



Systeme Distribue avec CORBA 2

Yassamine Seladji

yassamine.seladji@gmail.com

17 mars 2020

L'architecture de CORBA

L'architecture **CORBA** se compose essentiellement :

- ▶ du bus logiciel ORB.
- ▶ des souches : stub et squelette.
- ▶ du POA (Portable Object Adapter).
- ▶ de l'interface IDL.
- ▶ des IOR (Interoperable Object Reference).
- ▶ Des services.

La mise en place d'une application CORBA

- ▶ La définition du contrat IDL.
- ▶ La pré-compilation du contrat IDL.
- ▶ L'implémentation du serveur.
- ▶ L'implémentation du client.
- ▶ L'exécution répartie de l'application.

La mise en place d'une application CORBA

- ▶ **La définition du contrat IDL.**
- ▶ La pré-compilation du contrat IDL.
- ▶ L'implémentation du serveur.
- ▶ L'implémentation du client.
- ▶ L'exécution répartie de l'application.

La définition du contrat IDL.

- ▶ L'interface IDL est l'interface de communication client/serveur.
- ▶ L'IDL contient la signature des méthodes de l'objet distant.
- ▶ Une interface par objet distant.
- ▶ L'interface IDL est un fichier avec l'extension **.idl**.

La définition du contrat IDL.

- ▶ L'interface IDL est l'interface de communication client/serveur.
- ▶ L'IDL contient la signature des méthodes de l'objet distant.
- ▶ Une interface par objet distant.
- ▶ L'interface IDL est un fichier avec l'extension **.idl**.

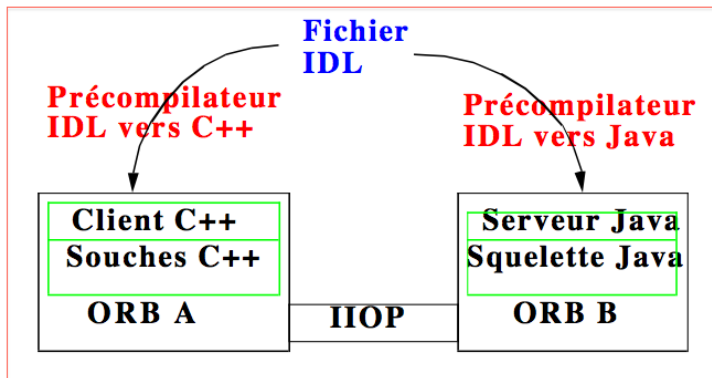
exemple : bourse.idl

```
module bourse {  
    interface Bourse {  
        double obtenirPrix (in String symbole);  
    };  
};
```

La mise en place d'une application CORBA

- ▶ La définition du contrat IDL.
- ▶ La pré-compilation du contrat IDL.
- ▶ L'implémentation du serveur.
- ▶ L'implémentation du client.
- ▶ L'exécution répartie de l'application.

La pré-compilation du contrat IDL.



La pré-compilation du contrat IDL.

Le précompilateur IDL vers Java : **idlj**

Côté Client :

- ▶ `idlj -fclient fichier.idl.`
- ▶ Le pré-compilateur génère :
 - ▶ Le fichier stub.
 - ▶ L'interface Java.
 - ▶ Le fichier Helper : Contient les méthodes qui permettent de faire le cast de la référence de l'objet CORBA en son type Java associé.
 - ▶ Le fichier Holder : Contient les opérations qui permettent de manipuler les paramètres **out** et **inout**.

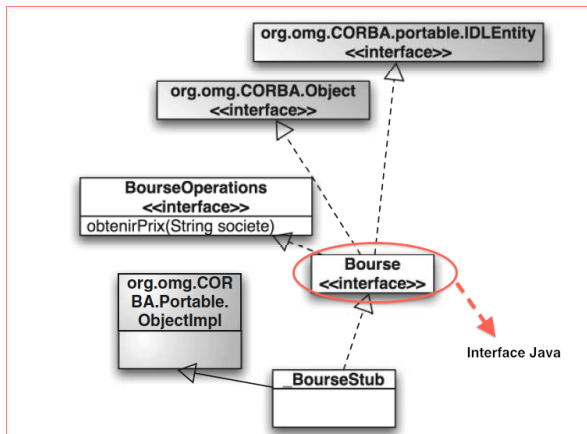
La pré-compilation du contrat IDL.

```
idlj -fclient bourse.idl.
```

- BourseOperations
- BourseHolder
- BourseHelper.
- Bourse.java
- _BourseStub.java

La pré-compilation du contrat IDL.

