TECHNISCHE UNIVERSITÄT ILMENAU
Fakultät für Informatik und Automatisierung
Institut für Praktische Informatik und Medieninformatik
Fachgebiet Telematik

Seminar „Verteilte Anwendungen“

Anleitung, Aufgabenstellung und Protokoll zur Rechnerübung

Common Object Request Broker Architecture (CORBA)

19. Januar 2004

1 Einführung

Die Aufgabe des Seminars besteht darin, die Kenntnisse aus der Vorlesung zur Programmierung CORBA-konformer Applikationen zu vertiefen. In der Rechnerübung kommt der ORB des Java Development Kit (JDK) von Sun Microsystems zum Einsatz. Sun Microsystems bezeichnet die CORBA-basierte Umgebung des JDK als *Java IDL*. Weitere Informationen zum Thema Java IDL findet man unter http://java.sun.com/products/jdk/idl/index.html.

Die Entwicklung verteilter Applikationen mit Java IDL erfolgt in 4 Schritten:

1. Schreiben der IDL-Interface-Beschreibung
2. Generierung des korrespondierenden Java Interface, der Stubs und Skeletons mit dem IDL-Compiler `idlj`
3. Entwicklung der Client/Server-Applikation und deren Übersetzung mit dem Java-Compiler `javac`
4. Start der Applikation, Server und Client, gegebenenfalls inklusive Name Server, `tnameserv`

2 Aufgabenstellungen

2.1 Hello-Anwendung

Im folgenden werden die einzelnen Schritte zur Erstellung von verteilten Anwendungen mit Hilfe von Java IDL an dem einfachen Beispiel der Hello-Anwendung behandelt.

Verzeichnis: `~/Corba/hello`

Dateien:
```
~/Corba/hello/Hello.idl
~/Corba/hello/HelloImpl.java
~/Corba/hello/HelloServer.java
~/Corba/hello/HelloClient.java
```

2.1.1 IDL-Beschreibung

Entwerfen Sie ein Interface im Verzeichnis `Corba/hello` , Datei `~/Corba/hello/Hello.idl` für die einfache Hello-Anwendung, bei der ein String an den Server geschickt, dort komplettiert und wieder zurückgeschickt wird! Das Interface soll Bestandteil des Moduls `HelloApp` sein.

2.1.2 Generieren des Java Interface, der Stubs und Skeletons

Generieren Sie das korrespondierende Java Interface mit Hilfe des IDL-Compiler des JDK!

```
% cd ~/Corba/hello
% idlj -fall Hello.idl
```

Der IDL-Compiler generiert neben dem Java Interface `Hello.java` die Dateien `_HelloImplBase.java` für die Server-Seite, `HelloHelper.java`, `HelloHolder.java`, `_HelloStub.java` für die Client-Seite sowie `HelloOperations.java` für Client und Server.

`_*ImplBase.java`	Server Skeleton
`_*Stub.java`	Client Stub
`*Helper.java`	zusätzliche Funktionen, wie zum Beispiel `narrow()`
`*Holder.java`	Behandlung von IDL-Definitionen, die sich nicht so leicht auf Java-Semantik abbilden lassen
`*Operations.java`	Bestandteil des korrespondierenden Java-Interface

2.1.3 Entwickeln und Übersetzen der Client/Server-Applikation

2.1.3.1 Implementierung des Interface

Die Java-Klasse `HelloImpl`, Datei `~/Corba/hello/HelloImpl.java`, implementiert das Interface! Diese Klasse enthält keinen Code zur Kommunikation mit einem ORB-Interface.

Ergänzen Sie die Implementierung der Operation `sayHelloTo()`!

2.1.3.2 Server-Implementierung

Der Server ist dafür verantwortlich, ein Objekt der Klasse `HelloImpl` zu starten und dieses über den Object Adapter beim ORB anzumelden. Bei Verfügbarkeit eines Name Service erfolgt hier auch das Eintragen des Hello-Dienstes unter einem bestimmten Namen.

Verschaffen Sie sich einen Überblick über den Inhalt der Datei `~/Corba/hello/HelloServer.java`! Welche Schritte sind notwendig, um den Service über den ORB bereitzustellen?

Übersetzen Sie den Hello-Server mit Hilfe von `javac`!

```
% cd ~/Corba/hello
% javac HelloImpl.java HelloServer.java
  HelloApp/*.java
```

2.1.3.3 Client-Implementierung

Der Hello-Client ruft den Dienst des Hello-Service über den ORB auf.

Verschaffen Sie sich einen Überblick über den Inhalt der Datei `~/Corba/hello/HelloClient.java`! Welche Schritte unternimmt der Client, um den Dienst über den ORB in Anspruch nehmen zu können?

Ergänzen Sie den Aufruf der Operation `sayHelloTo()` und die Ausgabe des Ergebnisses!

Übersetzen Sie den Hello-Client mit Hilfe von `javac`!

```
% javac HelloClient.java HelloApp/*.java
```


2.1.4 Test der verteilten Anwendung

Starten Sie den Name Service auf einer benachbarten Maschine, ihrem ***Server Host***!

```
% rlogin serverHost
<Paßwort>
% tnameserv -ORBInitialPort 83xx &
```

Setzen Sie ORBInitialPort auf: **8330 + numerischer Anteil Ihres Logins**!

Starten Sie den Hello-Server auf der entfernten Maschine, Ihrem ***Server Host***!

```
% cd ~/Corba/hello
% java HelloServer -ORBInitialHost serverHost
                   -ORBInitialPort 83xx
```

Starten Sie den Hello-Client auf Ihrer lokalen Maschine!

```
% cd ~/Corba/hello
% java HelloClient -ORBInitialHost serverHost
                   -ORBInitialPort 83xx
```

2.2 Arithmetik-Anwendung

Es werden zwei Vektoren mit 5 ganzen Zahlen addiert. Der Client stellt die zwei Vektoren zur Verfügung und sendet diese über das Netzwerk an den Server. Der Diensterbringer hat die Aufgabe, die Addition durchzuführen und den Ergebnisvektor an den Client zurück zu geben. Der Dienstbenutzer gibt das Ergebnis schließlich aus.

Implementieren Sie diese einfache Arithmetik-Anwendung als CORBA-konforme Applikation mit Hilfe von Java IDL! Orientieren Sie sich bei Ihrer Lösung an der Implementierung der Hello-Anwendung.

Verzeichnis: `~/Corba/arithmetic`

Dateien: `~/Corba/arithmetic/Arithmetic.idl`