Lab 10: BPEL

Introduction: In this lab, we will learn how to create and execute BPEL-based Web services using NetBeans and the Sun Application Server. We will see how to deploy a simple BPEL process as part of a "composite application" (a service-based component interface and implementation) to the application server and how to test the composite service's interface with JUnit.

The tutorial is based on Sun's synchronous hello-world BPEL composite application example available at `http://www.netbeans.org/kb/55/helloworldca.html`.

Introduction

As commercial technology for service-oriented architecture (SOA) matures, open source alternatives are also starting to appear. As we discussed in class, the Web Services Business Process Execution Language (WS-BPEL) provides a standard platform-independent method for specifying Web service orchestrations. There are at least three major open source efforts to enable BPEL-based development:

- Sun's NetBeans Enterprise Pack (`http://www.netbeans.org/products/enterprise/`) offers a graphical environment for BPEL design and XML editing as well as Java code generation based on the Service Component Architecture (SCA) specification for service components.
- The Eclipse SOA Tools Platform Project (`http://www.eclipse.org/stp`) offers a graphical business process modeling tool that generates BPEL, a BPEL to Java translation engine, and tools and frameworks for deploying services to runtime containers.
- JBoss jBPM (`http://www.jboss.com/products/jbpm`) provides a business process modeling language, an execution engine for the language, visual modeling tools, and support for BPEL.

The Sun tools are the most mature and easy to use, but the Eclipse and JBoss tools are catching up. Here we'll focus on the BPEL and composite application support in NetBeans 6.0. NetBeans is a bit resource hungry but runs quite well on a 2 GHz Pentium 4 with 1 GB of RAM running Ubuntu Gutsy Gibbon (7.10). Its main benefits are close integration with Java EE 5 (JPA, EJB 3.0, etc.) and strong support for Web services development, including asynchronous Web Services via JAX-WS 2.0. With this API, we don't have to create threads on our own — the library handles it for us.

NetBeans installation

1. Download NetBeans 6.0 Beta 2 from `http://download.netbeans.org/netbeans/6.0/beta2/` or the local URL provided by the instructor.
2. The most convenient setup of NetBeans is in your home directory:

   ```
   % cd ~
   % chmod 755 netbeans-6.0beta2-linux.sh
   % ./netbeans-6.0beta2-linux.sh
   ```

   You can simply accept the defaults.

In the Gnome environment on Ubuntu, NetBeans is automatically added to the Applications → Programming menu and a shortcut is created on the desktop.

Hello World test

To get the feel of NetBeans, let's just do a standard Java hello world program before building our first BPEL module. Fire up the IDE and go to File → New Project → Java → Java Application. Give the project a name such as "Hello World," select "Set as Main Project" and "Create Main Class." The `.java` file will be created and opened in the source view. Just replace the comment in `main()` with your hello world message:

```
System.out.println("Hello World!");
```

To compile, use Build → Build Main Project, or you can press F11 or click the build icon in the toolbar. You should see a successful compile, then you can run with Run → Run Main Project, or you can press F6 or the run icon in the toolbar.

Synchronous BPEL application

First we try a synchronous BPEL application. This is a quick version of the full tutorial, "Developing a Hello World Composite Application," at `http://www.netbeans.org/kb/55/helloworldca.html`. If you get stuck or want to see more details, go there. They even have videos you can watch.

Create the project

1. Set up the server runtime. Click the "Services" tab in the left panel and expand "Servers." If you installed correctly you should see "Glassfish V2." If not, go to `http://www.netbeans.org/kb/60/soa/synchsample.html#configure-sjas` and follow the directions to set up the application server runtime. Otherwise, right click on the server and select "Start." It will take a minute or two. You should see the message "Application server startup complete" and in the console, and you should see a little green arrow icon next to the server in the "Services" tab.

2. Create your BPEL Module project. Go to File → New Project → Samples → Service Oriented Architecture → Synchronous BPEL process. Name the project "SynchronousSample." You should see two projects, a BPEL module called SynchronousSample and a "composite application" called SynchronousSampleApplication.

BPEL, WSDL, and XSD

Our BPEL module is going to be invoked through a Web service endpoint that we provide to remote clients. To allow clients to access our service, we have to implement the logic of the service and publish a specification of the service's interface so that developers of client service consumers can properly format their XML messages and interpret responses.

WSDL (Web Services Description Language) is the main way we specify the characteristics of a Web service endpoint. A WSDL specification contains

- An abstract description of the Web service interface, containing a list of operations (actions the client can invoke) and corresponding input/output messages;
- A concrete definition, binding operations to specific transport technology such as SOAP.

The input and output messages for our service have to be defined using XSD (XML Schema Definition). We can either *embed* the XSD in our WSDL document or keep it in a separate `.xsd` file and *import* the definition in the WSDL document. NetBeans likes us to keep the XSD separate, so it is separate in this sample BPEL project.

See Erl, T. (2005), *Service Oriented Architecture (SOA): Concepts, Technology, and Design*, Prentice-Hall, for more detail on how WSDL and XSD are used to specify Web service endpoints.

1. Look at `SynchronousSample.bpel` in the Source panel and the Design panel. The BPEL source code is generated automatically by NetBeans using the graphical business process designer, or it can be exported by another tool.

   This is an extremely simple process that simply copies its input (a string) to its output.

2. Look at `SynchronousSample.wsdl` in the Source, WSDL, and Partner panels. The WSDL specification and its graphical visualization in the Partner panel tell us how the service works.

3. Lastly, look at `SynchronousSample.xsd`. This schema definition defines the request and response messages used to communicate with the Web service we're designing.

4. Let's modify the business process slightly. In the BPEL design view open the "Structured Activity" panel of the palette and drag an "If" node just before the "Assign" node. In the BPEL Mapper view, select Operator → == and String → String Literal. Set the string literal text to "Hello World," then connect the "any1" input of the "Equal" operator to the "paramA" value of the input message and the "any2" input of the "Equal" operator to the "Hello World" literal. Connect the resulting Boolean to the output of the If activity.

5. Now go back to the main design pane and move the assignment node into the "true" flow of the conditional in the If node.

6. Drag a new assignment node from the palette to the "false" flow of the conditional in the If node. In the mapper, create a concatenation of the literal "Hello " with the "paramA" input string and set the "resultType" of the output to the concatenation result.

Composite application

The Java code for our sample business process needs to be packaged for deployment on the application server.

In Service Component Architecture (SCA), an emerging industry-led open specification for service component creation, deployment, and composition, BPEL modules are deployed inside of so-called "Composite Applications." A composite application is similar in some ways to an EJB.

1. Make sure the Output window is visible, then right click on the composite application project (SynchronousSampleApplication) and select "Deploy." Select the Glassfish V2 application server. Watch for errors or "BUILD SUCCESSFUL."

2. To test the new composite application with JUnit, create a unit test by right clicking on the "Test" node under the composite application project and selecting "New Test Case." Name the test case "MyTestCase."

3. When prompted by the test case wizard, select the WSDL and Operation1 operation.

4. Open the "Input" test data and replace the input string with your name.

5. Open the "Output" test data and note that it is empty.

6. Right click on "TestCase0" and select "Run." The test should execute and the "JUnit Test Results" panel should show that the test passed.

7. Now run MyTestCase. The test should fail since the observed output is not the same as Output.xml. Go ahead and save the observed result as the expected result, and make sure it is what you expect from the logic you created.

That's all! We have only scratched the surface of the NetBeans tools for Web services and SOA, but this will help you get started.

There are many more examples and tutorials on BPEL and SOA development with NetBeans available at `http://www.netbeans.org/kb/trails/soa.html`.