Script **generated by TTT**

Title: Distributed_Applications (21.05.2012)

Date: Mon May 21 09:16:57 CEST 2012

Duration: 44:45 min

Pages: 17

Phases of RPC based distributed applications

We distinguish between 3 phases:

a) design and implementation

b) binding of components

c) invocation: a client invoking a server operation.

Component binding

Mediation and brokering

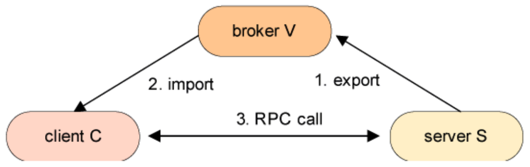
Possible terms for a mediation component are: registry, broker or trader; Corba uses the term object request broker.

Functionality of a broker

servers register their available service interfaces with the broker ("export interface").

the broker supplies the client with information in order to localize a suitable server and to determine the correct service interface ("import interface").

Client-to-server binding



Broker information

Handling client requests

Broker may either just provide the service interface to the client or act as a mediator between client and server.

direct communication between C and S.

indirect communication between C and S; communication between C and S is only possible via broker V (or several brokers).

A broker manages information about the available, exported interfaces.

server names ("white pages")

service types ("yellow pages")

behavioral or functional attributes

static attributes: functionality of the provided services, cost, required bandwidth.

dynamic attributes: current server state.

Mediation and brokering

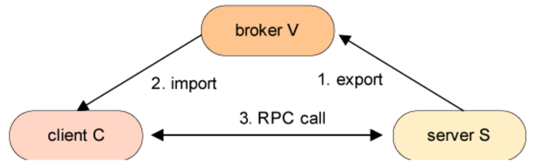
Possible terms for a mediation component are: registry, broker or trader; Corba uses the term object request broker.

Functionality of a broker

servers register their available service interfaces with the broker ("export interface").

the broker supplies the client with information in order to localize a suitable server and to determine the correct service interface ("import interface").

Client-to-server binding



Broker information

Handling client requests

Broker may either just provide the service interface to the client or act as a mediator between client and server.

direct communication between C and S.

indirect communication between C and S; communication between C and S is only possible via broker V (or several brokers).

Phases of RPC based distributed applications

We distinguish between 3 phases:

a) design and implementation

b) binding of components

c) invocation: a client invoking a server operation.

Component binding

Mediation and brokering

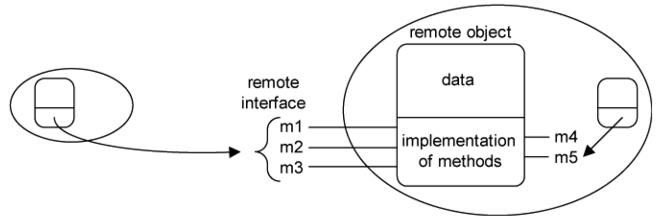
Definition: Remote object is an object whose method can be called by an object residing on another Java Virtual Machine (JVM), even on another computer.

Definition: Remote interface

is a Java interface specifying the methods of a remote object.

Definition: Remote method invocation (RMI) allows object-to-object communication between different Java Virtual Machines (JVM), i.e. it is the action of invoking a method of a remote interface on a remote object.

The method calls for local and remote objects have the same syntax.



Remote Method Invocation (RMI)

RMI supports communication among objects residing on different Java virtual machines (JVM). RMI is an RPC of the object-oriented Java environment.

Definitions

RMI characteristics

RMI architecture

Locating remote objects

Developing RMI applications

Parameter Passing in RMI

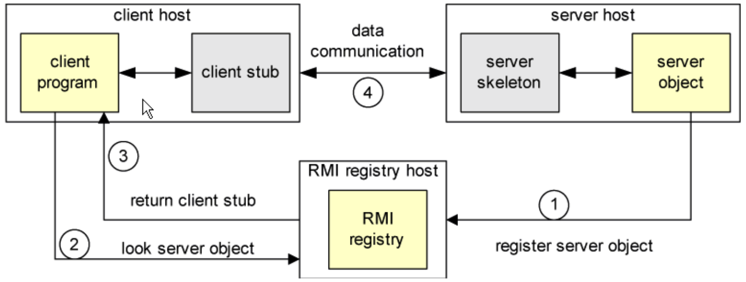
Distributed garbage collection

How does RMI work

Java RMI uses

a registry to provide naming services for remote objects,

stub and skeleton to facilitate communications between client and server.

