RMI

- ❑ Remote Method Invocation: Fernaufe über Netzwerk
- ❑ Transparente Objektreferenzen auf Objekte auf anderen Rechnern
- ❑ Kopieren von Objekten (Übergabe "by value") auf andere Rechner
- ❑ Ähnlich wie unter Spring: Fernaufrufe mit Stubs
- ❑ remote-fähige Klasse muß Typ java.rmi.Remote implementieren
  (Nur Kennzeichnung: Typ "Remote" ist leer implementiert)
- ❑ Parameterübergabe:
  - by value: default
  - by reference: wenn Übergabe-Typ zu java.rmi.Remote kompatibel ist

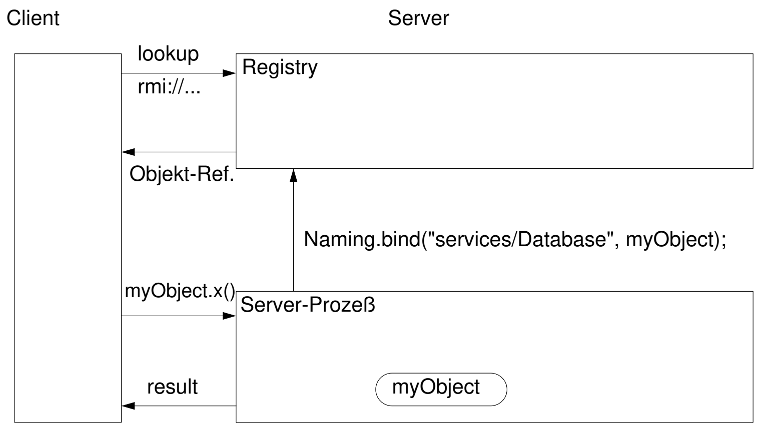
RMI

- ❑ Registry: kann Objekte mit Namen registrieren (Object Request Broker)
- ❑ Arbeitet nur lokal, Objekte über Rechnernamen der Registry und lokalen Pfad ansprechbar
- ❑ Namensschema: URL
- ❑ Beispieladresse für ein Objekt:

`rmi://oobp.informatik.uni-erlangen.de/services/Database`

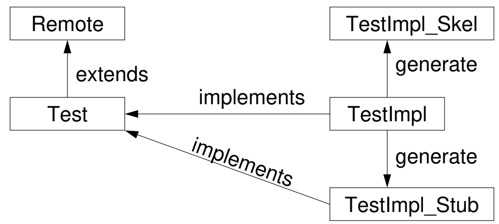
- ❑ Security-Einschränkung von netscape: WWW-Server und remote-Objekte müssen auf dem gleichen Rechner liegen.

RMI



RMI

- ❑ Wie unter Spring können nur Interfaces als Remote-Typ verwendet werden.
- ❑ Die Stubs werden aus den Implementationen des Interfaces erzeugt



RMI — Beispiel

❏ Interface:

```
public interface Test extends java.rmi.Remote {
    String call(String message) throws java.rmi.RemoteException;
}
```

❏ Server:

```
public class TestImpl extends UnicastRemoteServer implements Test
{
      public String call(String message) throws RemoteException
      {
            return  "Hello " + message;
      }
}

TestImpl t=new TestImpl();
Naming.bind("rmi://oobp.informatik.uni-erlangen.de/testit", t);
```

❏ Client:

```
Test t = (Test)java.rmi.Naming.lookup("rmi://oobp.informatik/testit");
String s = t.call("hello");
```


RMI — Object-Serialization

❏ Nötig zum Verschicken von Objekten: Objekt-Marshalling

❏ In Java: Call by value bei nicht-remote Objekten

❏ Object-Streams können Objekte in Bytestrom packen und aus Bytestrom extrahieren.

❏ Z.B. Abspeichern von Objekten:

```
FileOutputStream f = new FileOutputStream("/tmp/objects");
ObjectOutput s = new  ObjectOutputStream(f);
s.writeObject("Today");
s.writeObject(new Date());
...

FileInputStream in = new FileInputStream("/tmp/objects");
ObjectInputStream s = new ObjectInputStream(in);
String today = (String)s.readObject();
Date date = (Date)s.readObject();
```


RMI — Tabulator

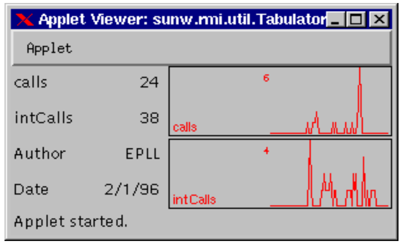
❏ Statistik-Information kann mit Tabulatoren angezeigt werden

❏ Ähnlich wie Spring

❏ Dazu vorhanden: Tabulator-Klasse

❏ Beispiel html-Seite:

```
<applet code=sunw.rmi.util.Tabulator width=300 height=120>
<param name=url value="rmi://oobp.informatik.uni-erlangen.de/myObj">
</applet>
```



RMI — Tabulator

❏ Deklaration: Kompatibilität zu Tabular

```
public class MyClass implements sunw.rmi.util.Tabular, ... {

      public TabularInfo[] rmiGetTabularInfo(boolean needTags) {
            TabularSupport ts = new TabularSupport(true);
            ts.graphDeltas("calls", calls);
            ts.graphDeltas("intCalls", intCalls);
            ts.show("Author", "EPLL");
            ts.show("Date", "2/1/96");
            return (ts.list());
      }

}
```