Compte-rendu TP n°5 : RMI

Fichier XML Original :

```
<clients>
    <client>
        <pseudo> michel </pseudo>
        <mdp>mot de passe</mdp>
    </client>
    <client>
        <pseudo> MonsieurDenis </pseudo>
        <mdp>ILoveXML</mdp>
    </client>
    <client>
        <pseudo>Charles</pseudo>
        <mdp>Bonjour</mdp>
    </client>
    <client>
        <pseudo>Henri</pseudo>
        <mdp>Pointer</mdp>
    </client>
</clients>
```

Fichier XML Après exécution:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<clients>
  <client>
    <pseudo> michel </pseudo>
    <mdp>mot de passe</mdp>
  </client>
  <client>
    <pseudo> MonsieurDenis </pseudo>
    <mdp>ILoveXML</mdp>
  </client
  <client>
    <pseudo>Charles</pseudo>
    <mdp>Bonjour</mdp>
  </client>
  <client>
    <pseudo>Henri</pseudo>
    <mdp>Pointer</mdp>
  </client>
  <client>
    <pseudo>henri</pseudo>
    <mdp>thierry</mdp>
  </client>
</clients>
```

RMIServeur.java :

```java
package Serveur;

import Common.Client;
import Common.Clients;
import org.w3c.dom.Document;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ArrayList;
import java.util.List;

public class rmiServeur {

   public static void main(String[] args) {

      try {

         XmlOperations xmlOD = new XmlOperations();

xmlOD.readXMLFile("C:\\Users\\Clementoni\\Documents\\ENSIM\\4A\\Programmati
on concurrente\\TP5\\src\\Serveur\\clients.xml");
         xmlOD.getClients();

         Registry reg = LocateRegistry.createRegistry(1099);
         reg.bind("xmlOP", xmlOD);
         System.out.println("C'est bon " + xmlOD);
      } catch (Exception e) {
         System.out.println("ERREUR");
         e.printStackTrace();
      }
   }

}
```

RMIClient.java :

```java
package Client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;

import Common.XmlOperationsI;

public class rmiClient {

   static String text;
   static Scanner sc = new Scanner(System.in);

   public static void main(String[] args) {
      try {
         Registry reg = LocateRegistry.getRegistry("localhost", 1099);
         XmlOperationsI xmlOD = (XmlOperationsI) reg.lookup("xmlOP");
         System.out.println("OD=" + xmlOD);
         System.out.println(xmlOD.addUser("michel","forever"));
         System.out.println(xmlOD.addUser("bastien","forever"));
         System.out.println(xmlOD.removeUser("michel"));
```

```
        do{
           //get the command line from user
           System.out.println("Veuillez saisir une commande " +
                 " :\n P_EXIST  <pseudo> \nU_EXIST  <pseudo> <mdp>\nADD
<pseudo> <mdp>\nREMOVE  <pseudo>\n");
           text= sc.nextLine();


           String commande= text.substring(0,text.indexOf(' '));

           if(commande.equals("P_EXIST")){
              String pseudo= text.substring(text.indexOf('
')+1,text.lastIndexOf(' '));

              System.out.println(xmlOD.pseudoExist(pseudo));
           }
           if(commande.equals("U_EXIST")){
              String pseudo= text.substring(text.indexOf('
')+1,text.lastIndexOf(' '));
              String mdp= text.substring(text.lastIndexOf(' ')+1);

              System.out.println(xmlOD.userExist(pseudo,mdp));
           }
           if(commande.equals("ADD")){
              String pseudo= text.substring(text.indexOf('
')+1,text.lastIndexOf(' '));
              String mdp= text.substring(text.lastIndexOf(' ')+1);

              System.out.println(xmlOD.addUser(pseudo,mdp));
           }
           if(commande.equals("REMOVE")){
              String pseudo= text.substring(text.indexOf(' ')+1);
              System.out.println(xmlOD.removeUser(pseudo));

           }

        }while (!text.equals("STOP"));

      } catch (Exception e) {
        e.printStackTrace();
      }
   }

}
```

Client.java :

