WebServices with BEA WebLogic

WebServices in a Commercial Product

Exercise

Pizzas

WebServices with BEA WebLogic

WebServices in a Commercial Product

Introduction

BEA WebLogic is BEA's flagship application server product. It is a fully-featured product and provides support for the whole J2EE specification as well as extra facilities for clustering, transaction handling, etc.

From release 6.1, WebLogic also supports SOAP and WSDL.

In this exercise you will take a brief look at WebLogic and also see how SOAP makes it possible for various technologies to interoperate.

Setting Up

There are a number of housekeeping tasks you need to do before you can get started.

Installing the Software

In this session, you will require the following:

- MS SOAP Toolkit 2.0 sp2
- Microsoft Windows Script 5.6
- BEA WebLogic Server 6.1

Each of the above are distributed as windows installers.

Install MS SOAP Toolkit

Installing this toolkit is another straightforward task using the installer supplied on your CD in **Resources\SoapToolkit20.exe**.

Install MS Windows Script

If you are using Windows 2000 or above, this facility is part of the standard Operating System, otherwise simply double-click the supplied installer **Resources\scr56en.exe**.

Install BEA Weblogic 6.1

Simply double-click the supplied installer **Resources\WebLogic 6.1\weblogic610_win.exe**. You should accept all the default settings offered to you by the installer. In particular, this exercise assumes that WebLogic is installed under the directory **C:\bea**.

Pizzas!

WebLogic exposes a large proportion of its infrastructure for the WebServices developer to use. In particular, WebLogic exposes Enterprise Java Beans[†] as WebServices.

In this section, you will create a Stateless Session EJB (a close analogue to a Servlet) and expose it as a WebService.

Populate Development Directories

Open a Command Prompt window and then execute the following command to set up a new directory for you to 'play' in (in this sheet the DOS command prompt is shown as **>**, *you should not type this directly*. This command assumes that your CD-ROM is accessible via drive Z:):

```
> xcopy /e/i "Z:\Exercises\1 Pizzas\framework" C:\pizzas
```

This directory contains a number of empty files for you to edit, alongside a number of 'boilerplate' files that you can examine but that should not be changed.

You should do the majority of your work within the **C:\pizzas** directory.

WebLogic Server 6.1 bundles the Open Source 'Ant' tool that you can use for building EJB projects. Because this is a non-trivial framework to configure (given the available time), we have provided many of the necessary files for you…you will have copied them from your CD-ROM earlier.

Create the EJB

An EJB consists of two separate Java interfaces and a single Java class file.

- Home interface
  Provides lifecycle support of the bean (creation, removal, etc.).
- Remote interface
  Provides a means of advertising the business logic methods implemented by the Enterprise Java Bean.
- Bean class
  The implementation class of the EJB.

[†] An Enterprise Java Bean may be thought of as a "more evolved servlet" that lives within an environment that provides higher functionality support services than those provided to a simple servlet.

Create the Home Interface

In the C:\pizzas\pizzaEJB directory, create the following in the file **PizzaHome.java**:

```
package pizzas.pizzaEJB;

public interface PizzaHome
  extends javax.ejb.EJBHome
  {
  Pizza create()
    throws javax.ejb.CreateException, java.rmi.RemoteException;
  }
```

Create the Remote Interface

The Remote Interface should be created in the file **Pizza.java** in the same directory as the file above:

```
package pizzas.pizzaEJB;

import java.rmi.RemoteException;

public interface Pizza
  extends javax.ejb.EJBObject
  {
  public float getStandardPizza ()
    throws RemoteException;
  public float getCustomPizza (String [] toppings)
    throws RemoteException;
  }
```

