

# Java Web Services Menggunakan Apache Axis

**Ade Anom A.**

adeanom@yahoo.com  
http://www.a3rex.info

## ***Lisensi Dokumen:***

*Copyright © 2005 IlmuKomputer.Com*

*Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.*

## **Definisi Web Services**

Web Service adalah sekumpulan *application logic* beserta object-object dan method-method yang dimilikinya yang terletak di suatu server yang terhubung ke internet sehingga dapat diakses menggunakan protocol HTTP dan SOAP (*Simple Object Access Protocol*). Dalam penggunaannya, web service dapat digunakan dari hanya untuk memeriksa data user yang *login* ke sebuah *web site* ataupun untuk digunakan pada transaksi perbankan *on-line* yang rumit.

Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat *platform (platform-neutral)* dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*).

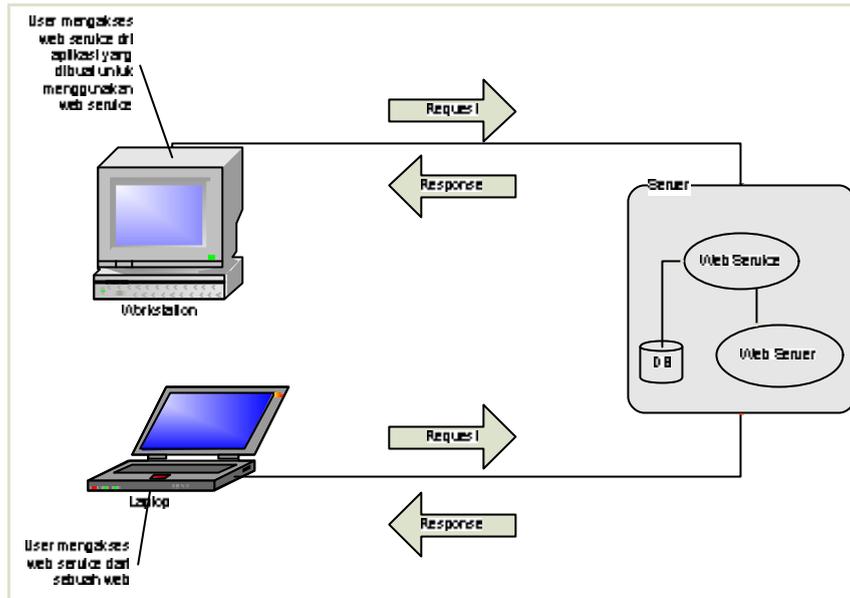
Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain daripada itu, penggunaan SOAP menjadikan method-method dari object-object yang ada dalam sebuah web service dapat di akses dari aplikasi lain seperti halnya aplikasi tersebut mengakses method lokal.

Pada gambar 1 terlihat struktur sederhana sebuah web service yan terletak pada sebuah server. User dapat menggunakan aplikasi yang ada di komputernya yang dibuat untuk mengakses web service tersebut, atau user juga dapat mengakses web site yang menjadi interface dari web service itu.

Disini bahasa pemrograman dan tampilan yang digunakan untuk membuat aplikasi tersebut bisa bermacam-macam. Misallnnya aplikasi yang mengakses tersebut dibuat menggunakan Java, PHP, maupun .Net tetap bisa mengakses web service yang ada karena data yang dipertukarkan antara client dan server berupa data XML yang standar.

## **Apache Axis**

Apache Axis adalah salah satu dari banyak implementasi Web Service Toolkit untuk Java. Project ini diawali dari tim yang mengerjakan Apache SOAP dengan tim dari SOAP4J yang bekerja sama untuk membuat *tool* yang dapat memudahkan proses pembuatan web service menggunakan Java.