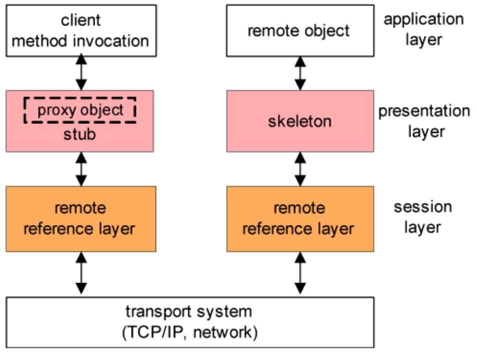


RMI works as follows

1. a server object is registered with the RMI registry
2. a client looks through the RMI registry for the remote object
3. once the remote object is located, its stub is returned to the client
4. the remote object can be used in the same way as a local object

communication between client and server is handled by stubs and skeletons.

RMI architecture



Stub/Skeleton layer

Layer intercepts method calls by the client and redirects these calls to the remote object.

Object serialization/deserialization; hidden from the application.

Remote Reference layer

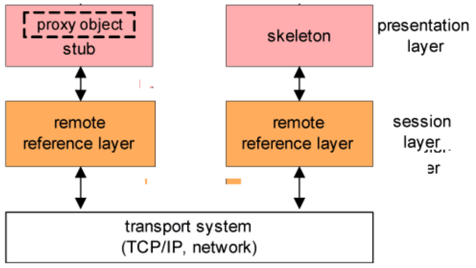
Connects client and remote objects exported by the server environment by a 1-to-1 connection link.

The layer provides JRMP (Java Remote Method Protocol) via TCP/IP.

Mapping of stub/skeleton operations to the transport protocol of the host; it interfaces the application code with the network communication.

The layer supports the method `invoke` .

RMI architecture



Stub/Skeleton layer

Layer intercepts method calls by the client and redirects these calls to the remote object.

Object serialization/deserialization; hidden from the application.

Remote Reference layer

Connects client and remote objects exported by the server environment by a 1-to-1 connection link.

The layer provides JRMP (Java Remote Method Protocol) via TCP/IP.

Mapping of stub/skeleton operations to the transport protocol of the host; it interfaces the application code with the network communication.

The layer supports the method `invoke` .

```
Object invoke (Remote obj, java.lang.reflect.Method method, Object
[ ] params, long opnum) throws Exception
```

Remote Method Invocation (RMI)

RMI supports communication among objects residing on different Java virtual machines (JVM). RMI is an RPC of the object-oriented Java environment.

