Distributed Programming II

A.Y. 2015/16

Final Test 2

All the material needed for this test is included in the folder where you have found this file, which will be called `[root]`. Copy your solution of Assignment 4 (ant script, WSDL, source code, and customization file if any) into this folder (using the appropriate sub-directories `wsdl`, `src`, and `custom`).

The test consists of a programming exercise (6 points) and a question (4 points). The exam can be passed only if the programming exercise solution passes the mandatory tests (at least two of the tests of part 1). In this case, the proposed mark will be the sum of the points got for the test and a base evaluation mark in the range 16-20, assigned on the basis of the evaluation of the submitted assignments (16 points granted because your assignments passed the mandatory tests at submission time and 4 extra points based on the evaluation of one extra aspect of your assignments).

Programming exercise

1. Modify your service(s) developed in Assignment 4 so that, when a request for creating a new process is received, if the request refers to a workflow that is not among the ones known to the service that has received the request, that service must contact the service provided with Assignment 3, asking for the description of the unknown workflow (service composition). If neither the service provided with Assignment 3 knows the workflow, your service has to report an error to the client without creating any process. Instead, if the service provided with Assignment 3 returns the information about the unknown workflow, your service must add that workflow to its list of known workflows and create the requested new process, associating it to the newly added workflow. This entails that, since that time onwards, your service must use that workflow without having to request it any more. Moreover, the service that returns information about workflows and processes must include the newly added workflow into the list of workflows returned to clients.

   Note that when a new workflow is added to the list of known workflows only the newly created process refers to that workflow. Note also that your services must keep the same interfaces they had originally, i.e. your client1 and client2 must still work with the new version of your web services.

   Check that your server can execute operations concurrently and modify it if necessary so that it can execute up to 5 requests concurrently.

2. Make a copy of the java sources of your web service(s) modified as requested in point 1, move this copy to the package `it.polito.dp2.WF.sol4.server2` (storing the files into the folder `[root]/src/it/polito/dp2/WF/sol4/server2`) and modify this copy according to the following specifications, in order to create a new version of your web services. Store the WSDL of the new version of your services in the `[root]/wsdl` folder. The new web services must keep the time of last modification of the information about workflows and processes, and return this time in the response of each operation that provides information about workflows or processes. Note that there is a **single** time of last modification for the whole WorkflowMonitor. When a new workflow is added to the list of known workflows or when a new process is created, this time must be updated. Change the URLs of these new web services so that the web service that includes the operations for creating new processes is published at the URL http://localhost:7080/wfcontrol, while the web service that includes the operations for reading all the available information is published at the URL http://localhost:7081/wfinfo (if the service is the same, it must be

published at both URLs). Of course, the new web services must provide the WSDL as usual. Update the *ant* script `[root]/sol_build.xml` so that it also compiles this new version of your web services as part of the `build-server` target.

3. Make a copy of the java sources of your client1 and client2, move this copy to the packages `it.polito.dp2.WF.sol4.client3` and `it.polito.dp2.WF.sol4.client4` respectively (storing the files into the corresponding folders under `[root]/src/`) and modify these copies so that (a) they can interact with the new version of your web services developed in point 2 and (b) the WorkflowMonitor created by the factory of your client3 also implements the Refreshable interface (the same one used for the client developed in Assignment 3; note that this interface is available inside `[root]/src/lab3.jar`). Add a new target named `build-test2-clients` to your *ant* script `[root]/sol_build.xml`. This new target must build client3 and client4. Of course, you can assume that server2 has already been started when this target is invoked.

Apart from these new specifications, all the specifications given for Assignment 4 still apply.

In particular, the classes for point 1 must be developed in the same packages used for the solution of Assignment 4 and stored in the same directories as the solution of Assignment 4. Make sure that the *ant* script `[root]/sol_build.xml` still works for the compilation of your new solutions and modify it as necessary. As in Assignment 4, you can assume that, when your clients are compiled and run, your web services have already been deployed (i.e. your `WorkflowServer` applications have already been started). Similarly, you can assume that when your server is compiled and run the server provided with Assignment 3 is already deployed.

Question

Explain if and how in your server modified in point 1 you avoid that two copies of the same workflow are created in your server if two clients concurrently request the creation of processes referring to the same unknown workflow. In your explanation, refer to the code of your server and copy the most significant lines of code of your server that show what you are explaining.

The answer to this question must be written in a text file named `[root]/answer.txt`

Correctness verification

In order to pass the exam your solution must pass at least the mandatory tests that are included in the archive (the sources are available in the folder `it.polito.dp2.WF.lab4.tests`). These tests can be run by the ant script `buildTest2.xml` included in the *.zip* file, which also compiles your solution by calling your ant script and runs the servers and the clients as necessary. The command for running the tests is

```
ant –f buildTest2.xml run-final-test –Dseed=XXXX
```

The other targets in `buildTest2.xml` are for internal use only. The tests are divided into two parts, each one including a number of junit tests. In order to pass the exam it is necessary to pass at least two if the junit tests of part 1. Part 1 tests point 1 while part 2 tests points 2 and 3.

In the tests for part 1, your server is started with testcase 0 while the server of Assignment 3 is started with testcase 1. Then, client2 is used to request the creation of a new process that refers to workflow NormalSale, which is unknown to your server but is known to the server of Assignment 3. The operation is expected to succeed and a new workflow is expected to be added to your server. Finally, client1 is used to check that your server provides information coherent with the operation performed.

In the tests for part 2, your server2 is started as in part1 and your client3 and client4 are used in a way similar to what is done in part 1.

Submission format

A single *.zip* file must be submitted, including all the files that are part of your solution (including the files of your solution of Assignments 4 that you are re-using). The *.zip* file to be submitted must be produced by issuing the following command (from the `[root]` directory):

```
ant –f buildTest2.xml make-final-zip
```

Important: check the contents of the zip file named `solution.zip` after having run the command and leave the zip file where it has been generated!