

# Programmazione con Oggetti Distribuiti: Java RMI

Vittorio Scarano

II edizione - Marzo 2010



*A mio padre e a mia madre, ispiratori*

*A mia moglie, dolce compagna, amica e confidente*

*Ai miei tre figli, gioielli e luce dei miei occhi*





# **EMERGENCY**

EMERGENCY è un'associazione italiana indipendente e neutrale. EMERGENCY offre assistenza medico-chirurgica gratuita e di elevata qualità alle vittime civili delle guerre, delle mine antiuomo e della povertà.

Dal 1994 più di 3 milioni di persone sono state curate gratuitamente in ospedali, cliniche e centri di riabilitazione di EMERGENCY dove inoltre è garantita la formazione del personale medico locale.

Tre ospedali, un centro di maternità e 29 posti di primo soccorso in Afghanistan, un ospedale e un posto di primo soccorso in Cambogia, un centro di riabilitazione in Iraq, un poliambulatorio per migranti a Palermo, un ospedale e un centro pediatrico in Sierra Leone, una clinica pediatrica e un centro di cardiocirurgia in Sudan, due centri pediatrici in Darfur e in Repubblica Centrafricana, e un centro di Ostetricia e Ginecologia in Nicaragua in costruzione.

Ecco gli ospedali che Emergency ha voluto, ha costruito e tuttora gestisce grazie al contributo di migliaia di persone.

EMERGENCY

Via Gerolamo Vida 11  
20127, Milano  
tel. 02/881881 - fax 02/86316336

<http://www.emergency.it>  
e mail: [info@emergency.it](mailto:info@emergency.it)

Conto corrente postale  
intestato a "EMERGENCY Ong Onlus"  
n. 2842 6203  
IBAN: IT37 Z076 0101 6000 0002 8426 203

Conto corrente bancario  
intestato a "EMERGENCY Ong Onlus"  
IBAN: IT 02 X 05018 01600 000000130130  
presso Banca Etica, Filiale di Milano

*L'autore devolve interamente i propri proventi in favore dei progetti di Emergency.*



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	I sistemi distribuiti . . . . .	2
1.2	Un modello di riferimento: Open Distributed Processing . . . . .	3
1.2.1	Le caratteristiche di un sistema distribuito . . . . .	4
1.2.2	I requisiti non funzionali di un sistema distribuito . . . . .	5
1.2.3	La trasparenza di un sistema distribuito . . . . .	7
1.2.4	I diversi punti di vista ( <i>viewpoints</i> ) per un sistema distribuito . . . . .	10
1.3	Il Middleware ad Oggetti Distribuiti . . . . .	13
1.3.1	Il progenitore: Remote Procedure Calls (RPC) . . . . .	15
1.3.2	Da RPC al Middleware ad Oggetti Distribuiti . . . . .	15
1.3.3	Esempi rappresentativi: CORBA, Java RMI e .NET . . . . .	16
1.4	Il Middleware ad Oggetti Distribuiti nel Modello a Componenti . . . . .	17
	Note bibliografiche . . . . .	19
	Spunti per lo studio individuale . . . . .	20

## **Introduzione a Java RMI** **25**

<b>2</b>	<b>Dai socket agli oggetti remoti</b>	<b>25</b>
2.1	Introduzione . . . . .	26
2.2	Un breve richiamo sulla programmazione con i socket . . . . .	26
2.2.1	I socket TCP . . . . .	27
2.2.2	Gli stream . . . . .	27
2.2.3	HelloWorld con i socket . . . . .	28
2.2.4	Un esempio di client-server con i socket . . . . .	31
2.3	Da un oggetto locale . . . . .	34
2.4	. . . all'oggetto remoto . . . . .	36
2.4.1	Il server e la interfaccia remota . . . . .	37
2.4.2	Il client . . . . .	39
2.4.3	Lo strato stub/skeleton . . . . .	39
2.4.4	La sequenza delle invocazioni . . . . .	43
2.4.5	Per lanciare la applicazione . . . . .	43
2.5	Indirizzamento dell'oggetto remoto . . . . .	45
2.5.1	Il Client . . . . .	45
2.5.2	Lo Skeleton . . . . .	47
2.6	Alcuni commenti conclusivi . . . . .	49
2.7	Approfondimenti . . . . .	49
2.7.1	Il funzionamento di <code>getOutputStream()</code> . . . . .	49
2.7.2	Conversione di <code>Byte unsigned</code> . . . . .	50
	Note bibliografiche . . . . .	50
	Spunti per lo studio individuale . . . . .	51