Definitions

RMI characteristics

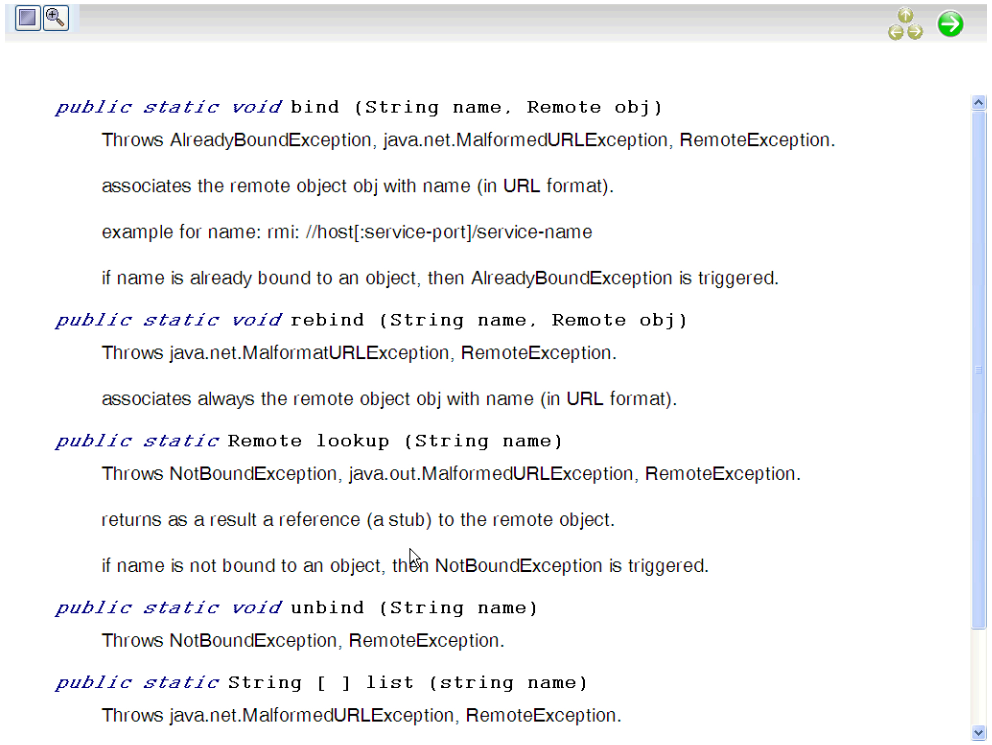
RMI architecture

Locating remote objects

Developing RMI applications

Parameter Passing in RMI

Distributed garbage collection



public static void `bind (String name, Remote obj)`

Throws AlreadyBoundException, java.net.MalformedURLException, RemoteException.

associates the remote object obj with name (in URL format).

example for name: rmi: //host[:service-port]/service-name

if name is already bound to an object, then AlreadyBoundException is triggered.

public static void `rebind (String name, Remote obj)`

Throws java.net.MalformatURLException, RemoteException.

associates always the remote object obj with name (in URL format).

public static `Remote lookup (String name)`

Throws NotBoundException, java.out.MalformedURLException, RemoteException.

returns as a result a reference (a stub) to the remote object.

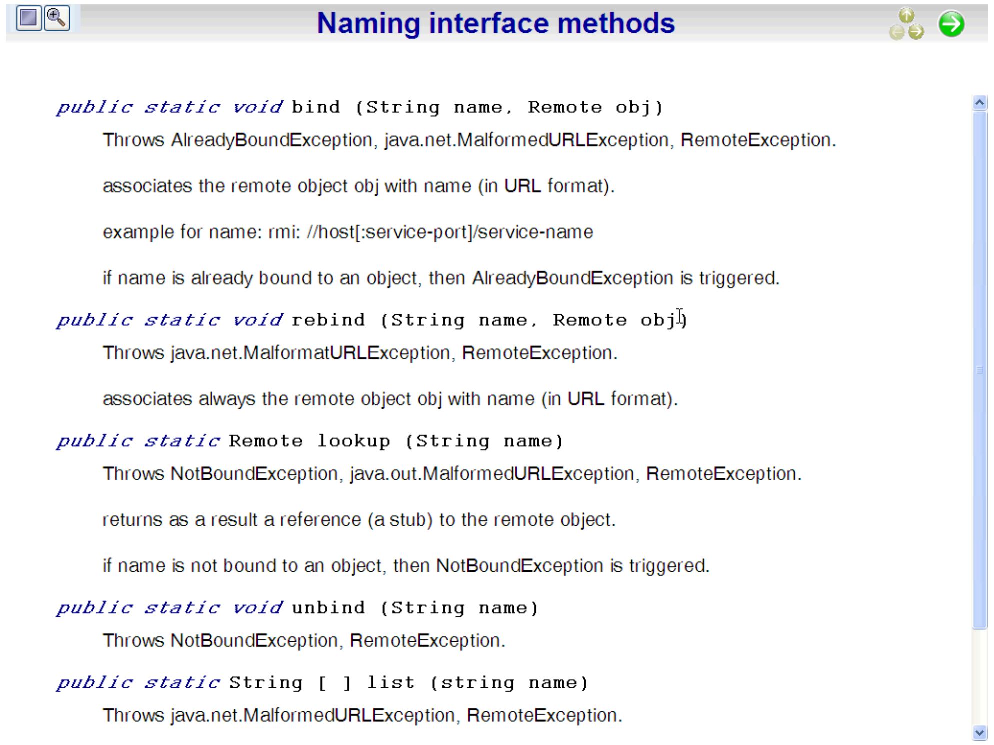
if name is not bound to an object, then NotBoundException is triggered.

public static void `unbind (String name)`

Throws NotBoundException, RemoteException.

public static `String [ ] list (string name)`

Throws java.net.MalformedURLException, RemoteException.



