

# ITU-VOOP-F2002

## Persistens

Lars Thorup

[lars.thorup@acm.org](mailto:lars.thorup@acm.org), Mar 26, 2002

1

## Dagens overblik

- Siden sidst
- Persistens
  - Serialisering
  - Netværkstransport
  - RMI
- Til næste gang

2

## Siden sidst

- Assert i Java 1.4:

```
class App {
    public static void main(String[] args) {
        assert 4 > 3 : "arithmetic catastrophe";
        assert 3 > 4 : "expected failure";
    }
}
$ javac -source 1.4 $<
$ java App # default is no assertion checking
$ java -enableassertions App
```

3

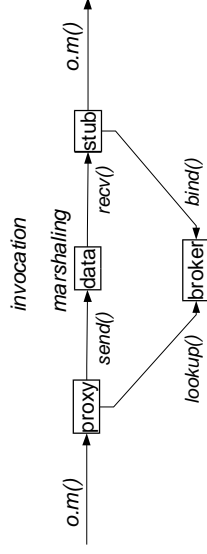
## Persistens – Definition

- Definition på Persistens
  - Få data til at overleve det kørende program.
- Eksempler
  - Lagring, fx i filer eller databaser
  - Transport, fx over netværk.

4

## Persistens – Elementer

- Elementer i Persistens
  - Marshalling = ind- og udpakning
  - Invocation = metode-kald
  - Brokering = objekter identitet & placering



5

## Persistens – Hvordan?

- Marshalling: kodes manuelt eller brug Serializable
- Invocation: kodes manuelt eller brug RMI
- Brokering: kodes manuelt eller brug RMI
- RMI : Remote Method Invocation

6

## Manuel Marshaling

- Kontrol over dataformat
- En masse toString()-kald
- Java-uaafhængigt
- Kontrol over versionering
- Ingen behov for fælles klasse-filer
- Kun for simple datastrukturer

7

## Serializable (1)

- Kun Java
- For komplekse datastrukturer
  - Der foretages "deep copy"
  - Pas på performance
- Implements Serializable
- Bruger InputStream og OutputStream

8

## Serializable – eksempel (1)

```
package dk.itu.voop.rmi;
import java.io.Serializable;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
public class SerializableDemo {
    public static class Data implements Serializable {
        String str;
        int i;
    }
    public static void main(String[] args) throws Exception {
        Data data = new Data();
        data.str = "dava";
        data.i = 4711;
        ... { næste slide } ...
    }
}
```

9

## Serializable – eksempel (2)

```
// write data
String name = "tmp/data.dat";
FileOutputStream fileOutputStream = new FileOutputStream(name);
ObjectOutputStream objectOutputStream =
    new ObjectOutputStream(fileOutputStream);
objectOutputStream.writeObject(data);
objectOutputStream.close();

// read data
FileInputStream fileInputStream = new FileInputStream(name);
ObjectInputStream objectInputStream =
    new ObjectInputStream(fileInputStream);
Data data2 = (Data)objectInputStream.readObject();
System.out.println(data.str + " == " + data2.str);
System.out.println(data.i + " == " + data2.i);
```

10

## Serializable – indkodning

```
$ od -c /tmp/data.dat
0000000  _  i  \0 005  s  r  \0  %  d  k  .  i  t  u  i  a  t  i  i  z
0000020  o  o  p  e  m  i  .  S  e  r  i  a  l  i  z  a  t  i  o  n
0000040  a  b  l  e  D  a  t  a  \0 215 217  s
0000060 022  Z  è  u  & 002  I  \0 001  i  L  \0 003  s
0000100  t  r  t  \0 022  L  j  a  v  a  \0 004  /
0000120  S  t  r  i  n  g  ;  x  p  \0  \0 022  g  t  \0 004
0000140  d  a  v  a
0000144
```

11

## Serializable (2)

- Marker instans variable "transient"
  - Forhindrer at de kommer med
- Kræver at class-filer er på begge sider
  - i samme version
  - eller kompatibelt,
  - inkl JDK
- Kun objekt-delning indenfor samme stream

12

## Externalizable

- Som manuel
- Men foregiver at være Serializable
- Kun Java
- Næsten fuld kontrol over dataformat
  - readExternal() og writeExternal()
- Eksplicit versionsstyring
  - brug Externalizable, skriv versionsnr forrest

13

## Netværkstransport

- Antagelse: IP-netværk
- Afsender og modtager
- Connection-based eller connection-less
- Protokolstak
- Lyttetråd

14

## Afsender og modtager

- Identifikation
  - Maskine og port
- Maskinnavn eller IP-nummer
  - Linux: hostname eller hostname -i
  - Windows: ipconfig
  - Har et IP-nummer for hvert netkort
  - Offentlige IP-numre
  - Private IP-numre (192.168.x.x, 10.100.x.x)

15

## Afsender og modtager

- Portnummer
  - 0–1023 er reserverede standardporte
    - 80 = HTTP (web)
    - 25 = SMTP (email)
    - 22 = SSH (terminal)
  - 1024–65536 kan bruges frit (men undgå sammenfald...)
    - 3306 = mysql (database)
    - 6000 = X (gui)

