Java RMI
Remote Method Invocation

Tópicos

- Definição
- Arquitetura
- Tutorial

Definição

- RMI: paradigma de comunicação através do qual objetos de processos diferentes podem comunicar-se uns com os outros.
  - O objeto de um processo pode invocar métodos de um objeto de outro processo

Definição

- RMI Java: Extensão de software à máquina virtual Java remota (JVM) que permite, sintaticamente, a invocação de métodos em uma outra JVM. Propicia a implementação natural do projeto Java OO em ambiente distribuído.

Aplicações distribuídas com RMI Java

- Tipicamente cliente-servidor
- Servidor cria objetos remotos com os quais clientes interagem
- Localização de objetos remotos
- Comunicação com objetos remotos
- Carregamento dinâmico de código de classes

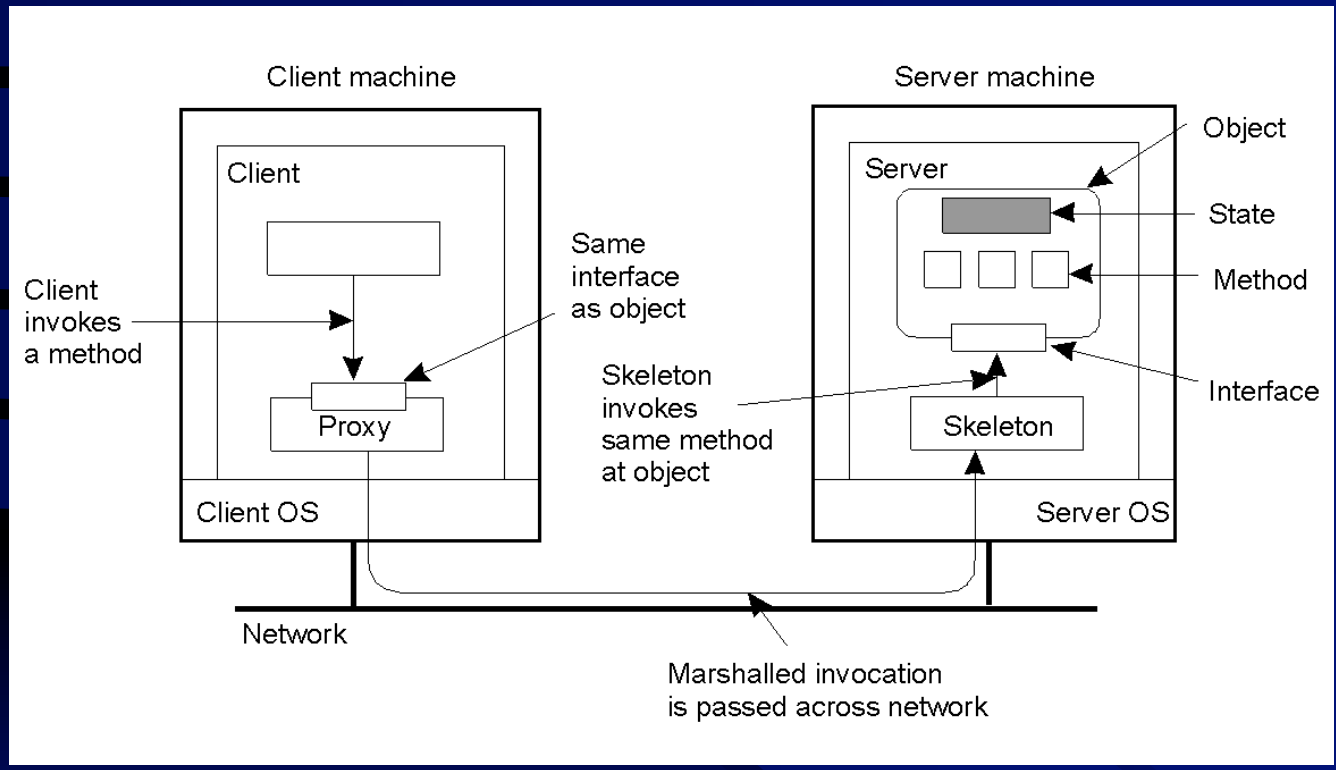
Objetos remotos x objetos locais *(semelhanças)*