<b>3</b>	<b>Presentazione di Java Remote Method Invocation</b>	<b>53</b>
3.1	Introduzione . . . . .	54
3.2	Gli obiettivi della progettazione di Java RMI . . . . .	54
3.3	Il modello a oggetti distribuiti di Java RMI . . . . .	56
3.3.1	La struttura delle classi di Java RMI . . . . .	56
3.3.2	Il meccanismo di invocazione remota . . . . .	58
3.3.3	La differenza tra il modello a oggetti locale e quello remoto . . . . .	61
3.4	La architettura di Java RMI . . . . .	62
3.4.1	I tre layer della architettura . . . . .	62
3.4.2	Distributed Garbage Collection . . . . .	66
3.4.3	Caricamento dinamico delle classi . . . . .	67
3.5	Approfondimenti . . . . .	68
3.5.1	L'eterogeneità nelle tecnologie ad oggetti distribuiti . . . . .	68
3.5.2	La trasparenza degli oggetti distribuiti . . . . .	70
3.5.3	La sicurezza in Java . . . . .	74
3.5.4	Il meccanismo di marshalling usato da Java RMI . . . . .	75
	Note bibliografiche . . . . .	76
	Spunti per lo studio individuale . . . . .	78
<b>4</b>	<b>Un primo esempio con Java RMI</b>	<b>81</b>
4.1	Introduzione . . . . .	82
4.2	Il processo di creazione di un programma Java RMI . . . . .	82
4.2.1	Definizione della interfaccia remota . . . . .	82
4.2.2	Implementazione del server . . . . .	83
4.2.3	Compilazione del server . . . . .	83
4.2.4	Compilazione con lo stub compiler rmic . . . . .	84
4.2.5	Servizio di naming: rmiregistry . . . . .	85
4.2.6	Esecuzione del server . . . . .	85
4.2.7	Registrazione del server sul servizio di naming . . . . .	88
4.2.8	Implementazione del client . . . . .	88
4.2.9	Compilazione ed esecuzione del client . . . . .	89
4.3	L'esempio HelloWorld . . . . .	89
4.3.1	Definizione della interfaccia remota . . . . .	89
4.3.2	Implementazione del server . . . . .	90
4.3.3	Compilazione del server . . . . .	91
4.3.4	Compilazione con lo stub compiler rmic . . . . .	91
4.3.5	Servizio di naming: rmiregistry . . . . .	91
4.3.6	Esecuzione del server . . . . .	92
4.3.7	Registrazione del server sul servizio di naming . . . . .	92
4.3.8	Implementazione del client . . . . .	92
4.3.9	Compilazione ed esecuzione del client . . . . .	93
4.4	La sicurezza e la policy del Security Manager . . . . .	93
4.5	Commenti conclusivi . . . . .	94
	Note bibliografiche . . . . .	95
	Spunti per lo studio individuale . . . . .	96
	<b>Programmazione con Java RMI</b>	<b>99</b>
<b>5</b>	<b>Design Pattern con Java RMI</b>	<b>99</b>
5.1	Introduzione . . . . .	100