```
package Common;

public class Client {
    private String pseudo;

    private String mdp;

    public Client(){
    }

    public void setPseudo(String pseudo){
        this.pseudo = pseudo;
    }

    public void setMdp(String mdp){
        this.mdp = mdp;
    }

    public String getPseudo(){
        return this.pseudo;
    }

    public String getMdp(){
        return this.mdp;
    }

}
```

Clients.java :

```
package Common;

import java.util.ArrayList;
import java.util.List;

public class Clients {

    private List<Client> clientList = new ArrayList<Client>();

    public void addClient(Client cli){
        clientList.add(cli);
    }

    public void setClientList(List<Client> clientList){
        this.clientList = clientList;
    }

    public List<Client> getClientList(){
        return this.clientList;
    }

}
```

XMLOperations.java :

```
package Serveur;

import java.io.File;
import java.io.IOException;
import java.rmi.*;
import java.rmi.server.*;
import java.util.ArrayList;
import java.util.List;

import Common.Client;
import Common.XmlOperationsI;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class XmlOperations extends UnicastRemoteObject implements
XmlOperationsI {

   private List<Client> clientList = new ArrayList<Client>();
   private Document xmlDoc = null;


   protected XmlOperations() throws IOException,
ParserConfigurationException, SAXException {
   }

   // Read the XML File fileName and puts its content inside xmlDoc
Document.
   public void readXMLFile(String fileName) throws
ParserConfigurationException, IOException, SAXException {
      File xmlFile = new File(fileName);
      DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document doc = builder.parse(xmlFile);
      this.xmlDoc = doc;
   }

   // Gets the client list from the XML File
   public void getClients(){
      NodeList clientNodes = xmlDoc.getElementsByTagName("client");
      List<Client> clientListLocal = new ArrayList<Client>(); // Will
replace the existing list

      for(int i = 0 ; i < clientNodes.getLength() ; i++){
         Node clientNode = clientNodes.item(i);
         clientListLocal.add(new Client());

         if(clientNode.getNodeType() == Node.ELEMENT_NODE){
            Element clientElement = (Element) clientNode;
clientListLocal.get(i).setPseudo(clientElement.getElementsByTagName("pseudo
").item(0).getTextContent());

clientListLocal.get(i).setMdp(clientElement.getElementsByTagName("mdp").ite
m(0).getTextContent());


            /*DEBUG ONLY
            System.out.println("Nom du client " + i + " : " +
clientListLocal.get(i).getPseudo());
            System.out.println("Mdp du client " + i + " : " +
clientListLocal.get(i).getMdp());
             */
         }

      }
      clientList = clientListLocal; // Replacing the old list
      System.out.println("La liste contient " + clientListLocal.size() + "
clients.\n");
   }

   // Check if pseudo already exists in database
   public Boolean pseudoExist(String pseudo) throws RemoteException {
      for(int i = 0 ; i < clientList.size() ; i++){
         if(clientList.get(i).getPseudo().equals(pseudo))
            return true; // This pseudo is already in Database
      }
         return false;
   }

   // Check if user already exists in database
   public Boolean userExist(String pseudo, String mdp) throws
RemoteException {
      for(int i = 0 ; i < clientList.size() ; i++){
         if(clientList.get(i).getPseudo().equals(pseudo) &&
clientList.get(i).getMdp().equals(mdp))
            return true; // These pseudo and password are already in
Database
      }
      return false;
   }

   // Add user to database
   public String addUser(String pseudo, String mdp) throws RemoteException
{

      // Cant add same user twice
      if (userExist(pseudo,mdp)) return null;
      if(pseudoExist(pseudo)) return null;

      // Adding user to the clientList
      Client michel = new Client();
      michel.setPseudo(pseudo);
      michel.setMdp(mdp);
      clientList.add(michel);

      // Adding user to XML File
      Element root = xmlDoc.getDocumentElement();
      Element clientElement = xmlDoc.createElement("client");
      root.appendChild(clientElement); // Adding a new client tag
```

```
        // Adding the user's pseudonym
        Element pseudoElement = xmlDoc.createElement("pseudo");
        pseudoElement.appendChild(xmlDoc.createTextNode(pseudo));
        clientElement.appendChild(pseudoElement);