16

## Connection

- Connection-based
  - TCP/IP, class `ServerSocket`
  - Request-response
  - Stream-oriented
  - Som en telefon
- Connection-less
  - UDP/IP, class `DatagramSocket`
  - Enkeltpakker
  - Som postvæsenet

17

## TCP – ServerSocket

- **Server:**

```
ServerSocket ss = new ServerSocket(port);
while(!ss.isClosed()) {
    Socket s = ss.accept();
    ... s.getInputStream().read() ...
    ... s.getOutputStream().write() ...
}
```
- **Client:**

```
Socket s = new Socket(machine, port);
... s.getInputStream().read() ...
... s.getOutputStream().write() ...
```

18

## UDP – DatagramSocket

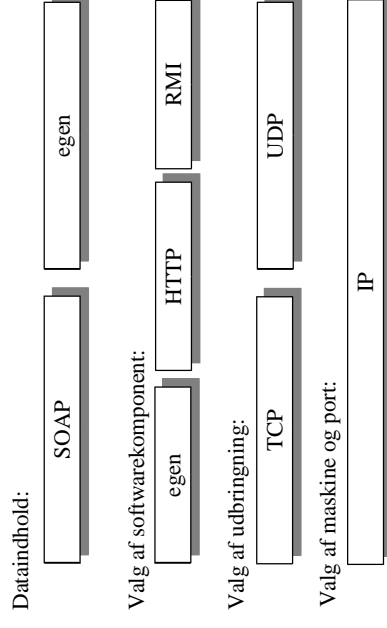
- **Server:**

```
while(!ss.isClosed()) {
    ds = new DatagramSocket(port);
    p = new DatagramPacket(...);
    ds.receive(p);
    ...
}
```
- **Client:**

```
p = new DatagramPacket(..., machine, port);
ds = new DatagramSocket();
ds.send(p);
```

19

## Protokolstak



20

## Lyttetråd

- Modtagelse af pakker
- Ikke blot tilføje en listener som med GUI
- Hvis serveren selv kan sende til klienten
  - Klienten må have separat lyttetråd der venter
- Alternativt skal klienten polle serveren
  - Men så skal man have en timer-tråd

21

## RMI

- Til Invocation og Brokering.
- Kun Java
- Ligesom CORBA, (D)COM og SOAP/UDDI
- Lidt mere opsætning end sockets
- Kræver class-filer (samme version eller compatibelt, inkl JDK) på begge sider
- Programmet "rmiregistry" er broker
- Baseret på Proxy-pattern

22

## RMI – eksempel (1)

```
package dk.itu.voop.rmi;
import java.rmi.Remote;
import java.rmi.RemoteException;
interface CommonInterface extends Remote {
    int length(String str) throws RemoteException;
}
```

23

## RMI – eksempel (2)

```
package dk.itu.voop.rmi;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Common
    extends UnicastRemoteObject
    implements CommonInterface
{
    public Common() throws RemoteException {
    }
    public int length(String str) throws RemoteException {
        return str.length();
    }
}
```

24

## RMI – eksempel (3)

```
package dk.itu.voop.rmi;

import java.rmi.RMISeccurityManager;
import java.rmi.Naming;
import java.rmi.server.UnicastRemoteObject;

public class Server {
    public static void main(String[] args) throws Exception {
        Common common = new Common();
        String commonId = "rmi://127.0.0.1:2001/theCommon";
        Naming.bind(commonId, common);
        System.out.println("Server is running...");
    }
}
```

25

## RMI – eksempel (4)

```
package dk.itu.voop.rmi;

import java.rmi.RMISeccurityManager;
import java.rmi.Naming;

public class Client {
    public static void main(String[] args) throws Exception {
        System.setSecurityManager(new RMISeccurityManager());
        // find the remote object
        String commonId = "rmi://127.0.0.1:2001/theCommon";
        CommonInterface common =
            (CommonInterface)Naming.lookup(commonId);
        // call the remote method
        String str = "xpqf";
        System.out.println(
            "length(" + str + ") = " + common.length(str));
    }
}
```

26

## RMI – eksempel (5)

```
# buildrmi.sh
javac dk/itu/voop/rmi/CommonInterface.java
javac dk/itu/voop/rmi/Common.java
javac dk/itu/voop/rmi/Server.java
javac dk/itu/voop/rmi/Client.java
rmic dk.itu.voop.rmi.Common

# runserver.sh
rmiregistry 2001 &
java dk.itu.voop.rmi.Server

# runclient.sh
java dk.itu.voop.rmi.Client
```

27

## Persistens – sikkerhed

- Sikre at man benytter en åben port
  - Og at den godt må være åben
- Undgå at serveren kan gøre uheldige ting
  - Med vilje
  - Eller pga en fejl i server-koden
- Valider klienter
- Kryptere kommunikationsstrømmen.
  - Brug secure stream:  
<http://java.sun.com/products/jdk/1.2/docs/guide/rmi/>

28

## Til næste gang

- Implementer persistens i tegnekonferencesystemet.
- Nyt krav!

## Nyt krav

- **filformat:** systemet skal kunne læse MacroShops standard filformat for tegninger, det XML-baserede MacroShopDraw-format.

- Eksempel

```
<drawing>
<line anchorx="3" anchory="3" endx="10" endy="10" />
<text anchorx="0" anchory="10">
  Oles carport
</text>
</drawing>
```