

RMI DEMO

Assignment 3: implement a simple RMI

Enrico Magnago 186957

Introduction:

This project is an implementation of a simple Remote Method Invocation.

The server application binds a *DocumentTimestampper* instance that appends the time stamp information to any input *Document* instance.

The client creates a *Document* instance and asks the server to append the time stamp, then both documents are printed on standard output.

Implementation:

The project is composed of 3 gradle modules: *Client*, *Server* and *Common*.

The *Common* module contains the structures that are shared between *Client* and *Server* : the *Document* class and the *DocumentTimestampper* interface. This interface extends both *Remote* and *Serializable*. It provides the signature of a method that accepts a *Document* as input and returns a *Document* as output, the returned object should be equal to the input one but with an additional *String* that represents the timestap. *Document* is implemented as a simple *List* of *String* objects. The *Common* modules contains a constant *String* which represents the name where the *DocumentTimestampper* can be found.

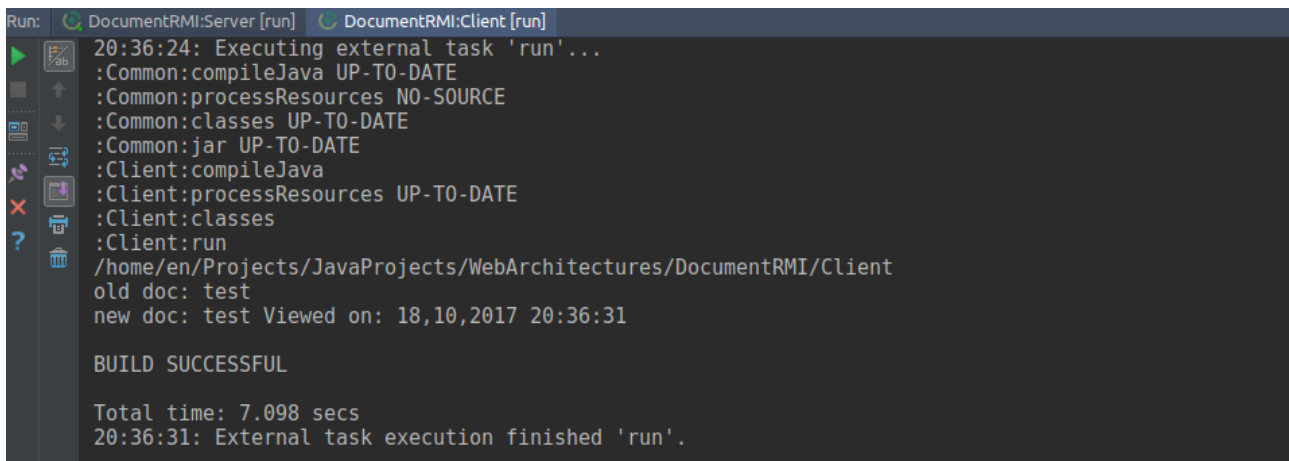
The *Client* module contains the *Client* class which first sets the security policy, then creates a *SecurityManager*. The client tries to connect to the server on the default port on localhost and then asks for the *DocumentTimedstamper* instance registered at the name specified in *Common*. Once the client has acquired that stub it creates a simple *Document* which is used as input to the *addTimestamp* method, the input *Document* internally gets serialized, sent to the server instance which adds the timestamp and the returned *Document* is sent back to the client which deserializes the received object and shows it as the return value of the function. The client by printing both *Document* instances highlights that a copy of the input has been performed in the process.

The *Server* module contains the class *Server* which is an implementation of the *DocumentTimestampper* interface and extends *UnicastRemoteObject* so that in order to get an instance of this class is sufficient to create an instance of *Server*. The server provides an implementation of the *addTimestamp* method as required by the *DocumentTimestampper* interface. In the main methods the *Server*, like the *Client*, sets the security policy and creates a *SecurityManager* instance. After this initial set up it creates a registry on the port 1099 which is the default one and it binds an instance of *Server* to the name specified in *Common*.

Deployment:

In oder to deploy the application is sufficient to perform the run task on the *Server* module and then on the *Client* module. The *Server* will stay active waiting for requests from the clients, while the client will quit as soon as it has received the answer from the server and printed the 2 documents on standard output.

Note:



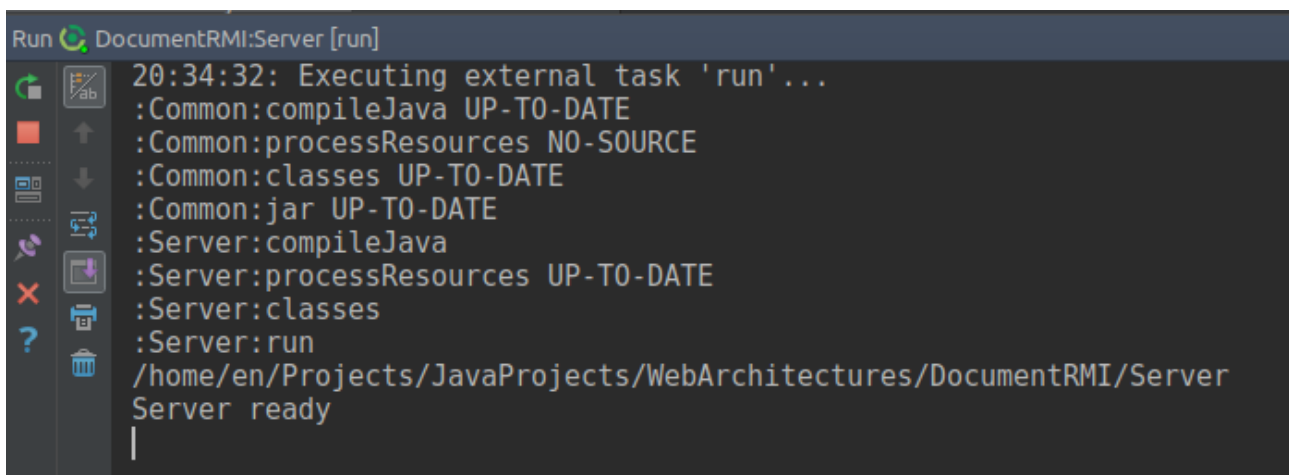
```
Run: DocumentRMI:Server [run] DocumentRMI:Client [run]
20:36:24: Executing external task 'run'...
:Common:compileJava UP-TO-DATE
:Common:processResources NO-SOURCE
:Common:classes UP-TO-DATE
:Common:jar UP-TO-DATE
:Client:compileJava
:Client:processResources UP-TO-DATE
:Client:classes
:Client:run
/home/en/Projects/JavaProjects/WebArchitectures/DocumentRMI/Client
old doc: test
new doc: test Viewed on: 18,10,2017 20:36:31

BUILD SUCCESSFUL

Total time: 7.098 secs
20:36:31: External task execution finished 'run'.
```

In this image the execution of the client can be seen, in particular the 2 strings printed by the java application: *“old doc: test”* which corresponds to the document the client has sent to the server and the second one *“new doc: test Viewed on:”* which is the document received as response from the server.

The following image shows the corresponding server execution



```
Run DocumentRMI:Server [run]
20:34:32: Executing external task 'run'...
:Common:compileJava UP-TO-DATE
:Common:processResources NO-SOURCE
:Common:classes UP-TO-DATE
:Common:jar UP-TO-DATE
:Server:compileJava
:Server:processResources UP-TO-DATE
:Server:classes
:Server:run
/home/en/Projects/JavaProjects/WebArchitectures/DocumentRMI/Server
Server ready
|
```

The server once it is ready waits and answers to client requests.