Naming interface methods

public static void `bind (String name, Remote obj)`

Throws AlreadyBoundException, java.net.MalformedURLException, RemoteException.

associates the remote object obj with name (in URL format).

example for name: rmi: //host[:service-port]/service-name

if name is already bound to an object, then AlreadyBoundException is triggered.

public static void `rebind (String name, Remote obj)`

Throws java.net.MalformatURLException, RemoteException.

associates always the remote object obj with name (in URL format).

public static `Remote lookup (String name)`

Throws NotBoundException, java.out.MalformedURLException, RemoteException.

returns as a result a reference (a stub) to the remote object.

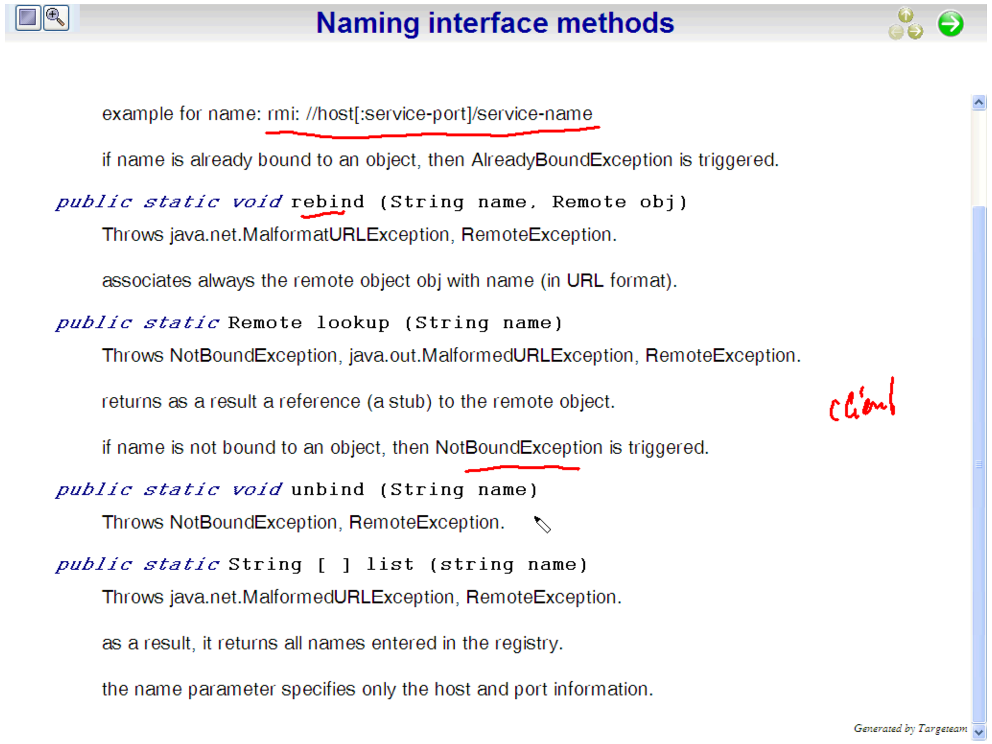
if name is not bound to an object, then NotBoundException is triggered.

public static void `unbind (String name)`

Throws NotBoundException, RemoteException.

public static `String [ ] list (string name)`

Throws java.net.MalformedURLException, RemoteException.



Naming interface methods

example for name: rmi: //host[:service-port]/service-name

if name is already bound to an object, then AlreadyBoundException is triggered.

public static void `rebind (String name, Remote obj)`

Throws java.net.MalformatURLException, RemoteException.

associates always the remote object obj with name (in URL format).

public static `Remote lookup (String name)`

Throws NotBoundException, java.out.MalformedURLException, RemoteException.

returns as a result a reference (a stub) to the remote object. client

if name is not bound to an object, then NotBoundException is triggered.

public static void `unbind (String name)`

Throws NotBoundException, RemoteException.

