JAVA DAYS '99



Die Beschwörung der Öllampe oder Programmierung eines Jini Service

Andreas Dörr
Freier IT-Berater / Consultant

Sun microsystems

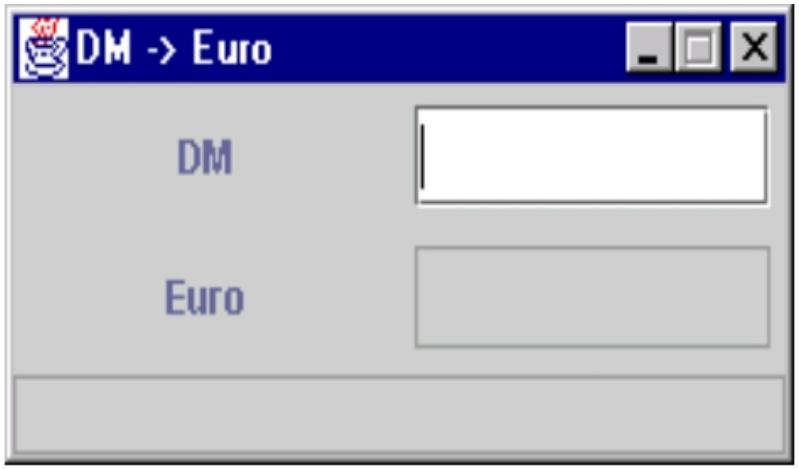
Konzepte

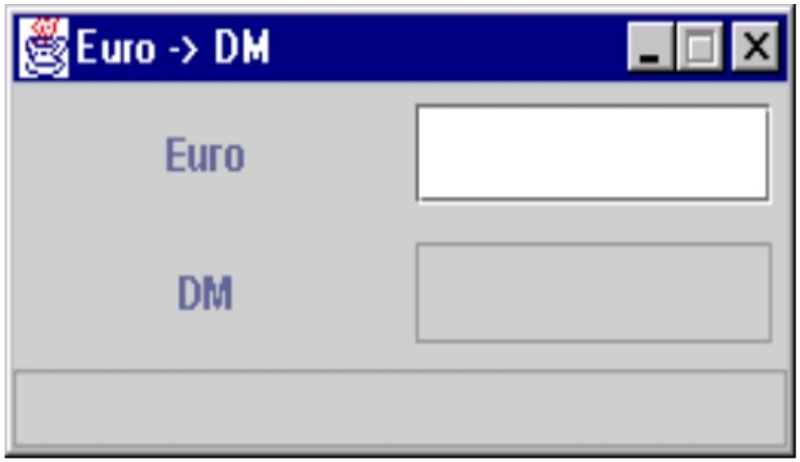


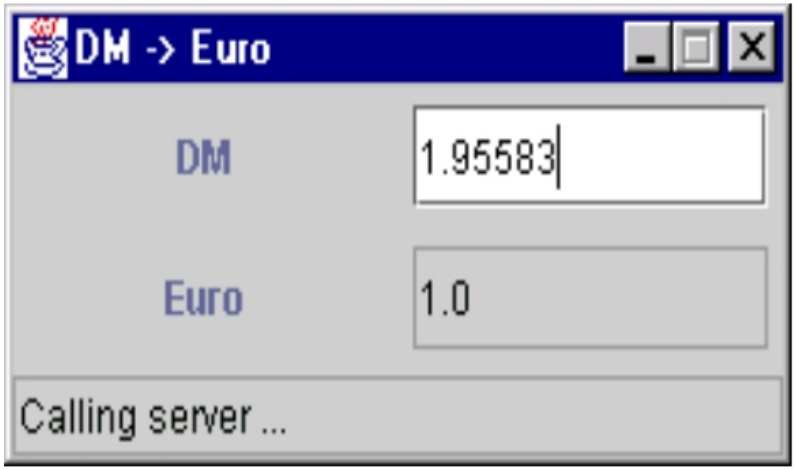
- Discovery
- Lookup
- Leasing
- Remote Events
- Transactions

