A Mini Project Report

On

Client-Server Using RMI Method

Submitted in partial fulfillment of requirements for the Course
CSE18R272 - JAVA PROGRAMMING

Bachelor's of Technology

In

Computer Science and Engineering

Submitted By

TVD.Hanumann
9918004117

V.RaviTeja
9918004118

Under the guidance of

Dr. R. RAMALAKSHMI

(Associate Professor)

Department of Computer Science and Engineering

Kalasalingam Academy of Research and Education

Anand Nagar, Krishnankoil-626126

APRIL 2020

ABSTRACT

Our project is to say how the system works.The System works like client server and the project is to explain about how the server and client connects. And I am writing client server program using RMI i.e Remote Method Invocation. In this project I will say how to connect client and server using a Remote object and after connecting how to upload,download,making directories,delete directories and to list the directories in files.

DECLARATION

I hereby declare that the work presented in this report entitled "**Client-Server using RMI Method**", in partial fulfilment of the requirements for the course CSE18R272- Java Programming and submitted in **Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education** (**Deemed to be University**) is an authentic record of our own work carried out during the period from **Jan 2020** under the guidance of Mr. **Dr. R. Ramalakshmi** (Associate Professor).

The work reported in this has not been submitted by me for the award of any other degree of this or any other institute.

V.RaviTeja
9918004118
TVD.Hanumann
9918004117

ACKNOWLEDGEMENT

First and foremost, I wish to thank the Almighty God for his grace and benediction tocomplete this Project work successfully. I would like to convey my special thanks from the bottom of my heart to my dear Parents and affectionate Family members for their honest support for the completion of this Project work.

I express deep sense of gratitude to "Kalvivallal" Thiru. T. Kalasalingam B.com., Founder Chairman, "Ilayavallal" Dr.K.Sridharan Ph.D., Chancellor, Dr.S.ShasiAnand, Ph.D., Vice President (Academic) , Mr.S.ArjunKalasalingam M.S., Vice President (Administration) , Dr.R.Nagaraj Vice-Chancellor, Dr.V.Vasudevan Ph.D., Registrar , Dr.P.Deepalakshmi Ph.D., Dean (School of Computing) . And also a special thanks to Dr. A. FRANCIS SAVIOUR DEVARAJ. Head Department of CSE, Kalasalingam Academy of Research and Education forgranting the permission and providing necessary facilities to carry out Project work.

I would like to express my special appreciation and profound thanks to my enthusiastic Project Supervisor Dr.R.Ramalakshmi Ph.D, Associate Professor at Kalasalingam Academy of Research and Education [KARE] for her inspiring guidance, constant encouragement with my work during all stages. I am extremely glad that I had a chance to do my Project under my Guide, who truly practices and appreciates deep thinking. I will be forever indebted to my Guide for all the time he has spent with me in discussions. And during the most difficult times when writing this report, he gave me the moral support and the freedom I needed to move on.

V.RaviTeja
9918004118
TVD.Hanumann
9918004117

TABLE OF CONTENTS

1. ABSTRACT i

2. CANDIDATE'S DECLARATION ii

3. ACKNOWLEDGEMENT iii

4. TABLE OF CONTENTS iv

5. LIST OF FIGURES v

Chapter 1 INTRODUCTION 1
1.0.1 Objectives 1

Chapter 2 RMI METHOD 2
2.0.1 Steps to write RMI method 2
2.0.2 How to Compile the programs 3

Chapter 3 CONCLUSION 6

REFERENCES 7

APPENDIX 8

LIST OF FIGURES

2.1 Uploading Command 3
2.2 Downloading Command 4
2.3 listof directories command 4
2.4 Making Directory Command 5
2.5 Remove Directory Command 5

Chapter 1

INTRODUCTION

Our project is to say how the system works.When we are searching in google the user is client and the information is sent to the server in some where and it returns an information what you need .This is the basic method we are using in our systems.

This project is to explain about how the server and client connects.And I am writing client server program using RMI i.e Remote Method Invocation. In this project I will say how to connect client and server using a Remote object and after connecting how to upload,download,making directories,delete directories and to list the directories in files. These client-server are connected by a port number(Any Number) and server number(For ex:127.0.0.1).The port is same for client and server methods when the server port number is given an 80 then the client port number number should be also 80 then it connects to communicate with each other.

1.0.1 Objectives

List the objectives of the project work...