Create the Bean Class

The Bean class file must be in the same directory as the other Java source files and should have the name **PizzaBean.java**. The content should be as follows:

```
package pizzas.pizzaEJB;

public class PizzaBean
  implements javax.ejb.SessionBean
  {
  private final float
    STANDARD_COST = 7.50f,
    TOPPING_COST = 0.5f;

  public void ejbActivate () {  }
  public void ejbRemove () { }
  public void ejbPassivate () { }
  public void setSessionContext (javax.ejb.SessionContext ctx) { }

  public void ejbCreate ()
    throws javax.ejb.CreateException
    { }
```

```
public float getStandardPizza ()
  {
  return (STANDARD_COST);
  }

public float getCustomPizza (String [] toppings)
  {
  return (getStandardPizza () +
          (toppings.length * TOPPING_COST));
  }
}
```

Build and Deploy the EJB

BEA WebLogic 6.1 makes use of the open source community's Ant tool to control the development process. This tool is configured via the file **build.xml**, which is one of those that you copied earlier. You should examine this file and the other XML files that are associated with the Ant build process: **build-ejb.xml** and **build-ws.xml**.

You should also briefly examine the other two XML files that you copied before: **ejb-jar.xml** and **weblogic-ejb-jar.xml (**they are simply text files that can be opened with any editor). Together, these define the deployment descriptor for the Pizza EJB that you have developed. These files tell WebLogic which Java class files comprise the bean, how session/transaction handling should be performed on behalf of the bean, etc.
Notice that the bean is associated with the name *GetYourPizzasHere* in the file weblogic-ejb-jar.xml. This will become important for clients, as you shall see.

To use Ant to build and deploy the bean is easy. It is necessary to ensure that ant is executable via the command line (this only needs to be done once when a Command Prompt window is initially opened, but it needs to be done each time a new Command Prompt window is opened). Type:

```
> C:\bea\wlserver6.1\config\mydomain\setEnv.cmd
```

To build and deploy the EJB, simply type:

```
> ant -verbose
```

This will create the **pizzas.ear** "enterprise archive" file and then copy it to the WebLogic deployment directory **C:\bea\wlserver6.1\config\mydomain\applications**.

The next time WebLogic is started, it will deploy the bean and be ready to handle SOAP requests for the bean's services.

Create the Java Client

Execute the following command:

```
> cd C:\pizzas\jClient
```

Create the following in the file **Client.java**:

```
package pizzas.jClient;

import java.util.Properties;
import java.io.PrintWriter;
import javax.naming.Context;
import javax.naming.InitialContext;

import pizzas.pizzaEJB.*;

public class Client
  {
  private static final PrintWriter
    out = new PrintWriter (System.out, true);
  public static void main (String [] args)
    throws Exception
    {
    String url = "http://localhost:7001";

    if (args.length > 1)
      {
      out.println("Usage: java pizzas.jClient.Client url");
      return;
      }
    else if (args.length == 1)
      url = args [0];

    final Properties
      h = new Properties();

    h.put (Context.INITIAL_CONTEXT_FACTORY,
          "weblogic.soap.http.SoapInitialContextFactory");
    h.put ("weblogic.soap.wsdl.interface",
          Pizza.class.getName());
    h.put ("weblogic.soap.verbose",
           "false");

    final Context
      context = new InitialContext (h);
    final Pizza
      pizzaHouse = (Pizza) context.lookup (url +
        "/pizzas/GetYourPizzasHere/GetYourPizzasHere.wsdl");
    out.println ("Cost of a standard pizza: $" +
                 pizzaHouse.getStandardPizza ());
    final String []
      toppings = { "mushrooms", "pepperoni" };
    out.print ("Cost of a pizza with");
    for (int i = 0; i < toppings.length; i ++)
      out.print (" " + toppings [i]);
    out.println (": $" + pizzaHouse.getCustomPizza (toppings));
    }
  }
```

