

Re: Developing Web Services in JAVA using Apache Soap on a tomcat server

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2004-09/1084.html>

From: William Brogden (wbrogden_at_bga.com)

Date: 09/11/04

Date: Fri, 10 Sep 2004 23:42:12 -0500

On 10 Sep 2004 12:52:37 -0700, John Cox <coxjohn@hotmail.com> wrote:

- > *Trying to work with Apache Soap and java server but not understand*
- > *what it means to Compile the Java class and put it somewhere in your*
- > *web server's classpath for tomcat server. Here are the directions*
- > *that I am following but keep getting a 404 error when trying to*
- > *contact the service through a perl client.*
- >
- > *Installing Apache SOAP*
- >
- > *Apache SOAP can be used as both a client and provider of SOAP web*
- > *services. A server-side installation of Apache SOAP involves placing*
- > *some .jar files in your classpath. You will need a separate web server*
- > *that supports Servlets and Java Server Pages, such as Apache's Tomcat*
- > *(<http://jakarta.apache.org/tomcat/>).*
- >
- > *The Apache SOAP homepage, <http://xml.apache.org/soap/index.html>, has*
- > *links to both source-only and precompiled distributions of the*
- > *toolkit. Installing the precompiled binary distribution is as simple*
- > *as downloading a Zip archive and extracting it into a directory.*
- >
- > *On the client, three .jar files from the distribution (soap.jar,*
- > *mail.jar, and activation.jar) must be present in your classpath. Also*
- > *present must be any Java API for XML Parsing (JAXP) aware XML parser,*
- > *such as Xerces Version 1.4 (<http://xml.apache.org/xerces-j/>).*
- >
- > *Assuming that you installed Apache SOAP .jar files in the C:\book\soap*
- > *directory, set your SOAP_LIB environment variable to C:\book\soap\lib.*
- > *Adding the .jar files to your classpath then entails:*
- >
- > *set CLASSPATH = %CLASSPATH%;%SOAP_LIB%\soap.jar*
- > *set CLASSPATH = %CLASSPATH%;%SOAP_LIB%\mail.jar*
- > *set CLASSPATH = %CLASSPATH%;%SOAP_LIB%\activation.jar*

Note that Tomcat does not pay any attention to the System environment CLASSPATH – see the class loader how-to in the docs installed with Tomcat.

>
> *Or, in the Unix Bourne shell (/bin/sh):*
>
> *CLASSPATH = \$CLASSPATH;\$SOAP_LIB/soap.jar*
> *CLASSPATH = \$CLASSPATH;\$SOAP_LIB/mail.jar*
> *CLASSPATH = \$CLASSPATH;\$SOAP_LIB/activation.jar*
>
> *The exact steps for a server installation will depend on which web*
> *application server you are using, but the process is essentially the*
> *same. The first step is to ensure the same three .jar files are*
> *located in your application server's classpath.*
>
> *If your application server supports the use of web application*
> *archives (WAR files), simply use the soap.war file that ships with*
> *Apache SOAP. Apache Tomcat supports this. The Apache SOAP*
> *documentation includes detailed installation instructions for Tomcat*
> *and a number of other environments.*
>
> *If you intend to use the Bean Scripting Framework (BSF) to make*
> *script-based web services, you need to ensure that bsf.jar and js.jar*
> *(a BSF JavaScript implementation) are also in the web application*
> *server's classpath.*
>
> *The vast majority of problems encountered by new Apache SOAP users are*
> *related to incorrect classpaths. If you encounter problems writing web*
> *services with Apache SOAP, be sure to start your debugging by checking*
> *your classpath!*
> *The Hello Server*
>
> *We're going to do the same things we did in Perl: create the code,*
> *deploy the service, and use the service. Example 3-12 shows the Java*
> *code for the Hello class.*
>
> *Example 3-12: Hello.java*
>
> *package samples;*
> *public class Hello {*
> *public String sayHello(String name) {*
> *return "Hello " + name;*
> *}*
> *}*
>
> *Compile the Java class and put it somewhere in your web server's*
> *classpath.*
>
> *Deployment descriptor for samples.Hello*
>
> *<dd:service xmlns:dd="http://xml.apache.org/xml-soap/deployment"*
> *id="urn:Example1">*
> *<dd:provider type="java"*
> *scope="Application"*

```
> methods="sayHello">
> <dd:java class="samples.Hello"
> static="false" />
> </dd:provider>
> <dd:faultListener>
> org.apache.soap.server.DOMFaultListener
> </dd:faultListener>
> <dd:mappings />
> </dd:service>
>
>
> Example 3–16. hw_jclient.pl, the Perl client for the Java Hello World
> server
>
> #!/usr/bin/perl -w
> # hw_jclient.pl – java Hello client
> use SOAP::Lite;
> my $name = shift;
> print "\n\nCalling the SOAP Server to say hello\n\n";
> print "The SOAP Server says: ";
> print SOAP::Lite
> -> uri('urn:Example1')
> -> proxy('http://localhost:8080/soap/servlet/rpcrouter James')
> -> sayHello($name)
> -> result . "\n\n";
>
>
> output I get from running the perl client
>
> C:\Documents and Settings\Administrator\Desktop>test.pl James
>
> Calling the SOAP Server to say hello
>
> The SOAP Server says: 404 /soap/servlet/rpcrouter%20James at
> C:\Documents and Se
> ttings\Administrator\Desktop\test.pl line 7
```

The "/servlet/" usage sounds like it expects the "invoker" servlet to be running – Tomcat used to come with the "invoker" turned on but this ceased to be true partway through Tomcat 4 releases. This is a source of endless confusion – check your default web.xml and the soap web.xml for mention of "invoker"

--
Using Opera's revolutionary e-mail client: <http://www.opera.com/m2/>