BBj and Web Services

By David Wallwork

The term "Web services" has set the computing world abuzz during the last few years. Almost any current computer magazine contains an article about or a reference to Web services. Yet, most people do not understand just what a Web service is, or what it does. This article provides simple explanations of several key Web service terms. Additionally, this article demonstrates a BBj program that uses a Web service to translate phrases from English to French, German, and Spanish.

Background

In many ways, using Web services is similar to using HTML. When a user requests an HTML page from a remote Web server, the Web browser on the user's machine communicates with the remote Web server by transferring ASCII data. Often, a program that is running on the Web server has generated the HTML file that displays in the user's browser. The Web browser cannot tell (and does not need to know) what language was used to write the program that generated the HTML. The language and the operating system of the remote program are transparent to the Web browser. Web services provide this same transparency using Remote Procedure Calls (RPC).

A Remote Procedure Call (RPC) is any mechanism that allows a program running on one machine to invoke code that is actually running on some other machine. There have been a number of implementations of RPC (CORBA, RMI, DCOM, traditional client/server implementations etc.). However, none of these implementations provide the level of transparency seen when using HTML.

Web services is a set of standards (W3C) for doing RPC. When programs use these standards, the data transferred between the programs is all ASCII data in XML format. This allows a program (the consumer program) to invoke code (the provider program) that is running on another machine, even when that provider program is written in a different language.

Although this sounds rather daunting, in practice it is quite simple. The key to making this work is the Web Service Definition Language (WSDL). WSDL is a specific form of XML, which describes the methods provided by a Web service. Every modern language provides tools that understand WSDL and can use a WSDL document to generate the 'glue' that allows a program to participate in the Web services world.

When an organization publishes a Web Service, they provide a WSDL document to describe that Web service. When a developer wants to consume the service, they will use their language-specific tools to analyze the WSDL file and to generate code, called program "stubs," in their language of choice. When developers want to invoke a method of the Web service, they call a method on the stub and receive a return value from that stub. The stub translates the request made in the developer's language into XML data and sends that data to server offering the Web service. The stub receives a response that is also in XML and translates that response to a data type understood by the calling program. The following section shows how easy this process is.

Translating from English to German, French, and Spanish within BBj

The BBj program listed in Table 1 below uses a public Web service to translate a phrase from English to French, German, and Spanish.

Table 1

```
REM instantiate Web service stubs
  service! = new wsdl.babelfish.BabelFishService_Impl()
  port! = service!.getBabelFishPort()

REM make a phrase to be translated
  phrase$ = "Not all that glitters is gold"

REM translate phrase to French
  translationMode$ = "en_fr"
  print port!.babelFish(translationMode$, phrase$)
```

```
REM  result: pas tout ce qui scintille est or

REM translate phrase to German
  translationMode$ = "en_de"
  print port!.babelFish(translationMode$, phrase$)

REM  result:  nicht alles, das funkelt, ist Gold

REM translate phrase to Spanish
  translationMode$ = "en_es"
  print port!.babelFish(translationMode$, phrase$)

REM  result: no todo que brilla es oro
```

For this BBj program to run correctly, the class files for wsdl.bablefish package must be available on the machine running BBj Services. The following sections provide information about how to generate the class files from the published WSDL document that describes the BabelFish Web service.

Finding Web Services
The BabelFish Web service is one of many published Web services at www.xmethods.com. This site contains descriptions for a number of different Web services. Each description has the URL for the corresponding WSDL document. The language tools use WSDL documents to generate the stubs needed to use the Web service.

Generating Consumer Side Stubs for a Web Service
The Web services Developer Pack contains the tools required to generate the consumer stubs. This developer pack is downloadable from:

java.sun.com/webservices/downloads/webservicespack.html

After installing jwsdp, there are four steps required to generate all of the consumer side stubs used in the example provided in Table 1.

Step 1. Create a file named babelfish.xml in the /jaxrpc-1.0.3/bin directory containing the XML code from Table 2.

Table 2

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
 xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
 <wsdl name="babelfish"
   location="http://www.xmethods.net/sd/2001/BabelFishService.wsdl"
   packageName="wsdl.babelfish">
   <typeMappingRegistry>
   </typeMappingRegistry>
 </wsdl>
</configuration>
```

Step 2. Run the following command from the /jaxrpc-1.0.3/bin directory:

wscompile -gen babelfish.xml

Running this command creates consumer side stubs. These stubs are the generated class files that perform the actual communication with the Web service provider. The stubs convert Java parameters into XML parameters and exchange SOAP (Simple Object Access Protocol) messages with the Web service provider.

Step 3. Create a jar file that contains the generated class files by issuing the following command from the /jaxrpc-1.0.3/bin directory:

[JAVA_HOME]\bin\jar -cf babelfish.jar wsdl

where [JAVA_HOME] is the directory where your Java SDK is installed.

Step 4. Place this new jar file in the CLASSPATH for BBj Services by following the instructions below:

1. Start the BBj Enterprise Manager.
2. Connect to the appropriate BBj Services.

3. Double click on the Java Settings node.
4. Select the CLASSPATH tab.
5. Click the Add button.
6. Select the babelfish.jar file.
7. Restart BBj Services.

Providing Web Services in BBj

BASIS is currently in the process of writing an extension to the Java BBjBridge to provide a simple structure for publishing BBj programs as a Web service. TechCon2003 promises a demonstration of this program, paving the way for customer availability. Publishing a Web service requires proper configuration and is somewhat more difficult than consuming a Web service. However, after proper configuration, publishing a BBj program as a Web service becomes surprisingly easy! Come to TechCon2003 to learn more about Web services.