CSE 532 Distributed Operating Systems

Remote Method Invocation

Batuhan Düzgün - 20163505004

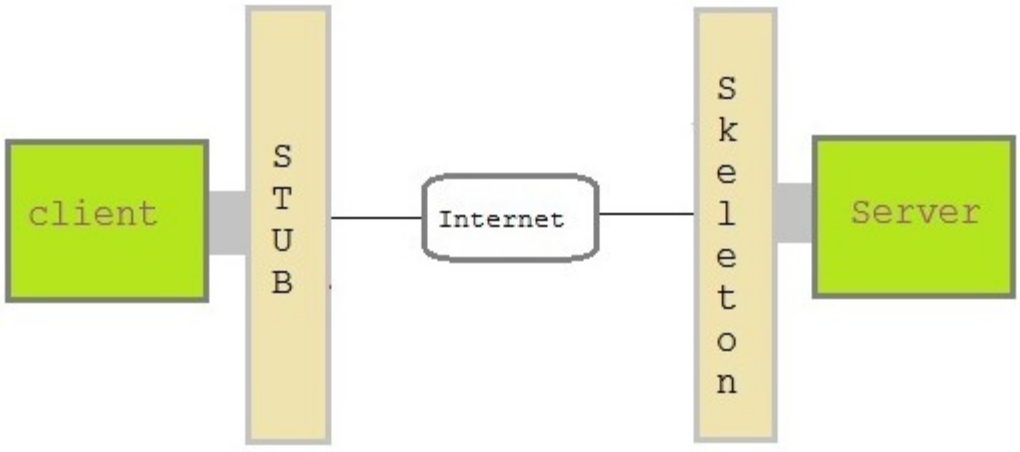
1. Introduction

This example illustrates the basics of using Remote Method Invocation. There will be a distributed calculator program which depends on RMI mechanism. This calculator program will have four basic functions which they are named as “Sum”,”Substract”,”Divide” and ”Multiply”. Client program will accept two numbers from the command line, parse the ASCII text to convert them to numbers, and will call a function which can be one of four calculation functions via RMI. Server program will accept this request and will make calculation. After that, the server program will return a value to the client. This application will be developed in a Linux Operating System which is named as Ubuntu. Java is used to develop the project. Eclipse is used as main IDE.

2. Remote Method Invocation Definition

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



4. Simple RMI Application involves following steps

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

5. Remote Interface

A remote interface specifies the methods that can be invoked remotely by a client. A interface must extend the **Remote** interface of **java.rmi** package.

```
package com.rmi.calculator.remote.functions;

import java.rmi.Remote;

// INFO: "RemoteCalculatorFunctions" Interface have to extend "java.rmi.Remote" Interface
public interface RemoteCalculatorFunctions extends Remote{

    // INFO: Abstract and Remote functions of Calculator
    public double sum(double firstNumber, double secondNumber) throws RemoteException;
    public double substract(double firstNumber, double secondNumber) throws RemoteException;
    public double divide(double firstNumber, double secondNumber) throws RemoteException;
    public double multiply(double firstNumber, double secondNumber) throws RemoteException;

}
```

6. Implementation of Remote Interface

It has to extend "**UnicastRemoteObject**" and has to implement "**Remote Interface**" which is named "RemoteCalculatorFunctions" for this example. A piece of code blocks of implementation class:

```
    @Override
    public double sum(double firstNumber, double secondNumber) throws RemoteException {

        double result = firstNumber + secondNumber;

        printLog("SUM OPERATION", firstNumber, secondNumber, result);

        return result;
    }

    @Override
    public double substract(double firstNumber, double secondNumber) throws RemoteException {

        double result = firstNumber - secondNumber;

        printLog("SUBSTRACT OPERATION", firstNumber, secondNumber, result);

        return result;
    }
```

7. Server Program

You need to create a server application and host RMI service 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 RMI service.

```
package com.rmi.calculator.server;

import java.rmi.Naming;

public class CalculatorServer {

    public static void main(String args[]) {

        try {
            RemoteCalculator remoteCalculator = new RemoteCalculator();
            Naming.rebind("RemoteCalculator", remoteCalculator);
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

8. Client Program

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 **RemoteCalculatorFunctions**. All remote method invocation is done on this object.

```
RemoteCalculatorFunctions remoteCalculator =
        (RemoteCalculatorFunctions) Naming.lookup("rmi://" + host + "/RemoteCalculator");
```

Remote Calls :

```
switch (operationType) {

    case 1:
        result = remoteCalculator.sum(firstNumber, secondNumber);
        break;
    case 2:
        result = remoteCalculator.substract(firstNumber, secondNumber);
        break;
    case 3:
        result = remoteCalculator.divide(firstNumber, secondNumber);
        break;
    case 4:
        result = remoteCalculator.multiply(firstNumber, secondNumber);
        break;
    default:
        throw new Exception("ERROR(1005): Invalid Calculation Operation Type!");
```

9. Start RMI Registry

```
batuhan@batuhan-Vostro-5470:~$ cd /home/batuhan/workspace/RMIBasedCalculator/bin
batuhan@batuhan-Vostro-5470:~/workspace/RMIBasedCalculator/bin$ rmiregistry &
[1] 20209
batuhan@batuhan-Vostro-5470:~/workspace/RMIBasedCalculator/bin$
```

10. Run Server Application

```
CalculatorServer [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Oct 27, 2016, 12:43:37 AM)
[ ********** SUM OPERATION ********* ]
[First Argument : 20.0]
[Second Argument: 54.0]
[Result         : 74.0]
[ ********************************** ]
[ ********** SUM OPERATION ********* ]
[First Argument : 20.0]
[Second Argument: 54.0]
[Result         : 74.0]
[ ********************************** ]
[ ********** DIVIDE OPERATION ********* ]
[First Argument : 20.0]
[Second Argument: 54.0]
[Result         : 0.37037037037037035]
[ ********************************** ]
[ ********** SUBSTRACT OPERATION ********* ]
[First Argument : 20.0]
[Second Argument: 54.0]
[Result         : -34.0]
[ ********************************** ]
[ ********** MULTIPLY OPERATION ********* ]
[First Argument : 20.0]
[Second Argument: 54.0]
[Result         : 1080.0]
[ ********************************** ]
```

11. Run Client Application

```
<terminated> CalculatorServer [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Oct 27, 2016, 12:50:04 AM)
[Result Value: 1080.0 ]
```

```
<terminated> CalculatorServer [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Oct 27, 2016, 12:49:02 AM)
[Result Value: 0.37037037037037035 ]
```