Gambar 1. Web Service

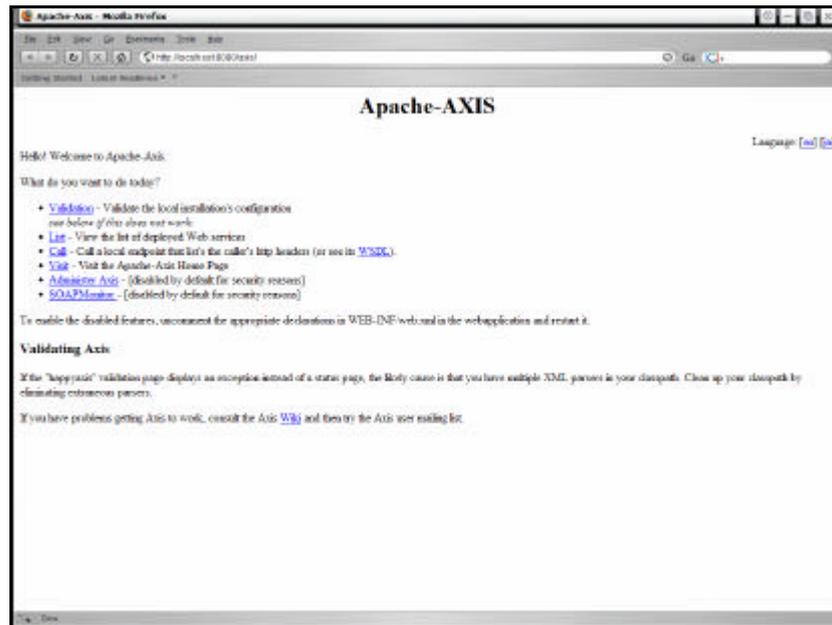
### Instalasi Axis

Untuk dapat menginstal Axis, terlebih dahulu anda harus menginstal Java SDK dan web server Java EE. Disini penulis menggunakan server Apache Tomcat 5.5.9 (<http://jakarta.apache.org/tomcat/>). Setelah itu anda dapat men-download Apache Axis dari <http://ws.apache.org/axis>, disini penulis menggunakan Axis versi 1.2.

Jika anda telah selesai menginstal Tomcat, maka langkah selanjutnya adalah memindahkan folder webapps/axis dari dalam file distribusi axis ke direktori webapps milik Tomcat.

Setelah itu, langkah yang sangat dianjurkan adalah menambahkan semua file .jar di dalam folder WEB-INF/lib milik Axis ke dalam CLASSPATH anda. Hal ini untuk memudahkan dalam proses kompilasi web service serta dalam penggunaan beberapa tools Axis nantinya. Penulis mengansumsikan bahwa anda telah melakukan hal ini dalam tulisan ini.

Berikutnya jalankan Tomcat, dan buka browser anda dan ketikkan <http://localhost:8080/axis/> di location bar. Disini penulis asumsikan bahwa anda tidak merubah konfigurasi web server anda untuk menggunakan port 8080. Bila tampil tampilan seperti pada gambar 2 maka Axis belum terinstal dengan benar, atau server belum dijalankan.



Gambar 2. Halaman awal Apache Axis

Selanjutnya klik link validate untuk memeriksa apakah semua library yang dibutuhkan oleh Axis bisa ditemukan. Ketika pertama kali menginstal Axis, penulis menemukan bahwa Axis membutuhkan sebuah file bernama activation.jar yang tidak dapat ditemukan. Jika anda menemukan masalah yang sama silahkan download file tersebut dari <http://java.sun.com/products/javabeans/glasgow/jaf.html>.

## Metode Deployment di Axis

Sebelum kita membuat sebuah web service, ada baiknya untuk mengetahui beberapa metode untuk men-deploy sebuah web service menggunakan Axis. Metode-metode tersebut adalah :

### 1. Metode JWS

Metode ini adalah metode yang paling mudah dalam men-deploy sebuah web service di Axis. Dengan metode ini yang perlu anda lakukan hanyalah membuat sebuah program Java seperti biasa dan simpan file java tersebut di bawah direktori webapps/axis di Tomcat kemudian rubah akhiran .java menjadi .jws. Sebagai contoh kita membuat sebuah file bernama "hello.java", maka rubah nama file tersebut menjadi "hello.jws". Untuk mengakses WSDL dari aplikasi tersebut, maka kita bisa membuka URL <http://localhost:8080/axis/hello.jws?wsdl>.