5.2	L'Adapter . . . . .	100
5.2.1	Un esempio di Adapter: un contatore remoto . . . . .	100
5.2.2	Contatore remoto: il server . . . . .	101
5.2.3	Contatore remoto: il client . . . . .	105
5.2.4	Alcuni commenti all'esempio . . . . .	106
5.3	La Factory . . . . .	106
5.3.1	Un esempio di Factory: HelloWorld multilingua . . . . .	107
5.3.2	HelloWorld multilingua: il server . . . . .	107
5.3.3	HelloWorld multilingua: il client . . . . .	110
5.3.4	Alcuni commenti all'esempio . . . . .	111
5.4	L'Observer . . . . .	111
5.4.1	Un esempio di Observer: la callback per le architetture client-server . . . . .	112
5.4.2	Awareness con callback: la architettura . . . . .	113
5.4.3	Awareness con callback: lato client . . . . .	114
5.4.4	Awareness con callback: lato server . . . . .	117
5.4.5	Alcuni commenti all'esempio . . . . .	119
	Note bibliografiche . . . . .	119
	Spunti per lo studio individuale . . . . .	120
<b>6</b>	<b>Una chat con Java RMI</b>	<b>121</b>
6.1	Introduzione . . . . .	122
6.1.1	Le funzionalità di una chat . . . . .	122
6.2	Chat con i socket . . . . .	123
6.2.1	Un esempio con socket TCP . . . . .	123
6.2.2	Un esempio con socket UDP Multicast . . . . .	128
6.2.3	Commenti e ulteriori sviluppi . . . . .	131
6.3	Una chat client-server con Java RMI . . . . .	133
6.3.1	La architettura software . . . . .	133
6.3.2	Il server . . . . .	133
6.3.3	Il client . . . . .	136
6.3.4	Commenti e ulteriori sviluppi . . . . .	139
6.4	Una chat peer2peer . . . . .	141
6.4.1	Un primo esempio di peer . . . . .	141
6.4.2	Commenti e ulteriori sviluppi . . . . .	146
6.5	Approfondimenti . . . . .	147
6.5.1	UDP e indirizzi IP di gruppo . . . . .	147
	Note bibliografiche . . . . .	148
	Spunti per lo studio individuale . . . . .	149
<b>7</b>	<b>Alcuni esercizi</b>	<b>151</b>
7.1	Introduzione . . . . .	152
7.2	Una chat CS per "Cuori solitari" . . . . .	152
7.2.1	Il testo dell'esercizio . . . . .	152
7.2.2	Una soluzione . . . . .	152
7.3	Una chat CS per un docente dispotico . . . . .	159
7.3.1	Un confronto tra differenti strategie . . . . .	160
7.3.2	Una soluzione . . . . .	161
	Spunti per lo studio individuale . . . . .	168

<b>Argomenti Avanzati su Java RMI</b>	<b>173</b>
<b>8 Gestione dinamica di RMI</b>	<b>173</b>
8.1 Introduzione . . . . .	174
8.2 Caricamento dinamico delle classi . . . . .	174
8.2.1 Il ClassLoader . . . . .	174
8.2.2 Caratteristiche del caricamento dinamico . . . . .	175
8.2.3 Gli scenari di utilizzo . . . . .	175
8.3 Stub e Skeleton: la evoluzione . . . . .	183
8.3.1 Stub e Skeleton versione JDK 1.1 . . . . .	183
8.3.2 Stub versione JDK 1.2 . . . . .	187
8.3.3 JDK 5: niente Stub e Skeleton! . . . . .	189
Note bibliografiche . . . . .	193
Spunti per lo studio individuale . . . . .	194
<b>9 RMI e Java Enterprise Edition</b>	<b>195</b>
9.1 Introduzione . . . . .	196
9.2 Java Enterprise Edition . . . . .	196
9.2.1 Le architetture multi-tier . . . . .	196
9.2.2 La architettura di Java EE . . . . .	197
9.3 Corba e IIOP . . . . .	198
9.4 Java RMI-IIOP . . . . .	200
9.4.1 Java RMI e Corba . . . . .	200
9.4.2 Le differenze tra RMI e RMI-IIOP . . . . .	200
9.4.3 Un esempio in Java RMI-IIOP . . . . .	201
Note bibliografiche . . . . .	206
Spunti per lo studio individuale . . . . .	207

