Corba CD Server mit JAVA

Constantin Wolber

Benötigte Tools

- JDK 1.3 oder höher
- JacORB
- Optional Ant (Einfacher fürs konfigurieren von JacORB)

Installation von JacOrb

JacOrb kann man sich als Source oder als vorkompilierte Version unter http://www.jacorb.org runterladen

Nach dem entpacken kann man im JacORB Verzeichnis einfach ant aufrufen. Ant kompiliert dann diverse Sourcen nochmals und passt die Batch bzw sh Scripte an.

Unter JACORB_HOME/bin liegen dann die fertig konfigurierten scripte für den ORB. Insofern bietet es sich an den Path um dieses Verzeichnis zu erweitern.

Wichtige ORB Tools

idl: Der Idl Compiler
ns: Der Corbanameserver
dior: Tool zum ausgeben einer lesbaren IOR
fixior: Tool zum verändern einer IOR (relevant wenn hostname != offizieller Name)

JacORB verwenden

Um den JacORB dann als ORB Ansteller des Standard SunOrb zu verwenden müssen noch folgende Anpassungen gemacht werden.

jacorb.properties.template nach ins HOME Verzeichnis kopieren und in jacorb.properties umbenennen.

In dieser Datei muss nur eine Änderung gemacht werden und muss

```
ORBInitRef.NameService=http://www.x.y.z/~user/NS_Ref
```

entsprechend dem Ort wo die .ref Datei gespeichert wird geändert werden. Ich greife direkt auf den Nameservice zu mittels:

```
ORBInitRef.NameService=corbaloc::127.0.0.1:9101/NameService
```

JacOrb verwenden II

Als letztes muss dann jede CORBA Anwendung noch mit den Parametern:

-Djava.endorsed.dirs=JACORB_HOME\lib
-Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton

gestartet werden und schon benutzt man nicht mehr den Standard ORB sondern den JacORB

ORB initialisieren Server

Der Server wird ähnlich wie in C++ folgendermaßen initialisiert:

```
ORB orb = org.omg.CORBA.ORB.init(args, null);
POA rootPOA = POAHelper.narrow(orb.resolve_initial_references
                ("RootPOA"));
POAManager poaMgr  = rootPOA.the_POAManager();
org.omg.CORBA.Policy [] policies =
{
    rootPOA.create_id_assignment_policy
        (IdAssignmentPolicyValue.USER_ID),
    rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT)
};

POA factoryPOA = rootPOA.create_POA( "CdServer", poaMgr,
    policies);
CdServerImpl factoryServant = new CdServerImpl(factoryPOA);
factoryPOA.activate_object_with_id( new String("CdServer").
    getBytes(), factoryServant );
```

Activator zuweisen

Hier weicht das ganze anscheinend von der C++ Version ab. Dieser Activator war unter C++ nicht nötig.

```
factoryPOA.the_activator( new CdAdapterActivatorImpl( orb ) );
```

Die Klasse CdAdapterActivatorImpl extendiert _AdapterActivatorLocalBase

In der Methode unknown_adapter(POA parent, String name) werden die Policies für den servantManager erstellt und dann mittels

```
POA newPOA = parent.create_POA(name, parent.the_POAManager(),
                policies);
set_servant_manager(new CDServantActivatorImpl( orb ) );
```

eingebunden und der servantManager wird gesetzt

ServantManager

Die Servant Manager Klasse implementiert die Methoden etherealize und incarnate.

Hier gibt es keine Änderungen gegenüber C++

Cd Server Implementierung

Hier werden dann die Methoden create und shutdown implementiert.

Die create Methode kann man fast genau wie in C++ implementieren, nur die shutdown Methode weist einige Besonderheiten auf.

Java lässt es nicht zu, das man einen Thread aus sich selbst heraus beendet. Insofern muss ein neuer Thread, welcher auch in einer neuen ThreadGroup läuft erstellt werden. Dieser kann dann den ORB mittels shutdown beenden

Der Client

Die Programmierung des Clients ist sehr einfach. Um an ein Cd Object heranzukommen sind nur folgende Zeilen nötig:

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
NamingContextExt nc = NamingContextExtHelper.narrow
          (orb.resolve_initial_references("NameService"));
CdServer factory = CdServerHelper.narrow(nc.resolve
          (nc.to_name("CdServer.service") ));
Cd cd = factory.create("CD1");
```

Fazit

- Einige Probleme durch Portierung
- Wenig Doku zu CORBA und Java
- dürftige Doku zu JacORB

+ Methoden des CORBA Pakets und Java genau wie unter C++

Ich habe am Anfang als ich mich damit auseinandergesetzt die Methode objectId_to_String vermisst, allerdings gibt es ja in Java die Klasse String und einer der Konstruktoren lautet String(byte[] bytes). Damit erreicht man genau das selbe.