Java et CORBA

- Intégré dans le JDK1.2 … : Java IDL
- … hormis `idltojava` à chercher au java developer connection (~300Ko) Solaris, Win32
- On utilise les paquetages `org.omg.CORBA` et `org.omg.COSNaming` du JDK1.2

Architecture CORBA

- Travaux de l'OMG (Object Management Group).
- CORBA (Common Object Request Broker Architecture) = une architecture, des interfaces, des protocoles, des services pour faire communiquer des objets répartis sur diverses machines.

Architecture CORBA (suite)

- Caractéristiques importantes : les objets peuvent avoir été écrits dans des langages de programmation distincts (essentiellement C++, Ada95, Smalltalk, COBOL, C et évidemment Java).
- CORBA = des spécifications, rien que des spécifications mais toutes les spécifications

ORB = Object Request Broker

- ORB = ?
- ORB = Bus logiciel = ?
- ORB = pseudo-objet de bootstrap donnant accès à la mécanique (l'univers) CORBA (Orfali)
- ORB = un protocole orienté objet
- Parmi ces services : le Naming Service

naming Service = Service de nommage

- Les objets sont décrits par des noms.
- Noms rangés par arborescence

Initialisation : code Java

- "ORB initialization is a bootstrap call into the CORBA world."

```
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;

class LeClientOuLeServeur {
   public static void main (String args[]){

    Properties props = new Properties ();
    props.put("org.omg.CORBA.ORBInitialPort", "900");
    props.put("org.omg.CORBA.ORBInitialHost",
"localhost");
    ORB orb = ORB.init( args, props);

    …
}
```

- Est équivalent à :

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

De l’ORB au Naming Service

- Les méthodes `ORB.init(…)` du JDK lance un service de nommage NamingService (qui, par défaut écoute localement sur le port 900 cf.diapo précédente)
- On récupère ce service de nommage par :

```
ORB orb = ORB.init(...);
org.omg.CORBA.Object refNServ =
      orb.resolve_initial_references("NameService");
NamingContext nc =
      NamingContextHelper.narrow(refNServ);
```

IDL = Interface Definition Language

- méta langage
- langage de spécification (de déclarations) CORBA
- Syntaxe proche de C++
- Passage des paramètres en `in`, `out`, `inout`.
- Module ~ paquetage Java ~ espace de nommage

IDL : un exemple

- code Java

```
// Count.idl
module Counter {
    interface Count {
        attribute long sum;
        long increment();
 };
};
```

De l’idl vers Java

- génération :
  - d’une interface Java
  - d’une classe Helper
  - d’une classe Holder
- et très souvent (`idltojava`)
  - d’un stub
  - d’un skeleton

Fichiers générés

- les interfaces Java = traduction en Java des interfaces idl = le contrat que doit implanter le serveur = ce que le client peut demander au serveur
- classe Helper = {méthodes} pour :
  - convertir des objets CORBA en type Java (Cast) : `narrow()`
  - lire/écrire des objets CORBA : `write()`, `read()`

Fichiers générés (suite)

- classe Holder : implémente les passages out et inout (inexistant en Java natif).

Stub avec `idltojava`

- Stub de la forme _*NomInterface*`Stub`
  - = le représentant coté client du serveur distant.
  - convertit les requêtes du client en appel au serveur distant
  - attend la réponse
  - la renvoie au client.

Skeleton avec `idltojava`

- Skeleton de la forme _*NomInterface*`ImplBase`
  - attend les requêtes
  - déploie les arguments (marshalling)
  - demande l'exécution au serveur
  - envoie les réponses.
- sert de classe de base pour le serveur !!

Finalement, pour le(s) programmeur(s), il faut :

- créer le fichier `idl`
- lancer `idltojava` sur ce fichier
- implanter le serveur (classe `...Impl`) en le dérivant du squelette `_...ImplBase`.
  - Serveur (français) = servant (US)
- écrire le lanceur du serveur
  - lanceur serveur (français) = server (US)
- écrire le client

A l'exécution :

- lancer le service de nommage par `tnameserv &`
- exécuter le lanceur de serveur
- exécuter le client

Démo : calculatrice CORBA : coté serveur

- l'interface IDL : `Calculette.idl`
- génération du code Java : `idltojava Calculette.idl`

  => génération de l'interface Java, du stub et du skeleton
- implémentation de l'interface Java (i.e. le Serveur) : `CalculetteImpl.java`
  - dérive du Skeleton

Démo : calculatrice CORBA : coté serveur

- implémentation du lanceur du serveur : `LanceCalculette.java`
  - initialisation de l ’ORB et recherche du NamingService
  - enregistrement (rebind()) du serveur auprès de ce NamingService
  - attente des clients

Démo : calculatrice CORBA : coté client

- Appel dynamique par DII (Dynamic Invocation Interface)
- client application indépendante : `DIIClientCalculette.java`

Démo : calculatrice CORBA : compilation

- `javac LanceCalculette.java`
- `javac DIIClientCalculette.java`

Démo : calculatrice CORBA : exécution

- `tnameserv &`
- `java LanceCalculette`
- `java DIIClientCalculette`

Calculette : applet

- Ecriture de l ’applet `DIIAppletClientCalculette.java`
- lancement par `appletviewer DIIAppletClientCalculette.html`

Bibliographie

- Un excellent tutorial d'introduction à Java IDL : `http://java.sun.com/products/jdk/1.3/docs/guide/idl/jidlExample.html`
- Technologie Java IDL à Javasoft : `http://java.sun.com/products/jdk/idl/index.html`
- Le site de l'OMG : `http://www.omg.org`

Bibliographie (suite)

- Java Enterprise in a nutshell. David Flanagan, Jim Farley, William Crawford & Kris Magnusson; ed O'Reilly. ISBN 1-56592-483-5. Chapitre Java IDL
- Client/Server Programming with Java and CORBA second edition. Robert Orfali, Dan Harkey; ed Wiley. ISBN 0-471-24578-X

Bibliographie (suite)

- Au coeur de CORBA avec Java. Jérôme Daniel, ed Vuibert. ISBN 2-7117-8659-5
- Java Programming with CORBA. Andreas Vogel, Keith Duddy, ed Wiley. ISBN 0-471-24765-0