WEB SERVICES EXAMPLE USING JWSDP 2 AND TOMCAT
*(from **Code Diaries**)*

Aim
The aim of this tutorial is to write a simple 'Hello World' web service and client using the JWSDP2 framework on Tomcat. Leave any questions or problems as comments and I will endeavour to answer them. See Web Services Example Using Axis2 and Tomcat for an alternative to JWSDP.

Assumptions
This article assumes that you have the JWSDP 2 installed and configured on Tomcat. Please see Setting up JWSDP2 on Tomcat for more details. For purposes of this example Tomcat is running on port 8080 and the project directory is JWSDP2TomcatHelloWorld.

Versions used in this example

Sofware/Component	Image
Windows XP SP2	N/A
JWSDP 2	jwsdp-2_0-windows-i586.exe
tomcat 6	apache-tomcat-6.0.18.zip
JDK 1.5.0	N/A

Links to these files can be found here

Although this is a quick 'helloworld' tutorial we will not sacrifice neatness for speed - so all the source files, class files etc will be kept in separate directories and as clean as possible. You will thank me for this later.

Create this directory structure in your working directory- similar to something you would get from eclipse. The structure used in this example is given below. The **JWSDP2TomcatHelloWorld** will be referred to as your project or working directory. Create this structure.

JWSDP2TomcatHelloWorld
_|_client_bin
_|_client_src
_|__|_config.xml
_|___JWSDP2TomcatHelloWorldClient.java
_|_web_bin
_|__|_WEB-INF
_|____|_classes
_|__.__.__|_jaxrpc-ri.xml
_|__.__.__|_web.xml
_|_web_src
_.__|_HelloWorldServiceImplementation.java
_.__|_HelloWorldServiceInterfacejava

The JWSDP2 in installed in the **C:\Sun\jwsdp-2.0** directory for this tutorial.

COMPILING THE WEBSERVICE CODE

1. We don't really need an interface for this example, but I have added one for completeness, because we will use these exact same files for the JWSDP example. In this tutorial we will expose the *HelloCallEcho* function

```
1. package helloworld;
2.
3. import java.rmi.RemoteException;
4.
5. public interface HelloWorldServiceInterface extends java.rmi.Remote{
6.     public String HelloCallEcho(String hellostring)throws java.rmi.RemoteException;
7.     public int HelloCallAdd(int helloNumber1, int helloNumber2)throws java.rmi.RemoteException;
8. }
```

   Save this as HellowWorldServiceInterface.java in your **web_src** directory.

   Next create the implementaion

```
1. package helloworld;
2.
3. public class HelloWorldServiceImplementation implements HelloWorldServiceInterface{
4.
5.     public String HelloCallEcho(String hellostring) {
6.         return "Web service echoing back - "+hellostring;
7.     }
8.
9.     public int HelloCallAdd(int helloNumber1, int helloNumber2) {
10.        return helloNumber1 + helloNumber2;
11.    }
12. }
```

   Save this as HelloWorldServiceImplementation.java under in your **web_src** directory

2. Open a command prompt and cd into your project directory (JWSDP2TomcateHelloWorld in this example).

3. Compile the code

```
..workspace\JWSDP2TomcatHelloWorld>javac -d web_bin\WEB-INF\classes
web_src\helloworld\*.java
```

4. Your **web_bin** directory should have some class file in there.
5. Create the jaxrpc-ri.xml file in the **web_bin\META-INF** directory.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <webServices xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
3.    version="1.0"
4.    targetNamespaceBase="http://mysite.org/wsdl"
5.    typeNamespaceBase="http://mysite.org/types"
6.    urlPatternBase="/Hello">
7.
8.    <endpoint name="JWSDPTomHelloWorld"
9.       displayName="Hello World from JWSDP 2"
10.       description="This is a Hello World Echo service"
11.       wsdl="/WEB-INF/JWSDPTomHelloWorld.wsdl"
12.       interface="helloworld.HelloWorldServiceInterface"
13.       implementation="helloworld.HelloWorldServiceImplementation"/>
14.
15.    <endpointMapping
16.       endpointName="JWSDPTomHelloWorld"
17.       urlPattern="/hello"/>
18.
19. </webServices>
```

6. Create the web.xml file in the same **WEB-INF** directory.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app>
3.    <display-name>Hello Service</display-name>
4.    <description>
5.       Returns a String saying Hello
6.    </description>
7.
8.    <session-config>
9.       <session-timeout>60</session-timeout>
10.    </session-config>
11.
12. </web-app>
```

7. Cd into web_bin directory and jar it all up using,

```
..workspace\JWSDP2TomcatHelloWorld\web_bin>jar -cvf HelloWorldServices.jar *
```

8. Now use the wsdeploy tool to create the war file that will be deployed to tomcat

```
..workspace\JWSDP2TomcatHelloWorld\web_bin>c:\Sun\jwsdp-
2.0\jaxrpc\bin\wsdeploy.bat -o JWSDP2TomcatHelloWorld.war
HelloWorldServices.jar
```

DEPLOYING THE SERVICE

1. Copy the **jwsdp2tomcathelloworld.war** file to your tomcat webapps directory. After a few seconds tomcat would have extracted the directory.

