Java RMI
(*Remote Method Invocation*)

Prof. Orlando Loques - IC/UFF

versão 16-05-2001

Ref: Apresentação de Sérgio Teixeira de Carvalho, IC-UFF

Programação Java RMI

Conceitos

Mecanismo

Exemplo de Invocação Remota

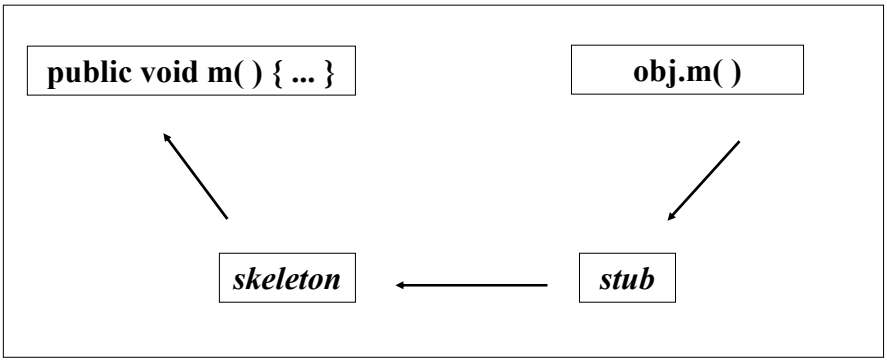
Request-Release com Monitor Remoto

Sumário

Java RMI - Conceitos

- **Invocação de métodos como fundamento**
- **Obtém referência do objeto remoto**
- **Invoca métodos diretamente ao objeto remoto**
- **Semelhança com o RPC (*Remote Procedure Call*)**

Mecanismo



compilador RMI (*rmic*) gera *stub* e *skeleton*

Mecanismo

Classe UnicastRemoteObject

- **objetos remotos herdam métodos desta classe**
- **permite que objetos sejam "exportados"**

RMI Registry

- **serviço que registra e recupera referências a objetos através do nome**

Exemplo: Definição da Interface Remota

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MsgIntf extends Remote {
        String mostraMsg() throws RemoteException;
}
```

Ex: Implementação de Método Remoto

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
import java.rmi.server.UnicastRemoteObject;

public class MsgImpl extends UnicastRemoteObject
                     implements MsgIntf {

    public MsgImpl() throws RemoteException {
    }

    public String mostraMsg () {
         return ("Ok!");
    }
        .
        .
        .
```

Exemplo: Registro do Objeto Remoto

```
        .
        .
        .
     public static void main(String args[]) {
      if (System.getSecurityManager() == null) {
        System.setSecurityManager(new
                        RMISecurityManager());
      }
      try {
        MsgIntf msg = new MsgImpl();
        Naming.rebind("rmi://murici/MsgSrv", msg);
        System.out.println("Msg ativo...");
      }
      catch (Exception e) { }
    }
}
```

Cliente Invocando Método Remoto

```
import java.rmi.Naming;
import java.rmi.RemoteException;

public class MsgCliente {

    public static void main (String[] args) {
       String mensagem;

       MsgIntf msg = (MsgIntf) Naming.lookup
                                  ("rmi://murici/MsgSrv");
       mensagem = msg.mostraMsg();
       System.out.println (mensagem);
    }
}
```

Interface Remota do Monitor

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MonitorIntf extends Remote {
   boolean ocupado = false;

   public void request() throws RemoteException;
   public void release() throws RemoteException;
}
```

Implementação do Monitor Remoto

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
import java.rmi.server.UnicastRemoteObject;

public class MonitorImpl extends UnicastRemoteObject
                         implements MonitorIntf {
   private boolean ocupado = false;

   public MonitorImpl() throws RemoteException { }

   public synchronized void request() {
    while (ocupado) {
      try { wait(); } catch (InterruptedException e) { } }
    ocupado = true;
   }

   public synchronized void release() {
      ocupado = false;
      notifyAll();          .
   }                        .
                            .
```

Registro do Monitor Remoto

```
        .
        .
        .
    public static void main(String args[]) {
      if (System.getSecurityManager() == null) {
        System.setSecurityManager(new
                        RMISecurityManager());
      }
      try {
        MonitorIntf monitor = new MonitorImpl();
        Naming.rebind("rmi://murici/Monitor", monitor);
        System.out.println("Monitor ativo no Registry..");
      }
      catch (Exception e) { }
    }
}
```

Criação das Threads

```
import java.rmi.Naming;
import java.rmi.RemoteException;

public class RequestRelease {
   public static void main(String[] args) {
      MonitorIntf m = null;

      try {
        m = (MonitorIntf) Naming.lookup(//murici/Monitor);
      } catch (Exception e) { }

      Usuario us_1 = new Usuario(m);
      Usuario us_2 = new Usuario(m);

      us_2.start();
      us_1.start();
   }
}
```

Definição da Classe Usuário (*Thread*)

```
public class Usuario extends Thread {
   private MonitorIntf monitor;

   public Usuario (MonitorIntf m) {
      monitor = m;
   }

   public void run() {
      for (int i = 0; i < 5; i++) {
         monitor.request();
                    ... usa recurso ...
         monitor.release();
      }
   }
}
```