        // Adding the user's password
        Element mdpElement = xmlDoc.createElement("mdp");
        mdpElement.appendChild(xmlDoc.createTextNode(mdp));
        clientElement.appendChild(mdpElement);

        // Regenerate XML File
        try {
            regenerateXMLFile();
        } catch (ParserConfigurationException e) {
            throw new RuntimeException(e);
        } catch (TransformerConfigurationException e) {
            throw new RuntimeException(e);
        } catch (TransformerException e) {
            throw new RuntimeException(e);
        }

        return "Utilisateur ajouté";
    }

    // Removes a user from database
    /*public String removeUser(String pseudo) throws RemoteException {


        // Removing client from list
        for(int i = 0 ; i < clientList.size() ; i++){
            if(clientList.get(i).getPseudo().equals(pseudo)){
                clientList.remove(i);
            }
        }

        // Removing from XML File

        NodeList nodeList =  xmlDoc.getElementsByTagName("client");
        for (int i = 0 ; i < nodeList.getLength() ; i ++){
            Node node = nodeList.item(i);
            if(node.getNodeType() == Node.ELEMENT_NODE){
                Element clientElement = (Element) node;
                NodeList textNodes =
clientElement.getElementsByTagName("pseudo");
                for(int j = 0 ; j < textNodes.getLength() ; j++){
                    Element pseudoElement = (Element) textNodes.item(i);

                    if(pseudoElement.getTagName().equalsIgnoreCase(pseudo)){
                        clientElement.removeChild(pseudoElement);
                    }
                }
            }
        }
        xmlDoc.normalize();
        // Regenerate XML File
        try {
            regenerateXMLFile();
        } catch (ParserConfigurationException e) {
            throw new RuntimeException(e);
        } catch (TransformerException e) {
            throw new RuntimeException(e);
        }
        return "Utilisateur supprimé";
    }
    */


    public String removeUser(String pseudo){

        // Removing client from list
        for(int i = 0 ; i < clientList.size() ; i++){
            if(clientList.get(i).getPseudo().equals(pseudo)){
                clientList.remove(i);
            }
        }

        // Removing client from XML File
        try{

            // Set clients as root
            Element root = xmlDoc.getDocumentElement();
            // Making a NodeLists of Clients
            NodeList clients = xmlDoc.getElementsByTagName("client");

            for(int i=0;i<clients.getLength();i++){
                Element client=(Element) clients.item(i);


if(client.getElementsByTagName("pseudo").item(0).getTextContent().equals(ps
eudo)){
                    root.removeChild(client);
                }
            }
            xmlDoc.normalize();

            // Regenerate XML File
            try {
                regenerateXMLFile();
            } catch (ParserConfigurationException e) {
                throw new RuntimeException(e);
            } catch (TransformerException e) {
                throw new RuntimeException(e);
            }


        }catch(Exception e){
        }
        return "Utilisateur supprimé";

    }

    // Regenerate the XML File. Is used by addUser() and removeUser()
methods.
    private void regenerateXMLFile() throws ParserConfigurationException,
TransformerException {

