Software Architectures

Assignment 3: Service Oriented Architectures

Assistants: Kennedy Kambona, Janwillem Swalens
Mail: {kkambona, jswalens}@vub.ac.be
Office: {10F730, 10F719}

Deadline: 24^{th} April 2014, 23:59

This exercise gives you the opportunity to experiment with some typical technologies used in *Service Oriented Architectures*. You will use the Business Process Execution Language (BPEL) to orchestrate two simple web services, and use the resulting web service with the extended Web Portal application from the previous assignment (# 2)[1].

NOTE: This assignment uses all kind of technologies you have most likely not used before. **Start early** with the technical part!

Assignment

This assignment consists of an implementation and a reporting part. Your implementation artifacts (project files) and a brief report have to be handed in.

Deadline: 24th April 2014 at 23:59. The deadline is fixed and will not be extended

Deliverables: The report and implementation should be handed in as a single ZIP file. Submit the ZIP file on the Software Architectures course page in PointCarré, by clicking on *Assignments (Opdrachten) > Assignment 3*.
The file should follow the naming schema `(Firstname-Lastname_)*3.zip`, for example: `Kennedy-Kambona_Janwillem-Swalens_3.zip`.

Team work You are allowed to work alone, or in a team of two. It is preferrable that you continue with your previous team-mate. Only one of you should submit the report on PointCarré, but be sure to mention both names in the report!

Grading The exercises will be graded and can become subject of an additional defense.

[1] If you did not work on assignment 2, you have to use the original version of the Web portal application.

Context

This assignment **reuses** the scenario of assignment 2. You are asked to extend the web portal with functionality to query the `LibrarySearch` web service, which itself will query two other web services.

Similar to the last assignment, you have to change the web portal implementation to provide yet another data backend. The new backend will be restricted to search for books. It has to be implemented as an addition to the existing layers, i.e., the book search has to use the configured database layer in addition to the new data source to query for books.

The `LibrarySearch` web service orchestrates two existing library web services using a BPEL process. The two web services are the library web service of the Soft lab, and the national library of Belgium. Both web services are provided as Tomcat web applications, and should not be changed. However, the BPEL process has to be optimized by you.

Exercise 1: BPEL Processes

For the first exercise, you have to modify the existing LibrarySearch BPEL process[2].

- Change the BPEL process to invoke the two library web services **in parallel**. The different structured activities in BPEL have been briefly introduced in the exercise session slides and are documented in the WS-BPEL 2.0 specification[3].
  - Give a brief summary of your changes in the report. Indicate BPEL constructs you used, and document changes in other files as well, if necessary.
  - Make it clear why you chose to parallelize certain activities and not others. This means, describe briefly which data dependencies between the different actions need to be taken into account when parallelizing requests.
  - What other technique could be used to optimize the invocation of multiples services apart from parallelization? Briefly state how this could be done in the BPEL process.
- Also propose a strategy to handle the different structures of the result data from the two web services NationalLibrary and SoftLibrary.
  - What would the best solution be if additional multiple services are going to use the `LibrarySearch` web service?
  - An implementation of this solution is optional.

Exercise 2: Integration with Legacy Software

The extended web portal application from the previous assignment 2 only has access to a local database of books at the moment. To allow users to access a larger number of

[2] LibrarySearch/LibrarySearch.bpel

[3] `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html#_Toc164738514`

books, especially the books in the Soft library and in the national library of Belgium, you will integrate the `LibrarySearch` web service into the web portal.

- Add it as another data backend, which only includes the functionality to search for books. In the end, searching in the web app for books should combine local results and the external web service results. Thus, the web portal should be able to use either of the two database layer implementations from the last assignment, and the new backend which queries the BPEL Web service i.e. (the original `(db.sql AND db.webservice)` OR your new implementation `(db.flatfile AND db.webservice)`)
- Generating the Web service client code is NOT optional! (In case of trouble, see the mentioned tutorials in part 4 of "Tools Setup" below, or feel free to make an appointment with the assistants)
- Describe your implementation decisions in the report:
  - What did you change in the web portal application?
  - How did your refactoring from the last assignment help or hinder you with this task?
  - Which pattern or architectural style did you use to make this task (if any)?
  - With respect to Exercise 1 (BPEL Process): How did you implement the data mapping of the result in the web portal application? Which element in your architecture is responsible, and what are the benefits and drawbacks of your design?

Exercise 3: Architecture

Give an overview of the architecture of the system.

- Chose an appropriate visualization, for example an UML or block diagram. The diagram should allow to see the connections between the relevant elements in this architecture, e.g., the involved web services, and the relevant parts of the web portal application.
- Also minimally depict your architecture which results from your design choices in exercise 2.
- **Briefly** describe your diagram and name the relevant concepts/architectural styles. **Do not** go into the details for responsible classes etc., the architecture has to show logical components, not primarily implementation details. Often you will have no control over available web services, so treat the library web services as if they where provided by external providers.

Package the Result

Please upload a single ZIP-archive to PointCarré "Assignment" option (Assignment 3). It should contain your report as a PDF, the changed `LibrarySearch` web service (packaged as an archive), and the web portal (also packaged as an archive). Make sure that the uploaded file use this naming schema:
First-LastName_First-LastName-3.zip

Tools Setup

To manually install the tools and configure the project you have sets of instructions to follow. After this is done, proceed to the next section (Generating a Web Service Client).

