



UNIVERSITA' DEGLI STUDI DI BERGAMO
Facoltà di Ingegneria

Informatica Industriale

Prof. Davide Brugali

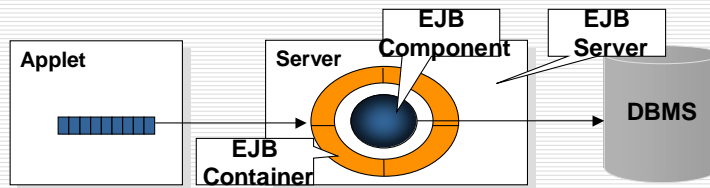
2.4 – Java 2 Enterprise Edition

Enterprise JavaBeans

- EJB is a standard set of APIs:
 - a business component architecture for Java
 - third-party vendors provide implementations.
- Objectives:
 - standardise Java application servers;
 - allow reuse of business components without access to their source code;
 - make component and application development easy;
 - let ISVs handle the hard technical problems of enterprise computing.

EJB architecture

- An EJB component contains business logic:
 - EJB defines naming conventions for standard services.
- An EJB server provides the execution environment:
 - threads, load balancing, middleware, two-phase commit.
- An EJB container manages the component:
 - distribution, creation, persistence, transactions etc.



2.4 - J2EE

Informatica Industriale - Prof. Davide Brugali

3/27

Deploying an EJB component

- EJB components are deployed as JAR archive files:
 - the JAR file contains a "manifest" that describes the run-time properties of the component.
- Properties include:
 - access control lists (security);
 - transactional behaviour;
 - your own properties (called environment properties):
- Containers and servers include a deployment console to set EJB properties:
 - GUI tools that can also monitor running servers and components.

2.4 - J2EE

Informatica Industriale - Prof. Davide Brugali

4/27

Transactions

- ❑ Some EJB servers support two-phase commit:
 - allows many components and databases across the network to participate in a single “all-or-nothing” transaction.
- ❑ Clients can start and commit transactions.
- ❑ Components can start and commit transactions.
 - the deployer defines how this is done;
 - the component developer does not have to worry about transactions.

EJB Security

- ❑ The deployer specifies the security options for the component:
- ❑ You can define roles with access privileges:
 - can be interfaced with traditional database userid/password;
 - you can control whether the component executes under its own privileges or under those of the client.
- ❑ Access control lists specify who can access:
 - the entire component;
 - each service of the component.

EJB Communications

- ❑ EJB uses Java RMI (Remote Method Invocation) to access components.
- ❑ RMI makes a distributed Java programme look like a local one:
 - simple, transparent distribution;
 - any Java method can be called across the network;
 - any Java object can be sent across the network;
 - distributed garbage collection.
- ❑ Clients and servers write almost no code for communications

EJB and CORBA

- ❑ The EJB standard defines how to implement all EJB services on top of CORBA middleware:
 - EJB Transactions map to CORBA Transaction Services;
 - EJB Security maps to the CORBA Security Services;
 - EJB Naming maps to CORBA Naming Services.
- ❑ EJB is "CORBA made easy":
 - you don't have to build a component architecture on top of CORBA middleware.

Portability versus interoperability

- ❑ EJB specifies portability:
 - develop a component for one server, then deploy it in another server.
- ❑ CORBA provides interoperability:
 - EJB components will be able to communicate across multiple vendors' servers if the servers are all based on CORBA.

Java 2 Enterprise Edition

- ❑ It is a middleware framework specifically designed to build enterprise applications. It is structured as a four-tier architecture.
- ❑ The **Client Tier** typically consists of a Web browser or a stand-alone application
- ❑ The **Web Tier** manages the dynamic content creation of the web pages using Servlets.
- ❑ The **Business Tier** hosts the EJB components (Business Objects) that implement application-specific business logic.
- ❑ The **Server Tier** is the bridge between the Business Tier and heterogeneous enterprise information systems such as enterprise resource planning, mainframe transaction processing, and database systems.

J2EE Services

- ❑ **Naming Services** provide EJB components with naming and directory functionalities, such as searching for objects using their attributes.
- ❑ **Deployment Services** provide facilities to associate deployment descriptors to components and to manipulate them. These descriptors define how the component can be configured, customized, and installed when it is reused in a new environment.
- ❑ **Security Services** provide mechanisms for the identity authentication and resource access authorization of software components and users.




❑ UNIVERSITA' DEGLI STUDI DI BERGAMO
Facoltà di Ingegneria

Mobilità del codice


Sistemi distribuiti e mobilità

La diffusione e l'evoluzione dei sistemi distribuiti genera la necessità di meccanismi che permettano di spostare il codice tra i nodi della rete.