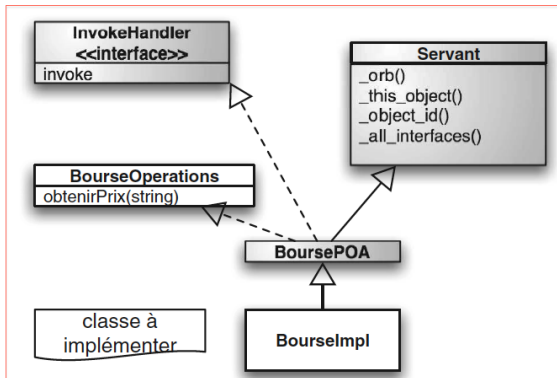
Hiérarchie des interfaces CORBA pour le client.



La pré-compilation du contrat IDL.

Côté serveur :

- ▶ idlj -fserver bourse.idl.
- ▶ Le pré-compilateur génère :
 - ▶ Le fichier BoursePOA.



La mise en place d'une application CORBA

- ▶ La définition du contrat IDL.
- ▶ La pré-compilation du contrat IDL.
- ▶ **L'implémentation du serveur.**
- ▶ L'implémentation du client.
- ▶ L'exécution répartie de l'application.

L'implémentation du serveur.

- ▶ L'implémentation de l'objet distant.
- ▶ L'implémenter d'une classe qui hérite de **BoursePOA**.
- ▶ Aucun constructeur spécial n'est requis.

L'implémentation du serveur.

- ▶ L'implémentation de l'objet distant.
- ▶ L'implémenter d'une classe qui hérite de **BoursePOA**.
- ▶ Aucun constructeur spécial n'est requis.

```
package bourse;

public class BourseImpl extends BoursePOA{

    public double obtenirPrix(String symbole) {
        return 45.0 ;
    }

}
```

L'implémentation du serveur.

L'écriture du code du serveur.

- 1 Initialisation de l'ORB.
- 2 La création de l'objet.
- 3 L'activation du POAManager : afin d'obtenir une référence.
- 4 L'obtention et l'enregistrement de la référence.

L'implémentation du serveur.

L'écriture du code du serveur.

1 Initialisation de l'ORB.

```
Properties props = new Properties();  
props.put("org.omg.CORBA.ORBInitialPort", "9999");  
props.put("org.omg.CORBA.ORBInitialHost", "localhost");  
ORB orb = ORB.init(args,props);
```

2 La création de l'objet.

3 L'activation du POAManager : afin d'obtenir une référence.

4 L'obtention et l'enregistrement de la référence.

L'implémentation du serveur.

L'écriture du code du serveur.

- 1 Initialisation de l'ORB.
- 2 La création de l'objet.

```
BourseImpl bourse = new BourseImpl();
```

- 3 L'activation du POAManager : afin d'obtenir une référence.
- 4 L'obtention et l'enregistrement de la référence.

L'implémentation du serveur.

L'écriture du code du serveur.

- 1 Initialisation de l'ORB.
- 2 La création de l'objet.
- 3 L'activation du POAManager : afin d'obtenir une référence.

```
POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPoa"));  
rootpoa.the_POAManager().activate();
```

- 4 L'obtention et l'enregistrement de la référence.

L'implémentation du serveur.

L'écriture du code du serveur.

- 1 Initialisation de l'ORB.
- 2 La création de l'objet.
- 3 L'activation du POAManager : afin d'obtenir une référence.
- 4 L'obtention de la référence.

```
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(bourse);  
Bourse href = BourseHelper.narrow(ref);
```

L'implémentation du serveur.

L'écriture du code du serveur.

- 1 Initialisation de l'ORB.
- 2 La création de l'objet.
- 3 L'activation du POAManager : afin d'obtenir une référence.
- 4 L'obtention de la référence.

```
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(bourse);  
Bourse href = BourseHelper.narrow(ref);
```

- 5 L'enregistrement de la référence.

```
OutputStream file = new FileOutputStream("bourse.ior");  
DataOutputStream out = new DataOutputStream(file);  
String ior = orb.object_to_string(href);  
out.writeBytes(ior);  
out.close();
```

L'implémentation du serveur.

L'écriture du code du serveur.

```
public class BourseServer {  
  
    public static void main(String[] args) throws Exception{  
  
        Properties props = new Properties();  
        props.put("org.omg.CORBA.ORBInitialPort", "9999");  
        props.put("org.omg.CORBA.ORBInitialHost", "localhost");  
        ORB orb = ORB.init(args, props);  
  
        BourseImpl bourse = new BourseImpl();  
  
        org.omg.CORBA.Object a = orb.resolve_initial_references("RootPOA");  
        POA rootpoa = POAHelper.narrow(a);  
        rootpoa.the_POAManager().activate();  
  
        org.omg.CORBA.Object ref = rootpoa.servant_to_reference(bourse);  
        Bourse href = BourseHelper.narrow(ref);  
  
        OutputStream file = new FileOutputStream("bourse.ior");  
        DataOutputStream out = new DataOutputStream(file);  
        String ior = orb.object_to_string(href);  
        out.writeBytes(ior);  
        out.close();  
        System.out.println("serveur prêt");  
        orb.run();  
    }  
}
```

Le service d'annuaire.

