Java RMI

Remote method invocation(RMI) allow a java object to invoke method on an object running on another machine. RMI provide remote communication between java program. RMI is used for building distributed application.

Concept of RMI application

A RMI application can be divided into two part,**Client** program and **Server** program. A **Server** program creates some remote object, make their references available for the client to invoke method on it. A **Client** program make request for remote objects on server and invoke method on them. **Stub** and **Skeleton** are two important object used for communication with remote object.

Stub

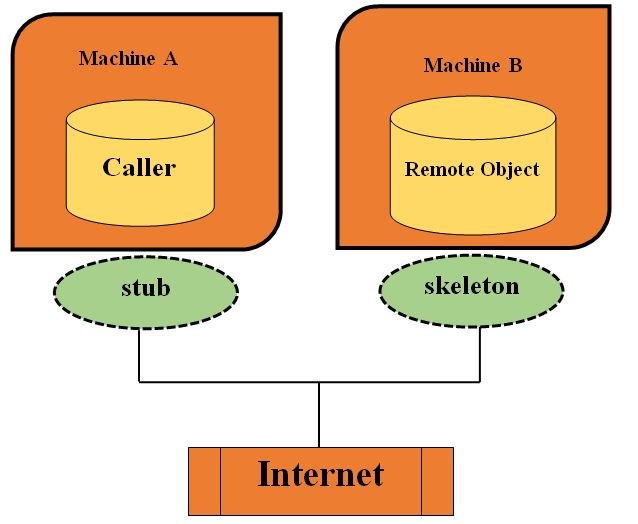
In RMI, a stub is an object that is used as a Gateway for the client-side. All the outgoing request are sent through it. When a client invokes the method on the stub object following things are performed internally:

1. A connection is established using Remote Virtual Machine.
2. It then transmits the parameters to the Remote Virtual Machine. This is also known as Marshals
3. After the 2nd step, it then waits for the output.
4. Now it reads the value or exception which is come as an output.
5. At last, it returns the value to the client.

Skeleton

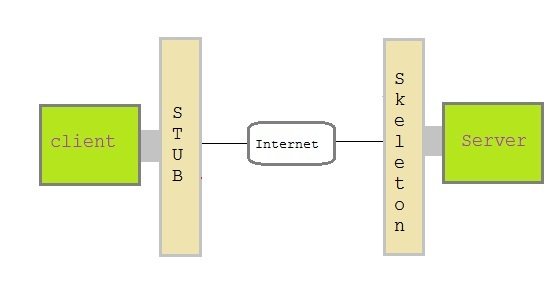
In RMI, a skeleton is an object that is used as a Gateway for the server-side.All the incoming request are sent through it. When a Server invokes the method on the skeleton object following things are performed internally:

1. All the Parameters are read for the remote method.
2. The method is invoked on the remote object.
3. It then writes and transmits the parameters for the result. This is also known as Marshals.



Stub and Skeleton

**Stub** act as a gateway for Client program. It resides on Client side and communicate with **Skeleton** object. It establish the connection between remote object and transmit request to it.



Skeleton object resides on server program. It is responsible for passing request from **Stub** to remote object.

Creating a Simple RMI application involves following steps

* Define a remote interface.
* Implementing remote interface.
* create and start remote application
* create and start client application

Define a remote interface

A remote interface specifies the methods that can be invoked remotely by a client. Clients program communicate to remote interfaces, not to classes implementing it. To be a remote interface, a interface must extend the **Remote** interface of **java.rmi** package.

import java.rmi.\*;

public interface AddServerInterface extends Remote

{

public int sum(int a,int b);

}

Copy

Implementation of remote interface

For implementation of remote interface, a class must either extend **UnicastRemoteObject** or use exportObject() method of **UnicastRemoteObject** class.

import java.rmi.\*;

import java.rmi.server.\*;

public class Adder extends UnicastRemoteObject implements AddServerInterface

{

Adder()throws RemoteException{

super();

}

public int sum(int a,int b)

{

return a+b;

}

}

Copy

Create AddServer and host rmi service

You need to create a server application and host rmi service **Adder** in it. This is done using rebind() method of **java.rmi.Naming** class. rebind() method take two arguments, first represent the name of the object reference and second argument is reference to instance of **Adder**

import java.rmi.\*;

import java.rmi.registry.\*;

public class AddServer {

public static void main(String args[]) {

try {

AddServerInterface addService=new Adder();

Naming.rebind("AddService",addService); //addService object is hosted with name AddService

}

catch(Exception e) {

System.out.println(e);

}

}

}

Copy

Create client application

Client application contains a java program that invokes the lookup() method of the **Naming** class. This method accepts one argument, the **rmi** URL and returns a reference to an object of type **AddServerInterface**. All remote method invocation is done on this object.