1. To develop a code on RMI Method
2. To implement a project for to invoke methods from distance on remote objects (these objects are located on other systems). In order to use a remote object, we have to gain a reference for that object. The methods that will be used for that object are invoked in the same way as the local objects.

Chapter 2

RMI METHOD

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 JVM(JAVA VIRTUAL MACHINE). The RMI provides remote communication between the applications using two objects.

2.0.1 Steps to write RMI method

1. Creating the Remote Interface
2. Developing a class which implements the interface from 1.
3. Develop the server component (process).
4. Develop the client component (process).

To write these steps we need to write some java packages like
1.java.io.*;
2.java.rmi.*;

rmi.server;

rmi.server.UnicastRemoteObject;

rmi.RemoteException;
3.java.rmi.registry.*;

rmi.registry.Locateregistry;

rmi.registry.Registry;

This project is about the file transfer using java RMI method.In that it has upload ,download,make directories,list the directories,delete directories.

2.0.2 How to Compile the programs

1.Compile all the 4 classes.
2.Run the server class RMIServer.java, pass a portnumber available on your local machine to run the server on.
3.Set the server location (local host and port number) to your environment variables.
4.Run the client code:
4.1.To upload a file
java RMIClient.java
upload <path on client> <C:/ServerStorage/filename>
4.2.To download a file.

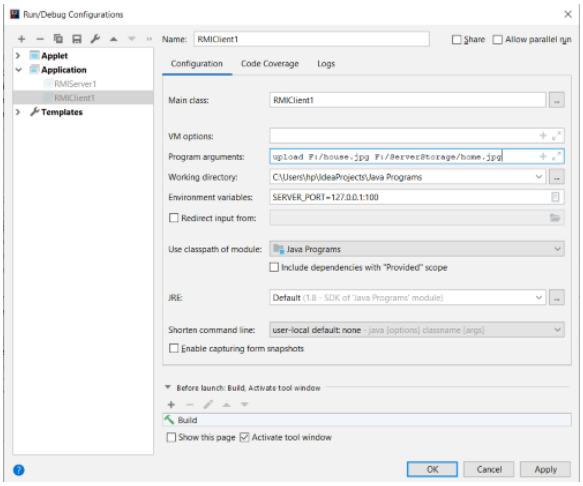

Figure 2.1: Uploading Command

java RMIClient.java
download <C:/ServerStorage/existing filename on server> <path on client/newfilename>

4.3.To list the directories and file present on the server.
java RMIClient.java

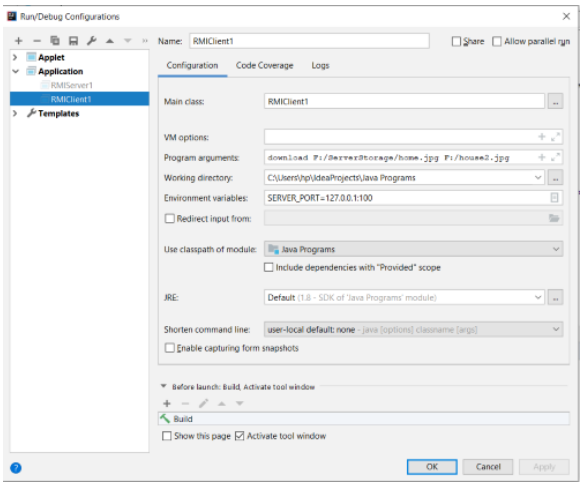

Figure 2.2: Downloading Command

dir <C:/ServerStorage/path existing directory on server>
4.4.To create a new directory.

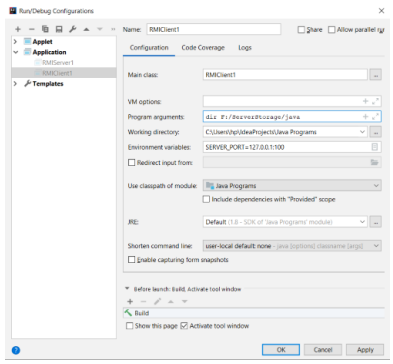

Figure 2.3: listof directories command

java RMIClient.java
mkdir <C:/ServerStorage/new directory name>
4.5.To delete an existing directory.
java RMIClient.java
rmdir <C:/ServerStorage/existing directory name>
4.6.To shutdown the client.
java RMIClient.java shutdown

