RMI client and server

Maurizio Franchi

2 October 2015

Introduction

In this report I discuss two projects: `Square Root` and `Notebook`. In each of two projects there are a server and a client. The server and the client communicate through the `RMI API`.

Implemention

In this section I explain how I implemented the Square Root and the Notebook project.

Square Root

For the Square Root I decide to create two distinct java project one for the server called `SqrtServer` and another for the client called `sqrtClient`.
In the `Server` I create the java packege called `sqrtserver` in which there are a interface called `ISquareRoot` and a java class called `SqrtImpl`.
In the java interface there is the following function:

```
double getSquare(int a) throws RemoteException;
```

This function called `getSquare` is double and accept a integer value a.

In the java class `SqrtImpl` through the following code

```
LocateRegistry.createRegistry(1099);
```

I create a registry on the default port, then through the code below:

```
ISquareRoot stub = (ISquareRoot)UnicastRemoteObject.exportObject(sqrt, 0);
```

displays all methods that implements and then through the following code:

```
System.out.println("Ready for RMI's");
```

I print that the server is ready.

In the `Client` I created two different packages:

- `sqrtserver` in which there is the java interface `ISquareRoot` (the same interface presents in the Server);
- `sqrtclient` in which there is the java class `SqrtUser`.

In the java class `SqrtUser` there are the main in which I set to integer s1 and s1 is the number 25 and I set also s2 as double. After that:

- I get to the register that I create in the server;
- I set s2 as the square root of s1;
- I print the result.

Notebook

For the Notebook, as for the Square Root, I decide to create two distinct java project one for the server called `NotebookServer` and another for the client called `notebookclient`. In the `Server` I create the java packege called `notebookserver` in which there are two interfaces called `INotebook` and `Server` ans two java classes called `ServerImpl` and `Notebook`.
In the java interface `INotebook` there is the two following functions:

```
public void add(String s) throws RemoteException;
public void read() throws RemoteException;
```

in which the `add` function, that accepts only string, add a string t the Notebook and the function `read` read the string in the Notebook.

In the second interface `Server` there are the function below:

```
public void sign(INotebook n) throws RemoteException;
```

This function create a remote method sign for the Notebook.

In the java class `Notebook` through the following code:
`public LinkedList<String> str = new LinkedList<>();` I create a list of strings and then through the function `add` and `read` the class add and read the list of the strings.

In the java class `ServerImpl` through the following code:

```
Server engine = new ServerImpl(1);
Server stub = (Server) UnicastRemoteObject.exportObject(engine, 0);
Server engine2 = new ServerImpl(2);
Server stub2 = (Server) UnicastRemoteObject.exportObject(engine2, 0);
```

I create two different server `engine` and `engine2` and put the servers in the registry.

After that through the following code:
`Registry registry = LocateRegistry.createRegistry(1099);` I create the regestry in the default port.
Then through the following code:

```
System.out.println("s1 and s2 bound");
```

I print that the two server are ready.

In the `Client` I created two different packages:

- `notebookserver` in which there is the java interface `INotebook`, `Server` and `Notebook` (the same interfaces and class present in the Server);
- `notebookclient` in which there is the java class `NotebookUser`.

In the java class `NotebookUser` there are the main in which I verify if there is the SecurityManager and if there isn't I create it. After that:

- I get to the register that I create in the server;
- I recall the two server;
- through the function sign I sign the two servers to the Notebook

After that I print the result.

Deployment

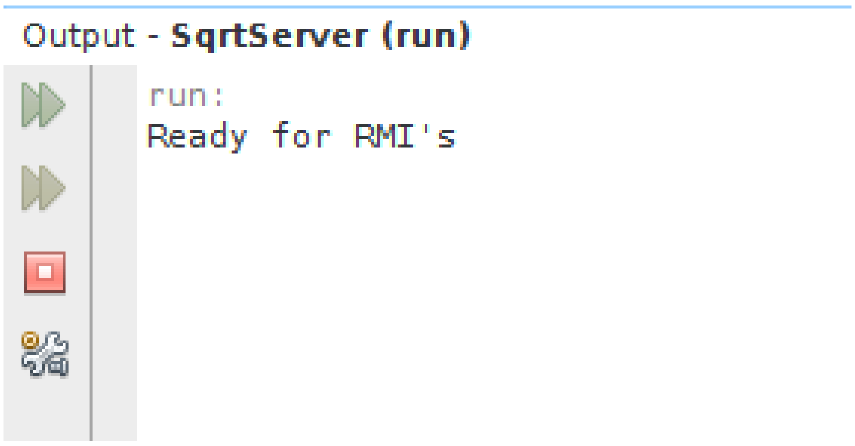
Square Root

To start the two project Server and Client I create for each project their jar with the command `Clean and built` and then I go to the property of each project, I click on Run and in the field VM Options I put:

```
-cp /path in with there is notebookclient/notebookclient/src;/path in with there is no
```

