


Programma operativo Regione Lombardia/Ministero del Lavoro/Fondo Sociale Europeo, Obiettivo 3 Misura C3

Progetto ID 24063 "Moduli e contenuti professionalizzanti inseriti nei corsi di laurea e diplomi universitari dell'Università degli Studi di Brescia"

Azione ID 41654

"Formazione teorico–pratica nell'ambito di moduli professionalizzanti orientati all'Informatica del terzo anno di corso della Laurea in Ingegneria dell'Informazione (nuovo ordinamento)"

Insegnamento a supporto del quale si svolge l'azione formativa integrativa:
Ingegneria del Software A
Modulo n.2 " Strumenti CASE e specifiche OMG"

1

Introduzione a CORBA

Sabrina De Capitani di Vimercati
decapita@ing.unibs.it
Dipartimento di Elettronica per l'Automazione
Universita' di Brescia

2

Sommario

- Applicazioni distribuite e middleware
- Tecnologie ad oggetti distribuiti
- La Object Management Architecture
  - **CORBA**
- Remote Method Invocation (RMI)
- Cenni DCOM
- Cenni SOAP

3

Applicazioni Distribuite e Middleware

- Le moderne applicazioni aziendali devono soddisfare le esigenze di dinamicità delle aziende tramite:
  - uso di applicazioni distribuite…
  - …basate su tecnologie object-oriented e component-based e …
  - … capaci di sfruttare **Internet**

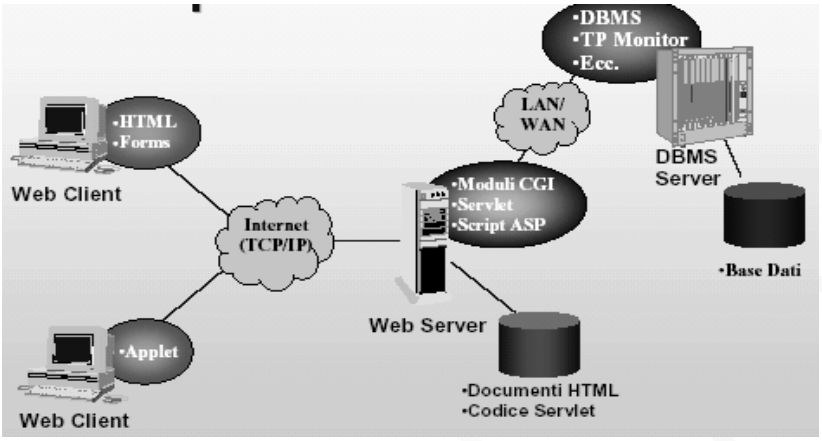
4

Applicazione Distribuita

- **Definizione**
  - Applicazione costituita da una o più *processi* che sono eseguiti in parallelo su macchine distinte connesse da una rete di comunicazione
- I processi che costituiscono una applicazione distribuita cooperano sfruttando i servizi forniti dalla rete di comunicazione

5

Applicazione Distribuita: Esempio



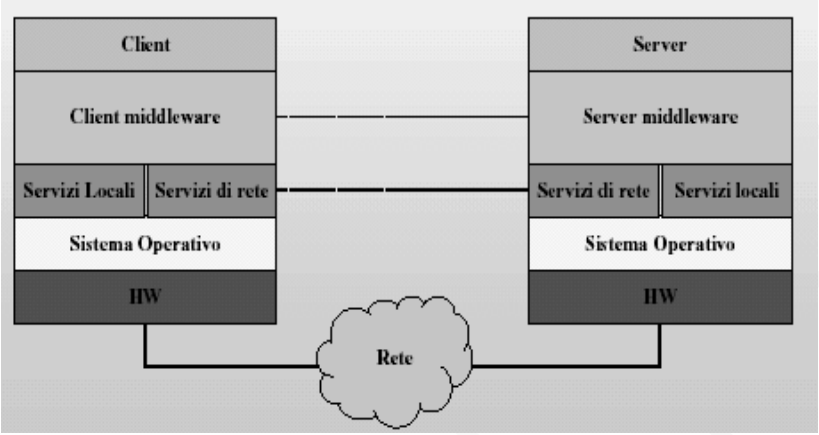
6

Middleware: Una Definizione

- Insieme di servizi trasversali che permettono ad applicazioni ed utenti finali di interagire sfruttando una rete di comunicazione

7

Middleware Client/Server



8

C/S Middleware: Servizi di Comunicazione di Base

- I middleware C/S forniscono servizi di comunicazione di base e servizi ausiliari
  - **Remote Procedure Call (RPC)**
  - Remote-Data Access (RDA)
  - Message-Oriented Middleware (MOM)