# Prefazione

## Obiettivi

Questo libro su Java RMI ha due obiettivi. Il primo è quello di fornire allo studente il collegamento tra i tradizionali corsi di programmazione di rete, che sono basati sulla comunicazione basata su TCP/IP ed i socket, e i corsi di programmazione di applicazioni distribuite che tipicamente usano strumenti di alto livello come il modello a componenti distribuite (ambienti Enterprise) oppure il modello orientato ai servizi (ambienti Web Services). Quello che manca, e che si intende fornire con questo libro, è il collegamento che permette allo studente di comprendere in quale maniera Java Enterprise Edition, ad esempio, realizza la infrastruttura di comunicazione tra oggetti, utilizzando Java Remote Method Invocation oppure quanto del modello delle architetture orientate ai servizi (Service Oriented Architecture) derivi direttamente dai modelli di invocazione remota strettamente accoppiati (come Java RMI).

Quindi, il viaggio che si intende far intraprendere allo studente illustrerà come la programmazione di rete serve alla invocazione remota di metodi che servono poi a realizzare la infrastruttura di comunicazione. In un certo senso, si vuole usare la programmazione di socket e la invocazione remota di metodi come la programmazione assembly per la programmazione con un linguaggio evoluto, illustrando i meccanismi e le tecnologie coinvolte nell'ambiente Java. Tutto questo passerà anche attraverso la presentazione di alcuni concetti fondamentali per le architetture di sistemi distribuiti per contestualizzare la presenza di Java RMI.

Il secondo obiettivo è quello di fornire le basi per la programmazione distribuita di oggetti, sviluppando quindi un certo numero di esempi pratici che guideranno lo studente, a partire dalla implementazione con i socket, a realizzare semplici applicazioni che però serviranno ad evidenziare problematiche tipiche dei sistemi client-server e dei sistemi peer2peer.

Dal punto di vista dello stile di programmazione, la scelta è stata quella di evitare di sovraccaricare il codice, senza presentare, ad esempio, interfacce grafiche ma limitandosi alla shell in modo da limitare la quantità di codice da presentare. Il cosiddetto *coding style* di Java viene utilizzato, anche se, per ragioni di spazio e di efficacia grafica, molti dei commenti nel codice sono ridotti all'osso, e mancano del tutto i commenti JavaDoc. Questo stile di programmazione (utilizzato per rendere più agevole lo studio e la presentazione grafica) non è da ritenere un suggerimento, anzi! Sia ben chiaro allo studente che commenti adeguati e informativi sono estremamente utili per la manutenzione e la evoluzione del software e che la presentazione stringata è esclusivamente per limitare la lunghezza del codice stampato.

## Struttura del libro

Il libro consta di tre parti. Nella prima si introduce Java Remote Method Invocation attraverso l'utilizzo dei socket. Nella seconda si affrontano i diversi esempi di programmazione

usando Java RMI, allo scopo di introdurre tecniche e metodi utili per la realizzazione di semplici applicazioni remote. Nella terza parte si trovano alcuni approfondimenti sulla struttura interna di Java RMI, con la capacità di gestire a run-time le invocazioni, e RMI-IIOP.

## A chi si rivolge il libro

Il libro può essere utilizzato come testo di riferimento per corsi universitari di Programmazione Distribuita di I livello (terzo anno) oppure di II livello (primo anno) in Informatica e Ingegneria Informatica. Si suppone che gli studenti abbiano una buona conoscenza di Java, con esperienza nell'uso di Integrated Development Environment (IDE) e qualche conoscenza di base di Ingegneria del Software. Tra le competenze specifiche di Java che sono necessarie, una qualche semplice esperienza con i socket (anche in altri linguaggi) e una semplice conoscenza dei thread.

Il materiale del libro può essere usato per un corso da 4-6 CFU, a seconda del livello di dettaglio e delle esercitazioni da fare in laboratorio (qualora previsto).