Walaupun metode ini sangatlah mudah, tetapi memiliki beberapa kekurangan, yaitu :

- Tidak mendukung penggunaan package di dalam aplikasi.
- Yang dibutuhkan pada waktu deployment adalah source code, padahal kemungkinan besar yang kita miliki adalah file class atau bahkan jar.
- Sulitnya untuk melakukan berbagai konfigurasi.

### 2. Metode WSDD (Web Service Deployment Descriptor)

Metode ini adalah metode standar dari Axis. Metode ini menggunakan sebuah file XML yang berisi berbagai informasi tentang web service yang akan di deploy, dan anda juga dapat mengatur method-method apa saja yang dapat diakses oleh client.

## Membuat Web Services beserta Client

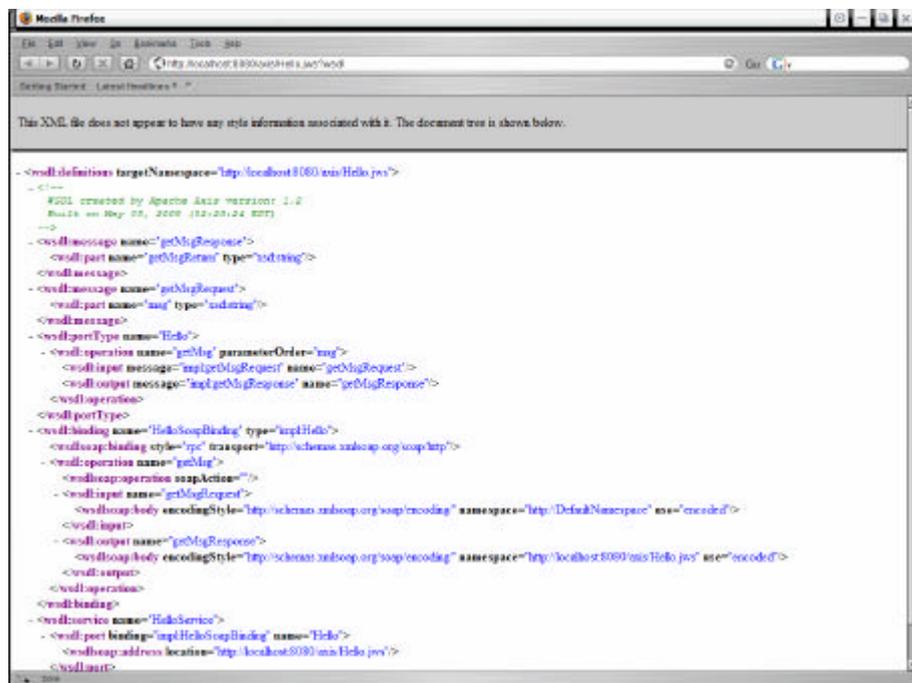
Sekarang setelah mengetahui metode-metode yang dapat digunakan untuk men-deploy sebuah web service, maka sekarang kita akan mencoba untuk membuat web service sederhana beserta client untuk mengakses web service tersebut. Kita juga akan mencoba kedua metode deployment diatas.

## 1. Metode JWS

Pertama kita akan coba untuk menggunakan metode deployment yang pertama, yaitu menggunakan JWS Endpoint.

```
1. public class Hello
2. {
3.     public String getMsg(String msg)
4.     {
5.         return("You say : " + msg);
6.     }
7. }
```

Setelah anda menulis program diatas, simpan source code tersebut di folder axis dan beri nama Hello.jws. Sekarang pastikan Tomcat anda sudah berjalan, dan buka URL <http://localhost:8080/axis/Hello.jws> di browser anda, bila ada pemberitahuan bahwa ada sebuah web service maka web service anda telah berhasil di deploy. Kemudian klik link yang ada untuk melihat WSDL dari web service anda, atau buka <http://localhost:8080/axis/Hello.jws?wsdl> di browser anda. Hasilnya kurang lebih seperti yang terlihat pada gambar 3.