9

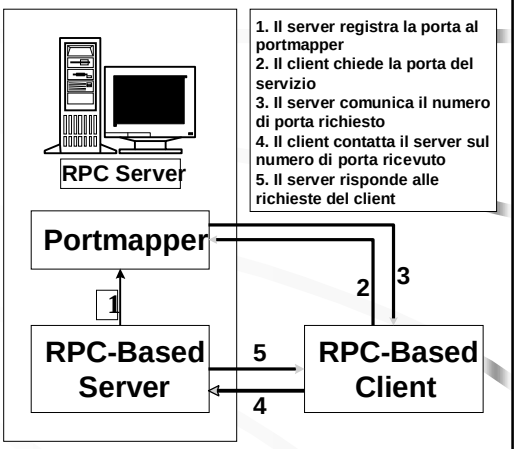
Remote Procedure Call

- Schema di funzionamento
  - il client invoca una procedura localizzata su un server remoto con sintassi analoga alle chiamate di procedura locali
  - il server esegue la procedura e rimanda indietro la risposta
- Ogni coppia richiesta/risposta è gestita separatamente

10

RPC: Funzionamento

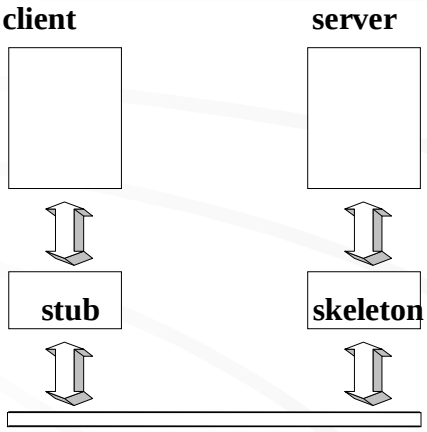
- *RPC (Remote Procedure Call)*: servizio che consente di chiamare una funzione di libreria residente su un'altra macchina della rete come se fosse locale



11

Ieri: Client/Server con RPC

- interfaccia locale del server remoto definita in *IDL* (*Interface Definition Language*)
- *stub* e *skeleton* eseguono il *marshalling* e l'*unmarshalling* dei dati



12

Oggi: Middleware per Applicazioni Client/Server OO

- **Obiettivo**
  - Approfittare dei vantaggi dell'approccio OO in ambito distribuito
- **Tecnologie**
  - OMG CORBA
  - Sun RMI
  - Microsoft DCOM

13

Applicazioni Distribuite OO

- Una applicazione distribuita OO è organizzata come una collezione di oggetti…
- …localizzati su macchine diverse…
- … che comunicano attraverso *invocazione remota di metodi*

14

Interface Definition Language

- In ambito distribuito, information hiding e compilazione separata diventano essenziali
- Non è ipotizzabile che sia nota l'implementazione delle applicazioni con le quali occorre comunicare
- Ogni oggetto deve quindi conoscere solo l'*interfaccia* degli oggetti con i quali comunicare
- È necessario un linguaggio con cui definire tali interfacce: **Interface Definition Language**

15

CORBA: una introduzione

- CORBA = Common Object Request Broker Architecture
- CORBA è una infrastruttura per gli oggetti distribuiti che permette agli stessi una interazione (1), anche se sono su piattaforme eterogenee (2).
  - (1) grazie all'*Object Management Architecture* (OMA);
  - (2) grazie all'*Interface Definition Language* (IDL).
- La OMA è il risultato del lavoro di una organizzazione no-profit: l'**OMG**

16

L'Object Management Group

- OMG è una organizzazione internazionale no-profit che include oltre 700 membri istituzionali
- Fondata nel 1989 per promuovere la diffusione e lo sviluppo di tecnologie orientate agli oggetti (OO)

17

La visione dell'OMG

- L'importanza dei dati, del loro reperimento e della loro elaborazione nelle moderne aziende
  - rendere i dati disponibili indipendentemente dalla loro "posizione" deve essere l'obiettivo delle moderne applicazioni
- Il singolo elaboratore non puo' più essere la piattaforma di riferimento……la piattaforma è oggi la **rete (Internet)**
  - → tecnologie che permettano di sviluppare applicazioni distribuite capaci di sfruttare questa piattaforma