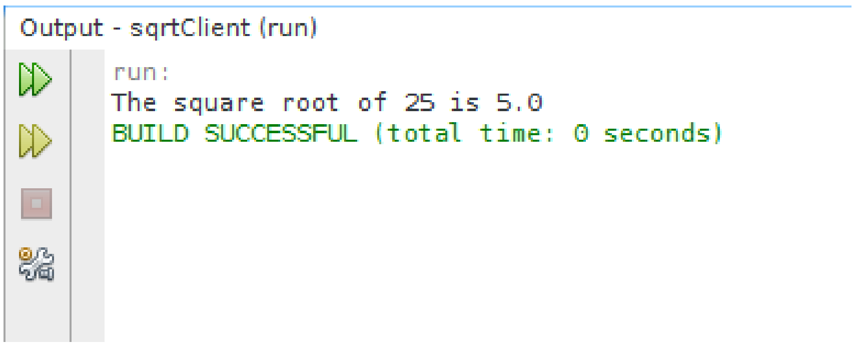
This id for the client. It is the same for the server.

The result is:



```
run:
Ready for RMI's
```

Figure 1: The output of the server of Square Root



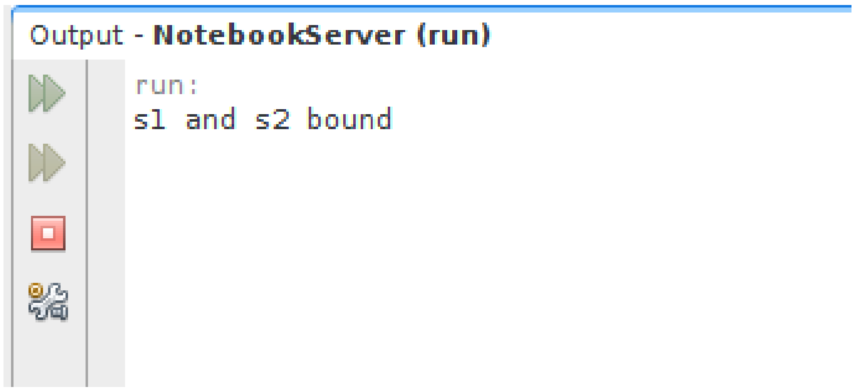
```
run:
The square root of 25 is 5.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 2: The output of the client of Square Root

Notebook

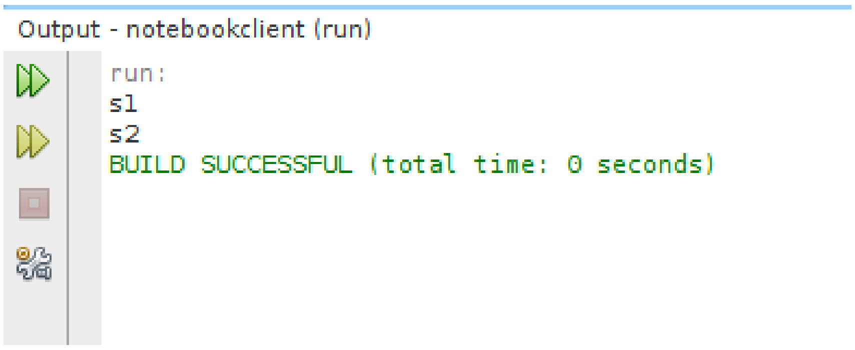
To start the two project Server and Client I create for each project their jar with the command `Clean and built` and then I go to the property of each project, I click on Run and in the field VM Options I put:
`-cp /path in with there is notebookclient/notebookclient/src;/path in with there is no`
This id for the client. It is the same for the server.

The result is:


Output - NotebookServer (run)

```
run:
s1 and s2 bound
```

Figure 3: The output of the server of Notebook


Output - notebookclient (run)

```
run:
s1
s2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4: The output of the client of Notebook