Exercice « CORBA-IDL » complet (J2SE 1.4 +)

Thème :
Ecrire un programme serveur permettant de calculer un montant T.T.C. à partir d'un montant H.T. et d'un taux de T.V.A., en utilisant l'architecture CORBA-IDL.

1: Proposer une interface du service en IDL (***I****nterface*** *D****efinition*** *L****anguage***).

2 : Donner la commande permettant d'engendrer les fichiers java client et serveur à partir de l'interface IDL, ainsi que la liste des fichiers créés.

3 : Ecrire la classe java de l'objet « servant » (la classe qui fait le calcul).

4 : Ecrire la classe du serveur (celle qui donne accès au service, tout en le masquant pour le client).

5 : Ecrire la classe du client.

6 : Donner les commandes permettant d'activer l'ORB sur la machine serveur, de lancer le serveur, et d'exécuter le client

Exercice « CORBA-IDL » complet (J2SE 1.4 +)

S O L U T I O N

Etape 1: *Ecrire l'interface du service en IDL (fichier* **`calcul.idl`***)* :

```
module calcul {

        interface calcul_montants {
                double calcul_ttc (in double mt_ht, in double taux);
        };
};
```

Etape 2 : *Commande permettant d'engendrer les fichiers java client et serveur à partir de l'interface IDL* :

D:\cnam\jdk\bin> **idlj -fall** "D:\cnam\jdev\mywork\corba\ calcul.idl"

Cette commande engendre les fichiers suivants (à partir de Corba 2.2 / J2SE 1.4):

- _calcul_montants**Stub** : souche ou talon de l'objet calcul_montants (côté client).
- calcul_montants : Interface Java de l'interface.
- calcul_montants**Operations** : Interface des méthodes distantes (définies dans l'interface IDL); ce fichier est partagé par la souche et le squelette.
- calcul_montants**POA** : Squelette de l'objet calcul_montants (côté serveur).
- calcul_montants**Helper** : Méthodes d'accès pour l'objet calcul_montants.
- calcul_montants**Holder** : Implante les passages de paramètres pour calcul_montants.

Etape 3 : *Classe java de l'objet « servant » (la classe qui fait le calcul)* :

```
package calcul;

// (importations ici, si cette classe est en tête du paquetage)

class Calcul_Servant extends calcul_montantsPOA {

        public Calcul_Servant() { }

        public double calcul_ttc (double mt_ht, double taux) {
                return mt_ht * (1 + taux/100);
        }
}
```

Etape 4 : *Classe du serveur* :

```
package calcul;

// Pour utiliser le service de nommage :
import org.omg.CosNaming.*;

// inclure le paquetage des exceptions pouvant être déclenchées
// par le service de nommage :
import org.omg.CosNaming.NamingContextPackage.*;

// Pour manipuler les objets CORBA :
import org.omg.CORBA.*;

// Classes nécessaires pour référencer le POA ( >= Corba 2.2 / J2SE 1.4) :
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;              // interface.

// Propriétés pour initialiser l'ORB :
import java.util.Properties;

public class Serveur {

public Serveur() { }

    public static void main(String args[]) {
        try {
                // créer et initialiser l'ORB qui intègre le service de noms :
                ORB orb=ORB.init (args, null);

                // obtenir la référence de rootpoa & activer le POAManager :
                POA rootpoa =
                        POAHelper.narrow (orb.resolve_initial_references("RootPOA"));
                rootpoa.the_POAManager().activate();

                // créer l'objet servant :
                Calcul_Servant calc= new Calcul_Servant ();

                // obtenir la référence CORBA du servant :
                org.omg.CORBA.Object ref = rootpoa.servant_to_reference (calc);
                calcul_montants href = calcul_montantsHelper.narrow (ref);

                // obtenir la référence du contexte de nommage :
                org.omg.CORBA.Object objRef =
                        orb.resolve_initial_references("NameService");

                // Utiliser NamingContextExt, qui est Interopérable :
                NamingContextExt ncRef = NamingContextExtHelper.narrow (objRef);
                // enregistrer le servant dans le service de nommage :
                String name = "calcul_ttc";          // nom arbitraire.
```

```
                NameComponent path [] = ncRef.to_name ( name );
                ncRef.rebind (path, href);

                System.out.println("Le Serveur à votre écoute…");

                // attendre les appels des clients…
                orb.run ();
        }
        catch(Exception exc) {
                System.err.println ("Erreur : "+exc);
                exc.printStackTrace (System.out);
        }
  }
} // fin du serveur.
```

Etape 5 : *Classe du client* :

```
package calcul;

import org.omg.CosNaming.*;      // inclure le service de nommage
import org.omg.CORBA.*;          // manipuler des objets CORBA
import org.omg.CosNaming.NamingContextPackage.*;

public class Client {

  public Client() { }

  public static void main (String args[]) {
        try {
                double mt_ht;
                double taux;
                double mt_ttc;
                mt_ht = Double.valueOf (args[0]);
                taux = Double.valueOf (args[1]);

                // créer et initialiser l'ORB
                ORB orb = ORB.init (args, null);

                // obtenir une référence au service de nommage
                org.omg.CORBA.Object objRef =
                        orb.resolve_initial_references ("NameService");

                // Utiliser NamingContextExt (au lieu de NamingContext), car
                //interopérable :

                NamingContextExt ncRef =
```

```
                    NamingContextExtHelper.narrow(objRef);

                // Demander la référence de l'objet au service de noms :

                String nom = "calcul_ttc";          // le nom “public” du service.
                calcul_montants calcul_ttc = calcul_montantsHelper.narrow
                    (ncRef.resolve_str(nom));

                // Faire appel a l'objet serveur et imprimer les résultats :

                mt_ttc = calcul_ttc.calcul_ttc (mt_ht, taux);

                System.out.println ("le montant ttc "+ mt_ttc);
        }
        catch (Exception exc) {
                System.out.println("Erreur : "+exc);
                exc.printStackTrace(System.out);
        }
  } // fin du main
}
```

Etape 6 : *Commandes permettant d’activer l'ORB sur la machine serveur, de lancer le serveur, et d’exécuter le client* :

start orbd -ORBInitialPort 1500
java calcul.Serveur -ORBInitialPort 1500
java calcul.Client ***100 20*** –ORBInitialHost localhost –ORBInitialPort 1500
.