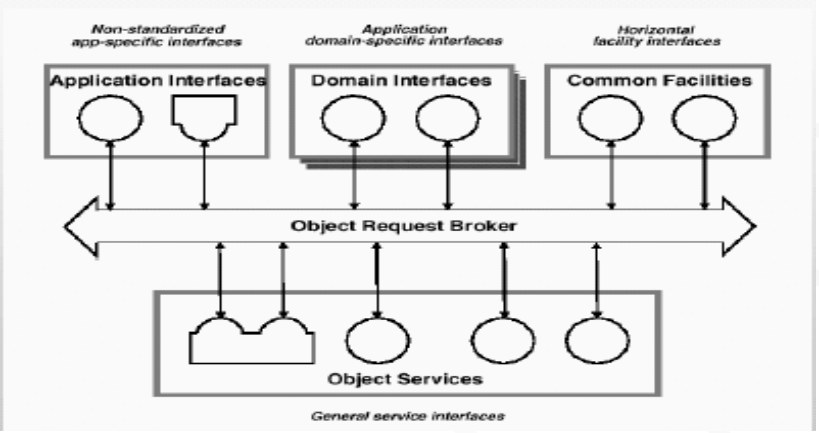
18

Lo scopo dell'OMG

- Problema
  - la complessità del software continua a crescere…
  - …e di conseguenza i costi di sviluppo
  - la distribuzione aggrava il problema
- Scopo dell'OMG è ridurre lo sforzo necessario a sviluppare applicazioni distribuite
  - per mezzo di una *architettura di riferimento*, multipiattaforma, capace di favorire il riuso di componenti e l'interoperabilità tra applicazioni

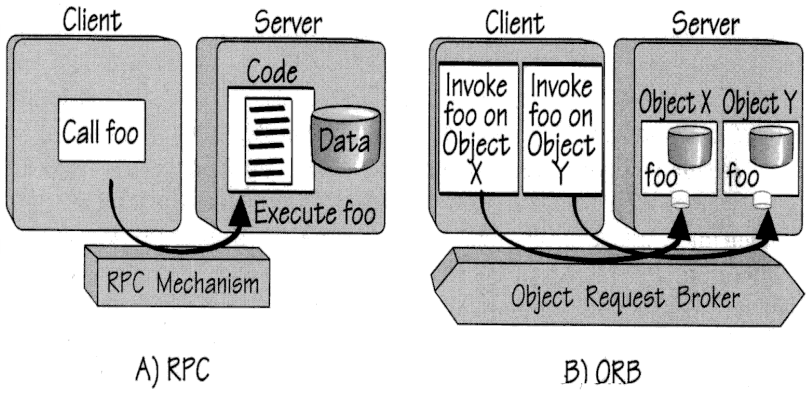
19

L'Object Management Architecture



20

CORBA Rispetto a RPC



21

Object Request Broker

- L'ORB è il cuore dell'OMA
- Fornisce l'infrastruttura di comunicazione attraverso la quale i componenti dell'applicazione distribuita cooperano
- Le primitive fornite dall'ORB sono indipendenti dalla piattaforma sulla quale sono in esecuzione i diversi componenti…
- … e dalla tecnologia con la quale questi sono implementati

22

Oggetti dei Servizi

- Gli oggetti dei servizi (Object Services) costituiscono un insieme di servizi di uso generico per l'implementazione di applicazioni distribuite
- L'OMA include la specifica delle interface di un insieme di oggetti che realizzano gli Object Services
- L'OMA non descrive alcuna implemetazione

23

Object Services: Una Elencazione (1)

- **Naming service**
  - fornisce la possibilità di avere il riferimento ad un oggetto noto il suo nome simbolico
  - il servizio puo' essere implementato in maniera distribuita
- **Event service**
  - Fornisce un servizio di comunicazione asincrono e multicast
    - un componente invia "eventi" su un *canale*
    - i componenti in ascolto sul canale ricevono gli eventi inviati
    - i canali possono essere collegati tra loro

24

Object Services: Una Elencazione (2)

- **Life cycle service**
  - fornisce metodi per copiare, muovere e cancellare insiemi di oggetti collegati in un grafo (object factory)
- **Relationship service**
  - fornisce metodi per “navigare” all’interno di un grafo di oggetti
- **Persistent object service**
  - fornisce servizi di persistenza che consentono di salvare lo stato di componenti per recuperarlo in fututo

25

Object Services: Una Elencazione (3)

- **Transaction service**
  - fornisce un supporto all’implementazione di sistemi transazionali dove ogni transazione coinvolge più componenti distribuiti
- **Concurrency control service**
  - fornisce metodi per coordinare l’accesso a risorse condivise da parte di componenti diversi
- **Externalization service**
  - permette il marshalling e l’unmarshalling dello stato dei componenti CORBA