        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource domSource = new DOMSource(xmlDoc);
```

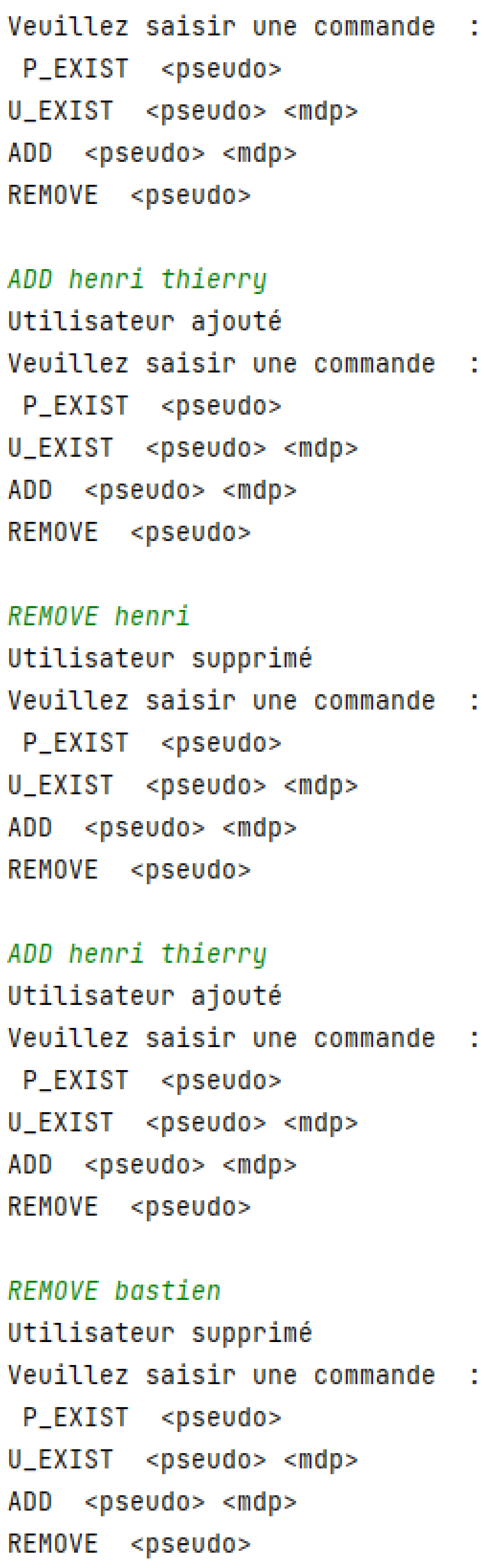
```
      // Generation will be on a test File
      // TODO : Change path to original file
      String xmlFilePath = new
String("C:\\Users\\Clementoni\\Documents\\ENSIM\\4A\\Programmation
concurrente\\TP5\\src\\Serveur\\clientsTest.xml");
      StreamResult streamResult = new StreamResult(new File(xmlFilePath));

      // Indenting document
      transformer.setOutputProperty(OutputKeys.INDENT, "yes");
      transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
amount", "2");

      //Generating XML File
      transformer.transform(domSource, streamResult);
   }

}
```

Exemple d’exécution :



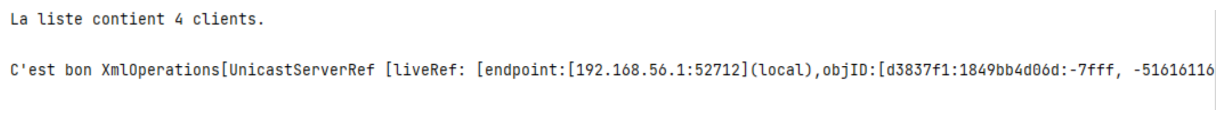
```
Veuillez saisir une commande  :
 P_EXIST  <pseudo>
U_EXIST  <pseudo> <mdp>
ADD  <pseudo> <mdp>
REMOVE  <pseudo>

ADD henri thierry
Utilisateur ajouté
Veuillez saisir une commande  :
 P_EXIST  <pseudo>
U_EXIST  <pseudo> <mdp>
ADD  <pseudo> <mdp>
REMOVE  <pseudo>

REMOVE henri
Utilisateur supprimé
Veuillez saisir une commande  :
 P_EXIST  <pseudo>
U_EXIST  <pseudo> <mdp>
ADD  <pseudo> <mdp>
REMOVE  <pseudo>

ADD henri thierry
Utilisateur ajouté
Veuillez saisir une commande  :
 P_EXIST  <pseudo>
U_EXIST  <pseudo> <mdp>
ADD  <pseudo> <mdp>
REMOVE  <pseudo>

REMOVE bastien
Utilisateur supprimé
Veuillez saisir une commande  :
 P_EXIST  <pseudo>
U_EXIST  <pseudo> <mdp>
ADD  <pseudo> <mdp>
REMOVE  <pseudo>
```



```
La liste contient 4 clients.

C'est bon XmlOperations[UnicastServerRef [liveRef: [endpoint:[192.168.56.1:52712](local),objID:[d3837f1:1849bb4d06d:-7fff, -51616116
```