Web Services i Java med Axis

Denne artikel beskriver hvordan man bruger Web Services i Java både server og client side ved hjælp af Apache Axis.

Den forudsætter kendskab til Java og generel programmering men ikke til Web Service seller Axis.

Skrevet den **05. Feb 2009** af **arne_v** I kategorien **Programmering / J2EE** | ★★★☆☆

Historie:
V1.0 - 09/02/2004 - original

Web Services

Web Services er en standard eller et sæt af standarder for kald fra en applikation til en anden applikation ved hjælp af web teknologi.

Det adskiller sig fra almindelig web brug ved at der ikke er et menneske med en browser i den ene ende, men en applikation.

Først lidt forkortelser:

XML = eXtensible Markup Language = standard for markup languages

SOAP = Simple Object Access Protocol = XML som beskriver metode og argumenter ved kald (også retur værdi)

WSDL = Web Service Descriptor Language = XML som erklærer metode og argumenter for web service

UDDI = Universal Directory and Discovery Interface = online katalog over tilgængelige web services

Og så en advarsel:

Web Services er en utrolig "hot" eller "in" teknologi idag. Alle snakker om dem. Og mange vil bruge dem til hvad som helst. Meget af det er ren øregas fra folk som ikke har forstået hvad web services reelt er. Jeg synes at web services er en fremragende teknologi til mange ting f.eks. Java - .NET interaktion, men man skal ikke bruge web services uden at overveje om det nu er det bedste valg til problem stillingen. Se også afsnit til sidst.

Jeg vil i de efterfølgende eksempler udelukkende vise SOAP over HTTP og kun RPC style. Det er efter min mening mest relevant. Og svarer i funktionalitet meget til klassisk RMI (Remote Method Invocation).

Axis

Der er flere forskellige Web Services toolkit til Java.

Jeg foretrækker Apache Axis.

Et kendt alternativ er SUN JWSDP (Java Web Services Developer Pack). Men jeg kan ikke lide pakningen af det. Og alle (inkl. bl.a. IBM, BEA, Borland og JBoss) siger at Axis performer bedre.

Axis kan hentes på:
http://ws.apache.org/axis/

Installationen består af:
- download ZIP
- UNZIP
- kopier axis dir fra axis-n.n/webapps til din-servlet-engine/webapps

Så vil du kunne connecte til:
http://localhost:8080/axis/

Jeg tester altid med Tomcat, men enhver servlet engine bør kunne bruges.

Til professionel brug vil man bundle de nødvendige jar filer ind i ens egen web app, men når du kommer så langt, så er det heller ikke noget problem.

Simpel server

Hvis man kun skal bruge simple data typer så er det nemt.
Man laver bare sin klasse, kalder den .jws i stedetfor .java og kopierer den til din-servlet-engine/webapps/axis. Så er den klar til at blive kaldt. Det kan ikke være meget nemmere.

Eksempel:

Calc.jws

```
public class Calc {
   public int add(int a, int b) {
       System.out.println("Vi er blevet bedt om at lave en add");
       return (a + b);
   }
   public int mul(int a, int b) {
       System.out.println("Vi er blevet bedt om at lave en mul");
       return (a * b);
   }
}
```

WSDL filen genereres automatisk ved deployment.

håndkodet client

Man kan selv skrive koden til at kalde web servicen med.

Eksempel:

TestCalc.java

```
import javax.xml.rpc.*;
import javax.xml.namespace.*;
```

```
public class TestCalc {
    public static int mul(int a, int b) throws Exception {
        // lookup service
        String endpoint = "http://localhost:8080/axis/Calc.jws";
        ServiceFactory servfact = ServiceFactory.newInstance();
        Service serv = servfact.createService(new QName("localhost"));
        // setup method
        Call cal = serv.createCall();
        cal.setTargetEndpointAddress(endpoint);
        cal.setOperationName(new QName("mul"));
        // setup arguments
        Object[] arg = new Object[2];
        arg[0] = new Integer(a);
        arg[1] = new Integer(b);
        // call
        return ((Integer)cal.invoke(arg)).intValue();

    }
    public static void main(String[] args) throws Exception {
        System.out.println(mul(2, 3));
        System.out.println(mul(3, 4));
        System.out.println(mul(4, 5));
    }
}
```

Avanceret server
Hvis man skal bruge egne data typer så skal man til at bruge deployment descriptor.

