Tribhuvan University

Institute of Engineering

Pulchowk Campus

Distributed Systems

Lab 2

IMPLEMENT RMI TO ADD TWO NUMBERS

Submitted by:

Bishal Katuwal

075BCT028

Submitted to:

Department of Electronics and Computer Engineering

Pulchowk Campus

Submitted on:

June 22, 2022

Title

Implement RMI to add two numbers

Basic Theory

RMI is a pure Java solution to Remote Procedure Calls (RPC) and is used to create the distributed applications in java. It is a component of the standard Java application programming interface (API), which uses object serialization and defines object interfaces as Java Interfaces. Since the original implementation depends on JVM class-representation mechanisms, RMI only supports making calls from one JVM to another. A typical implementation of RMI includes stub and skeleton objects as shown below:

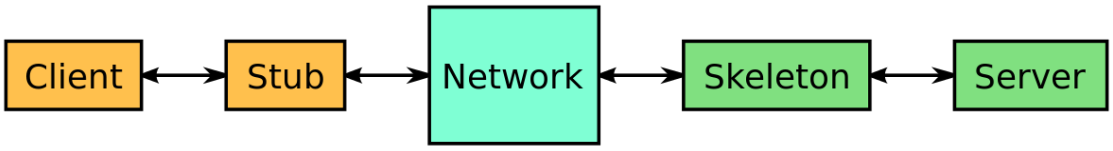

Figure 1: RMI in Java

Stub

The stub is a client side object on the client machine builds an information block and sends this information block to the server. This block consists of an identifier of the remote object to be used, method name which is to be invoked and parameters to the remote JVM.

Skeleton

The skeleton is a server side object that passes the request from the stub object to the remote object. It calls the desired method on the real object present on the server and it forwards the parameters received from the stub object to the method.

Steps in RMI

There are following steps in RMI:

1. Define remote interfaces
2. Implement client and remote objects.
3. Compile the source and generate stubs and skeletons.
4. Start the registry service by the rmiregistry tool.
5. Run the server application.
6. Run the client application.

Codes:

1. Creating the remote interface

```
import java.rmi.*;
public interface Adder extends Remote{
    public int add(int x,int y)throws RemoteException;
}
```

2. Providing the implementation of the remote interface

```
public class AdderRemote implements Adder{
    public int add(int a,int b){
        return a+b;
    }
}
```

3. Creating the stub and skeleton objects using the rmic tool

```
rmic AdderRemote
```

4. Starting the registry service by the rmiregistry tool

```
rmiregistry 5000
```

5. Creating and running the server application

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
public class MyServer extends AdderRemote{
    public Server() {}
    public static void main(String args[])
    {
        try
        {
            AdderRemote obj = new AdderRemote();
            Adder stub = (Adder) UnicastRemoteObject.exportObject(obj, 0);
            Registry registry = LocateRegistry.getRegistry();
            registry.bind("Bishal", stub);
        }
        catch(Exception ae)
        {
            System.out.println(ae);                }
        }
    }
}
```

6. Creating and running the client application

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class MyClient
{
    public static void main(String args[])
    {
        try
        {
            Registry registry = LocateRegistry.getRegistry(null);
            Adder stub = (Adder)registry.lookup("Bishal");
            int answer = stub.add(10,10);
            System.out.println(answer);
        }
        catch(Exception ae)
        {
            System.out.println(ae);
        }
    }
}
```

Source Code

Adder.java

```
import java.rmi.*;
public interface Adder extends Remote{
public int add(int x,int y)throws RemoteException;
}
```

AdderRemote.java

```
public class AdderRemote implements Adder{
    public int add(int a,int b){
        return a+b;
    }
}
```

MyClient.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class MyClient
{
    public static void main(String args[])
    {
        try
        {
            Registry registry = LocateRegistry.getRegistry(null);
            Adder stub = (Adder)registry.lookup("Bishal");
            int answer = stub.add(10,10);
            System.out.println(answer);
        }
        catch(Exception ae)
        {
            System.out.println(ae);
        }
    }
}
```

MyServer.java

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
public class MyServer extends AdderRemote{
    public Server() {}
    public static void main(String args[])
    {
        try
        {
            AdderRemote obj = new AdderRemote();
            Adder stub = (Adder) UnicastRemoteObject.exportObject(obj, 0);
            Registry registry = LocateRegistry.getRegistry();
            registry.bind("Bishal", stub);
        }
        catch(Exception ae)
        {
            System.out.println(ae);
        }
    }
}
```

Result

```
    20
```

Conclusion

In this way ”Lab2 : Implement RMI to add two numbers” was completed by implementing RMI through Java.