26

Object Services: Una Elencazione (4)

- **Query service**
  - permette l'invocazione di query su insieme di oggetti distribuiti
- **Security service**
  - fornisce servizi di identificazione, autenticazione, controllo di accesso, auditing e sicurezza nella comunicazione
- **Collection service**
  - fornisce servizi per gestire insiemi di componenti distribuite

27

Facilitazioni Comuni

- Le facilitazioni comuni (Common Facilities) sono un insieme di componenti che forniscono funzionalità applicative di uso comune
- Includono componenti per gestire:
  - stampa
  - accesso a DB
  - accesso a servizi di email
- L'OMA descrive solo le interfacce di questi componenti

28

Interfacce di Dominio

- Le interfacce di dominio (Domain Interfaces) specificano componenti che forniscono servizi specifici per particolari domini applicativi
- OMA si occupa di standardizzare solo le interfacce dei componenti che implementano tali servizi

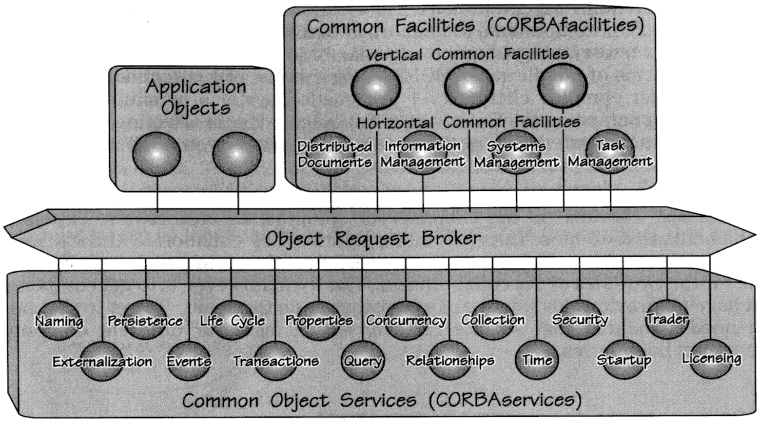
29

Oggetti delle Applicazioni

- Gli oggetti delle applicazioni (Application Interfaces) sono le interfacce che descrivono i componenti specifici dell'applicazione
- Poiche' l'OMG non sviluppa applicazioni la standardizzazione di tali interfacce è fuori dai propri compiti

30

Una Visione d'Assieme



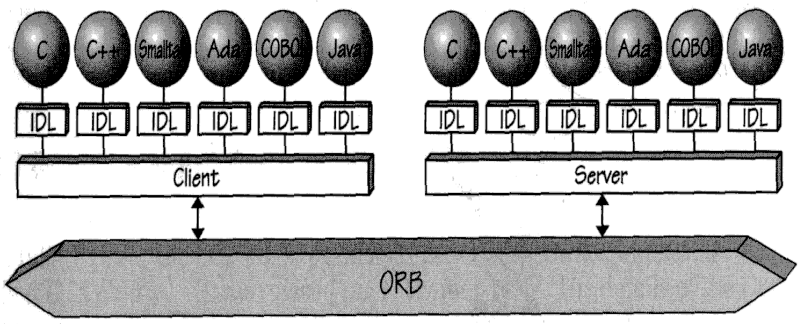
31

CORBA IDL

- Le interfacce che descrivono i componenti che costituiscono l'OMA sono definite tramite **l'Interface Definition Language**
- Per accedere ai servizi esportati da un componente, una applicazione client deve conoscere solo l'interfaccia
- Un componente la cui interfaccia sia specificata con il CORBA IDL puo' essere implementato con diversi linguaggi di programmazione

