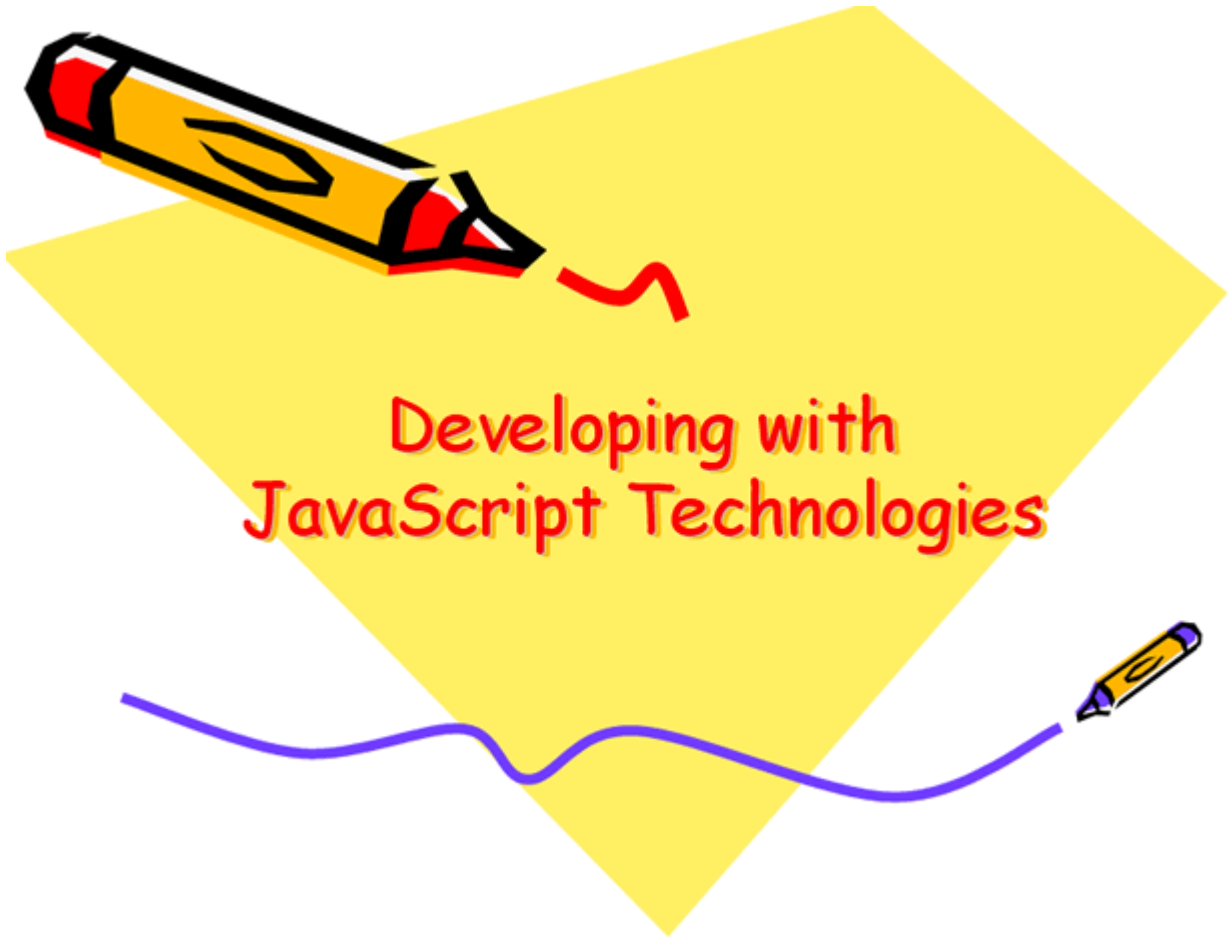
transentia

Exercise 6



WebServices in Tomcat

Introduction

Tomcat is the Apache project's Servlet engine and serves as the reference implementation for Servlet and Java Server pages technologies. It is also developing as a fine platform for WebServices.

In this exercise you will learn how to write a simple WebService—written in JavaScript and running within the 'Rhino' JavaScript interpreter—that is accessible using SOAP and see how to deploy it into Tomcat.

This exercise contains a lot of 'pre-canned' scripts…you should take a bit of time to examine each one before you run it so that you can get a better idea of what is happening "behind the scenes."

Setting Up

This exercise involves both server-side and client-side sotware.

Installing the Software

Since WebServices are based on SOAP which is itself based on XML, there are a (surprising) number of prerequisite pieces of software that need to be installed to establish a development framework.

In this session, you will require the following:

- Sun JDK 1.3.1
- Tomcat 3.2.1
- Soap 2.2
- Bean Scripting Framework 2.2
- Java Activation Framework 1.0.1
- Javamail 1.2
- Rhino 1.5R2
- Xerces 1.4.3

You have been given these on your CD-ROM.

Configuration

A bit of housekeeping is needed before development can begin.