- Podem ser parâmetro ou resultado de chamada de método
- A passagem de parâmetros de objetos locais é feita por referência na chamada de método local assim como a passagem de parâmetros de objetos remotos em chamada de método remoto também é feita por referência
- *Casting* : objetos remotos -> interfaces remotas

Objetos remotos x objetos locais *(diferenças)*

- A interação com objetos remotos (pelos clientes) é feita via interface (para *stub* do objeto remoto)
- A passagem de parâmetros e resultado na chamada de método remoto é por cópia para objetos locais
- Tipo de falha na chamada de método remoto é mais complexo (erro de comunicação é possível)

Arquitetura RMI



Extensões

- Coleta de lixo de objetos remotos
- Ativação de objetos persistentes sob demanda

RMI API

- java.rmi
  - *Remote*, RemoteException, Naming
- java.rmi.server
  - RemoteObject, RemoteStub, RemoteServer, UnicastRemoteObject
- java.rmi.registry
  - Registry

Interfaces Remotas Java RMI

- Estendem a interface *Remote*
- Seus métodos têm que tratar exceções do tipo *RemoteException*

```
import java.rmi.*
public interface myInterface extends Remote{
   int getVersion() throws RemoteException;
}
```

RMIregistry

- É o *binder* do Java RMI
- Deve executar em cada servidor de objetos
- Mantém uma tabela dos objetos remotos do servidor onde executa
- Acessado através de métodos da classe *Naming*
  - *void rebind (String name, Remote obj)*
  - *void unbind (String name, Remote obj)*
  - *Remote lookup (String name)*

RMIregistry

- Os métodos da classe Naming recebem nomes na forma de URLs
  - rmi://computerName:port/objectName

O objeto remoto/servidor

- Implementa a interface remota
- Estende a classe UnicastRemoteObject
- Registra-se no RMIregistry

```
import java.rmi.*;
import java.rmi.server.*;
public class exemplo_Servidor
extends UnicastRemoteObject
implements myInterface {
...}
```