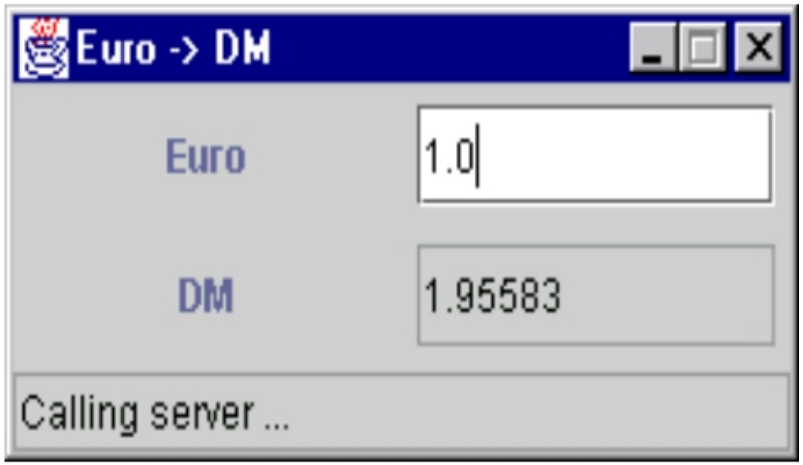
© 1999 Andreas Dörr

Die Anwendung









© 1999 Andreas Dörr

Am Anfang war? Die Schnittstelle!




```
public interface EuroConverterIF extends Remote
{
  static final double EXCHANGE_RATE = 1.95583;

  public double euro2Dm(double aEuro) throws
    RemoteException;

  public double dm2Euro(double aDM) throws
    RemoteException;
}
```

© 1999 Andreas Dörr

Aber jetzt kommt Jini - Noch nicht!




```
public class EuroConverterServer extends UnicastRemoteObject
                                 implements EuroConverterIF
{
    ...
    public double euro2Dm(double aEuro)
    {
       return (Math.rint((aEuro * EuroConverterIF.EXCHANGE_RATE)
          * 100000) / 100000);
    }

   public double dm2Euro(double aDM)
   {
       return (Math.rint((aDM / EuroConverterIF.EXCHANGE_RATE)
          * 100000) / 100000);
   }
   ...
}
```

© 1999 Andreas Dörr

Aber jetzt kommt Jini - Noch nicht! (2)




```
public class EuroConverterServer extends UnicastRemoteObject
                                 implements EuroConverterIF
{
   ...
   public static void main(String[] args)
   {
      System.setSecurityManager(new RMISecurityManager());
      try {
         EuroConverterIF srv = new EuroConverterServer();
         Naming.rebind("EuroConverterIF", srv);
      } catch (Exception e) {
         e.printStackTrace();
         System.exit(-1);
      }
   }
   ...
}
```

© 1999 Andreas Dörr

Der Dienstleister




```
public class ServiceProvider {
  ...
  private void registerDMToEuro() {
    DMToEuroRemote srvObj = new DMToEuroRemote();
    JoinManager jm = null;
    Entry[] attr = new Entry[1];
    ServiceInfo srvInf = new ServiceInfo("A.Doerr", "DM->Euro",
        "EuroConverter", "08/15", "Demo Ware Inc.", "1.0");
    attr[0] = srvInf;
    try {
      jm = new JoinManager(srvObj, attr, new SrvIDListener(),
                new LeaseRenewalManager());
    } catch(java.io.IOException e) {
      e.printStackTrace();
    }
  }
  ...
}
```

© 1999 Andreas Dörr

Die Klasse JoinManager



- Wickelt das Join Protokoll für einen Dienst ab
- Ermittelt die aktiven Lookup Services
- Registriert den Dienst bei diesen Lookup Services
- Erneuert die Leases für die Registrierung
- Sorgt für stets aktuelle Attribute

© 1999 Andreas Dörr

Die Klasse ServiceInfo



- Speichert allgemeine Informationen über einen Dienst.
- Name, Modell, Version
- Hersteller, Seriennummer

© 1999 Andreas Dörr

Das Interface Entry



- Alle Einträge in einem Lookup Service müssen von diesem Typ sein.
- Alle Felder müssen *public* Referenzen auf Objekte sein.
- Daraus folgt: Keine primitiven Datentypen (Integer statt int).
- Alle Felder werden getrennt serialisiert.

