Il linguaggio Java
Remote Method Invocation

Programmi d'esempio

Interfaccia remota

```
public interface Calculator
        extends java.rmi.Remote {
   public long add(long a, long b)
      throws java.rmi.RemoteException;

   public long sub(long a, long b)
      throws java.rmi.RemoteException;

   public long mul(long a, long b)
      throws java.rmi.RemoteException;

   public long div(long a, long b)
      throws java.rmi.RemoteException;
}
```

Oggetto remoto

```
import java.rmi.server.*;

public class CalculatorImpl
                extends UnicastRemoteObject
              implements Calculator {

  public CalculatorImpl()
                throws java.rmi.RemoteException {
    super();
  }

  public long add(long a, long b)
    throws java.rmi.RemoteException {
    return a + b;
  }

  public long sub(long a, long b)
    throws java.rmi.RemoteException {
    return a - b;
  }

  public long mul(long a, long b)
    throws java.rmi.RemoteException {
    return a * b;
  }

  public long div(long a, long b)
    throws java.rmi.RemoteException {
    return a / b;
  }
}
```


Server

```
import java.rmi.Naming;

public class CalculatorServer {

  public CalculatorServer() {
    try {
      Calculator c = new CalculatorImpl();
      Naming.rebind("localhost:1099/Calculator",
                                                    c);
  } catch (Exception e) {
      System.out.println("Trouble: " + e);
    }
  }

  public static void main(String args[]) {
    new CalculatorServer();
  }
}
```

Client

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class CalculatorClient {

    public static void main(String[] args) {
        try {
            Calculator c = (Calculator)
            Naming.lookup("rmi://localhost:1099/Calculator");
            System.out.println( c.sub(4, 3) );
            System.out.println( c.add(4, 5) );
            System.out.println( c.mul(3, 6) );
            System.out.println( c.div(9, 3) );
        }
        catch (MalformedURLException murle) {
            System.out.println();
            System.out.println("MalformedURLException");
            System.out.println(murle);
        }
        catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
            System.out.println(re);
        }
        catch (NotBoundException nbe) {
            System.out.println();
            System.out.println("NotBoundException");
            System.out.println(nbe);
        }
        catch (
            java.lang.ArithmeticException ae) {
            System.out.println();
            System.out.println( "ArithmeticException");
            System.out.println(ae);
        }
    }
}
```


Hello: interfaccia, implementazione

```
import java.rmi.*;

public interface Hello extends java.rmi.Remote {
    public String sayHello() throws RemoteException;
    public MessageObject getMessageObject() throws
                                                    RemoteException;
}
////////////////////////////////////////
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
import java.net.MalformedURLException;

public class HelloImpl extends UnicastRemoteObject
                                                    implements Hello {
    public HelloImpl() throws RemoteException {
        super();
    }
    public String sayHello() throws RemoteException {
        return "Hello!";
    }
    public MessageObject getMessageObject() throws
                                                    RemoteException {
        return new MessageObject();
    }
}
```

MessageObject

```
public class MessageObject implements Serializable {
  static  int number = 0;
  private int objNumber;

  public MessageObject()  {
   objNumber = number;
   System.out.println( "MessageObject: Class Number is #"
                       + number +    " Object Number is #"
                       +  objNumber );
   number = number + 1;
  }

  public int getNumberFromObject() {return objNumber;}

   public int getNumberFromClass() {return number;}
}
```


Hello: server

```
import java.net.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.LocateRegistry;

public class RMIServer {

   private static final String HOST_NAME   = "dini";

   public static void main( String[] args ) {
      try {
         rmi = new RMIServer();
      } catch ( java.rmi.UnknownHostException uhe ) {
         System.out.println( "Wrong name " + HOST_NAME);
      } catch ( RemoteException re ) {
         System.out.println( "Error starting service" ); System.out.println( "" + re );
      } catch ( MalformedURLException mURLe ) {
         System.out.println( "Internal error" + mURLe );
      } catch ( NotBoundException nbe ) {
         System.out.println( "Not Bound" ); System.out.println( "" + nbe );
      }
   }  // main

   public RMIServer() throws RemoteException, MalformedURLException,
                                                          NotBoundException {
      System.out.println( "Registry on host " + HOST_NAME);
      Hello h = new HelloImpl();
      System.out.println( "Remote HelloService implementation object created" );
      String urlString = "//" + HOST_NAME + ":" + "/" +  "HelloService";
      Naming.rebind( urlString, h );
      System.out.println( "Bindings Finished, waiting for client requests." );
   }
}
```

Hello: client