Sistemi operativi distribuiti

- 
- migrazione trasparente di processi ed oggetti
 - obiettivo: bilanciare il carico tra i calcolatori
 - approccio strutturato e complesso (poco flessibile)

Sistemi con mobilità del codice

- 
- mobilità visibile e controllabile dal programmatore
 - obiettivo: accedere a risorse distribuite nella rete
 - mobilità su vasta scala (internet)

Ambiti applicativi per la mobilità

La mobilità del codice è una soluzione flessibile ai problemi applicativi dei sistemi distribuiti in cui è richiesta l'autonomia dei componenti per adattarsi ad un ambiente dinamico.

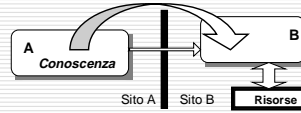
Ambiti applicativi:

- ☐ Reperimento di informazioni distribuite
- ☐ Documenti attivi
- ☐ Servizi avanzati per le telecomunicazioni
- ☐ Controllo e configurazione di dispositivi remoti
- ☐ Gestione di workflow
- ☐ Reti attive
- ☐ Commercio elettronico

Modelli progettuali per la mobilità

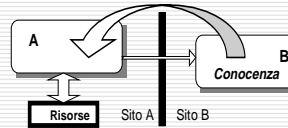
□ Valutazione remota

Il componente software A necessita delle risorse contenute nel sito B. A invia il codice relativo alla conoscenza in B, richiedendone l'esecuzione.



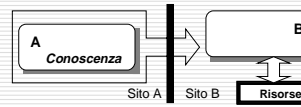
□ Codice su richiesta

Il componente A possiede le risorse ma necessita di conoscenza. A inoltra una richiesta a B per ottenere il codice con la conoscenza.



□ Agenti mobili

La conoscenza è posseduta dal componente software A che necessita delle risorse disponibili nel sito B. A migra presso il sito B.



Sperimentazione della mobilità del codice in Java

1° CASO

Sistema di mobilità con RMI

Mobilità applicata ad un problema di calcolo distribuito

RMI: Remote Method Invocation

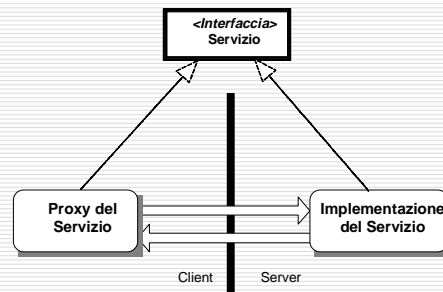
L'infrastruttura RMI estende il modello ad oggetti di Java agli ambienti distribuiti consentendo l'invocazione dei metodi per via remota.

L'architettura RMI è basata su di un principio importante:

Definizione del comportamento e implementazione sono concetti separati.

L'oggetto **Proxy** è un derivato dell'oggetto di implementazione.

Consente di invocare i metodi sull'oggetto remoto reale.



Il calcolo distribuito

Un calcolatore con elevata potenza di calcolo (server) deve essere impiegato per eseguire algoritmi complessi.

L'esecuzione degli algoritmi viene invocata dai client.

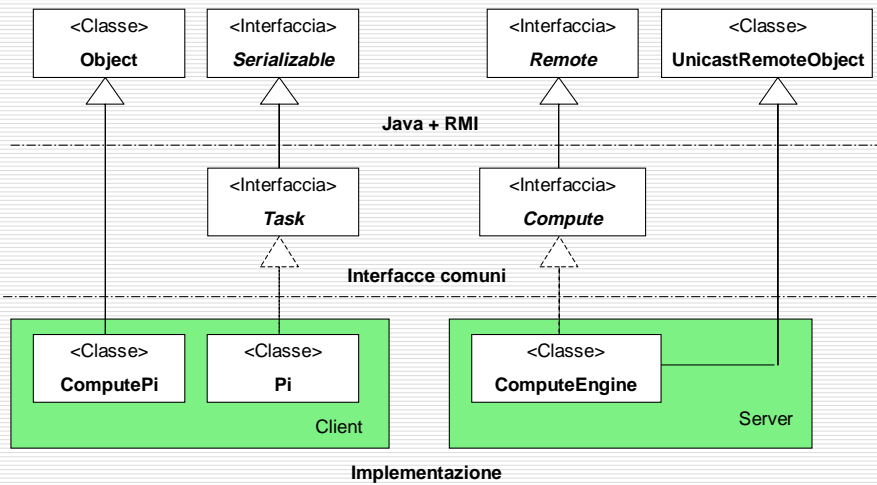
SOLUZIONE CLIENT-SERVER CLASSICA	SOLUZIONE CON MOBILITA' DEL CODICE
Tutti i servizi risiedono nel server. I client invocano l'esecuzione dei servizi inviando i parametri necessari	Il server fornisce il motore di calcolo I client inviano i servizi al server richiedendone l'esecuzione

Il calcolo distribuito in un ambiente dinamico:

- Servizi che possono evolvere nel tempo.
- Servizi disponibili solo a tempo di esecuzione.