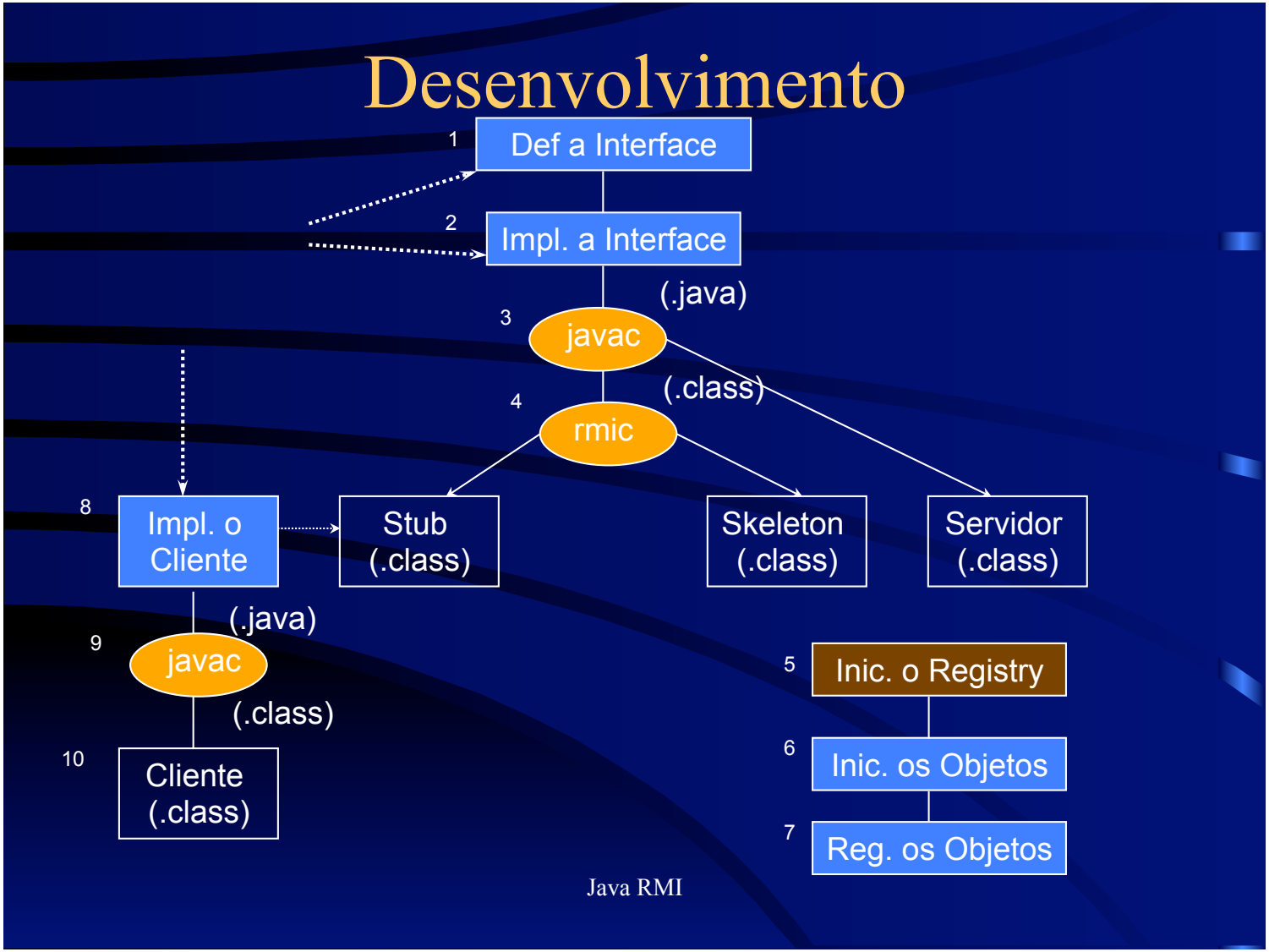
O cliente

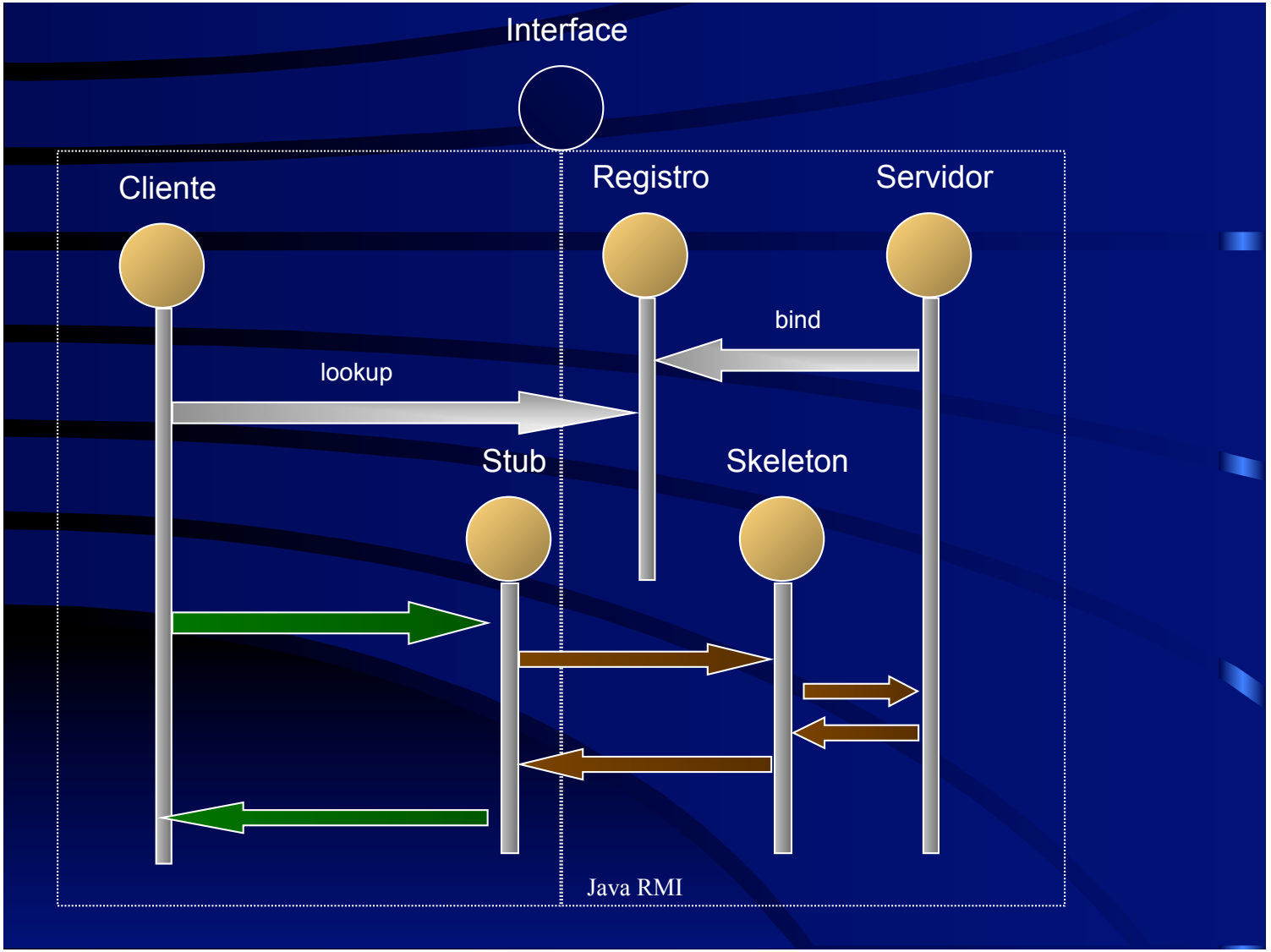
- Obtém uma referência remota a partir do RMIregistry
- Invoca os métodos remotos a partir da referência obtida

