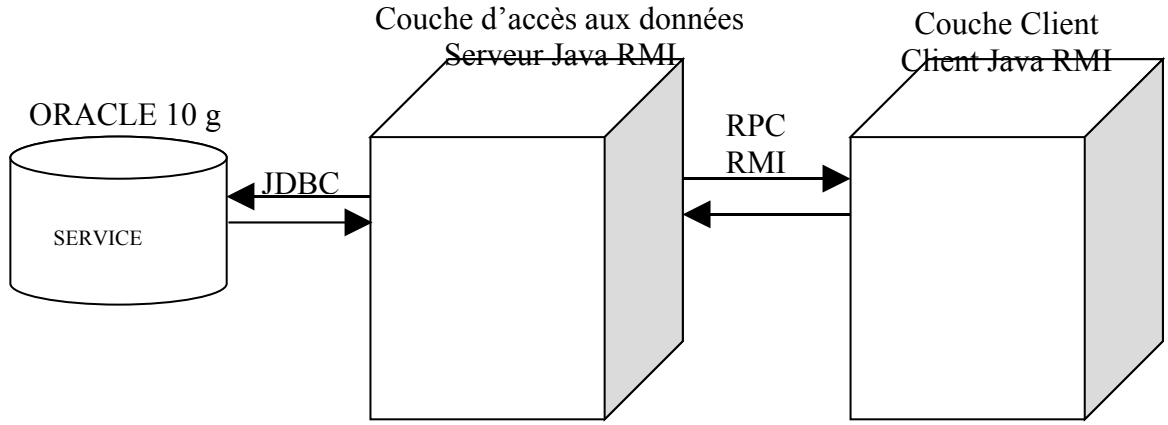
Java RMI, étapes de développement d'une application

Pour décrire les étapes de développement d'une application Java RMI, nous utiliserons une base de données ORACLE avec une table SERVICE qui représente les service au sein d'une société (Service Comptable, Service Informatique …).
L'objectif de cet exemple est de pouvoir insérer des services dans la table SERVICE à partir d'une application Java RMI :



Le développement d'une application Java RMI passe par les étapes suivantes :

1. Définir une Interface distante
   Il s'agit dans cette étape de définir une interface distante qui sera implémentée par la suite au niveau du serveur RMI. Dans notre exemple l'interface contient une seule méthode ServiceInsert qui prend en entrée un Code service et son libelle et les insère dans la table Service :

```
package gestionemployes;
import java.rmi.*;
public interface DataAccessInterface extends Remote {
public void ServiceInsert(String code, String libelle) throws RemoteException;
 }
```

2. : Créer une application Serveur
   a. Développer une classe qui implémente l'interface distante
      Dans cette étape nous développerons la couche d'accès aux données (Dans notre exemple les données sont représentés par la table SERVICE). Cette couche sera une Classe java (Serveur RMI), avec les méthodes suivantes :
      i. Constructeur de la classe : Le rôle du constructeur est d'initialiser et d'ouvrir une connexion avec la base de données
      ii. Méthode implémentant l'interface distante : Cette méthode implémente l'interface distante et insère les données dans la base de données.
      iii. Méthode principale (main) : C'est la méthode main principale nécessaire pour lancer notre serveur RMI en tant qu'application

```
package gestionemployes;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
import java.sql.*;
import oracle.sqlj.runtime.*;
import sqlj.runtime.*;
import sqlj.runtime.ref.*;
import oracle.jdbc.driver.OracleDriver;
public class DataAccess extends UnicastRemoteObject
                        implements DataAccessInterface {
   Connection conn = null;
   public DataAccess() throws RemoteException {
       try
       {
```

```
        DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
          conn = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:cnam", "NFP111", "nfp111");
      }
      catch(SQLException sqle){
      System.out.println(sqle.toString());
      }
       finally{
      }
   }
public void ServiceInsert(String code, String libelle) throws RemoteException {
    try
    {
      Statement statement = conn.createStatement();
      String query = "INSERT INTO SERVICE (CODE,LIBELLE ) VALUES ("+"'"+code+"'"+","+"'"+libelle+"'"+')';
      statement.executeUpdate(query);
      conn.commit();
      System.out.println("Service "+code+" inséré");
    }
    catch(SQLException sqle){
    System.out.println(sqle.toString());
    }
    finally{
    }
}
 /* Methode principale du service */
  public static void main(String args[])
  {
    //set the security manager
    try
      {
       System.setSecurityManager(new RMISecurityManager());
       //create a local instance of the object
       DataAccess Server = new DataAccess();
       //put the local instance in the registry
       Naming.rebind("//localhost/DataAccess" , Server);
       System.out.println("Server waiting.....");
      }
    catch (java.net.MalformedURLException me)
      {
        System.out.println("Malformed URL: " + me.toString());
      }
    catch (RemoteException re)
      {
        System.out.println("Remote exception: " + re.toString());
      }
  }
}
```