1 Eclipse and Tomcat

This assignment was designed using **Eclipse for Java EE Developers** and **Apache Tomcat 6.0.39**. If you do not have these two software for your project, follow instructions in the Assignment 2 pdf file from Pointcarre in the section 'Preparing Eclipse and Tomcat'. Also, follow the steps described in "Loading the Web Application" from the previous assignment Either way, before proceeding make sure that the Tomcat server runs the web portal application on localhost.

2 Preparing the Project in Eclipse

These steps indicate how to load and configure the project for this assignment in Eclipse.

1. Download the files for this assignment from PointCarré.
2. Import the WAR archive for the NationalLibrary and the SoftLabLibrary
   a) Choose File > Import... > Web > WAR file
   b) Select one of the WAR files, the target runtime should be the same Tomcat instance from last time.
   c) Finish the wizard, and repeat the same steps for the second WAR file[4].
3. Add the two applications to the Tomcat server. In the Servers-view right click on Tomcat > Add and remove...
4. The web portal application should also be configured to run on this Tomcat instance to be able to debug it.
5. For the BPEL process we will need another Tomcat instance, thus, the Eclipse Tomcat has to use different ports. Double click on the Tomcat server in the Servers-view > on the right hand side, change the ports like follows:

[4]If you get "Loading Descriptor" errors, it is because you have an older Java version. Luckily, these errors can be ignored and the services will still deploy.

- Tomcat admin port: 8105
- HTTP/1.1: 8180
- AJP/1.3: 8109

6. Now, the Web services and the portal application should be deployed and you can start the server (possibly in debugging mode).

7. The web portal should now be reachable via `http://localhost:8180/web_portal/`.

2.1 Accessing the Web services with soapUI (optional)

soapUI is a simple and convenient tool to interact with Web services for debugging. You can use this tool for instance to interact with the two NationalLibrary and SoftLibrary web services and investigate their result data.

1. soapUI can be downloaded from `http://sourceforge.net/projects/soapui/` (tested version: 4.6.4). Get it and extract the zip to a folder.

2. Run the web_portal application to start the `NationalLibrary` and `SoftLabLibrary` services

3. Navigate to the soapUI folder, start soapUI inside the `bin` folder, and add a new soapUI project. Enter one of the following URLs in the field Initial WSDL, and give the project a suitable name
   - `http://localhost:8180/NationalLibrary/services/LibraryService?wsdl`
   - `http://localhost:8180/SoftLabLibrary/services/soap?wsdl`

4. The new project contains an example request which can be sent and customized to test the web services.

3 Deploying the BPEL process

To deploy and test the BPEL process you will modify for exercise 1, you will use the Apache ODE BPEL engine deployed in Tomcat. This environment does not provide a lot tooling, but it is simple to setup and use to deploy.

1. You can download the WAR distribution of Apache ODE from `http://ode.apache.org/getting-ode.html` (tested version: 1.3.6).

2. Extract the ZIP archive and copy the ode.war to the webapps folder of your copy of Tomcat (created from assignment 2).
   Sidenote: The Tomcat instance started from Eclipse uses only the binaries of Tomcat, but the actual working directory is somewhere in Eclipse workspace. Thus, it is safe to reuse it.

3. Navigate to the Tomcat folder and start it from the command line.
   To work around a known issue, you will need to set an environment variable first.
   - For **Linux/Mac OS X**:
     export CATALINA_OPTS=−Dhttp.nonProxyHosts=localhost
     Then start Tomcat using bin/startup.sh
     If you run into permissions problems you may need to give this file execute permissions.
   - For **Windows**:
     set CATALINA_OPTS=−Dhttp.nonProxyHosts=localhost
     Then start Tomcat using bin/startup.bat
4. Copy the content of the LibrarySearch archive into the ODE processes folder: tomcat/webapps/ode/WEB−INF/processes/LibrarySearch
5. The process will be deployed automatically and can be seen at
   `http://localhost:8080/ode/`
6. You can get the WSDL for the web service in the URL
   `http://localhost:8080/ode/processes/LibrarySearchService?wsdl`
7. To test the BPEL process, delete the LibrarySearch folder and the LibrarySearch.deployed file, and then copy the changed version from the Eclipse workspace into the processes folder.

For NetBeans users, the GlassFish server also includes standard debugging support for BPEL processes, including breakpoints.[5].

4 Generating a Web Service Client

For the second exercise you will generate the client code for your new `LibrarySearch` webservice from the first exercise. Eclipse provides facilities to generate the client code for Web services, for instance see: `http://www.craigsprogramming.com/2011/03/tutorial-consume-any-web-service-using.html`.

Depending on your strategy for handling the result data, you might want to adapt the BookList type in LibrarySearchArctifacts.wsdl which is part of the `LibrarySearch` Web service. This file is also the input file you want to use for generating the client code.

Beside Eclipse, there are also command-line tools to achieve the same results, e.g., see `http://axis.apache.org/axis2/java/core/docs/userguide-creatingclients.html`.

[5] `http://soa.netbeans.org/soa/`