```
import java.net.*;
import java.io.*;

import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.LocateRegistry;

public class RMIClient {
     private static final String HOST_NAME   = "dini";
     // Instance of ourselves
      private static RMIClient  rmi;
     // Instance of the Root Object(s)
     private static Hello hello;

      public static void main ( String[] args )  {
           rmi = new RMIClient();
     }  // main

   public RMIClient() {
     Hello        h;
     String       helloString;
     MessageObject mo;

     try {
       h = (Hello)Naming.lookup( "rmi://" + HOST_NAME + ":" + "/HelloService" );
       System.out.println( "HelloService lookup successful" );
       helloString = h.sayHello();
       System.out.println( "The server says: " + helloString );
       for ( int i = 0; i< 10; i++ ) {
          mo = h.getMessageObject();
          System.out.println( "MessageObject: Class Number is #" +  mo.getNumberFromClass() +
                              " Object Number is #" + mo.getNumberFromObject() );
       }
     } catch ( java.rmi.UnknownHostException uhe ) {
       System.out.println( "Wrong host name " + HOST_NAME);
     } catch ( RemoteException re ) {
       System.out.println( "A remote Exception when requesting the HelloService");
       System.out.println( "" + re );
     } catch ( MalformedURLException mURLe ) {
       System.out.println( "There is a problem with the rmi-URL" );
       System.out.println( "" + mURLe );
     } catch ( NotBoundException nbe ) {
       System.out.println( "" + nbe );
     }
   }

}  // class RMIClient
```


Esecuzione

```
//RMIServer
Registry created on host computer dini
Remote HelloService implementation object created
Bindings Finished, waiting for client requests.
MessageObject: Class Number is #0 Object Number is #0
MessageObject: Class Number is #1 Object Number is #1
MessageObject: Class Number is #2 Object Number is #2
MessageObject: Class Number is #3 Object Number is #3
MessageObject: Class Number is #4 Object Number is #4
MessageObject: Class Number is #5 Object Number is #5
MessageObject: Class Number is #6 Object Number is #6
MessageObject: Class Number is #7 Object Number is #7
MessageObject: Class Number is #8 Object Number is #8
MessageObject: Class Number is #9 Object Number is #9

//RMIClient
HelloService lookup successful
The server says: Hello!
MessageObject: Class Number is #0 Object Number is #0
MessageObject: Class Number is #0 Object Number is #1
MessageObject: Class Number is #0 Object Number is #2
MessageObject: Class Number is #0 Object Number is #3
MessageObject: Class Number is #0 Object Number is #4
MessageObject: Class Number is #0 Object Number is #5
MessageObject: Class Number is #0 Object Number is #6
MessageObject: Class Number is #0 Object Number is #7
MessageObject: Class Number is #0 Object Number is #8
MessageObject: Class Number is #0 Object Number is #9
```

Compute: compute, task

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

public interface Compute extends Remote {
    Object executeTask(Task t)
                    throws RemoteException;
}

////////////////////////////////////////

package rmi.compute.compute;
import java.io.Serializable;

public interface Task extends Serializable
{
    Object execute();
}
```


Compute: implementazione, server

```
package rmi.compute.engine;
import java.rmi.*;
import java.rmi.server.*;
import rmi.compute.compute.*;

public class ComputeEngine
          extends UnicastRemoteObject
          implements Compute
{
  public ComputeEngine()
          throws RemoteException {
    super();
  }

  public Object executeTask(Task t) {
    return t.execute();
  }

  public static void main(String[] args) {
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(
        new RMISecurityManager());
    }
    String name = "//host/Compute";
      try {
        Compute engine = new ComputeEngine();
        Naming.rebind(name, engine);
        System.out.println("ComputeEngine
                                        bound");
      } catch (Exception e) {
        System.err.println("ComputeEngine
                                  exception: " +
                                e.getMessage());
        e.printStackTrace();
      }
  }
}
```

