Distributed Programming II

A.Y. 2012/13

Assignment n. 5

All the material needed for this assignment is included in the *.zip* archive where you have found this file. Please extract the archive to an empty directory (that will be called `[root]`) where you will work.

The assignment consists of developing a client for a web service called PJSInfoService using JAX-WS "dynamic proxy" programming based on WSDL. The service gives read access to information about the hosts of a cluster (without information about jobs). The service can be started on your own local machine by running the command

```
$ ant run-server
```

Once the service has been started, the WSDL can be obtained as usual at the URL where the service is available, i.e. `http://localhost:8181/PJSInfoService?wsdl`

The semantics of the operations made available by the service is intuitive: one operation (getHostNames) returns the list of available hosts while another operation (getHosts) returns all the information of a set of hosts given their names.

The client to be developed has to take the form of a library, similar to the ones developed in the second part of assignments 2 and 4. The library must implement all the interfaces and abstract classes defined in package `it.polito.dp2.PJS`, and must return the data obtained from the service (as there are no data about jobs and job groups, the library must always return empty job lists and empty job group lists).

The library must be written entirely in package `it.polito.dp2.PJS.sol5` with all its sources stored in folder `[root]/src/it/polito/dp2/PJS/sol5/`. As for assignments 2 and 4, The library must include a factory class named `it.polito.dp2.PJS.sol5.ClusterFactory`, which extends the abstract factory `it.polito.dp2.PJS.ClusterFactory` and, through the method `newCluster()`, creates an instance of your concrete class implementing the `Cluster` interface.

More precisely, the library must work in the following way: whenever the factory method `newCluster()` is invoked, the library invokes the service and makes a snapshot of the data available in that moment from the server. The returned `Cluster` will always return the data in this snapshot, without contacting the service again. Therefore, an update of the available data is possible only by creating a new `Cluster` via the factory. If for any reason the service cannot be contacted or some error in the interaction with the service prevents the download of reliable data when the `Cluster` is created, a `ClusterException` must be thrown.

The actual URL used by the library to contact the service must be customizable: the actual URL has to be read by the library as the value of the `it.polito.dp2.PJS.sol5.URL` system property.

The artifacts have to be generated from the WSDL using the ant script provided with this assignment. The generation of artifacts can be started by running the command

```
$ ant compile-wsdl
```

As you can see by inspecting the ant script, the `compile-wsdl` target gets the WSDL using the service location specified in the system property mentioned above.

As in assignments 2 and 4, the library must be robust and interoperable, without dependencies on locales. The data coming from the server cannot be considered as trusted.

Correctness verification

Before submitting your solution, you are expected to verify its correctness and adherence to all the specifications given here. In order to be acceptable for examination, your assignment must pass at least all the automatic mandatory tests. Note that these tests check just part of the functional specifications! In particular, they only check that the data extracted from the library are the same as the data provided by the web service.

Other checks and evaluations on the code will be done at exam time (i.e. passing all tests does not guarantee the maximum of marks).

All the automatic tests use the random data generator provided with this assignment. The *.zip* file of this assignment includes a set of tests like the ones that will run on the server after submission. The tests are similar to the ones used for Assignments 2 and 4 and they have the same test cases. Tests can be run by the ant script included in the *.zip* file, which also compiles your solution. Of course, before running the tests you must have started your server. Then, you can run the tests using the same command used for assignments 2 and 4.

Note: please ensure to run the server and the tests by manually setting the same seed on both of them: if you do will not do this the tests will very probably fail just because the seed used to initialize the server will be different with respect to the seed used to run the tests. This is an example about how to properly run the two commands

```
ant run-server -Dseed=123456
ant runFuncTest -Dseed=123456
```

Submission format

A single *.zip* file must be submitted, including all the files that have been produced. The *.zip* file to be submitted can be produced issuing the following command (from the `[root]` directory):

```
$ jar -cf lab5.zip src/it/polito/dp2/PJS/sol5/*.java
```

Note that the *.zip* file **must not** include the files generated automatically.