32

IDL Mapping



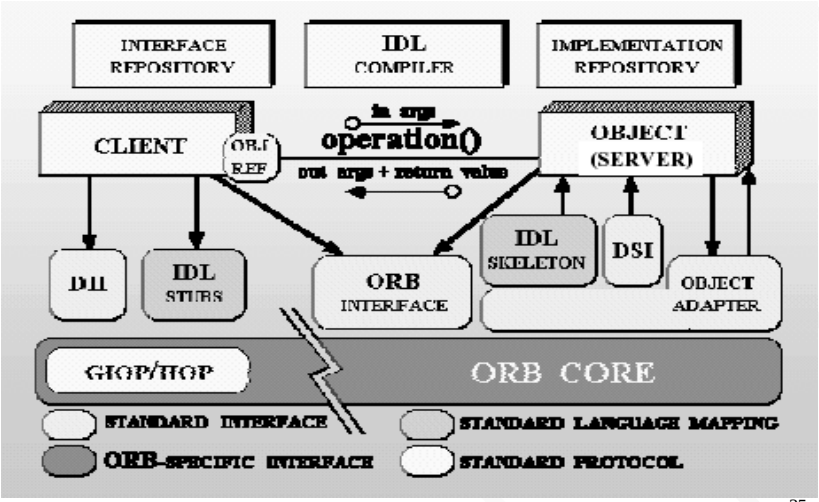
33

OMA e CORBA

- CORBA indica l'insieme delle interfacce dei modelli di riferimento che compongono l'OMA
- CORBA è anche il nome con il quale si indica l'architettura di riferimento di un ORB
- *CORBAServices* e *CORBAFacilities* sono i nomi tecnici per l'insieme di interfacce che specificano rispettivamente gli Object Services e le Common Facilities

34

CORBA (1)



35

CORBA (2)

- **Client**: componente che richiede il servizio
  - Possiede un riferimento al server (“obj ref”)
- **Server**: il componente che eroga il servizio
- **ORB Core**: implementa i meccanismi di base per la comunicazione trasparente tra client e server
  - reperire il server richiesto attivandolo, passare gli eventuali parametri e valori di ritorno…

36

CORBA (3)

- **ORB Interface**: interfaccia standard per l'accesso ai servizi dell'ORB
  - rende client e server indipendenti dalla specifica implementazione dell'ORB
- **Stub e skeleton**: generati a partire dall'interfaccia del server tramite l'IDL compiler
  - con ORB formano il tramite tra client e server

37

CORBA (4)

- **Dynamic Invokation Interface (DII):** specifica i servizi utilizzabili dal client per accedere ad un server del quale non sia staticamente nota l'interfaccia
- **Dynamic Skeleton Interface (DSI):** permette l'implementazione di server la cui interfaccia non sia nota staticamente

38

CORBA (5)

- **Object Adapter**: è il tramite principale tra ORB Core e server e viceversa. Si occupa di:
  - generare ed interpretare i riferimenti remoti, registrare il server, attivare e disattivare il server, invocare i metodi del server
  - è responsabile dei meccanismi e delle politiche usate nello svolgimento di tali operazioni
  - diversi tipi di OA possono essere forniti da una particolare implementazione di CORBA
- **GIOP e IIOP**: protocolli usati dall'ORB per trasferire le informazioni tra client e server

39

Invocazione Statica e Dinamica

- *static server invocation*: il client conosce l'interfaccia del server desiderato; l'ORB seleziona lo specifico server object che eseguirà il servizio
- *dynamic invocation:* il client specifica delle *server properties* a un modulo *Trader*, che rimanda l'OID del server adatto
- Il client comunica all'ORB la OID che ha ricevuto, l'ORB gli invia l'interfaccia del server

40

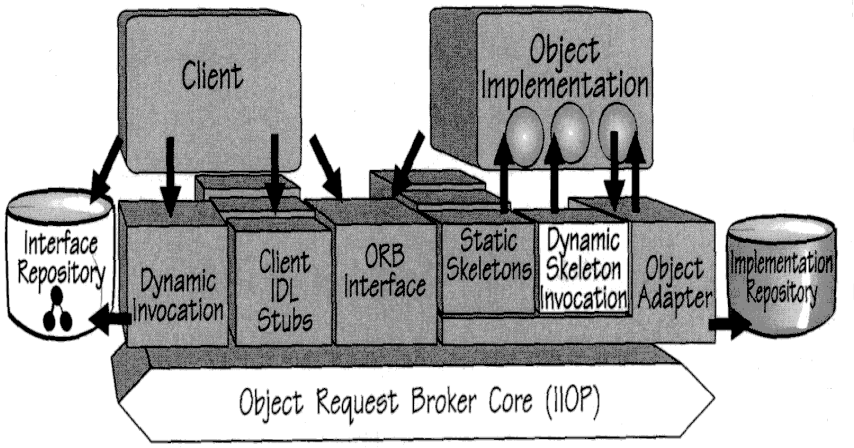
La Chiamata Dinamica in Dettaglio

L'*invocazione dinamica* CORBA consiste di tre fasi:

- il client identifica il server adatto attraverso un colloquio con il modulo *Trader* che gli restituisce uno *IOR (Interoperable Object Reference)*
- il client passa lo *IOR* all'*ORB* che usando l'*Interface Repository* recupera l'interfaccia dell'oggetto desiderato e la comunica al client
- il client *costruisce* ed *esegue* la chiamata

