UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática

Enterprise Application Integration 2009/2010
Assignment #3 – Application Integration with BPEL

Objectives

- To gain familiarity with the currently foremost important language for application integration and business process design language for web services: **BPEL**.

Due Date

- **30th November 2009**

Scheduled Effort

- **24 hours**, according to the following plan:
    - 8 hours → Reading and learning technologies (e.g. doing tutorials)
    - 12 hours → Coding and testing
    - 4 hours → Writing the report

PLEASE WRITE DOWN THE EFFECTIVE NUMBER OF HOURS SPENT IN EACH TASK
THE REPORTED EFFORT SHOULD BE PER STUDENT!
[This information is critical to in the future adjust the amount of work asked from students]

Final Delivery

- All source code for the project.
- A small report (5 pages max) about the implementation of the project. The report must specify the number of hours spent per student while working on the assignment. In appendix, you should add all the source code of your programs. Also, in the appendix, include all the diagrams for your business processes. Please use PDF. I will not open any word documents.

Software

For this assignment you will need to use the following components: 1) a BPEL engine; 2) a BPEL modeling tool; 3) a web services container.

I recommend (not enforce) that you use "**GlashFish ESB v2.1**", part of the OpenESB project (https://open-esb.dev.java.net/). **Please download the complete integrated package which includes both GlashFish ESB v2.1 AND NetBeans 6.5.1 (https://open-esb.dev.java.net/Downloads.html)**. This set of tools is highly integrated, being quite easy to develop web services and integrate them using BPEL.

Some of you may be tempted to use a more recent version of NetBeans, namely NetBeans 6.7. It's also available in that web site. But, as stated, it has not been thoroughly tested and may not be as stable.

It's also possible to use other sets of tools. Our experience with other tools is that: a) either they are highly integrated and quite good (e.g., IBM WebSphere; Oracle BPEL Process Manager) but, as a downside, they are very resource-hungry, making it difficult to do productive work; b) the tools are not so well integrated and require considerable hand-patching.

One very popular BPEL engine right now, for production-grade systems, is Active Endpoints (http://www.active-endpoints.com/). If you are looking into enriching your CV by learning a tool that is being used in industry, this should potentially be your choice. Nevertheless, I should warn you that I had significant trouble in the past to make it work with *JDK 6.0u16* + *JBoss*. The BPEL editor is also not so nice or intuitive as the one of Netbeans.

(*I must say that for this assignment I've used the following combination of tools:*

- *Java JDK 1.6.0u16*
- *JBoss 5.1.0GA + Eclipse, for hosting web services and the web tier*
- *GlashFish ESB, for modeling and orchestrating the business processes*

but again, this is probably an overkill…)

Other alternatives are also possible – several BPEL editors are available and many platforms support it. For instance, right now even Apache has a BPEL engine (http://ode.apache.org/). Nevertheless, if you don't have much time to loose, I still recommend that GlashFish ESB is your choice.

IMPORTANT WARNING
It's quite common after deploying and undeploying projects in GlashFish several times that leftovers an inconsistencies arise with what is on you application server. A huge number of exceptions will start to creep-in at that time.

I STRONGLY RECOMMEND that you make a copy of the complete setup after you have installed it in your machine. Every time you need a "clean server", you can use the copy that you have created. Believe me, this is a life saver.

Bibliography

There are many online resources about **BPEL**. For learning the basics of this technology, the following approach is proposed:

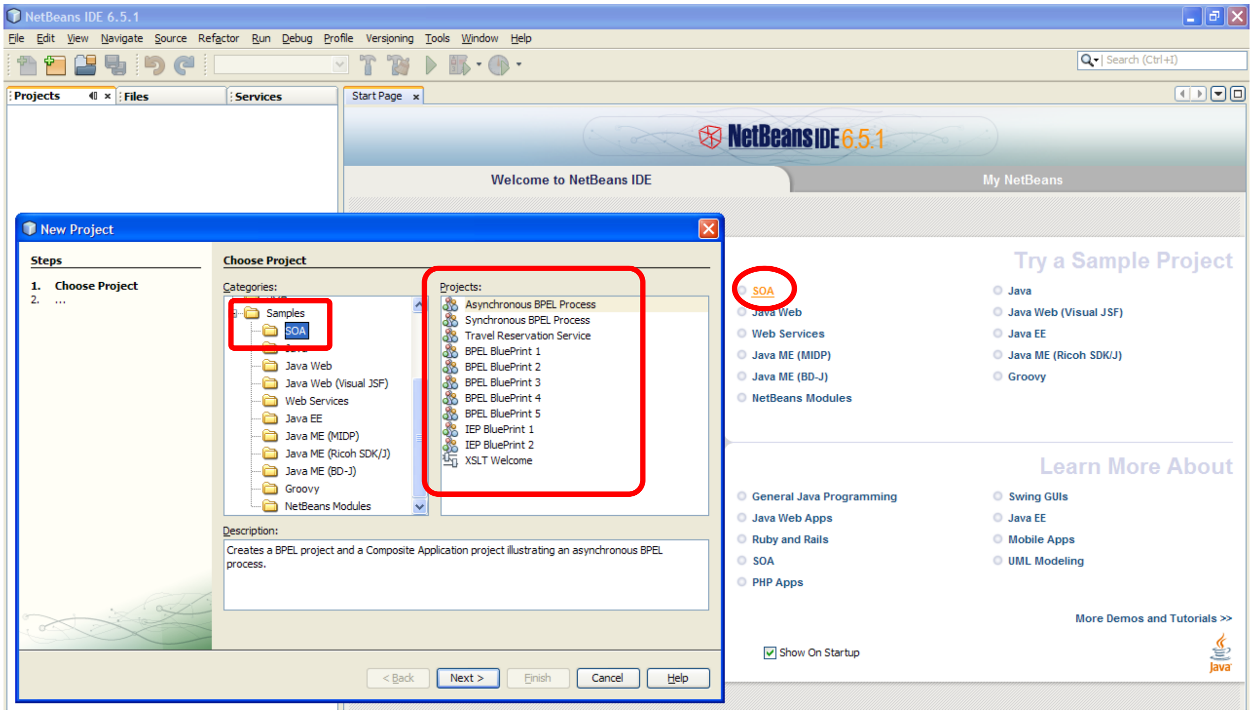
1. Read the articles "**Business processes in a Web services world**" and "**Business Process with BPEL4WS: Understanding BPEL4WS**". Read at least parts 1 and 2. The information is somewhat dated but you will get an overview of how things work. (Don't be scared by the number of links. The articles are actually short.)
   http://www.ibm.com/developerworks/library/ws-bpelwp/
   http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1/
   http://www.ibm.com/developerworks/webservices/library/ws-bpelcol2/

2. Watch the video "**Introduction to Sun GlashFish Enterprise Service Bus**" which demonstrates how to create this type of projects. Available at:
   http://developers.sun.com/docs/javacaps/tutorials/screencasts/Composite/start.html
   (the first part is about web services but afterwards it does cover BPEL!)

3. After you install GlashFish ESB, if you start NetBeans, you can see that it has a number of tutorials for helping you to get started (see image in next page). You should complete these tutorials.

4. Check out the **tutorials** and documentation for SOA on the NetBeans website:
   http://www.netbeans.org/kb/trails/soa.html
   (Very useful! This is a life saver!)

Important Note: Many of the documentation you will find on the web refers to BPEL 1.1. Currently, most BPEL engines are implementing features from BPEL-WS 2.0. Thus, it's only natural that you find that some primitives have been renamed, others disappeared and new ones appeared. This includes NetBeans and the Sun Application Server.



Creates a BPEL project and a Composite Application project illustrating an asynchronous BPEL process.

Assignment

On your last assignment you developed a 3-tier application called "**Low-Price Cameras Online**" (LPCO). As you may remember, it was mostly organized as a normal "application silo". In this assignment you are required to re-organize your application so that it follows the principles of a Service Oriented Architecture. What this means is that it still consists in the LPCO, but you are assuming you are a much larger enterprise, where which one of the modules of the application are actually separate departments of a company. The architecture of the application is thus changed in order to represent a large-scale system, possibly deployed in several machines geographically disperse.

The next figure illustrates the basic principle behind the assignment.