Gambar 3. WSDL dari Hello.jws

Perhatikan ketika pertama kali anda membuka URL <http://localhost:8080/axis/Hello.jws?wsdl> di browser, akan sedikit terasa sedikit lama untuk membuka halaman tersebut. Hal tersebut karena file JWS yang kita panggil akan dikompilasi terlebih dahulu. Hasil kompilasi tersebut bisa dilihat di folder WEB-INF/jwsClasses di folder Axis.

Untuk membuat client yang akan mengakses web service tersebut, mari kita buat program seperti berikut ini :

```
1. import org.apache.axis.client.Call;
2. import org.apache.axis.client.Service;
3. import org.apache.axis.encoding.XMLType;
4.
5. import javax.xml.rpc.ParameterMode;
6.
7. public class MsgClient
8. {
9.     public static void main(String[] args) throws Exception
10.    {
11.        String say = "Hello World";
12.
13.        String endPoint = "http://localhost:8080/axis/Hello.jws";
14.
15.        Service service = new Service();
16.        Call call = (Call) service.createCall();
17.        call.setTargetEndpointAddress(new java.net.URL(endPoint));
18.
19.        call.setOperationName("getMsg");
20.        call.addParameter("op1", XMLType.XSD_STRING, ParameterMode.IN);
21.        call.setReturnType(XMLType.XSD_STRING);
22.
23.        String msg = (String) call.invoke( new Object[]{say});
24.
25.        System.out.println(msg);
26.    }
27. }
```

Sekarang simpan file tersebut dengan nama MsgClient.java di folder tempat anda menyimpan source code anda, misalnya di C:/javacode. Setelah itu compile file tersebut dengan perintah berikut :

```
javac MsgClient.java
```

Akhirnya kita bisa jalankan aplikasi client tersebut, untuk mengetahui apakah web service kita bisa diakses dengan benar. Bila benar maka akan keluar hasil perhitungan berikut :

```
You say : Hello World
```

Sebelumnya, penulis telah menyebutkan bahwa metode ini tidak mendukung penggunaan package didalam program. Hal ini terbukti jika kita tambahkan package pada program Hello.jws diatas, maka ketika kita buka URL <http://localhost:8080/axis/Hello.jws?wsdl> maka akan tampil pesan kesalahan (Axis Fault) yang menyatakan bahwa Axis tidak dapat menemukan file Hello.class.

Bila anda perhatikan, pada program client tersebut terlihat cukup rumit. Hal itu bisa dipermudah dengan cara menggunakan tool WSDL2Java milik Axis. Anda cukup mengeksekusi perintah berikut ini di folder dimana anda akan menyimpan file-file client, untuk membuat beberapa file class pembantu.

```
java org.apache.axis.wsdl.WSDL2Java
http://localhost:8080/axis/Hello.jws?wsdl
```

Dengan begitu, Axis akan membuat class-class yang akan membantu kita membuat aplikasi client. Class-class tersebut berada di folder localhost/axis/Hello\_jws. Sekarang kita akan coba membuat aplikasi client menggunakan class-class tersebut.

```
1. import localhost.axis.Hello_jws.*;
2. public class HelloClient
3. {
4.     public static void main(String[] args) throws Exception
5.     {
6.         // Make a service
7.         HelloService service = new HelloServiceLocator();
8.         Hello port = service.getHello();
9.
10.        System.out.println(port.getMsg("Hello World"));
11.    }
12. }
```

Setelah menyimpan file tersebut dengan nama HelloClient.java dan mengkompilasinya. Maka anda dapat menjalankan program tersebut. Anda akan menemukan bahwa output yang dihasilkan akan sama dengan program sebelumnya.