Soluzione:
Mobilità del codice

Struttura dell'applicazione

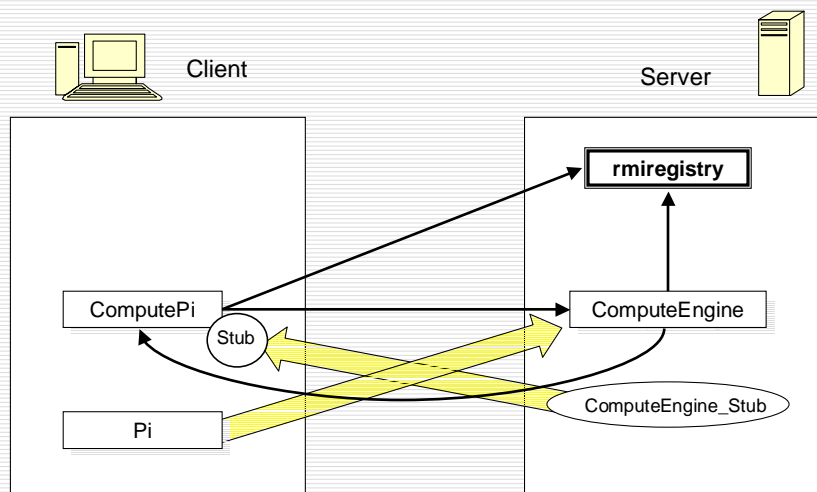


2.4 - J2EE

Informatica Industriale - Prof. Davide Bruggali

19/27

Interazione per il calcolo distribuito



2.4 - J2EE

Informatica Industriale - Prof. Davide Bruggali

20/27

Sperimentazione della mobilità del codice in Java

2° CASO

Sistema di mobilità con JINI

Architettura per la mobilità degli agenti

2.4 - J2EE

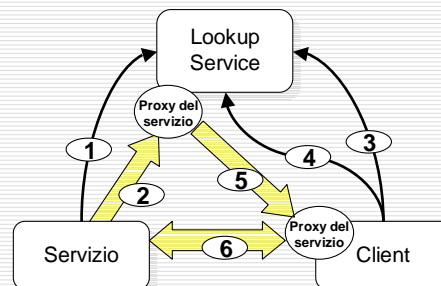
Informatica Industriale - Prof. Davide Brugali

21/27

L'architettura Jini

- Jini costituisce un'infrastruttura in cui client e servizi possono interagire, costituendo così una community.
- Scritto interamente in linguaggio Java, Jini impiega il meccanismo *Remote Method Invocation* per muovere gli oggetti all'interno della rete.
- Il **Lookup Service** (LUS) consente ai servizi di registrare la propria disponibilità ed ai client di accedere ai servizi attraverso i proxy

1	Discover	Il servizio scopre i LUS disponibili
2	Join	Il servizio invia un suo proxy al LUS
3	Discover	Il client scopre i servizi disponibili
4	Lookup	Il client richiede il servizio desiderato
5	Receive	LUS invia il proxy del servizio
6	Use	Il client interagisce con il servizio



2.4 - J2EE

Informatica Industriale - Prof. Davide Brugali

22/27

Agenti

Agente: termine impiegato per indicare una tecnologia utilizzata in ambiti relativi alle applicazioni di intelligenza artificiale.

Caratteristiche distintive degli agenti rispetto agli oggetti:

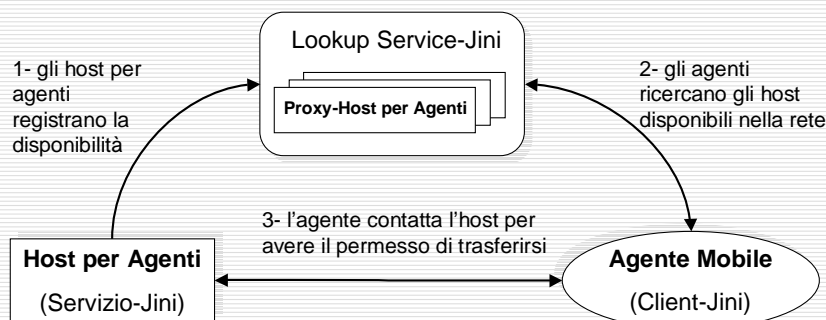
- sono **autonomi**
- **apprendono** e **si adattano**
- sono **mobili** e **persistenti**
- sono **orientati all'obiettivo**
- sono in grado di **collaborare** e di **comunicare**

Gli agenti tendono ad essere di piccole dimensioni:
non costituiscono da soli un'applicazione ma formano un'unità di lavoro insieme ad altri agenti e al componente fisico che li ospita.