2. Open a browser and navigate to 'http://127.0.0.1:8080/JWSDP2TomcatHelloWorld/hello.' You should see a table with some links. Click on the WSLD link to have a look at the WSDL.

GENERATING THE STUB CODE - REQUIRED FOR THE CLIENT.

1. Create the config.xml file in your client_src directory. Wscompile will use this to find the wsdl to create the stubs,

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <configuration
3.    xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
4.    <wsdl
5.       location="http://127.0.0.1:8080/JWSDP2TomcatHelloWorld/hello?WSDL"
6.       packageName="helloworld">
7.    </wsdl>
8. </configuration>
```

2. Open a prompt and cd to your working directory.

3. Generate the client stub by running wscompile.bat. This should be in your jwsdp install directory under 'jaxrpc/bin.' We use the -s to specify that all the stubs be installed in the client_src directory.

```
workspace\JWSDP2TomcatHelloWorld>C:\Sun\jwsdp-
2.0/jaxrpc/bin/wscompile.bat -gen -keep -d client_bin -s client_src
client_src/config.xml
```

COMPILING THE CLIENT

1. Write the actual client code,

```
1. public class JWSDP2TomcatHelloWorldClient
2. {
3.    public static void main (String args[])
4.       throws Exception
5.    {
6.       JWSDPTomHelloWorld_Impl service =  new JWSDPTomHelloWorld_Impl();
7.       HelloWorldServiceInterface stub = service.getHelloWorldServiceInterfacePort();
8.       String response = stub.helloCallEcho("client says hello JWSDP");
9.       System.out.println ( response );
10.    }
11. }
```

Save this as JWSDP2TomcatHelloWorldClient.java in the client_src directory.

2. Open a prompt to the project directory and compile it using extdirs to point to the jar files in your environment.

```
..workspace\JWSDP2TomcatHelloWorld>javac -cp client_src -d client_bin -extdirs
C:\Sun\jwsdp-2.0\saaj\lib;C:\Sun\jwsdp-2.0\jaxrpc\lib;C:\Sun\jwsdp-2.0\jwsdp-
shared\lib client_src\JWSDP2TomcatHelloWorldClient.java
```

This should place all the files into the 'client_bin' directory

RUNNING THE APPLICATION

1. CD into client_bin and run the client using the command below.

```
..workspace\JWSDP2TomcatHelloWorld\client_bin>java -
Djava.ext.dirs=C:\Sun\jwsdp-2.0\saaj\lib;C:\Sun\jwsdp-
2.0\jaxrpc\lib;C:\Sun\jwsdp-2.0\fastinfoset\lib;C:\Sun\jwsdp-2.0\jwsdp-
shared\lib;C:\Sun\jwsdp-2.0\jaxb\lib;C:\Sun\jwsdp-2.0\sjsxp\lib
JWSDP2TomcatHelloWorldClient
```

You should see a printout saying 'hello back' from the web service.

2. If you are getting errors have a look a the launcher.server.log in the tomcat log directory. You might see ClassNotFound Errors. Make sure the required jars are copied into your tomcat lib directory. MultipartMime errors need the mail.jar. See Setting up JWSDP2 with Tomcat 6 for more details.

LOOKING AT THE SOAP MESSAGES USING TCPMON

The way the monitor works is like a proxy. Your client connects to the monitor and the monitor connects to the web-service. Remember that Tomcat is running of port 8080.

1. Go back and edit the HelloworldServiceInterface_Stub.java. This is one of the files generated by wscompile and should be under the client_src\helloworld directory. Around line 40. Change the port to 8081 and save.

original	public HelloWorldServiceInterface_Stub(HandlerChain handlerChain) {
	super(handlerChain);
	_setProperty(ENDPOINT_ADDRESS_PROPERTY,
	"http://127.0.0.1:8080/JWSDP2TomcatHelloWorld/hello");

modified	public HelloWorldServiceInterface_Stub(HandlerChain handlerChain) {
	super(handlerChain);
	_setProperty(ENDPOINT_ADDRESS_PROPERTY,
	"http://127.0.0.1:8081/JWSDP2TomcatHelloWorld/hello");

2. Compile the client code as explained previously.
3. Open up another seprate command prompt and start up tcpmon.

```
...\anywhere>java -cp D:\downloads\axis-bin-1_4\axis-1_4\lib\axis.jar
org.apache.axis.utils.tcpmon
```

   Please refer to the final section in the Setting up Axis2 on Tomcat for more details on this step and why we are using the axis.jar core from Axis 1. This should startup tcpmon.

4. In the tcpmon gui, in the **listen port** type in 8081 and press add.

5. Go to the 8081 tab at the top.

6. Now open up a command prompt and run the recompiled client as explained in the previous section- You should see SOAP messages sent back and forth.