## 2. Metode WSDD

Untuk melihat contoh penggunaan metode ini, kita akan menggunakan dua buah source code yang merupakan sebuah class interface dan class yang berisi implementasi dari interface tersebut. Yang pertama yaitu Calc.java yang merupakan interface, dan yang kedua adalah CalcImpl.java. Class-class tersebut kita simpan di dalam folder Calculator di dalam direktori Axis. Setelah itu kompilasi file-file tersebut untuk mendapatkan file classnya.

```
1. package Calculator;
2.
3. public interface Calc
4. {
5.     public int add(int x, int y);
6.     public int sub(int x, int y);
7. }
```

```
1. package Calculator;
2.
3. public class Calc
4. {
5.     public int add(int x, int y)
6.     {
7.         return(x + y);
8.     }
9.
10.    public int sub(int x, int y)
11.    {
12.        return(x - y);
13.    }
14. }
```

Setelah selesai mengkompilasi, sekarang saatnya kita gunakan *tool* pertama dari Axis yang sangat membantu kita untuk membuat file WSDL, yaitu Java2WSDL. Tool tersebut akan membuat sebuah file WSDL yang sesuai dengan class/interface yang kita berikan. Untuk mengeksekusi tools tersebut, maka kita harus memberikan beberapa parameter yang akan berguna untuk membuat file WSDL. Parameter-parameter tersebut antara lain :

- Nama file WSDL (calc.wsdl)
- URL web service (<http://localhost:8080/axis/services/calculator>)
- Target namespace untuk WSDL (urn:calculator)
- Mapping Java package ke namespace (Calculator urn:calculator)
- Nama class (Calculator.Calc)

Untuk mengeksekusi tool tersebut maka ketikkan perintah seperti berikut ini :

```
java org.apache.axis.wsdl.Java2WSDL -o calc.wsdl -l
```

```
http://localhost:8080/axis/services/calculator -n urn:calculator  
-pCalculator urn:calculator Calculator.Calc
```

Jika program tool tersebut telah dieksekusi dengan benar, maka akan terdapat sebuah file calc.wsdl yang telah dibuat untuk kita gunakan nanti.

Langkah berikutnya yaitu menggunakan file WSDL tadi untuk membuat server-side wrapper code dan stub code untuk memudahkan kita membuat aplikasi client untuk mengakses web service tersebut. Untuk membuat code-code tersebut, maka kita akan menggunakan tool WSDL2Java yang telah kita gunakan sebelumnya.

File-file yang akan kita generate akan ditempatkan di package Calculator.ws untuk memisahkannya dari code asli kita.

Berbeda dengan contoh sebelumnya, dimana tool ini hanya kita berikan parameter file WSDL yang akan digunakan, sekarang kita akan memberikan beberapa parameter lain untuk membuat file-file yang kita butuhkan. Parameter-parameter tersebut antara lain :

- Base output directory (-o .)
- Scope of deployment (-d Session, nilainya bias Session, Request, dan Application)
- Aktifkan server-side generation (-s)
- Package untuk menyimpan code (Calculator.ws)
- Nama file WSDL (calc.wsdl)

Perintahnya adalah :

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p  
Calculator.ws calc.wsdl
```

Setelah mengeksekusi perintah tersebut, maka pada folder calculator/ws akan ada beberapa file .java dan dua buah file .wsdd. File-file wsdd tersebut berguna untuk proses deployment dan undeployment dari web service.

Sebelum mendeploy web service yang telah kita buat, maka terlebih dahulu kita harus merubah isi dari file CalculatorSoapBindingImpl.java yang telah kita generate tadi agar dapat terhubung ke class implementasi yang kita buat sebelumnya. Baris-baris yang berubah, tercetak tebal pada source code dibawah, yaitu pada baris 10, 14, dan 18.