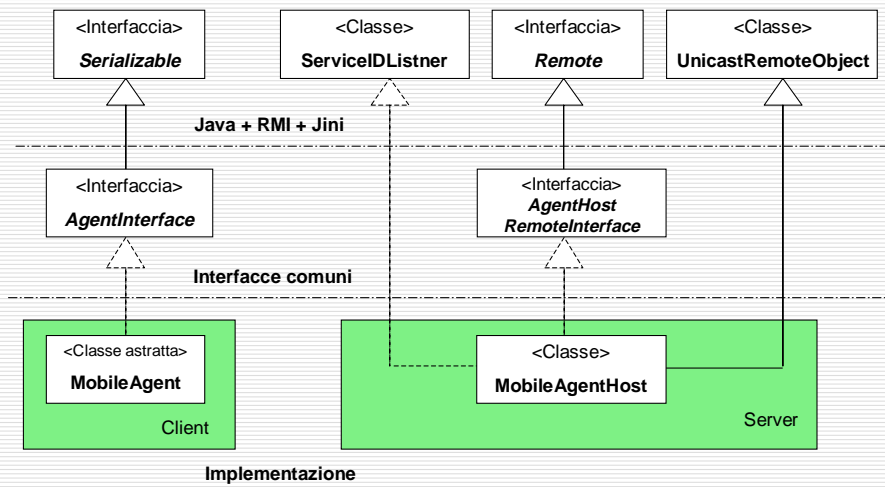
Implementare agenti mobili in Jini

Due tipologie di componenti software:

- gli host per agenti: forniscono il supporto all'esecuzione degli agenti, consentendo loro di accedere alle risorse del calcolatore.
- gli agenti mobili: possiedono un obiettivo da raggiungere ed includono la conoscenza. Necessitano delle risorse possedute dagli host.



Struttura dell'applicazione

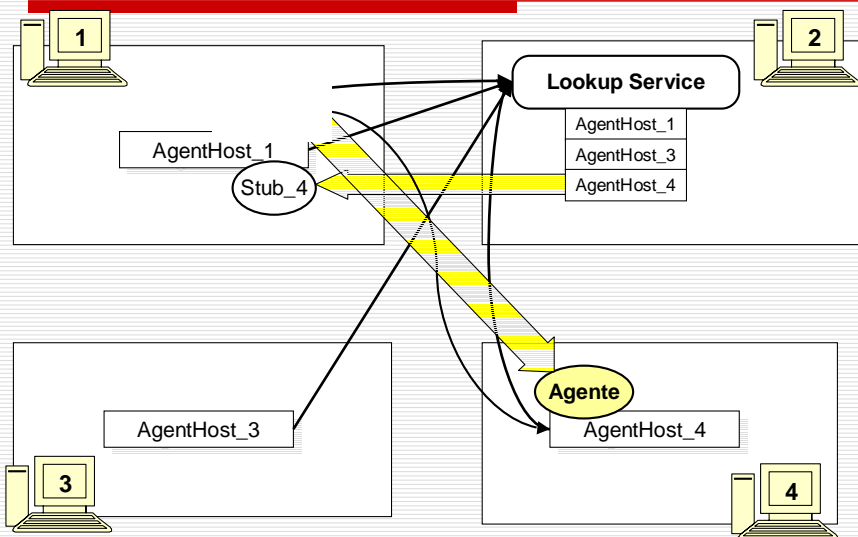


2.4 - J2EE

Informatica Industriale - Prof. Davide Brugali

25/27

Agenti liberi nel sistema distribuito



2.4 - J2EE

Informatica Industriale - Prof. Davide Brugali

26/27

Confronto

architettura con RMI: la mobilità viene consentita dall'unione della tecnica di invocazione dei metodi remoti abbinata alla serializzazione degli oggetti. Il fornitore dei servizi è unico e noto a priori.

- **architettura con Jini:** la mobilità usufruisce di una struttura più flessibile, i servizi possono essere molteplici e localizzati in punti. È quindi possibile localizzare i servizi a tempo di esecuzione.