei funktionieren.)

Kompilierung des Clients:

:/> *cd src*

:/src/> *javac WsClient.java*

Start des Clients:

:/src/> *java WsClient*

Hallo Welt! Ich bin's ... der/die/das _ein_Vorname_!!!

INT_PROST_1
INT_PROST_2
INT_PROST_3
INT_PROST_4
INT_PROST_5

In Zeile 10 des Clients wird der Parameter "_ein_Vorname_" an den WebService uebergeben.

System.out.printf("\n%s\n\n",port.bePoliteAndSayHello("_ein_Vorname_"));

Reminder: Obwohl die Methode eigentlich sayHelloWorld(...) heisst wurde der der "operationName" in der Annotation *@WebMethod(operationName="be-polite-and-say-hello")* der Klasse WSJavaSixProofOfConcept des Servers auf "be-polite-and-say-hello" gesetzt, was letztendlich in bePoliteAndSayHello(..) resultiert. Man ist hier vollkommen frei in der Wahl des WebService-Methoden-Namens, der wie man sieht unabhaenhig vom eigentlichen Methoden-Namen gewaehlt werden kann.

Das vom WebService return'te

return "Hallo Welt! Ich bin's ... der/die/das " + name + "!!!";

wird empfangen und per System.out.printf(..) letztendlich ausgegeben.

Dann wird fuenf mal die (ebenfalls via *@WebMethod(operationName="was_anderes")* / eigentlich *sayInternationalProst()*) parameterlose Methode *beInternationalAndSalute()* aufgerufen, die in zufaelligen Sprachen fuenf mal "zum Wohle!" wuenscht. Der Client beendet sich dann.

71desa1bif@hft-stuttgart.de

Quellen:

http://java.sun.com/javase/technologies/webservices.jsp
http://www.tutego.com/blog/javainsel/
http://www.galileocomputing.de/openbook/javainsel7/javainsel_18_007.htm
http://www.theserverside.de/webservice-in-java/
https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html
http://www.sherzad.com/

71desa1bif@hft-stuttgart.de

Anhang:

WSJavaSixProofOfConcept.java

```
import javax.jws.*;
import javax.jws.soap.SOAPBinding;

@WebService(name="degiorgi_ws", targetNamespace = "foo")
@SOAPBinding(style = SOAPBinding.Style.RPC)
public class WSJavaSixProofOfConcept
{
  String[] intProsts = new String[29];

  public WSJavaSixProofOfConcept() {
    intProsts[0]  = new String("Iechyd da (Welsh)");
    intProsts[1]  = new String("Chia (Vietnamese)");
    intProsts[2]  = new String("Cheers (English)");
    intProsts[3]  = new String("Sagliginiza (Turkish)");
    intProsts[4]  = new String("Choc-tee (Thai)");
    intProsts[5]  = new String("Arriba, abajo, al centro, para adentro (Spanish)");
    intProsts[6]  = new String("Salud (Spanish)");
    intProsts[7]  = new String("Zivio Ziveli (Serb)");
    intProsts[8]  = new String("Slainte (Scottish)");
    intProsts[9]  = new String("Na zdorovje (Russian)");
    intProsts[10] = new String("A sia saide (Portuguese)");
    intProsts[11] = new String("Ye zdrówko (Polish)");
    intProsts[12] = new String("Sanitas bona (Latin)");
    intProsts[13] = new String("Kong gang ul wi ha yo (Korean)");
    intProsts[14] = new String("Kampai (Japanese)");
    intProsts[15] = new String("Salute (Italian)");
    intProsts[16] = new String("Prost (German)");
    intProsts[17] = new String("L'chaim (Hebrew)");
    intProsts[18] = new String("Mazel tov (Yiddish)");
    intProsts[19] = new String("Okole maluna (Hawaiian)");
    intProsts[20] = new String("Stin ijiasas/Jamas (Greek)");
    intProsts[21] = new String("À votre santé (French)");
    intProsts[22] = new String("Je via sano (Esperanto)");
    intProsts[23] = new String("Fee sihetak (Egyptian)");
    intProsts[24] = new String("Proost (Dutch)");
    intProsts[25] = new String("Gan bei (Mandarin)");
    intProsts[26] = new String("Quapp'lah (Klingon)");
    intProsts[27] = new String("Schol (Flemish)");
    intProsts[28] = new String("pozdravljau s dnem rojdenija (Russian)");
  }

  @WebMethod(operationName="be-polite-and-say-hello")
  public String sayHelloWorld( @WebParam(name="yourname") String name )
  {
    return "Hallo Welt! Ich bin's ... der/die/das " + name + "!!!";
  }

  @WebMethod(operationName="be-international-and-salute")
  public String sayInternationalProst()
  {
    return intProsts[(int)(Math.random() * 29)];
  }
}
```