3. Créer une application cliente qui appelle les méthodes du serveur à distance
   Cette couche est une classe java (RMI Client) qui insère le service Test dans la base de données en passant par le serveur Java RMI :

```
package gestionemployes;
import java.rmi.*;
import java.rmi.server.*;
import gestionemployes.DataAccessInterface;
public class BusnessLayer {
   public BusnessLayer() {
   }
   public static void main(String[]  args)
   {
      String code = args[0];
      String libelle = args [1];
      //set the security manager for the client
     System.setSecurityManager(new RMISecurityManager());
     //get the remote object from the registry
     try
       {
        System.out.println("Security Manager loaded");
```

```
          String url = "//localhost/DataAccess";
           DataAccessInterface remoteDataAccess = (DataAccessInterface)Naming.lookup(url);
          System.out.println("Atteindre l'objet Remote");
           remoteDataAccess.ServiceInsert(code,libelle);
            System.out.println("Insertion faite");
            System.out.println("Le code remote est : "+code);
        }
       catch (RemoteException exc)
        {
         System.out.println("Error in lookup: " + exc.toString());
        }
       catch (java.net.MalformedURLException exc)
        {
         System.out.println("Malformed URL: " + exc.toString());
        }
       catch (java.rmi.NotBoundException exc)
        {
         System.out.println("NotBound: " + exc.toString());
        }
        catch (Exception exc)
         {
          System.out.println("NotBound: " + exc.toString());
         }
    }
}
```

Pour tester l'application Java RMI, il faut

1. Ajouter le répertorie des classes Serveur dans la variable d'environnement CLASS_PATH de la machine Serveur

2. Modifier la stratégie de sécurité du Client
   Pour que le client puisse se connecter au serveur RMI , il faut modifier son fichier de stratégie : **<Répertoire Jdeveloper>\jdk\jre\lib\security** et ajouter :

```
grant {
permission java.net.SocketPermission "*:1024-65535", "connect,accept";
permission java.net.SocketPermission "*:80", "connect";
};
```

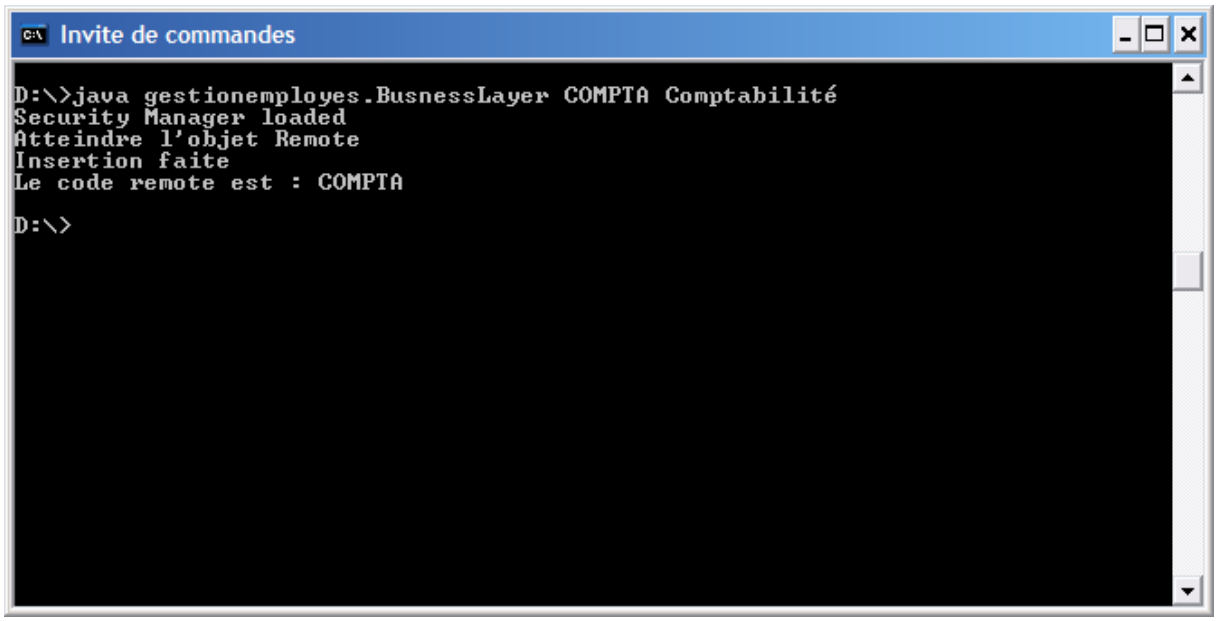
3. Créer les Stubs avec l'outil rmic
   Pour cela il faut lancer à partir d'une ligne de commande, la commande rmic avec comme argument le nom de la classe Serveur :

```
c:\> rmic gestionemployes.DataAccess
```

4. Démarrer le registre avec l'outil rmiregistry
   A partir d'une ligne de commande, lancer l'outils rmiregistry :

```
c:\> rmiregistry
```

5. Lancer le serveur pour créer les objets :
   Exécuter l'application Serveur
6. Tester le client :
   Lancer l'application Client, et insérer des services dans la table SERVICE.



Si on intérroge la base de données pour voir si les données sont insérées :