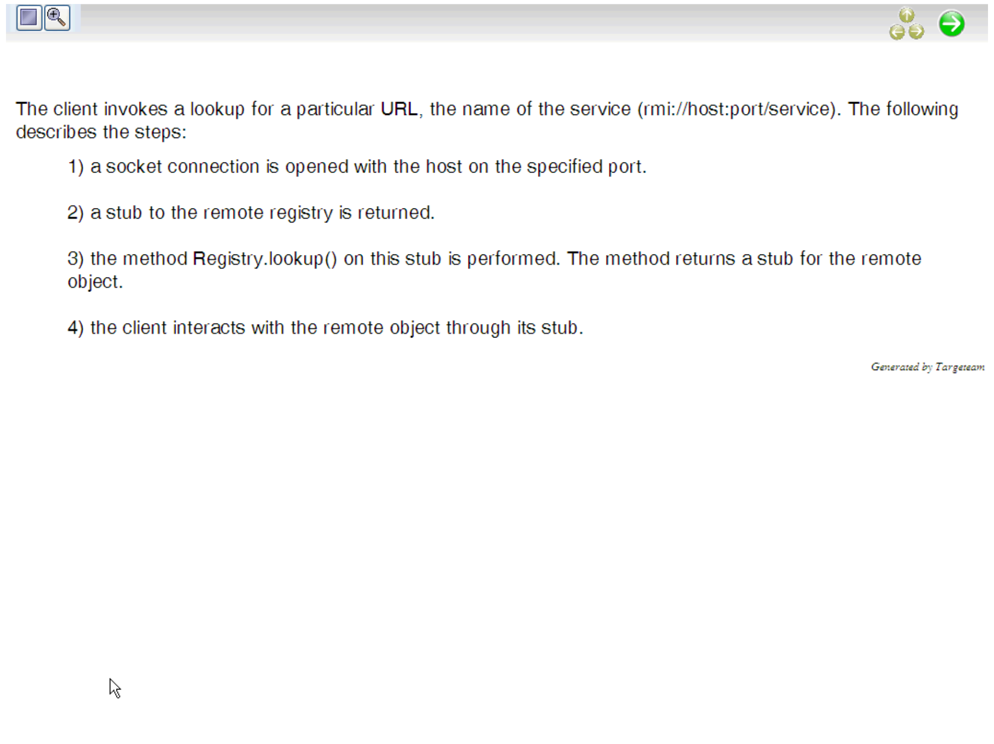
public static `String [ ] list (string name)`

Throws java.net.MalformedURLException, RemoteException.

as a result, it returns all names entered in the registry.

the name parameter specifies only the host and port information.

Generated by Targeteam



The client invokes a lookup for a particular URL, the name of the service (rmi://host:port/service). The following describes the steps:

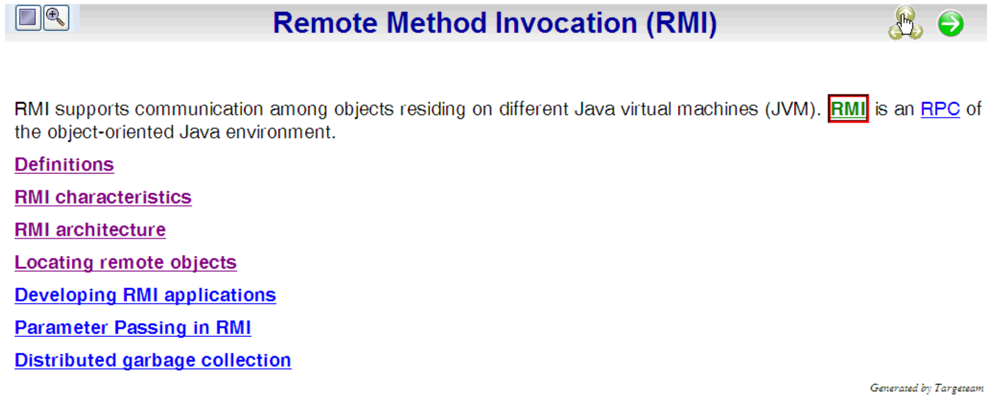
1) a socket connection is opened with the host on the specified port.

2) a stub to the remote registry is returned.

3) the method Registry.lookup() on this stub is performed. The method returns a stub for the remote object.

4) the client interacts with the remote object through its stub.

Generated by Targeteam



Remote Method Invocation (RMI)

RMI supports communication among objects residing on different Java virtual machines (JVM). RMI is an RPC of the object-oriented Java environment.

Definitions

RMI characteristics

RMI architecture

Locating remote objects

Developing RMI applications

Parameter Passing in RMI

Distributed garbage collection

Generated by Targeteam