

Here is a quick and easy step guide for creating a Web Service (WS) stub and a WS ties on Tomcat (J2EE web container) without reading lots of documentation.

We use standard JAX-PRC API with Java Web Service Development Pack (JWSDP) to deploy the WS WAR file directly on Tomcat web container, (compare with Apache Axis where you had to deploy your web service inside the Axis instead of deploying it directly into Tomcat web container),

Create a Web service on Tomcat

Assume that you have installed the JWSDP from “<http://java.sun.com/webservices/>” and defined the Windows environment variable :

```
%JWSDP_HOME%  
%TOMCAT_HOME%
```

All commands in this guide assumes to be run from the WS project directory

Description of the project directory structure for this web service project

Build	Contain all compiled files
Config	Web services configuration
Dist	WS WAR file for deploy on Tomcat
Doc	Generated javadoc for WS client stub
Src	Source files for the WS test client and WS server service
Web	Web directory structure for the WS WAR file

1. Write a Web service endpoint interface

The endpoint interface declares the methods that a WS client can invoke on the WS server.

```
package server;  
  
import java.rmi.Remote;  
import java.rmi.RemoteException;  
import java.lang.*;  
  
public interface wstest extends Remote {  
  
    public String getEchoString(String strv) throws RemoteException;  
  
    public int getEchoInt(int intv) throws RemoteException;  
  
    public int add (int i, int j) throws RemoteException;  
  
    public double divide (int i, int j) throws RemoteException;  
  
}
```

Save this interface to file src/server/wstest.java

2. Write a Web service implementation class

This is the implementation class that implements the endpoint interface

```
package server;  
import java.lang.*;  
  
public class wstestImpl implements wstest  
{  
    public String getEchoString(String strv) {  
        return strv;  
    }  
  
    public int getEchoInt(int intv) {
```

```

        return intv;
    }

    public int add (int i, int j) {
        return i+j;
    }

    public double divide (int i, int j) {
        return (double)i/(double)j;
    }

    public static void main (String[] args){
        wstestImpl wstest = new wstestImpl();
        System.out.println(" 5+2 = "+wstest.add(5,2));
        System.out.println(" 5/2 = "+wstest.divide(5,2));
    }
}

```

Save this class to file src/server/wstestImpl.java

3. Write a web service configuration file

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- configuration file for JWS DP wsdeploy tool -->

<webServices xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd" version="1.0"
    targetNamespaceBase="http://peteryeung.homeip.net/targetNamespaceBase/operation/message"
    typeNamespaceBase="http://peteryeung.homeip.net/typeNamespaceBase/type">

    <!-- the endpoint name becomes the service name in the WSDL -->
    <endpoint name="wstestService"
        interface="server.wstest"
        implementation="server.wstestImpl"/>

    <endpointMapping endpointName="wstestService" urlPattern="/wstestService"/>

</webServices>

```

Save this xml configuration to file config/jaxrpc-ri.xml

Create a web.xml file without content

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app>
</web-app>

```

Save this xml to file config/web.xml

4. Install WS class library files into Tomcat environment

```

@echo copy all JAX-RPC libs to tomcat
xcopy %JWS DP_HOME%\jaxrpc\lib %TOMCAT_HOME%\shared\lib\*.jar

@echo copy all SAAJ libs to tomcat
xcopy %JWS DP_HOME%\saa j\lib %TOMCAT_HOME%\shared\lib\*.jar

@echo copy all SAAJ libs to tomcat
xcopy %JWS DP_HOME%\jwsdp-shared\lib %TOMCAT_HOME%\shared\lib\*.jar

@echo copy all parser libs to tomcat
xcopy %JWS DP_HOME%\jaxp\lib\endorsed %TOMCAT_HOME%\shared\lib\*.jar

```

5. Compile my web service source

```

md build
javac -d build src/server/*.java

```

6. Compose a web directory

```
md web\WEB-INF
md web\WEB-INF\classes

xcopy/S build\server\* web\WEB-INF\classes\server\

copy config\jaxrpc-ri.xml web\WEB-INF
copy config\web.xml web\WEB-INF
```

7. Create a WAR file from the composed web directory

```
jar -cvf build/myWEB-INF.war -C web\ .
```

8. Create a Webs service WAR file

```
md dist
md build\tmpdir

%JWSDP_HOME%\jaxrpc\bin\wsdeploy.bat -verbose -o dist/myWStest.war build/myWEB-INF.war
```

9. Deploy the WS WAR file to Tomcat

Lazy deployment by copying the `myWStest.war` WAR file into Tomcat webapps directory

```
copy dist\myWStest.war %TOMCAT_HOME%\webapps
```

Start the Tomcat

Test the WSDL file on Tomcat by following command

```
start explorer http://localhost:8080/myWStest/wstestService?WSDL
```

Create a Web service client

10. Define where WSDL is located

Define where the URL for the WSDL file is located by specifying it in the `config/wscompile_config.xml` file. Define also the package name for the WS client stub code generation in the xml file.