41

Schema della Dynamic Invocation



42

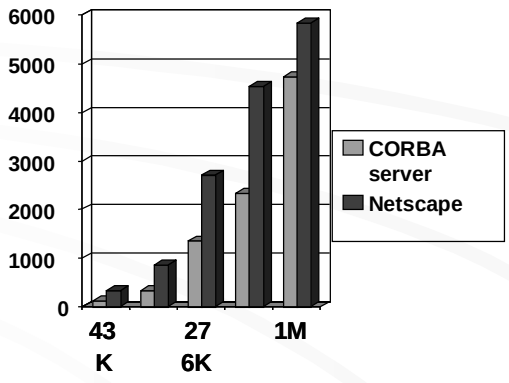
JOE: interfaccia Java/CORBA

- permette agli applet o ai servlet Java di comunicare con un ORB per reperire e invocare i server CORBA
- accesso tramite Web a sistemi *legacy* con *wrapper* CORBA
- *piena integrazione* del sistema informativo in intranet
- disponibile con JDK 1.1

43

Prestazioni di CORBA (PowerBroker)

- confronto tra *Netscape* che trasferisce file via *HTTP* e un client CORBA che li trasferisce via *IIOP*
- Netscape è penalizzato dal tempo di visualizzazione



44

Corba e Java (1)

I passi per implementare una applicazione client/server Java sono i seguenti:

- **Creazione di un contratto**: utilizzare IDL per descrivere un *contratto* per un oggetto CORBA
- **Compilazione IDL**: un compilatore IDL riceve la descrizione dell'interfaccia ed esegue una corrispondenza tra IDL ed il linguaggio in cui è implementato l'oggetto
  - classi stub
  - classi skeleton
  - classi di servizio

45

Corba e Java (2)

- **Implementazione:** scrittura del server e del client sfruttando le classi generate dal compilatore IDL e le classi di libreria che implementano le funzionalità dell'ORB
- **Compilazione:** compilare server e client con un compilatore Java standard
- **Esecuzione:** eseguire l'applicazione

46

Esempio di Programmazione (1)

Creazione contratto

```
module ContoCorrente
{
  interface Conto
  {
      attribute string numero_conto;
      void deposito(in long amount);
      long prelievo(in long amount);
      long saldo();
  };
};
```

47

Esempio di Programmazione (2)

Compilazione

- **Classi lato Client**
  - `_ContoStub.java`
  - `Conto.java`
- **Classi lato Server**
  - `ContoPOA.java`
  - `ContoPOATie.java`
  - `ContoOperations.java`
- **Classi di supporto**
  - `ContoHelper.java`
  - `ContoHolder.java`

48

Esempio di Programmazione (3)

Implementazione

```
Public class ContoImpl
         extends ContoPOA
{
  private long _saldo;
  public string numeroConto;
  public Conto(string _numeroConto)
  {
     _saldo = 0;
     numeroConto = _numeroConto;
  }
```


Esempio di Programmazione (4)

```
public void deposito(int amount)
{
    _saldo += amount;
}
public int prelievo(int amount)
{
    _saldo -=  amount;
    return _saldo;
}
public int saldo()
{
    return saldo;
}
```

Esempio di Programmazione (5)

```
string numeroConto()
{
   return _numeroConto;
}
void numeroConto(string arg)
{
   _numeroConto = arg;
}
}
```

51

Scrittura Server (1)

```
public class Server
{
  public static void main(String[] args)
  {  //inizializzazione org
     org.omg.CORBA.ORB orb =
  orb.omg.CORBA.ORB.init(args, null);
     try
     { //creazione riferimento POA
       org.omg.PortableServer.POA poa =
  org.omg.PortableServer.POAHelper.narrow(orb
  .resolve_initial_references("RootPOA"));
```

52

Scrittura Server (2)

```
//attivazione oggetto nel POA
org.omg.CORBA.Object o =
  poa.servant_to_reference(new contoImpl("0393"));
poa.the_POAManager.activate();
//pubblicazione OR
PrintWriter ps = new PrintWriter(new
  FileOutputStream(new File(args[0])));
ps.println(orb.object_to_string(o));
ps.close();
}
Catch(Exception e)
{e.printStackTrace();}
orb.run(); // attivazione ORB
}}
```

53

Scrittura Client (1)

```
public class Client
{
  public static void main(string args[])
  {
     try
     { //inizializzazione ORB
       org.omg.CORBA.ORB orb =
  orb.omg.CORBA.ORB.init(args,null);
       Conto contoBancario;//dichiara. variabile
       BufferedReader in = new BufferedReader(new
  FileReader(args[0])); //reperimento OR
       string IORString = in.readline();
```