71desa1bif@hft-stuttgart.de

WsServer.java

```
import javax.swing.JOptionPane;
import javax.xml.ws.Endpoint;

public class WsServer
{
  public static void main( String[] args )
  {
    Endpoint endpoint = Endpoint.publish( "http://localhost:8080/services", new
WSJavaSixProofOfConcept());
    JOptionPane.showMessageDialog( null, "Kill Server!" );
    endpoint.stop();
  }
}
```

WsClient.java

```
import foo.DegiorgiWs;
import foo.WSJavaSixProofOfConceptService;

public class WsClient
{
  public static void main( String[] args )
  {
    DegiorgiWs port = new WSJavaSixProofOfConceptService().getDegiorgiWsPort();

    System.out.printf("\n%s\n\n",port.bePoliteAndSayHello("_ein_Vorname_"));
    System.out.printf("%s\n",port.beInternationalAndSalute());
    System.out.printf("%s\n",port.beInternationalAndSalute());
    System.out.printf("%s\n",port.beInternationalAndSalute());
    System.out.printf("%s\n",port.beInternationalAndSalute());
    System.out.printf("%s\n",port.beInternationalAndSalute());
  }
}
```

exemplarisch "services.xml" - die wsdl-Datei aus der (automatisch) die Client-Stubs generiert werden

```
<?xml version="1.0" encoding="UTF-8"?><definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="foo"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

targetNamespace="foo" name="WSJavaSixProofOfConceptService">
  <types></types>
  <message name="be-polite-and-say-hello">
    <part name="yourname" type="xsd:string"></part>
  </message>
  <message name="be-polite-and-say-helloResponse">
    <part name="return" type="xsd:string"></part>
  </message>
  <message name="be-international-and-salute"></message>

  <message name="be-international-and-saluteResponse">
    <part name="return" type="xsd:string"></part>
  </message>
  <portType name="degiorgi_ws">
    <operation name="be-polite-and-say-hello" parameterOrder="yourname">
      <input message="tns:be-polite-and-say-hello"></input>
      <output message="tns:be-polite-and-say-helloResponse"></output>
    </operation>
```


71desa1bif@hft-stuttgart.de

```
    <operation name="be-international-and-salute" parameterOrder="">

      <input message="tns:be-international-and-salute"></input>
      <output message="tns:be-international-and-saluteResponse"></output>
    </operation>
  </portType>
  <binding name="degiorgi_wsPortBinding" type="tns:degiorgi_ws">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"></soap:binding>
    <operation name="be-polite-and-say-hello">
      <soap:operation soapAction=""></soap:operation>
      <input>

        <soap:body use="literal" namespace="foo"></soap:body>
      </input>
      <output>
        <soap:body use="literal" namespace="foo"></soap:body>
      </output>
    </operation>
    <operation name="be-international-and-salute">
      <soap:operation soapAction=""></soap:operation>
      <input>

        <soap:body use="literal" namespace="foo"></soap:body>
      </input>
      <output>
        <soap:body use="literal" namespace="foo"></soap:body>
      </output>
    </operation>
  </binding>
  <service name="WSJavaSixProofOfConceptService">
    <port name="degiorgi_wsPort" binding="tns:degiorgi_wsPortBinding">

      <soap:address location="http://localhost:8080/services"></soap:address>
    </port>
  </service>
</definitions>
```


71desa1bif@hft-stuttgart.de