

Vernetzte Systeme, Übungsstunde 7: Hinweise

Gabor Cselle, gabor@student.ethz.ch, 27. Mai 2005, HG F 26.3

Organisatorisches

- Umfrageresultate
- Alle Hinweisblätter verfügbar auf <http://www.gaborcselle.com/vs>
- Vorberechnung Serie 9, Nachberechnung Serien 5 und 6 (kurz!)

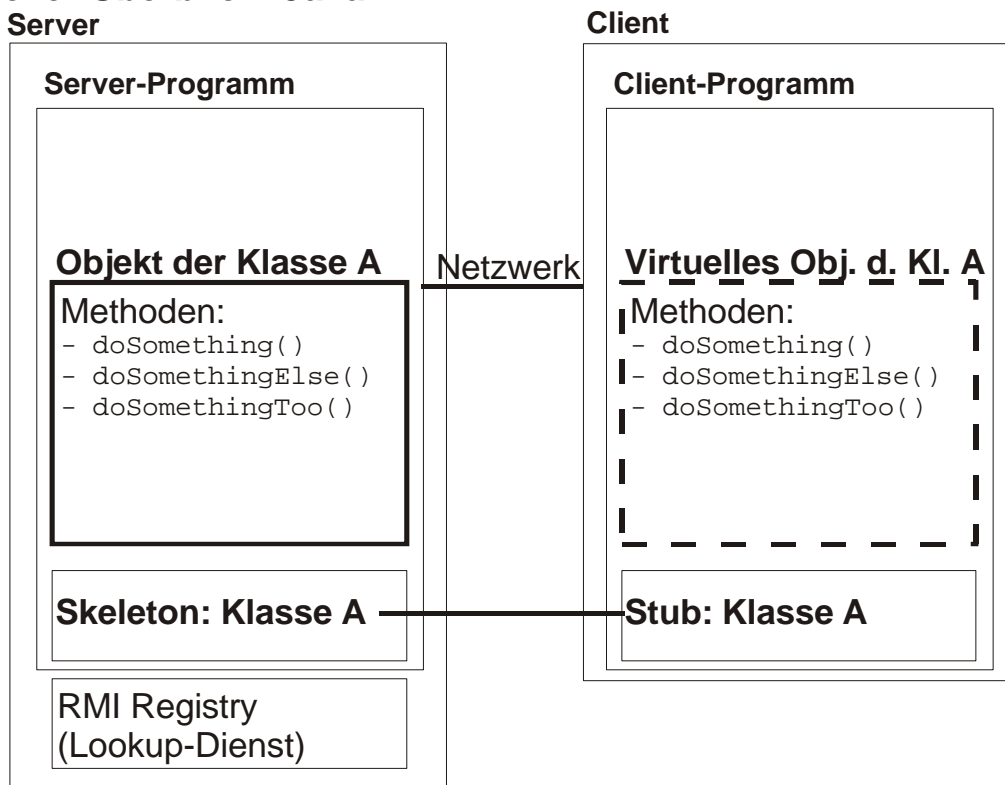
Interaktiv

Auf welcher Ebene des TCP/IP-Referenzmodells findet man RPC und RMI-Protokolle?

- A. RPC basiert immer auf HTTP.
- B. RPC ist auf der Application-Ebene (über TCP und UDP).
- C. RPC passiert direkt oberhalb der Network-Ebene (IP-Ebene).

Vorberechnung Übung 9

Konzeptueller Überblick: Java RMI



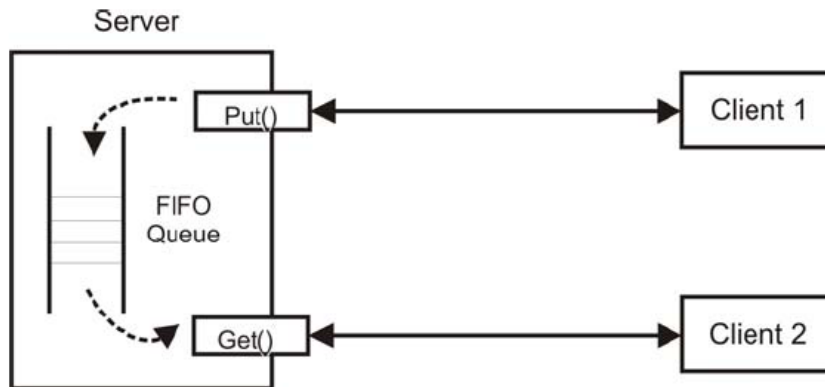
Zu beachten:

- Alle Methoden des Remote-Objektes, die von anderen aufgerufen werden sollten, müssen in einem Interface spezifiziert sein. Dieses Interface muss Remote extenden.
- Um den Skeleton und den Stub zu generieren, muss man den rmic-Compiler laufen lassen. (Zusätzlich zum Kompilieren des Java-Codes).
Beispiel: Aufruf für Klasse "MessagePoolImpl" in Package "server":

```
$ rmic server.MessagePoolImpl
```

- Server: RMI Registry muss laufen. Diesen startet man auf Windows mit "start rmiregistry" auf der Kommandozeile. Auf Linux mit "rmiregistry &".

Aufgabe in Übungsserie



Zu beachten:

- Das Aufgabenblatt wurde am Donnerstag nochmal geändert – schaut bitte, dass ihr die aktuellste Version habt.
- Bei Fehlern werden spezielle Exceptions geworfen, die aber alle Subklassen von `RemoteException` sind, z.B. `QueueEmptyException`, `QueueFullException`. Diese sind schon vordefiniert. Jedoch müsst diese Fehler abfangen und korrekt handhaben. (Siehe auch Anmerkungen zu Exceptions auf Vorlesungshomepage / Java-Tutorial.)

Beispiel von Aufgabenblatt:

```
public interface Calculator extends Remote{
    public int addOne(int i) throws RemoteException;
}

public class CalculatorImpl implements Calculator{
    public CalculatorImpl() throws RemoteException {
        UnicastRemoteObject.exportObject(this);
    }
    public int addOne(int i) throws RemoteException {
        return i+1;
    }
}

public class CalculatorServer {
    public static void main(String[] args){
        try{
            CalculatorImpl c = new CalculatorImpl();
            Naming.rebind("calc" , c);
            System.out.println("Calculator Server Ready!");
        }
        catch (...) {}
    }
}

public class CalculatorClient { (...)}
    try {
        Calculator cal = (Calculator ) Naming.lookup("rmi://server/calc");
        int output = cal.addOne(input);
        System.out.println(
            "The output of addOne(" + input + ")" + " is " + output);
    }
    catch (...) {}
}
```

Aufgabe für Euch:

- Definiert Interface für den Queue-Server
- Schreibt den Code für den MessageGetClient, um Referenz auf Server-Objekt zu erhalten.

Lösung:

Interface für Queue-Server (siehe auch Skeleton):

```
public interface MessagePool extends Remote{
    public void put(String msg) throws RemoteException;
    public String get() throws RemoteException;
}
```

Code für den MessageGetClient, um Referenz auf Server-Objekt zu erhalten:

```
MessagePool pool = (MessagePool ) Naming.lookup(args[0]);
```