

Quote of the Day Tutorial

**Practical created by Oliver Malham,
NeSC Training Team**

**Document by
Oliver Malham and Guy Warner,
NeSC Training Team**

Contents

Introduction.....	2
Environment.....	2
Service.....	2
Write interface and implementation .java files	3
Compile the code	3
Create the "WEB-INF\jaxrpc-ri.xml" fil.....	4
Create the "WEB-INF\web.xml" file	4
Package it all up into a portable WAR file	5
Make the WAR file deployable	5
Deploy your web service	5
Client.....	6
Generate the glue components	6
Write the client.....	6
Run the Client	8
What Now?	8
Appendix – Installing Your own Machine.....	8

Introduction

This practical will guide you through the creation and deployment of a Java web service and client. The following software packages will be used:

- Java 2 Enterprise Edition (J2EE) 1.4 (or above)
- Java Web Services Developer Pack (JWSDP) 1.1
- Tomcat-JWSDP

These packages are already installed on the classroom machines. The classroom machines have already been set up with the directory structure needed for the tutorial, under your “My Documents” directory. Please delete all the files in this directory except the file called “classpath.bat” (and any sub-directories), these files are now out of date.

When this document says to run a command, this command must be run at the command prompt. Commands are formatted like **this** in this document. To create java files the use of jedit is recommended. A shortcut to jedit is on the desktop of the classroom machines.

If you are using your own machine then please follow through the appendix on installing your own machine before starting the tutorial. It is assumed in this document that you are running under a Microsoft Windows version.

Environment

Developing and using Java web services requires a relatively complex set of environment variables. For occasional developers, it is convenient to set these variables in a batch file that you run every time you open a command-prompt window. Alternatively, these settings can be permanently added to the command-prompt shortcut. Within your tutorial directory is a file called “classpath.bat”. Open a command prompt (to be used for all future commands within this tutorial). Type

classpath.bat

This has set up the environmental variables (for example: PATH and CLASSPATH) needed during the tutorial. If you get errors during compilation or deployment, the first thing to check is usually that your CLASSPATH variable is correctly configured in the command prompt that you are currently using.

Service

The first part of this tutorial will show you how to construct and deploy a web service using Sun’s Java Web Services Developer Pack (JWSDP) and the JWSDP version of the Tomcat web application container.

Write interface and implementation .java files

From within your “QOTD Tutorial” folder, change firstly to the folder “service\src\qotd”. All the files you create in this section of the tutorial should be placed in this folder. The first task is to create a Java interface class for the service. Create a new (text) file, called "Qotd.java". Start by specifying the package and importing the classes that are needed:

```
package qotd; // This is just to keep the code neat

// This class enables remote method invocation
import java.rmi.Remote;

// This is for handling remote exceptions
import java.rmi.RemoteException;
```

Next define the interface itself. For this simple example service, this is as follows:

```
public interface Qotd extends Remote {
    public String getQuote() throws RemoteException;
}
```

That's it for the interface. Save this file. The next stage is to write the class that implements the functionality that has just been described in the interface. Create another new file, this time called "Qotd_Impl.java". The contents of this file should be:

```
package qotd; // Same package as the interface

public class Qotd_Impl implements Qotd {
    private static String quote = "Your quote";

    public String getQuote() {
        return this.quote;
    }
}
```

Save this file. That is all of the programming needed for this (admittedly trivial) web service.

Compile the code

The various files that have been created so far now have to be bundled up into a WAR file, which is a JAR file with a specific folder structure so that web application containers can deploy the software automatically.

Compile the two .java files by running this command in the "src" folder:

```
javac -d ../war\WEB-INF\classes\ qotd*.java
```

The "-d" flag and path tells the compiler to place the compiled ".class" files in the appropriate directory of the war file structure. A successful compile will produce no messages.

Create the "WEB-INF\jaxrpc-ri.xml" file

The jaxrpc-ri.xml file tells the wsdeploy utility how to generate the various pieces of code and configuration information in order for the portable WAR to be converted and deployed on a specific web application container.