54

Scrittura Client (2)

```
//creazione riferimento tramite iniz. con OR
contoBancario =
ContoHelper.narrow(orb.string_to_object(IORString
);
contoBancario.deposito(1000);
contoBancario.prelievo(100);
System.out.println("Il saldo del conto" +
contoBancario.numeroConto() + "e`" +
contoBancario.saldo());
}
catch (Exception e)
{e.printStackTrace();}
}
```

55

Sun RMI

- RMI costituisce la risposta SUN all'OMG
- Attraverso i servizi di base di RMI è possibile invocare metodi su un oggetto remoto
  - gli aspetti critici della distribuzione vengono mascherati al programmatore

56

RMI in Dettaglio

- Attraverso i servizi RMI è possibile invocare metodi su un *oggetto remoto* come se si trattasse di un oggetto locale
- Gli oggetti remoti si comportano come gli oggetti tradizionali
  - è possibile passare riferimenti ad oggetti remoti nelle chiamate a metodo
- La differenza principale riguarda il passaggio di parametri non remoti a metodi di oggetti remoti
  - il passaggio è fatto per copia
- Lo stesso vale per i valori di ritorno restituiti da metodi di oggetti remoti

57

CORVA vs RMI

- L'approccio è molto simile
- Vantaggi di CORBA
  - interoperabilità
  - standard aperto
- Vantaggi RMI
  - più semplice da usare (non richiede un IDL diverso da Java stesso)
  - è possibile trasferire oggetti per copia e non solo riferimenti ad oggetti e strutture dati
  - supporta un meccanismi di download trasparente del codice per le classi non residenti in locale

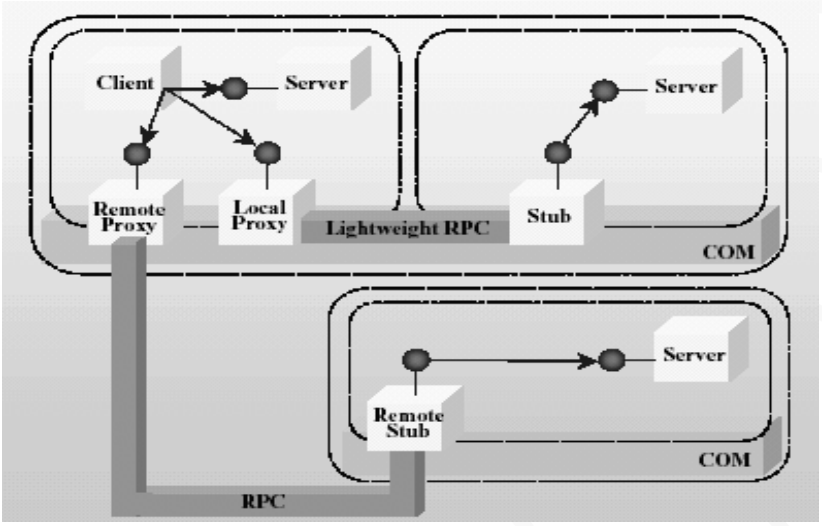
58

DCOM: Introduzione

- DCOM è la tecnologia Microsoft per il supporto di applicazioni formate da componenti distribuiti
- Simile a CORBA, DCOM fornisce un linguaggio di definizione di interfacce…
- … e definisce un protocollo standard di comunicazione che permette a componenti distribuiti di cooperare

59

L'Architettura DCOM



DCOM: Proxy e Stub

- Componenti COM che si occupano del marshalling e dell'unmarshalling dei parametri e del valore di ritorno dei metodi
- Il client accede all'interfaccia del proxy come tramite verso il componente server
- Lo stub riceve le invocazioni del proxy e si occupa di invocare il componente server

61

DCOM: Interfacce

- Il DCOM IDL viene utilizzato per la specifica delle interfacce dei componenti
  - DCOM IDL è simile al CORBA IDL
- A partire dalla specifica delle interfacce vengono generati proxy e stub
- Ogni interfaccia è univocamente identificata dal suo ID (IID)

62

DCOM: Componenti

- Ogni componente puo' implementare più interfacce
- DCOM, come COM, non supporta ereditarietà tra componenti
- Ogni componente è identificato univocamente da un GUID (Globally Unique ID)
- L'attivazione dei componenti è affidata ad un class factory

63

DCOM: Class Factory

- Un componente capace di creare nuovi componenti
  - identificato univocamente da un class id (CLSID)
- Implementa l'interfaccia standard IClassFactory