The URL in this example is: "<http://localhost:8080/myWStest/wstestService?WSDL>"

The package namespace is: "clientStub"

```
<?xml version="1.0"?>

<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <!-- WSDL URL and generated package name -->
  <wsdl location="http://localhost:8080/myWStest/wstestService?WSDL"
packageName="clientStub"></wsdl>
</configuration>
```

11. Generate client stub code from the WSDL location

Following command line reads the `config/wscompile_config.xml` that created in previous step.

```
%JWSDP_HOME%\jaxrpc\bin\wscompile -d build -gen:client -keep config/wscompile_config.xml
```

If you are sitting behind a http proxy e.g. www-proxy.ericsson.se:8080 then you can add httpproxy:www-proxy.ericsson.se:8080 as an extra argument:

```
%JWSDP_HOME%\jaxrpc\bin\wscompile -httpproxy:www-proxy.ericsson.se:8080 -d build -gen:client -keep config/wscompile_config.xml
```

12. Generate Javadoc for the client stub code

```
javadoc -d doc\clientStub build\clientStub\*.java
```

To view the Javadoc type:

```
Explorer.exe doc\clientStub\index.html
```

13. Create a client stub Jar lib API

```
md lib
jar -cvf lib/wsstub.jar -C build clientStub/
```

The new wsstub.jar lib will be placed inside the lib directory and be used by the WS client test code

14. Write WS client test code

Write a WS client test code to test the generated stub API

```
package client;
import clientStub.*;

public class wstestclient {

    public static void main (String[] args) {

        WstestService wstestService = new WstestService_Impl();
        Wstest wstest=null;

        try {
            wstest = wstestService.getWstestPort();
        }
        catch(javax.xml.rpc.ServiceException e)
        {
            e.printStackTrace(System.err);
        }

        // Invoke the webservices
        try {

            System.out.println(" wstest.getEchoInt(1) -> "+wstest.getEchoInt(1));

            System.out.println(" wstest.getEchoString(\"test\") -> "+wstest.getEchoString("test"));

            System.out.println(" 5+2 = "+wstest.add(5,2));
            System.out.println(" 5/2 = "+wstest.divide(5,2));

        }
        catch (java.rmi.RemoteException e)
        {
            e.printStackTrace(System.err);
        }
    }
}
```

Write the client java code to file `src/client/wstestclient.java`

15. Compile the WS client test code

In order to compile the WS client test code “wstestclient.java” the required JWSDP class library should be added into the system class path.

```
javac -g -d build -classpath
lib/wsstub.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-api.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-spi.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-impl.jar;
src/client/*.java
```

Note! Execute the command in the command prompt as one line

16. Run the WS test client code

```
java -cp lib/wsstub.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-api.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-spi.jar;
%JWSDP_HOME%\jaxrpc\lib\jaxrpc-impl.jar;
%JWSDP_HOME%\jwsdp-shared\lib\jax-qname.jar;
%JWSDP_HOME%\jwsdp-shared\lib\activation.jar;
%JWSDP_HOME%\jwsdp-shared\lib\mail.jar;
%JWSDP_HOME%\saaj\lib\saaj-impl.jar;
%JWSDP_HOME%\saaj\lib\saaj-api.jar;
%JWSDP_HOME%\jaxp\lib\endorsed\xercesImpl.jar;
%JWSDP_HOME%\jaxp\lib\endorsed\dom.jar;
build
client.wstestclient
```

Note! Execute the command in the command prompt as one line

Following text will be displayed in the command prompt screen:

```
wstest.getEchoInt(1) -> 1
wstest.getEchoString("test") -> test
5+2 = 7
5/2 = 2.5
```

Finally

If you don't want to do copy paste from this document you can download the zip file here with all sources code and bat [files here](#).

This example is tested on JWSDP 1.5

*By Peter Yeung
Technical product manager*