```
1. /**  
2.  * CalculatorSoapBindingImpl.java  
3.  *  
4.  * This file was auto-generated from WSDL  
5.  * by the Apache Axis 1.2 May 03, 2005 (02:20:24 EDT) WSDL2Java emitter.  
6.  */  
7.  
8. package Calculator.ws;  
9.  
10. import Calculator.CalcImpl;  
11.  
12. public class CalculatorSoapBindingImpl implements Calculator.ws.Calc {  
13.     public int add(int in0, int in1) throws java.rmi.RemoteException {  
14.         return calc.add(in0, in1);  
15.     }  
16.  
17.     public int sub(int in0, int in1) throws java.rmi.RemoteException {  
18.         return calc.sub(in0, in1);  
19.     }  
20. }
```

Setelah menyimpan kembali file CalculatorSoapBindingImpl.java tersebut, maka kita siap untuk

mendeploy web service tersebut. Yang pertama harus dilakukan adalah mengkompilasi semua file java didalam package Calculator.ws.

```
javac Calculator/ws/*.java
```

Jika telah terkompilasi semuanya, maka kita harus meletakkan semua file class di direktori Calculator dan Calculator/ws ke tempat yang dapat ditemukan oleh Axis. Ada 2 cara agar file-file tersebut dapat ditemukan, yaitu :

- Meletakkan semua class di dalam folder WEB-INF/classes, dengan susunan direktori yang sama dengan sebelumnya.
- Membuat file jar yang berisi semua class-class tersebut dan meletakkannya di direktori WEB-INF/lib.

Sekarang kita gunakan cara kedua, yaitu membuat file jar untuk diletakkan di direktori WEB-INF/lib.

```
jar cvf calc.jar Calculator/*.class Calculator/ws/*.class
```

Setelah itu pindahkan file calc.jar yang sudah dibuat ke direktori WEB-INF/lib. Sampai disini kita telah selesai membuat web service yang dibutuhkan, tetapi Axis belum mengenali web service tersebut. Untuk itu kita harus menggunakan file deploy.wsdd tadi yang telah dibuat.

```
java org.apache.axis.client.AdminClient Calculator/ws/deploy.wsdd
```

Jika proses selesai dilakukan, selanjutnya restart Tomcat anda. Dan web service siap digunakan. Jika anda masuk ke URL <http://localhost:8080/axis/servlet/AxisServlet>, maka akan terlihat bahwa Web Service yang telah kita buat telah terdaftar.



Gambar 4. Calculator web service siap digunakan

Sekarang untuk mengetes web service tersebut, kita akan membuat sebuah program client. Dari class-class yang telah digenerate sebelumnya di dalam package Calculator.ws, kita dapat menggunakan file Calc.class, CalcService.class, CalcServiceLocator.class, dan CalculatorSoapBindingStub.class untuk membantu kita membuat aplikasi client.

Copykan keempat file tersebut ke folder lain dengan tetap menjaga susunan direktori yang ada. Misalnya kita copykan ke C:/javacode, maka kita copykan file-file tersebut ke c:/javacode/Calculator/ws. Setelah itu anda bisa membuat code untuk client seperti dibawah ini.

```
1. import Calculator.ws.*;
2. public class CalcTest
3. {
4.     public static void main(String[] args) throws Exception
5.     {
6.         // Make a service
7.         CalcService service = new CalcServiceLocator();
8.
9.         // Use the service to get a stub to the service
10.        Calc calc = service.getcalculator();
11.
12.        System.out.println("10 + 16 = " + calc.add(10, 16));
13.        System.out.println("50 - 25 = " + calc.sub(50, 25));
14.    }
15. }
```

Simpan file tersebut dengan nama CalcTest.java dan compile. Setelah itu, jika anda coba eksekusi file tersebut maka akan tampil output seperti :

```
10 + 16 = 26
50 - 25 = 25
```

## BIOGRAFI PENULIS



**Ade Anom A.** Lahir di Sidoarjo, 26 April 1982. Menamatkan SMU di SMU 6, Bogor pada tahun 1999. Menyelesaikan program S1 pada jurusan Informatika di Universitas Gunadarma pada tahun 2004.

Berpengalaman menjadi application developer terutama yang berhubungan dengan pemrograman web dan Java pada beberapa perusahaan TI di Jakarta. Pernah menjadi asisten dan pengajar di laboratorium Internet LePKom Universitas Gunadarma.