Install the JDK

You should install the Sun Java Development Kit (this should be provided to you on your course CD-ROM) and note where it is installed.

Create and Populate the SOAP Home Directory

A (mostly) pre-installed and configured version of Tomcat and all associated software has been provided for you on your CD-ROM. There remains a small bit of work, however.

Open a new Command Prompt window and issue the following command to copy the directory Z:\Exercises\6 WebServices in Tomcat\SOAP to C:\SOAP:

```
C:\> xcopy /ieq "Z:\Exercises\6 WebServices in Tomcat\SOAP" C:\SOAP
```

Notes:

- This assumes that your CD-ROM device is Z:, you should use your real device letter instead
- Do not simply drag&drop from the CD-ROM; if you do, you will end up with a read-only directory structure and this will complicate things later on

You should end up with the following directory structure on your workstation's hard disk:

C:\SOAP
- bin
- bsf-2_2
- jaf-1.0.1
- jakarta-tomcat-3.2.1
- javamail-1.2
- rhino1_5R2
- soap-2_2
- xerces-1_4_3

Edit Utility Script

You have been provided with some utility scripts within C:\SOAP\bin. Using a text editor such as notepad, edit the supplied **setpath.bat**[1] file as follows: (for *path-to-jdk* use the place where you earlier installed the Java Development Kit):

```
@echo off
set JAVA_HOME=path-to-jdk
set PATH=%PATH%;%JAVA_HOME%\bin
set TOMCAT_HOME=C:\SOAP\jakarta-tomcat-3.2.1
```

[1] If you are using notepad, be aware that notepad will append a '.txt' extension to anything it saves. To stop this behaviour you should enclose the full desired filename in double quote marks when you type it into notepad's "Save As…" dialog box.

This will make your life a lot easier later on.

A Simple Calculator WebService

This example will show some of the versatility of Tomcat and SOAP.

Although services are traditionally implemented in languages such as C, C++ and (increasingly) Java, this need not always be the case. In this example you will develop a simple JavaScript service (which will be hosted by the Rhino interpreter).

Create Suitable Directories for the Exercise

Execute the following sequence of commands:

```
> mkdir C:\SOAP\Calc
> cd /d C:\SOAP\Calc
```

Create the Deployment Descriptor/Service Definition

The deployment descriptor tells Tomcat about a WebService and it's properties. In this example, the deployment descriptor *contains* the WebService source code…written in JavaScript, of course.

Create the following file. Call it **C:\SOAP\Calc\DeploymentDescriptor.xml**:

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

<isd:service
    xmlns:isd="http://xml.apache.org/xml-soap/deployment"
    id="urn:calculator">
  <isd:provider type="script" scope="Application"
                methods="plus">
    <isd:script language="javascript">
      function plus(x, y) {
        return x + y;
      }
    </isd:script>
  </isd:provider>
</isd:service>
```

Notice how the deployment descriptor contains the JavaScript code of the service within itself!

Create the Java Client Application

Create the following in the file **C:\SOAP\Calc\Client.java**:

```
import java.io.*;
import java.net.*;
import java.util.*;
```

```
import java.lang.*;
import org.apache.soap.*;
import org.apache.soap.rpc.*;

public class Client
  {
  public static void main (String [] args) throws Exception
    {
    Integer
      parX = new Integer (1),
      parY = new Integer (41);

    Call call = new Call ();
    call.setTargetObjectURI ("urn:calculator");
    call.setMethodName ("plus");
    call.setEncodingStyleURI (Constants.NS_URI_SOAP_ENC);
    Vector params = new Vector ();
    params.addElement (new Parameter ("x", int.class, parX, null));
    params.addElement (new Parameter ("y", int.class, parY, null));
    call.setParams (params);

    Response resp = call.invoke (new URL (args [0]), "");

    if (resp.generatedFault ())
      {
      Fault fault = resp.getFault ();
      System.out.println("Fault: ");
      System.out.println(" Fault Code = " +
        fault.getFaultCode ());
      System.out.println(" Fault String = " +
        fault.getFaultString ());
      }
    else
      {
      Parameter result = resp.getReturnValue ();
      System.out.println (parX + " + " + parY +
                          " = " + result.getValue ());
      }
    }
  }
```

You should compile this file with the following commands:

```
C:\SOAP\Calc> ..\bin\setpath.bat
C:\SOAP\Calc> ..\bin\setclasspath.bat
C:\SOAP\Calc> javac Client.java
```

Execution Time

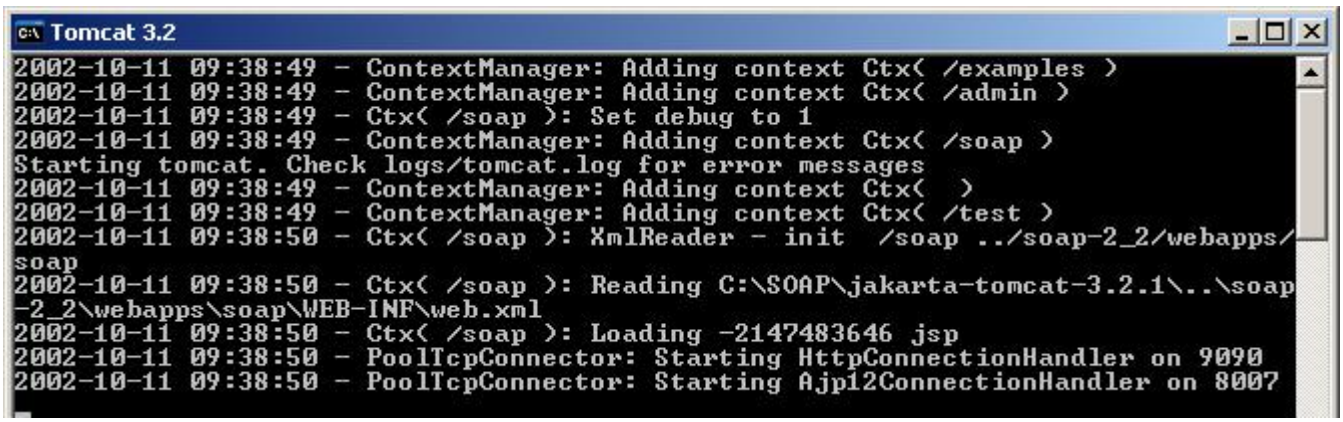
Time for all that hard work to pay off!

Deploy Tomcat

Ensure that you have Tomcat running:

```
C:\SOAP\Calc> %TOMCAT_HOME%\bin\startup.bat
```

You should see a new Command Prompt window open for the Tomcat server process. Ensure that the window contains lines similar to what is shown in the following screendump. In particular, you should ensure that the soap context is deployed correctly:

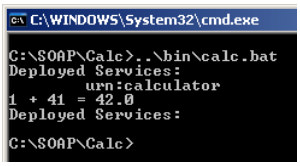


Execute the Client

You can now execute the calc.bat file that has been provided for you:

```
C:\SOAP\Calc> ..\bin\calc.bat
```

You should see the result of the calculation (as performed by the JavaScript server) printed on your screen, along with a number of deployment messages, as is shown in the following screendump:



Congratulations! You have successfully written a simple JavaScript server and its associated Java client.

Simple optional exercise

Modify the JavaScript server and the associated Java client so that they can also compute the difference between two integers…

For Your Interest: Making Things Visible

One of the oft-quoted major benefits of SOAP (the transmission protocol underlying WebServices) is that, being based on XML, all interactions proceed by the exchange of simple textual messages and thus is very easy to debug.

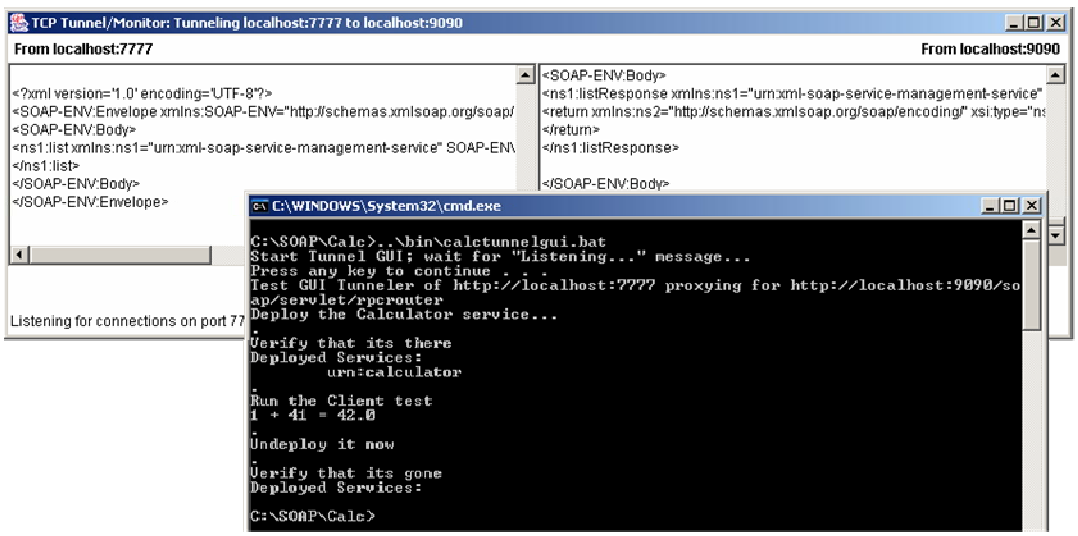
SOAP provides a simple tool called the TcpTunnelGui to show this feature.

Using the TunnelGui Tool

Execute this script:

```
C:\SOAP\Calc> ..\bin\calctunnelgui.bat
```

You will see results similar to the following:



You should find it quite easy to follow the interactions that take place between client and service (to make this even easier, you may want to insert *pause* statements at strategic points in the invoking batch file). This simple tool can be of great help during the development of a complex system.