Desenvolvimento

- Defina a Interface
- Implemente a Interface (Servidor)
- Gere os stubs
- Inicialize o Registry
- Inicialize/Registre o Servidor
- Implemente/Inicialize o Cliente

Desenvolvimento




Interface
Cliente
Registro
Servidor
bind
lookup
Stub
Skeleton
Java RMI

Tutorial / Interface

```
// Counter.java
import java.rmi.*;
public interface Counter
extends Remote {
  int nextValue() throws RemoteException;
}
```

Tutorial / Servidor

```
// example_Counter.java
import java.rmi.*;
import java.rmi.server.*;
public class example_Counter
extends UnicastRemoteObject
implements Counter {
private int value;
public int nextValue() throws RemoteException {
   return value++;}
public example_Counter() throws RemoteException {
   super();
   value = 0;}
}
```

Servidor / Registrando

```
// CounterServer.java
import java.rmi.*;
public class CounterServer {
public static void main(String args[])  {
try { Counter c = new example_Counter();
  Naming.rebind("Contador", c);
  System.out.println("Counter ready.");}
catch (Exception e) {
  System.out.println("Erro:"+e.getMessage());}
}
}
```

Tutorial / Cliente

```
// CounterCliente.java
import java.rmi.*;
public class CounterClient {
public static void main(String args[]) {
try {
   String host = args[0];
   Counter c = (Counter)
       Naming.lookup("rmi://"+host+"/Contador");
   System.out.println("Contador: "+c.nextValue());
} catch (Exception e) {
   System.out.println("Erro: "+e.getMessage());}
}
}
```

Tutorial / Executando

```
> javac Counter.java
> javac example_Counter.java
> javac CounterClient.java
> javac CounterServer.java
> rmic example_Counter
> start rmiregistry
> start java CounterServer
> java CounterClient <reg_host>
```

Referências

- Getting Started Using RMI

http://java.sun.com/products/jdk/1.2/docs/guide/rmi/getstart.doc.html

- RMI Specification. http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html
- Keith Edwards. *Core Jini*. Appendix A: An RMI Primer. Prentice Hall, 1999.