© 1999 Andreas Dörr

Der Kunde




```
public class ServiceConsumer
{
  public static void main(String[] args)
  {
    System.setSecurityManager(new RMISecurityManager());

    try {
      LookupLocator     lookup    =  new LookupLocator("jini://dijin");
      ServiceRegistrar  srvReg    =  lookup.getRegistrar();

      ServiceTemplate  srvTmpl  =  null;
      ServiceMatches   srvFound = null;
      ServiceItem      srvItem    =  null;
      Entry[]          srvInfo    =  new Entry[1];
      Class[]          srvClass   =  null;
      ...

    } catch (java.lang.Exception ex) {
      ...
    }
  }
}
```

© 1999 Andreas Dörr

Der Kunde (2)




```
public class ServiceConsumer
{
   public static void main(String[] args)
   {
      System.setSecurityManager(new RMISecurityManager());

      try {
         ...
        srvInfo[0]   = new ServiceInfo(null, null, "EuroConverter", null, null, null);
         srvClass   = new Class[] {Runnable.class};
         srvTmpl    = new ServiceTemplate(null, srvClass, srvInfo);

         srvFound = srvReg.lookup(srvTmpl, 2);

         for (int i = 0; i < srvFound.totalMatches; i++) {
            srvItem = srvFound.items[i];
            new Thread((Runnable)srvItem.service).start();
         }
      } catch (java.lang.Exception ex) {
         ...
      }
   }
}
```

© 1999 Andreas Dörr

Die Klasse LookupLocator



- Sucht einen Lookup Service mittels Unicast Discovery.
- Die Methode getRegistrar() liefert den Proxy des gefundenen Lookup Service.
- Die Klasse LookupDiscovery verwendet das Multicast Request Protokoll für den gleichen Zweck.

© 1999 Andreas Dörr

Das Interface ServiceRegistrar



- Schnittstelle zum Lookup Service.
- Gesucht wird mit Hilfe einer Schablone: ServiceTemplate
- Über die Methode lookup() wird ein bestimmter Service gesucht.
- Die Methode getEntryClass() liefert den Typ (die Klasse) eines Service.
- Mit getFieldValues() werden alle Services gesucht, die bestimmte Attribute besitzen.

© 1999 Andreas Dörr

Die Klasse ServiceMatches



- Wird mehr als eine Service gefunden, liefert der Lookup Service eine Instanz dieser Klasse zurück.
- Das Feld items enthält die Services als ein Array von ServiceItem Klassen.
- Das Feld totalMatches enthält die Anzahl der gefunden Services.

© 1999 Andreas Dörr

The End!!



- Die Mitwirkenden in der Reihenfolge des Erscheinens:
  - com.sun.jini.lookup.JoinManager
  - net.jini.core.entry.Entry
  - net.jini.lookup.entry.ServiceInfo
  - net.jini.core.discovery.LookupLocator
  - net.jini.core.lookup.ServiceRegistrar
  - net.jini.core.lookup.ServiceTemplate
  - net.jini.core.lookup.ServiceMatches
  - net.jini.core.lookup.ServiceItem

© 1999 Andreas Dörr

Weitere Informationen



- http://www.sun.com/jini
- http://www.jini.org
- Ken Arnold et. al.
  "The Jini(TM) Specification"
  Addision Wesley, ISBN 0201616343
- Andreas Dörr
  "Jini: Geist aus dem Netzwerk"
  iX 11/98

  "Die Geister, die ich rief."
  Programmierung eines Jini Service
  Java MAGAZIN 3.99

  "Alle guten Geister."
  Jini: Distributed Events und Leases
  Java MAGAZIN 4.99

© 1999 Andreas Dörr