RMI-based chat application: Papinho

Andon Tchechmedjiev, Jander Nascimento

March 14, 2011

1 Presentation

2 Remote interfaces

We will first define the remote interfaces we used, as they constitute the basis of our application. There is one remote interface for the server application and one for the client application.

The architecture we have chosen for our application is a classical server centric one. Namely, all the clients are synchronized to a central state kept on the server. All messages be they public or private messages, will always transit through the server, where logging will occur.

2.1 Chat server side

The server provides the services in the table 1. Those services are provided in order to allow the client to comunicate between other clients and the server itself.

Method	Summary
addClient	Add a client to the list of clients in the server side
removeClient	remove a client from the list of clients
sendMessage	broadcast certain message to all the clients, or for a specific client
clientNameChange	Change the user's login

Table 1: Server Side: Provided methods

2.2 Chat client side

The client must provide some methods so the server can inform the client about new messages, user renaming, etc.

Method	Summary
addClient	Add a client to the list of clients in the client side
removeClient	remove a client from the list of clients in the client side
receiveMessage	Add a new message to the main window
receivePrivateMessage	Add a new message to a private chat window
changeClientName	Change the username in the list of clients on the client side

Table 2: Client Side: Provided methods

3 History and persistence

In order to manage the persistence, the main design pattern used was a creational pattern *Proxy*. The main goal of this pattern is to decide which instance to generate. On the current implementation the persistence method used was the *serialization of the object to disk*, what is also called as *bean serialization*.

4 Graphical User Interface

The Graphical User Interface(GUI) allow the user to interact with the application in a more intuitive way. The GUI was implemented using Java Swing libraries. An example of the interface can be seem in the Figure 1

Figure 1: Main Window

5 Networking and security issues

Since Java Virtual Machine runs inside a sandbox, term coined by sun. The application must explicity tell the JVM which kind of connections are allows and which hosts are allowed to do so. To specify this information is necessary to write a policy file. In this file we specify the permission, host, port and actions.

In the example 1 the policy allows socket connections from any host to any port.

Listing 1: Policy file

```
grant {
  permission java.net.SocketPermission "*:*", "acce
};
```