Coté serveur :

- ▶ Lancer le service d'annuaire de Java **orbd** :
orbd -ORBInitialPort 1050
- ▶ Lancer l'ORB :

```
Properties props = new Properties();  
props.put("org.omg.CORBA.ORBInitialPort", "1050");  
props.put("org.omg.CORBA.ORBInitialHost", "localhost");  
ORB orb = ORB.init(args, props);
```

- ▶ Obtenir une référence sur l'annuaire :
- ▶ Créer la référence sur l'objet :
- ▶ Lier le nom à la référence de l'objet :

Le service d'annuaire.

Coté serveur :

- ▶ Lancer le service d'annuaire de Java **orbd** :
orbd -ORBInitialPort 1050

- ▶ Lancer l'ORB :

- ▶ Obtenir une référence sur l'annuaire :

```
NamingContextExt annuaireDistant =  
    NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
```

- ▶ Créer la référence sur l'objet :
- ▶ Lier le nom à la référence de l'objet :

Le service d'annuaire.

Coté serveur :

- ▶ Lancer le service d'annuaire de Java **orbd** :
orbd -ORBInitialPort 1050
- ▶ Lancer l'ORB :
- ▶ Obtenir une référence sur l'annuaire :
- ▶ Créer la référence sur l'objet :

```
BourseImpl bourse = new BourseImpl();  
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(bourse);  
Bourse href = BourseHelper.narrow(ref);
```

- ▶ Lier le nom à la référence de l'objet :

Le service d'annuaire.

Coté serveur :

- ▶ Lancer le service d'annuaire de Java **orbd** :
orbd -ORBInitialPort 1050
- ▶ Lancer l'ORB :
- ▶ Obtenir une référence sur l'annuaire :
- ▶ Créer la référence sur l'objet :
- ▶ Lier le nom à la référence de l'objet :

```
NameComponent[] nomLogic = annuaireDistant.to_name("bourse dz");  
annuaireDistant.rebind(nomLogic, href);
```

La mise en place d'une application CORBA

- ▶ La définition du contrat IDL.
- ▶ La pré-compilation du contrat IDL.
- ▶ L'implémentation du serveur.
- ▶ **L'implémentation du client.**
- ▶ L'exécution répartie de l'application.

L'implémentation du client.

L'écriture du code du client.

- 1 Création et initialisation de l'ORB.
- 2 Récupération de la référence IOR dans le fichier.
- 3 Récupérer l'objet et effectuer l'appel distant.

L'implémentation du client.

L'écriture du code du client.

- 1 Création et initialisation de l'ORB.

```
ORB orb = ORB.init(args, null);
```

- 2 Récupération de la référence IOR dans le fichier.
- 3 Récupérer l'objet et effectuer l'appel distant.

L'implémentation du client.

L'écriture du code du client.

- 1 Création et initialisation de l'ORB.
- 2 Récupération de la référence IOR dans le fichier.

```
InputStream file = new FileInputStream ("bourse.ior");  
BufferedReader reader = new BufferedReader(new InputStreamReader(file));  
String ior = reader.readLine();
```

- 3 Récupérer l'objet et effectuer l'appel distant.

L'implémentation du client.

L'écriture du code du client.

- 1 Création et initialisation de l'ORB.
- 2 Récupération de la référence IOR dans le fichier.
- 3 Récupérer l'objet et effectuer l'appel distant.

```
Bourse bourse = (Bourse) orb.string_to_object(ior);  
System.out.println(bourse.obtenirPrix("orange"));
```

L'implémentation du client.

L'écriture du code du client.

```
public class BourseClient {  
  
    public static void main(String[] args) throws Exception {  
  
        ORB orb = ORB.init(args, null);  
  
        InputStream file = new FileInputStream ("bourse.ior");  
        BufferedReader reader = new BufferedReader(new InputStreamReader(file));  
        String ior = reader.readLine();  
  
        Bourse bourse = (Bourse) orb.string_to_object(ior);  
        System.out.println(bourse.obtenirPrix("orange"));  
    }  
}
```


Le service d'annuaire.

Coté client :

- ▶ La récupération de l'annuaire :

```
Object refAnnuaire = orb.resolve_initial_references("NameService");  
NamingContextExt annuaireDistant = NamingContextExtHelper.narrow(refAnnuaire);
```

Le service d'annuaire.

