クライアント

WSDL から関連クラスを生成する方法

■手順

１．WSDL から 必要なファイルを作成
２．作成したファイルを利用して、サービスを利用する

■１．WSDL から 必要なファイルを作成

サーバのサービスを起動した状態で以下を実行

```
wsimport -d client http://localhost:8080/test?wsdl
```

または、

```
wsimport -keep -d client http://localhost:8080/test?wsdl
```

-keep オプションでソースを残せる

■２．作成したファイルを利用して、サービスを利用する

```
javac -cp client;. TestClient.java
```

実行は

```
java -cp client;. TestClient
```

動的にエンドポイントアドレスを変更する方法

アノテーションで
とするとできるらしいが、うまくいかない。

その場合は、
Java7 以降は、
でも OK.。
その後で、

```
((BindingProvider)port).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,"http://localhost:8080/test"
```

でアドレスを変更（たぶん、しなくてもすでに変わっていると思うけど・・・）

JAX-WS (WSDL) のバージョンが古い場合

サーバが提供する WSDL のバージョンが古い場合、 wsimport を実行すると

```
use="encoded" がサポートされていない
```

等のメッセージが表示されることがある。

その場合は、 Apache Axis 1（1.4 Final) を使う。

■準備

http://archive.apache.org/dist/ws/axis/1_4/
から axis 1.4 をダウンロードする。

■ WSDL から Java の生成

lib 以下の jar にクラスパスを通して、 org.apache.axis.wsdl.WSDL2Java を実行する。

```
    java       org.apache.axis.wsdl.WSDL2   Java       http://langrid.org/service_manager/wsdl/kyoto1
.langrid:KyotoUJServer?wsdl
```

バッチを作るなら以下の様な感じ

```
@echo off
setlocal

set base=%~dp0

set CLASSPATH=%base%/lib
set CLASSPATH=%CLASSPATH%;%base%/lib/axis-ant.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/axis.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/commons-discovery-0.2.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/commons-logging-1.0.4.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/jaxrpc.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/log4j-1.2.8.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/saaj.jar
set CLASSPATH=%CLASSPATH%;%base%/lib/wsdl4j-1.5.1.jar

java org.apache.axis.wsdl.WSDL2Java %*
```

■呼び出し

流れとしては

1.Locator を生成
2.Locator#get サービス (new URL(エンドポイント)) でサービスを取得

サービスの関数呼び出し

サーバ

大きく以下の二通りの方法がある。
・プログラム単体で Web サーバの役割も担う方法
・Tomcat を利用する方法

プログラムでサーバを立てる方法

http://itpro.nikkeibp.co.jp/article/COLUMN/20080801/311972/
http://unageanu.hatenablog.com/entry/20090722/1248257955

■手順

１．呼び出されるクラス（WebService）とメソッド（WebMethod) を作成
２．WSDL に必要なファイルを作成 ※この手順を省くと実行時に自動で作成するみたい
３．WebService のランチャーを作成

■１．呼び出されるクラス（WebService）とメソッド（WebMethod) を作成

```
javac -d server TestService.java
```

■２．WSDL に必要なファイルを作成

```
wsgen -d server -cp server test.TestService
```

または、

```
wsgen -keep -d server -cp server test.TestService
```

-keep オプションでソースファイルを残せる

■３．WebService のランチャーを作成

```
javac -d server -cp server ServiceLuncher.java
```

サービスを起動するには

```
java -cp server test.ServiceLuncher
```

Tomcat を利用する方法

http://kenichiro22.hatenablog.com/entry/20101015/1287130478
http://tdottjpn.blogspot.jp/2008/06/tomcatwebservice.html
http://www.melange.co.jp/blog/?p=1491

■ Tomcat に必要なライブラリの準備

Tomcat で SOAP のサービスを提供するにはライブラリが必要。
いくつか種類があるが、ここでは Metro を使う。
他にも CXF や Axis2 などがある。
比較は以下の URL 参照。
http://www.ibm.com/developerworks/jp/java/library/j-jws14/

Metro のダウンロード
https://metro.java.net/

Metro を Tomcat に組み込む

```
catalina.sh stop
ant -Dtomcat.home=<TOMCAT_INSTALL_DIR> -f <METRO_INSTALL_DIR>/metro-on-tomcat.xml install
catalina.sh start
```

TOMCAT_HOME/shared/lib 以下にファイルがコピーされる。

- webservices-tools.jar
- webservices-extra.jar
- webservices-extra-api.jar

・webservices-rt.jar

もし、tomcat に組み込みたくない場合は各アプリの WEB-INF/lib の中に上記の jar ファイルをコピーしてもよい。

■ファイル構成

```
└── WEB-INF
    │   web.xml
    │   sun-jaxws.xml
    │
    ├── classes
    │   └── test
    │           TestService.class
    │
    └── lib（このファイルは前述の通り、必要に応じて）
            webservices-extra-api.jar
            webservices-extra.jar
            webservices-rt.jar
            webservices-tools.jar
```

■ web.xml

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>JAX-WS Sample Application</display-name>
  <listener>
    <listener-class>com.sun.xml.ws.transport.http.servlet.WSServletContextListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>jax-ws Test</servlet-name>
    <servlet-class>com.sun.xml.ws.transport.http.servlet.WSServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jax-ws Test</servlet-name>
    <url-pattern>*.ws</url-pattern>
  </servlet-mapping>
</web-app>
```

■ sun-jaxws.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints version="2.0" xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime">
  <endpoint implementation="test.TestService" name="TestService" url-pattern="/test.ws"/>
</endpoints>
```

■ TestService.java

■ wsdl の確認

tomcat にアクセスして、wsdl が取得できるか確認する。

```
http://localhost:8080/soap/test.ws
```

URL の soap の部分は tomcat の設定次第。環境によって変わる。

WS-Security

http://www.ibm.com/developerworks/jp/java/library/j-jws13.html