## Gli strumenti utilizzati

**Composizione tipografica.** Per scrivere questo libro è stato utilizzato LaTeX ed in particolare la versione per Windows distribuita con MiKTeX 2.8 (<http://www.miktex.org>). TeXlipse 1.2.2 (<http://texlipse.sourceforge.net/>) è un plugin di Eclipse che è stato usato per facilitare la composizione.

Per la traduzione di programmi Java in LaTeX si è utilizzato, invece, Java2HTML versione 1.5.0 (<http://www.java2html.de>) che permette anche la conversione da Java in LaTeX, con il syntax-coloring ed è disponibile anche come plugin per Eclipse. Per la creazione delle immagini, si usa il plugin di Eclipse eDump 1.7.1 ([www.bdaum.de](http://www.bdaum.de)) che fornisce la possibilità di catturare immagini delle varie componenti di Eclipse con notevole flessibilità.

**Linguaggio e ambiente integrato di sviluppo.** In generale, si è utilizzata la ultima versione disponibile al momento della Java VM di Sun, vale a dire la versione Java SE Development Kit (JDK) 6 Update 11. Insieme ad essa, suggeriamo di scaricare in locale la documentazione Java disponibile sul sito, che porta una notevole quantità di informazioni e di guide per i vari argomenti.

Per la scrittura e la documentazione dei programmi di esempio si è usato Eclipse (<http://www.eclipse.org>), versione 3.4.1 con il plugin per eUML2 Free di Soyatec versione 3.2.1 (<http://www.soyatec.com/euml2/>).

## Convenzioni

**L'uso e l'abuso della lingua inglese.** In questo libro si cerca di evitare l'uso di forzate traduzioni di termini inglesi che sono oramai diventati di uso tecnico. Ben lungi dal prefigurarsi come sudditanza linguistica verso oltreoceano, questa scelta è motivata dal fatto che questo ben prepara gli studenti di oggi che, da professionisti dell'informatica di domani, si troveranno, necessariamente, a contatto con documentazione di tecnologia recente che sarà disponibile, probabilmente, solamente in inglese tecnico.

**Convenzioni tipografiche.** Per indicare una sessione di lavoro ad una shell (come ad esempio per la compilazione o esecuzione di un programma) si è scelto di utilizzare come esempio una semplice shell DOS<sup>1</sup>. Per rappresentare una sessione useremo la seguente convenzione:

```
Z:\>echo Ciao Ciao
```

```
Z:\>
```

Per indicare un listato di una classe Java di nome `Hello.java` si utilizzerà questo standard:

```
HelloWorld.java
```

```
1 // Una classe di esempio
2 public class HelloWorld {
3     public static void main(String[] args) {
4         System.out.println("Hello World!");
5     }
6 }
```

```
Fine: HelloWorld.java
```

In generale, comandi da shell vengono indicati in grassetto, come ad esempio, il compilatore Java **javac**.

---

<sup>1</sup>In alternativa, si può utilizzare un plugin per Eclipse chiamato WickedShell (<http://www.wickedshell.net>) che permette di creare shell molto flessibili come view di Eclipse.



# Prefazione alla seconda edizione

Questa edizione aggiunge alla precedente la terza parte, con gli argomenti avanzati: gestione dinamica e RMI-IIOP. Ovviamente, sono stati corretti anche gli errori, così come segnalati dai precisi e puntuali commenti che gli studenti del corso di Programmazione Distribuita del corso di Laurea in Informatica dell'Università di Salerno hanno fornito, durante l'anno accademico 2008-2009.

Ringrazio sentitamente tutti gli studenti che hanno inviato commenti, suggerimenti, correzioni, perchè il libro ne ha beneficiato notevolmente, e con questo, spero, ne beneficeranno anche i loro colleghi che lo useranno per lo studio negli anni successivi.

Anche i diritti di questa II edizione saranno interamente versati ad Emergency, nel ricordo di Teresa Sarti Strada, scomparsa il 1° settembre 2009.