import java.rmi.\*;

public class Client {

public static void main(String args[]) {

try{

AddServerInterface st = (AddServerInterface)Naming.lookup("rmi://"+args[0]+"/AddService");

System.out.println(st.sum(25,8));

}

catch(Exception e) {

System.out.println(e);

}

}

}

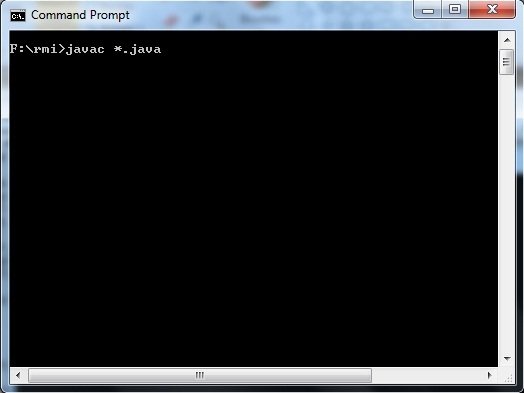
Copy

Steps to run this RMI application

Save all the above java file into a directory and name it as "rmi"

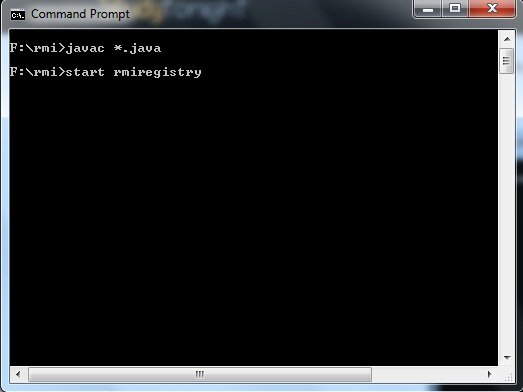
* compile all the java files

javac \*.java



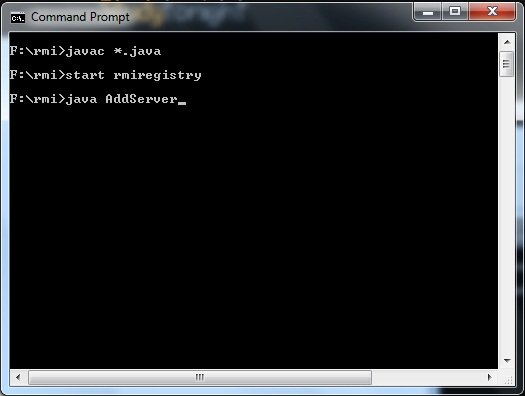
* Start RMI registry

start rmiregistry



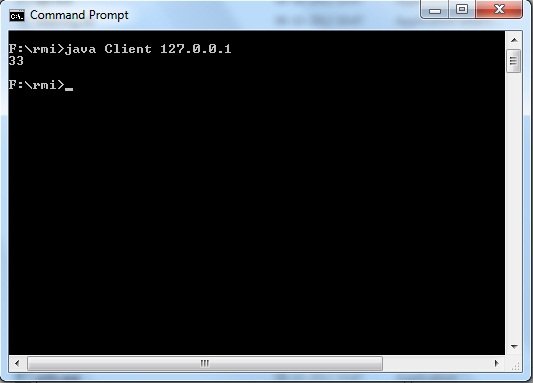
* Run Server file

java AddServer



* Run Client file in another command prompt abd pass local host port number at run time

java Client 127.0.0.1



Example:

**Program: Power.java**

import java.rmi.\*;

public interface Power extends Remote

{

public int power1()throwsRemoteException;

}

Copy

**Program: PowerRemote.java**

import java.rmi.\*;

import java.rmi.server.\*;

import java.util.Scanner;

public class PowerRemote extends UnicastRemoteObject implements Power

{

PowerRemote()throws RemoteException

{

super();

}

public int power1(int z)

{

int z;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the base number ::");

int x = sc.nextInt();

System.out.println("Enter the exponent number ::");

int y = sc.nextInt();

z=y^x;

System.out.println(z);

}

}

Copy

**MyServer.java**

import java.rmi.\*;

import java.rmi.registry.\*;

public class MyServer

{

public static void main(String args[])

{

try

{

Power stub=new PowerRemote();

Naming.rebind("rmi://localhost:1995/shristee",stub);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

Copy

**MyClient.java**

import java.rmi.\*;

public class MyClient

{

public static void main(String args[])

{

try

{

Power stub=(Power)Naming.lookup("rmi://localhost:1995/shristee");

System.out.println(stub.power1());

}

catch(Exception e){}

}

}

Copy