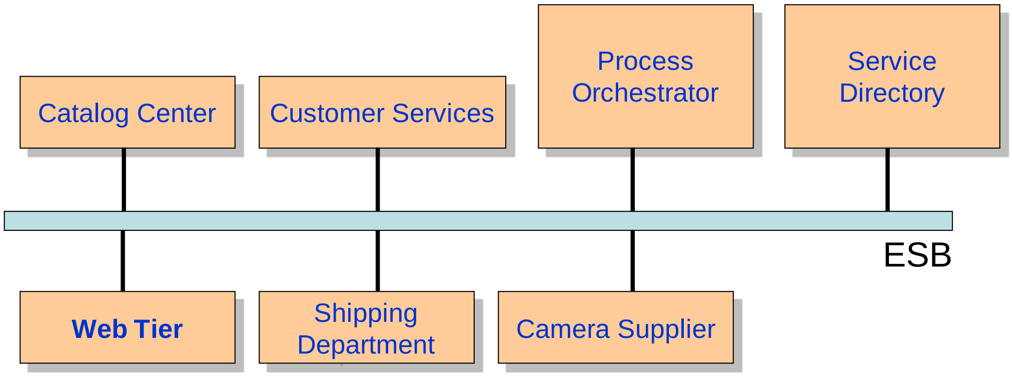


Figure 1 – Architecture of LPCO (SOA-like)

As you can see, all the different areas of the company are made available on an Enterprise Service Bus. In particular, all departments are made available as web services. A Process Orchestrator contains a set of BPEL business processes which implement the business rules in place in the organization. The modules shown on the picture are:

- **Web Tier**. Represents the user interface provided to clients. Corresponds to the functionality that you already had on the previous assignment. Note that instead of invoking EJBs, web activity will correspond to the invocation of business processes that are available on the "Process Orchestrator". In BPEL, business processes are always callable as web services.

- **Shipping Department**. Corresponds to the functionality implemented previously. Note that in this case the call back will correspond to the invocation of a business process on the Process Orchestrator.

- **Camera Supplier**. Corresponds to the functionality implemented previously. It allows you to find out the information about camera models.

- **Catalog Center**. It corresponds to the department in the company responsible for maintaining the catalog of cameras that are currently known. If you remember, any camera returned by a query performed to the "Camera Supplier" must be added to the catalog. Thus, the business process that actually calls the Camera Supplier web service will also need to call this one to update the catalog with new information. This service corresponds roughly to part of the database tier of assignment #2. Depending on your exact architecture, it may include parts that were implemented before on the business tier.

- **Customer Services:** Provides functionalities as authenticating users, keeping accounting information and alike. It's also in this service that you will maintain information about the orders currently submitted by clients and their completion status.

- **Process Orchestrator**: This corresponds to the "core business of the company". Every time there's web activity, it should result in an invocation to a business process. The business process will invoke other services in order to complete the required tasks. Likewise, if any other service needs to perform an activity, it should invoke a business process, never invoking the other services directly. It is expressly forbidden to do point-to-point integration having a web service (department) calling directly another department. The process orchestrator corresponds, roughly, to the business-tier of your previous assignment. Do note that "low-level" functionality should be implemented on the modules of the company and not at the process orchestrator. Business processes are written in BPEL.

- **Service Directory:** This corresponds to a directory where all services available in the company are registered and published. For this assignment you don't need to implement it.

You should try as much as possible reuse the code and web services of the previous assignment. In particular, if you were using JBoss, you can keep on doing so for the "external modules". You don't need (and shouldn't) migrate all the web services into GlashFish in order to complete the assignment[1]. The same thing applies to the web tier (it can still be run in JBoss). Your major preoccupation should be on the Process Orchestrator and on writing BPEL processes. Having said so, you will not be penalized if you decide to only use one application server, migrating the necessary code into it.

Finally, in this assignment you don't need to use a database or a persistence engine if you don't want to. Files or other similar mechanisms will do fine. Having said so, if you decide to re-use the persistence module from assignment #2, that will be a positive thing.

Good Luck!

[1] Think about this… If, for example, Camera Supplier is a department inside your company, with its functionality available as a web service in JBoss, you should be able to access it transparently. It shouldn't be required that everyone migrates to the same platform for having a SOA. In general, that would be unfeasible in a large organization.