Compute: client

```
package rmi.compute.client;

import java.rmi.*;
import java.math.*;
import rmi.compute.compute.*;

public class ComputePi {
  public static void main(String args[]) {
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(
           new RMISecurityManager());
    }
    try {
      String name = "//" + args[0] + "/Compute";
      Compute comp =
                  (Compute) Naming.lookup(name);
       Pi task =
              new Pi(Integer.parseInt(args[1]));
      BigDecimal pi =
           (BigDecimal) (comp.executeTask(task));
      System.out.println(pi);
    } catch (Exception e) {
      System.err.println("ComputePi exception: "
                              + e.getMessage());
      e.printStackTrace();
    }
  }
}
```


Compute: Pi.java

```
package rmi.compute.client;
import rmi.compute.compute.*;
import java.math.*;

public class Pi implements Task {

    /** constants used in pi computation */
    private static final BigDecimal ZERO =
        BigDecimal.valueOf(0);
    private static final BigDecimal  ONE =
        BigDecimal.valueOf(1);
    private static final BigDecimal FOUR =
        BigDecimal.valueOf(4);

    /** rounding mode to use during pi
        computation */
    private static final int roundingMode =
        BigDecimal.ROUND_HALF_EVEN;

    /** digits of precision after the decimal
        point */
    private int digits;

    /**
     * Construct a task to calculate pi to the
     * specified precision.
     */
    public Pi(int digits) {
        this.digits = digits;
    }

    /**
     * Calculate pi.
     */
    public Object execute() {
        return computePi(digits);
    }
// continua
```

Compute: Pi.java

```
    public static BigDecimal computePi(int digits) {
        int scale = digits + 5;
        BigDecimal arctan1_5 = arctan(5, scale);
        BigDecimal arctan1_239 = arctan(239, scale);
        BigDecimal pi = arctan1_5.multiply(FOUR).subtract(
                                    arctan1_239).multiply(FOUR);
        return pi.setScale(digits,
                           BigDecimal.ROUND_HALF_UP);
    }
    /**
     * Compute the value, in radians, of the arctangent of
     * the inverse of the supplied integer to the speficied
     * number of digits after the decimal point.  The value
     * is computed using the power series expansion for the
     * arc tangent:
     *
     * arctan(x) = x - (x^3)/3 + (x^5)/5 - (x^7)/7 +
     *      (x^9)/9 ...
     */
    public static BigDecimal arctan(int inverseX,
                                    int scale)
    {
        BigDecimal result, numer, term;
        BigDecimal invX = BigDecimal.valueOf(inverseX);
        BigDecimal invX2 =
            BigDecimal.valueOf(inverseX * inverseX);

        numer = ONE.divide(invX, scale, roundingMode);

        result = numer;
        int i = 1;
        do {
            numer =
                numer.divide(invX2, scale, roundingMode);
            int denom = 2 * i + 1;
            term =
                numer.divide(BigDecimal.valueOf(denom),
                             scale, roundingMode);
            if ((i % 2) != 0) {
                result = result.subtract(term);
            } else {
                result = result.add(term);
            }
            i++;
        } while (term.compareTo(ZERO) != 0);
        return result;

    }
}
```


Security policy

```
grant {
    permission java.net.SocketPermission
              "*:1024-65535", "connect,accept";
    permission java.net.SocketPermission
                                "*:80", "connect";
};

// questa policy permette al codice caricato
// da un qualunque code base di fare le
// seguenti cose:
// 1) connettersi a o accettare le connessioni
//    su porte non privilegiate su qualunque
//    host
// 2) connettersi alla porta 80 (HTTP)
```

Titolo

```
import java.io.*;
public class MiaClasse {
   public static void main(String[] args)
                                          throws IOException {
      // body
   }
}
```