64

DCOM: Component Server

- Un eseguibile o una DLL che si occupa di gestire un insieme di componenti DCOM
  - contiene la class factory e l'implementazione di tali componenti
- Tre tipi di component server
  - In-process
    - DLL caricata nello spazio di indirizzamento del client
  - Local
    - eseguibile che "gira" nella stessa macchina del client
  - Remote
    - eseguibile che "gira" su una macchina diversa da quella del client

65

DCOM: Service Control Manager

- Una parte di COM responsabile di:
  - reperire un component server dato il suo CLSID
  - far partire il server o connettersi al server in esecuzione
  - riportare le informazioni relative alla connessione al client
    - Sotto forma di un interfaccia ad un proxy

66

RMI – CORBA - DCOM

- RMI, CORBA e DCOM: tre approcci distinti per la soluzione dello stesso problema
- In pratica numerosi sforzi vengono oggi fatti per “unire” questi tre mondi
  - RMI – CORBA: una proposta di estensione di GIOP al fine di consentire il passaggio per copia di oggetti
  - RMI – DCOM: Microsoft fornisce strumenti per l’interoperabilità tra le due piattaforme
  - DCOM – CORBA: esistono dei ridge che consentono il colloquio tra componenti DCOM e CORBA

67

Problemi dei Protocolli basati su RPC

- Verbosità
  - Molti protocolli basati su RPC richiedono una considerevole banda (ad esempio, il ping-based lifecycle management di DCOM)
- Attraversamento firewall
  - Molte organizzazioni sono riluttanti nell’abilitazione di protocolli RMC come CORBA-IIOP o DCOM

68

Attraversamento Firewall

- Firewall partizionano la rete in “celle” difficili da penetrare dalle invocazioni remote di servizi
- Ad esempio, nell’architettura TCP/IP ad ogni servizio è associato un numero di porta
  - il blocco di tutte le porte tranne la 80 usata per le connessioni HTTP impedisce l’uso di protocolli come CORBA e DCOM
- L’abilitazione all’accesso di un e-service attraverso firewall richiede un intervento manuale

  → **SOAP**

69

SOAP: Introduzione

- SOAP è un protocollo “leggero” che risolve i problemi di verbosità e attraversamento dei firewall
- Proposto dall’XML Protocol Working Group del W3C (http://www.w3.org/2000/xp)

70

SOAP: Richiesta

```
POST /QuoteService HTTP/1.1
SOAP-Action="http://www.acme.com/GetQuote"
Content-Type= text/xml; charset="UTF-8"
Content-Length: nnnn
<!– SOAP invocation goes here -->
```

71

SOAP: XML Payload

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://schemas…"
xmlns:ACME="http://www.acme.com/soap" …>
  <SOAP-ENV:Header>
      <!– header entries-->
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
      <ACME:GetQuote>
            <ACME:OriginZIP> 90070 </ACME:OriginZip>
            <ACME:DestZip> 16804 </ACME:DestZip>
            <ACME:Weight> 500 </ACME:Weight>
      </ACME:GetQuote>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

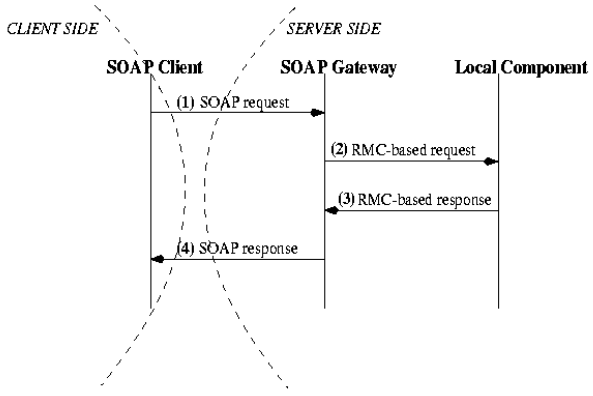
72

SOAP: Risposta

```
HTTP/1.1 200 OK
Content-Type= text/xml; charset="UTF-8"
Content-Length: nnnn
<SOAP-ENV:Envelope xmlns:SOAP-ENV
  ="http://schemas…"
xmlns:ACME="http://www.acme.com/soap" …>
  <SOAP-ENV:Body>
     <ACME:GetQuoteResponse>
          <ACME:Amount> 18 </ACME:Amount>
     </ACME:GetQuoteResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

73

Esecuzione di una Chiamata SOAP



74

Bibliografia

- RMI
  - http://java.sun.com
- CORBA
  - http://www.omg.org
- DCOM
  - http://www.microsoft.com/com
- SOAP
  - http://www.w3.org/TR

75