Create a new text file called jaxrpc-ri.xml under your WEB-INF folder. Enter the following XML:

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

<webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://nesc.ed.ac.uk/wsdl"
  typeNamespaceBase="http://nesc.ed.ac.uk/types"
  urlPatternBase="/ws">
  <endpoint
    name="Qotd"
    displayName="Quote Of The Day Service"
    description="A simple web service"
    interface="qotd.Qotd"
    implementation="qotd.Qotd_Impl"/>
  <endpointMapping
    endpointName="Qotd"
    urlPattern="/qotd"/>
</webServices>
```

This file achieves three principle tasks:

1. It tells wsdeploy about the namespaces that this service will use.
2. It defines a web service endpoint in terms of the Java interface that the service exposes.
3. An endpoint mapping is defined which tells the application container how to translate the endpoint into a URL.

Create the "WEB-INF\web.xml" file

The next stage is to create the web.xml file which is used by the application container to setup the web service.

Create a new text file under your WEB-INF folder, called "web.xml", and enter the following XML:

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

<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

<web-app>
    <display-name>Quote of the day service</display-name>
    <description>Returns something philosophical</description>
</web-app>
```

This simply tells the application container what to call the service and the description of what it does. Both of these are in a user-readable form that is used primarily for the application container management interface.

Package it all up into a portable WAR file

From within your “war” folder run the following command:

```
jar cvf temp.war *
```

This packages up all of the files and directory structure into a portable WAR file. The file created may be migrated to other web services environments if necessary. This tool is very similar in functionality to the linux command line utility tar.

Make the WAR file deployable

Still within the “war” folder run the “wsdeploy” command:

```
wsdeploy -o Qotd.war temp.war
```

This tool converts the portable WAR file into an implementation specific WAR file by generating all of the tie classes and a WSDL file from your Java interface class. If errors are encountered running this tool the “temp.war” should be rebuilt after fixing the errors and before rerunning wsdeploy.

Deploy your web service

The final step is to copy the deployable WAR file, Qotd.war, to the “webapps” folder of the JWSDP Tomcat installation. On the classroom PC’s this is “C:\tomcat-jwsdp-1.4\webapps” This will automatically deploy the web service contained in the WAR file into the tomcat server. You will have to restart the server in order for these changes to take effect. This is done by the following

```
%CATALINA_HOME%\bin\shutdown.bat
```

(Lots of error messages normally just mean the server wasn’t already running). Then

```
%CATALINA_HOME%\bin\startup.bat
```

By going to "http://localhost:8080/Qotd/qotd", the services WSDL file may be inspected.

Client

A web service is not much use without a client with which it can interact. The second part of this tutorial shows how to construct a simple command-line client for the service that has just written and deployed.

Generate the glue components

First, create a file called config.xml, in the "client" folder and add the following lines to it:

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

<configuration xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <wsdl location=" http://localhost:8080/Qotd/qotd?wsdl "
  packageName="qotd" />
</configuration>
```

This is the configuration file for the wscompile utility that will generate all of the glue code. The file tells wscompile which WSDL file to use (location can also refer to a local WSDL file), and what package name we would like the glue code placed in.

Next, we run the wscompile utility itself:

```
wscompile -gen:client -keep -d . config.xml
```

This gets wscompile to generate the client code (-gen:client), keep the generated source code so it can be examined later (-keep), and work in the current directory (-d .), using the configuration described in config.xml. As with previous commands no output is generated on successfully running.

This will create a "qotd" directory, which will contain a number of .class and .java files that are used to talk to the web service.

Write the client

Create a new file called qotdClient.java (still in the "client" folder).

First, the various RPC libraries and glue code that have just been generated must be imported:

```
// Standard libraries
import javax.xml.rpc.Service;
import javax.xml.rpc.Stub;

// Our glue code
import qotd.Qotd_Service_Impl;
import qotd.Qotd_PortType;
```

Next, create the client class and give it two methods:

```
public class qotdClient {
    public static String getQuote() throws Exception
    {

    }

    public static void main(String[] args)
    {

    }
}
```

Now, fill in the `getQuote` function so that it calls a remote function via web services. Add the following to the `getQuote` function:

```
Qotd_Service_Impl service = new Qotd_Service_Impl();
Qotd_PortType port = service.getQotdPort();
```

The `Qotd_Impl` class can be seen as the "iniator" of the web service, and `QotdPortType` can be seen as an "connector" to the web service. Now call the remote "getQuote" method, as defined in the WSDL, and return it to the caller:

```
String quote = port.getQuote();
return (quote);
```

And that is all that the `getQuote` client method needs. Now, create some code in the "main" method so that it calls the `getQuote` method:

```
try{
    String quote = getQuote();
    System.out.println(quote);
}
catch (Exception e) {e.printStackTrace();}
```

That is all of the code that is required. Compile the code with:

```
javac qotdClient.java
```

Run the Client

Run the client with the command:

```
java qotdClient
```

What Now?

If you have time at the end of the tutorial try accessing a WSDL from <http://xmethods.net> and write a client for it

Appendix – Installing Your own Machine

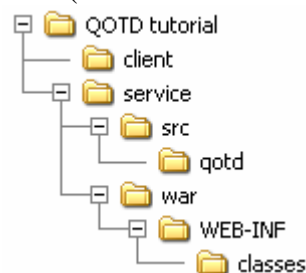
If you are setting up your own machine to use for the tutorial, please follow the below instructions:

1) You will need to install the below software, in the below order (the order is important):

- Java 2 Enterprise Edition (J2EE) 1.4 or above (<http://java.sun.com/j2ee/index.jsp>)
- Tomcat-JWS DP (http://java.sun.com/web_services/containers)
- Java Web Services Developer Pack (JWS DP) 1.1 (http://java.sun.com/web_services/jwsdp/index.jsp)

These software packages should already be installed on your tutorial machines. If not, or you are installing all the software on your own machine, please ensure that you install the JWS DP package **last**, and that when asked to select an application container you make sure that the Tomcat-JWS DP you have just installed is in the list and selected. If Tomcat-JWS DP is not in the list use the browse button and browse to the top folder of Tomcat-JWS DP.

2) Create the below folder structure (note that these names are case sensitive):



3) In the tutorial folder created a file called “classpath.bat”. This should be run at the command prompt you intend to run the tutorial exercises from. The contents of this

file should be as shown below. The values of WS_HOME and JWSDP_HOME should be adjusted according to where you installed J2EE and Tomcat – JWSDP respectively. Please note that some of the lines from the file have been split over several lines below so as to aid legibility (marked by the ↵ symbol). These lines must be typed as one single line.

```
@echo off
set WS_HOME=\Sun
set JWSDP_HOME=\tomcat-jwsdp-1.4

set CATALINA_HOME=%JWSDP_HOME%
set PATH=%WS_HOME%\Appserver\bin;%WS_HOME%\Appserver\jdk\jre;↵
%WS_HOME%\Appserver\jdk\bin;%JWSDP_HOME%\apache-ant;↵
%JWSDP_HOME%\apache-ant\bin;↵
%JWSDP_HOME%\jwsdp-shared\bin;↵
%JWSDP_HOME%\bin;↵
%JWSDP_HOME%\jaxr\bin;↵
%PATH%

set JAVA_HOME=%WS_HOME%\Appserver\jdk
set J2EE_HOME=%WS_HOME%\Appserver

set JAXRPC_HOME=%JWSDP_HOME%\jaxrpc
set JAXR_HOME=%JWSDP_HOME%\jaxr
set ANT_HOME=%JWSDP_HOME%\apache-ant

set JAVA_XML_HOME=%JWSDP_HOME%
set JAXM_HOME=%J2EE_HOME%\imq\demo
set JAXM_LIB=%J2EE_HOME%\imq\lib
set JAXP_HOME=%JWSDP_HOME%\jaxp
set JAXP_LIB=%JWSDP_HOME%\jaxp\lib
set JAXRPC_LIB=%JWSDP_HOME%\jaxrpc\lib
set XERCES_JAR=%JAXP_HOME%\lib\endorsed\xercesImpl.jar
set JSSE_HOME=%J2EE_HOME%\jdk\jre\lib
set SAAJ_HOME=%JWSDP_HOME%\saaaj

set CLASSPATH=.;%WS_HOME%\Appserver\jdk\lib\tools.jar;↵
%WS_HOME%\Appserver\jdk\lib\rt.jar;↵
%JAVA_HOME%\jre\javaws\javaws.jar;↵
%J2EE_HOME%\lib\activation.jar;↵
%JWSDP_HOME%\jwsdp-shared\lib\mail.jar;↵
%JAXRPC_HOME%;↵
%JAXR_HOME%;↵
%JAXRPC_HOME%\lib\jaxrpc-api.jar;↵
%JAXRPC_HOME%\lib\jaxrpc-impl.jar;↵
%JAXRPC_HOME%\lib\jaxrpc-spi.jar;↵
%SAAJ_HOME%\lib\saaaj-api.jar;↵
%SAAJ_HOME%\lib\saaaj-impl.jar;↵
```

```
%JAXP_HOME%;  
%JAXP_LIB%\jaxp-api.jar;  
%JAXP_LIB%\endorsed\xalan.jar;  
%JAXP_LIB%\endorsed\sax.jar;  
%JAXP_LIB%\endorsed\dom.jar;  
%J2EE_HOME%\lib\j2ee.jar;  
%XERCES_JAR%
```