CS454: Software Engineering 3

Lab 3 - RMI

Cairo University, Faculty of
Computers and Artificial Intelligence

Contents

RMI – Remote Method Invocation

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects ***stub*** and ***skeleton***.

Understanding Stub and Skeleton

A **remote object** is an object whose method can be invoked from another JVM.

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

CS454: Software Engineering 3

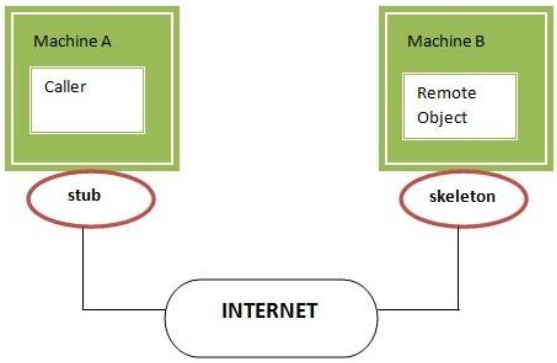
Lab 3 - RMI

Cairo University, Faculty of
Computers and Artificial Intelligence

skeleton

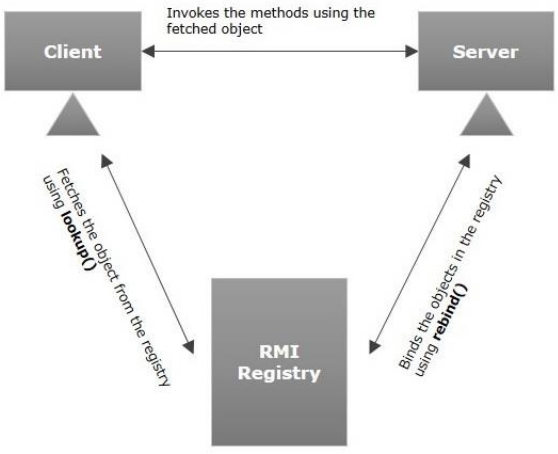
The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.



RMI Registry

RMI registry is a namespace on which all server objects are placed. Each time the server creates an object, it registers this object with the RMIregistry (using bind() or reBind() methods). These are registered using a unique name known as bind name.
To invoke a remote object, the client needs a reference of that object. At that time, the client fetches the object from the registry using its bind name (using lookup() method).
The following illustration explains the entire process:



Steps to create an RMI Program

The is given the 6 steps to write the RMI program.

CS454: Software Engineering 3

Lab 3 - RMI

Cairo University, Faculty of
Computers and Artificial Intelligence

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

RMI in Java

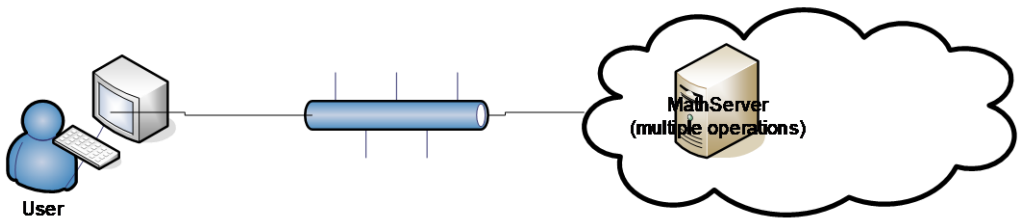
java.rmi.* AND **java.rmi.registry.***

To create a remote interface → extends **java.rmi.Remote** interface

To add the implementation of the remote interface→ extends **java.rmi.UnicastRemoteObject** and define a constructor that declares **RemoteException**

RMI Helloworld

In the rmi application, both client and server interacts with the remote interface. The client application invokes methods on the proxy object, RMI sends the request to the remote JVM. The return value is sent back to the proxy object and then to the client application.



Step 1: Create the remote interface

```
import java.rmi.Remote;

import java.rmi.RemoteException;


public interface IRemoteMath extends Remote {

     double add(double i, double j) throws RemoteException;
     double subtract(double i, double j) throws RemoteException;
}
```

Step 2: Provide the implementation of the remote interface

```
import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;
```

CS454: Software Engineering 3

Lab 3 - RMI

Cairo University, Faculty of
Computers and Artificial Intelligence

```
public class RemoteMathServant extends UnicastRemoteObject
implements IRemoteMath {

     protected RemoteMathServant() throws RemoteException {
          super();
          // TODO Auto-generated constructor stub
     }

     public double add ( double i, double j ) throws
RemoteException {
         return (i+j);
    }

    public double subtract ( double i, double j ) throws
RemoteException {
         return (i-j);
    }
}
```

Step 3: Compile the implementation class and create the stub and skeleton objects using the rmic tool

```
javac *.java
rmic AdderRemote
```

Step 4: Start the registry service by rmiregistry tool

```
rmiregistry 5000
```

Step 5: Create and start the remote application

```
import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;


public class MathServer {

     public static void main(String args[]){

         try{
             IRemoteMath remoteMath = new RemoteMathServant();
             Registry registry = LocateRegistry.getRegistry();
             registry.bind("Compute", remoteMath );
             System.out.println("Math server ready");
         }catch(Exception e) {
```

Cairo University, Faculty of
Computers and Artificial Intelligence


```
            e.printStackTrace();
        }
    }
}
```

```
java MyServer
```

Step 6: Create and start the client application

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;


public class MathClient {
    public static void main(String[] args) {
        try {

            Registry registry =
LocateRegistry.getRegistry("localhost");
            IRemoteMath remoteMath = (IRemoteMath)
registry.lookup("Compute");

            System.out.println( "1.7 + 2.8 = " +
remoteMath.add(1.7, 2.8) );
            System.out.println( "6.7 - 2.3 = " +
remoteMath.subtract(6.7, 2.3) );
        }
        catch( Exception e ) {
            System.out.println( e );
        }
    }
}
```

```
java MyClient
```

Run RMI applications from Eclipse

https://www.ejbtutorial.com/java-rmi/new-easy-tutorial-for-java-rmi-using-eclipse

CS454: Software Engineering 3

Lab 3 - RMI

Cairo University, Faculty of
Computers and Artificial Intelligence

Task

Write an RMI application for a client that add/view records of student data written in a file stored in the server.

The students data: name, id , level, group

The Server Functions: addStudent(Student s)

showAllStudents()

References

1. https://www.tutorialspoint.com/java_rmi/java_rmi_introduction.htm
2. https://www.javatpoint.com/RMI
3. https://www.ejbtutorial.com/java-rmi/new-easy-tutorial-for-java-rmi-using-eclipse