Eksempel:

Person.java

```
public class Person {
    private String name;
    private int age;
    public Person() {
        name = "";
        age = 0;
    }
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public void setName(String name) {
        this.name = name;
    }
```

```
    public void setAge(int age) {
        this.age = age;
    }
    public String toString() {
        return (name + " " + age);
    }
}
```

Sort.java

```
import java.util.*;

public class Sort {
    public Person[] sort(Person[] persons) {
        Arrays.sort(persons, new AgeComparator());
        return persons;
    }
}

class AgeComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        return (((Person)o2).getAge() - ((Person)o1).getAge());
    }
    public boolean equals(Object obj)  {
        return false;
    }
}
```

SortService.wsdd

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service name="SortService" provider="java:RPC">
        <parameter name="className" value="Sort"/>
        <parameter name="allowedMethods" value="*"/>
        <parameter name="scope" value="application"/>
        <typeMapping xmlns:ns="http://genpakke"
                     qname="ns:Person"
                     type="java:Person"

serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"

deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        <typeMapping xmlns:ns="http://genpakke"
                     qname="ns:ArrayOfPerson"
                     type="java:Person[]"

serializer="org.apache.axis.encoding.ser.ArraySerializerFactory"
```

```
deserializer="org.apache.axis.encoding.ser.ArrayDeserializerFactory"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </service>
</deployment>
```

Deployment:
(kopier først .class filer til classes eller .jar fil til lib som for enhver web app)
java -classpath axis.jar;jaxrpc.jar;commons-logging.jar;log4j-1.2.8.jar;commons-discovery.jar;saaj.jar org.apache.axis.client.AdminClient SortService.wsdd

WSDL filen genereres automatisk ved deployment.
client med genereret stub
Man kan også generere en client stub og kalde den i stedetfor at håndkode det hele.

Eksempel:

Generering:
java -classpath axis.jar;commons-logging.jar;log4j-1.2.8.jar;commons-discovery.jar;wsdl4j.jar;jaxrpc.jar;saaj.jar org.apache.axis.wsdl.WSDL2Java http://localhost:8080/axis/services/SortService?wsdl

TestSort.java

```
// import genereret beans
import genpakke.*;
// import genereret web service stubs
import localhost.axis.services.SortService.*;

public class TestSort {
    public static void main(String[] args) throws Exception {
        // setup input
        Person[] liste1 = new Person[3];
        liste1[0] = new Person();
        liste1[0].setName("Anders");
        liste1[0].setAge(20);
        liste1[1] = new Person();
        liste1[1].setName("Børge");
        liste1[1].setAge(25);
        liste1[2] = new Person();
        liste1[2].setName("Carsten");
        liste1[2].setAge(30);
        // lookup stub
        SortService service = new SortServiceLocator();
        Sort s = service.getSortService();
        // call stub
        Person[] liste2 = s.sort(liste1);
        // check output
        for(int i = 0; i < liste2.length; i++) {
            System.out.println(liste2[i].getName());
        }
    }
```

```
}
```

Videre
Den opmærksomme læser har opdaget at der stort set ikke er XML i ovenstående eksempler (undtagelsen er WSDD filen). Og sådan er det. SOAP og WSDL er nogle indviklede formater som egner sig til at blive læst og skrevet af programmer - de er ganske uegnede til at blive læst og skrevet af mennesker.

Til seriøs arbejde med Axis kan det absolut anbefales at bruge ant.

JBuilder Enterprise Edition har indbygget wizards som bruger Axis.

Web services kan kombineres med:
- sikkerhed baseret på IP adresse
- sikkerhed baseret på BASIC authentication med username/password
- sikkerhed baseret på digital signatur og/eller encryption

Men jeg vil ikke komme ind på disse ting i denne artikel.
Web Service eller noget andet ?
Hvis både client og server er Java og det er på LAN, så brug RMI (eller sockets for den sags skyld). Det performer meget bedre end web services.

Konverteringen mellem binære objekter og XML er faktisk ret dyr.

Men hvis det WAN og der evt. er firewalls imellem client og server, så begynder web services at ligne et godt bud, fordi HTTP er nemt at få gennem alle former for netværk (forudsat at det er TCP/IP selvfølgelig).

Og hvis det kun er den ene ende som er Java, men den anden ende er f.eks. Microsoft .NET, så er Web Services en af de få muligheder for at kunne lave kald på tværs af teknologi.

Kommentar af mora d. 20. Apr 2004 | 1

Meget god introduktion af axis og webservices.

Kommentar af simonvalter d. 11. Feb 2004 | 2

Det var godt af få noget information om hvornår det skal bruges. Deployment descriptoren kan godt virke lidt skræmmende når man ikke har læst om det før.. for den sags skyld også når man har ;) Alt ialt en god artikel der giver mulighed for at få lavet den første web service uden problemer, så kan man altid gå mere i dybden med det selv.

Kommentar af angam d. 06. Jun 2005 | 3

arne_v er min guru inden for java (og megen anden programmering), og så forklarer han tingene så selv jeg næsten kan forstå dem. Gennemgangen er krydret med gode eksempler - tak for det, arne_v

Kommentar af ismar d. 08. Dec 2006 | 4

Rigtigt god artikel.
"Web Service eller noget andet?" - delen hjælper til forståelse af teknologien.

Kommentar af jadirir d. 24. Jul 2005 | 5

Kommentar af thundergod d. 25. May 2004 | 6

Kommentar af mikkelbm d. 04. Dec 2004 | 7

Ganske glimrende...

Kommentar af alister_crowley d. 15. Mar 2005 | 8