Coté client :

- ▶ La récupération de l'annuaire :

```
Object refAnnuaire = orb.resolve_initial_references("NameService");  
NamingContextExt annuaireDistant = NamingContextExtHelper.narrow(refAnnuaire);
```

- ▶ La récupération de la référence de l'objet distant :

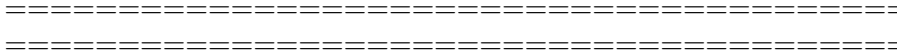
```
Bourse objetBourse = BourseHelper.narrow(annuaireDistant.resolve_str("bourse dz"));  
System.out.println(objetBourse.obtenirPrix("orange"));
```

La mise en place d'une application CORBA

- ▶ La définition du contrat IDL.
- ▶ La pré-compilation du contrat IDL.
- ▶ L'implémentation du serveur.
- ▶ L'implémentation du client.
- ▶ L'exécution répartie de l'application.

L'exécution répartie de l'application.

- ▶ l'exécution de l'application dépend des langages utilisés.
- ▶ Le lancement du serveur : Compilation et exécution du programme serveur.
- ▶ Le lancement du client : Compilation et exécution du programme client.



Mettre en place une application Hello world ! .

Ecrire une application répartie en CORBA tel que :

- ▶ le serveur implémente un objet distant **Hello**.
- ▶ Le service offert par le serveur est la méthode **hello()** qui renvoie le message Hello suivi du numéro du client. Exemple "Hello client 1".
- ▶ Le client fait un appel simple à la méthode **hello()**.

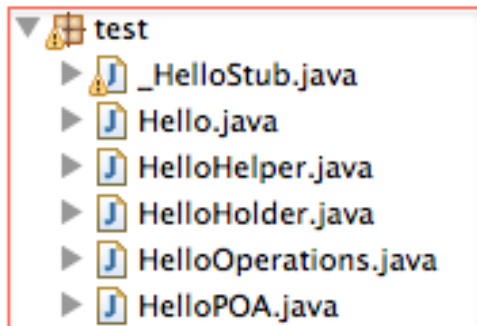
Mettre en place une application Hello world ! .

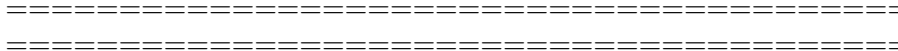
1 L'écriture de l'interface IDL.

```
module test {  
    interface Hello{  
        void hello();  
    };  
};
```

Mettre en place une application Hello world ! .

- 1 L'écriture de l'interface IDL.
- 2 La pré-compilation de l'interface.





Mettre en place une application Hello world ! .

- 1 L'écriture de l'interface IDL.
- 2 La pré-compilation de l'interface.
- 3 L'implémentation de l'objet distant côté serveur :

```
public class HelloImpl extends HelloPOA {  
  
    private int compteur;  
  
    public HelloImpl () {  
        compteur = 0;  
    }  
    public void hello() {  
        compteur++;  
        System.out.println("Hello! "+compteur);  
    }  
}
```

Mettre en place une application Hello world ! .

4 l'implémentation du serveur.

```
public class HelloServer {
    public static void main(String[] args) throws Exception {

        Properties props = new Properties();
        props.put("org.omg.CORBA.ORBInitialPort", "9999");
        props.put("org.omg.CORBA.ORBInitialHost", "localhost");
        ORB orb = ORB.init(args,props);

        org.omg.CORBA.Object a = orb.resolve_initial_references("RootPOA");
        POA rootpoa = POAHelper.narrow(a);
        rootpoa.the_POAManager().activate();
        System.out.println("serveur prêt");

        HelloImpl impl = new HelloImpl();
        org.omg.CORBA.Object ref = rootpoa.servant_to_reference(impl);
        Hello href = HelloHelper.narrow(ref);

        String ior = orb.object_to_string(href);
        System.out.println(ior);

        PrintWriter file = new PrintWriter("ior.txt");
        file.println(ior);
        file.close();

        orb.run();
    }
}
```

Mettre en place une application Hello world ! .

- 4 l'implémentation du serveur.
- 5 l'implémentation du client.

```
public class HelloClient {  
  
    public static void main(String[] args) throws Exception {  
        ORB orb = ORB.init(args, null);  
  
        BufferedReader br = new BufferedReader(new FileReader("ior.txt"));  
        String ior = br.readLine();  
        br.close();  
  
        org.omg.CORBA.Object obj = orb.string_to_object(ior);  
  
        Hello h = HelloHelper.narrow(obj);  
        h.hello();  
  
    }  
}
```