Start WebLogic

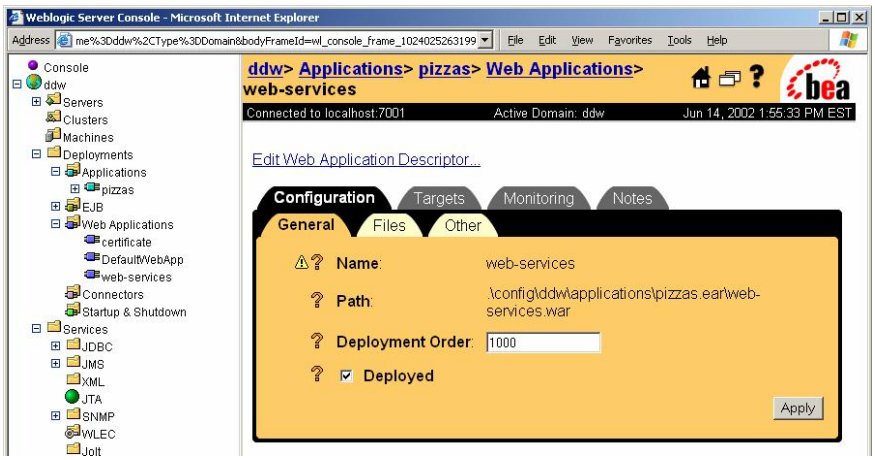
If WebLogic is not already running, execute the commands:

```
> cd C:\bea\wlserver6.1\config\mydomain
> .\startweblogic.cmd
```

Verifying the Deployment with the Console

WebLogic has a web-based console that allows one to examine deployments and the overall operation of the system.

Point your browser to the URL http://localhost:7001/console. You will be asked to provide a username and password (use *system* and *weblogic* respectively) and you will see the console. You can navigate around the system using the tree on the left of the window. The following picture shows an example:



Obtain the Client Proxy Jar File

Point your browser to the URL **http://localhost:7001/pizzas/GetYourPizzasHere/index.html**. You will see the following:

Save the identified **client.jar** archive to the following location (*you will first need to create the appropriate directory hierarchy*): **C:\pizzas\jClient\client.jar**. This file location will be inserted into the client's CLASSPATH environment variable later.

You should also examine the Web Services Description Language file that you can retrieve via the above URL (if you use Netscape, It may not display in the browser window so either save it to disk first, or use the appropriate “View Source” command to examine it once your browser has stopped downloading). You will see that the file describes everything about the WebService to make it possible for a “self-configuring” client to come along and interact with the service.

Compile the Java Client

As before, you will use the Ant tool to build the application. Type:

```
> cd C:\pizzas\jClient
> ant -verbose
```

Execute the Java Client against the EJB

At last! Execute the following command:

```
> java -classpath client.jar;. pizzas.jClient.Client
```

You should see the application run and return the price of various pizza combinations.

A Cross-Language Demonstration

Since WebLogic WebServices allow SOAP-based access, any SOAP-aware client can make use of a WebLogic Webservice. This is an *important* feature. The typical J2EE systems typically allow for Web, Java and CORBA clients but have not had (and still do not have) a convincing way of working with Microsoft COM-based technologies. This has long been a stumbling block for large enterprises. WebServices hold a lot of promise for the many enterprises that have a heavy investment in Microsoft COM-based technologies.

This section will show how Visual Basic Scripting can be used to access a Java-based EJB.

Establish a Working Directory

Execute the commands:

```
> cd C:\pizzas\vClient
```

Create the Visual Basic Script

Create the following in the file **pizzas.vbs**:

```
SET soapclient = WScript.CreateObject ("MSSOAP.SoapClient")
```

```
Call soapclient.mssoapinit
("http://localhost:7001/pizzas/GetYourPizzasHere/GetYourPizzasHere.
wsdl")

WScript.Echo "The cost of a standard Pizza is: $" &
soapclient.getStandardPizza()
```

Be careful of line breaks: there should be simply three lines of code in the source file.

Execute the Script

Ensure that WebLogic is running and that the pizzasEJB is still correctly deployed.

Within a Command Prompt execute the following simple command:

```
> cscript /nologo pizzas.vbs
```

You will see the now-familiar cost of a standard pizza displayed for you.

Do not underestimate the power implied in this simple exercise: you have created an enterprise-grade application in Java and invoked it from a simple script written in VBScript. This is a good example of cross-language interoperability. The solution would work through firewalls and is fairly simple to get going and to administer. WebServices really hold a lot of promise for the future.