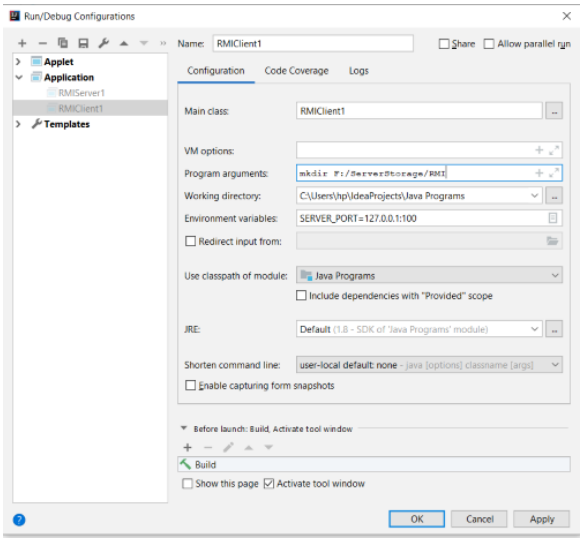

Figure 2.4: Making Directory Command

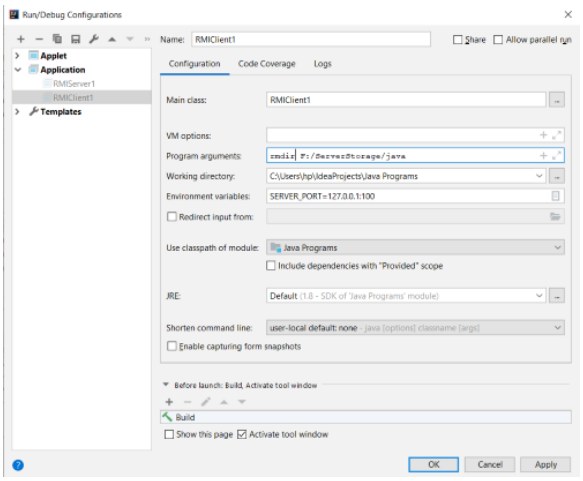

Figure 2.5: Remove Directory Command

Chapter 3

CONCLUSION

RMI provides a solid platform for truly object distributed computing.You can use RMI to connect to java components or to existing components written in other languages.As java proves itself in your environment,you can expand your java use and get all the benefits-no porting,low maintenance costs,and safe,secure environment.RMI gives you a platform to expand java into any part your system in an incremental fashion.As you add java,its benefits flow through all the java i your system.RMI makes this easy,secure,and powerful.

Appendices

SOURCE CODE

```
import java.io.*;
import java.rmi.*;
import java.rmi.registry.*;
public interface RmiInterface1 extends Remote
{

    public void uploadFileToServer(byte[] mybyte,
       ↪ String serverpath, int length) throws
       ↪ RemoteException;
    public byte[] downloadFileFromServer(String
       ↪ servername) throws RemoteException;
    public String[] listFiles(String serverpath) throws
       ↪  RemoteException;
    public boolean createDirectory(String serverpath)
       ↪ throws RemoteException;
    public boolean removeDirectoryOrFile(String
       ↪ serverpath) throws RemoteException;

}
```

```
import java.io.*;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class RmiImplementation1 extends
   ↪ UnicastRemoteObject implements RmiInterface,
   ↪ Serializable
{

    protected RmiImplementation1(String s) throws
       ↪ RemoteException
    {
        File storageDir = new File (s);
        storageDir.mkdir();
    }

    public void uploadFileToServer(byte[] mydata,
       ↪ String serverpath, int length) throws
```

```
    ↪ RemoteException
{

    try
    {
        File serverpathfile = new File(serverpath);
        FileOutputStream out=new FileOutputStream(
            ↪ serverpathfile);
        byte [] data=mydata;

        out.write(data);
            out.flush();
        out.close();

    }
    catch (IOException e)
    {

        e.printStackTrace();
    }

    System.out.println("Done writing data...");

}

public byte[] downloadFileFromServer(String
    ↪ serverpath) throws RemoteException
{

    byte [] mydata;

    File serverpathfile = new File(serverpath);
    mydata=new byte[(int) serverpathfile.length()];
    FileInputStream in;
    try
    {
        in = new FileInputStream(serverpathfile);
        try
        {
            in.read(mydata, 0, mydata.length);
```

```
            }
            catch (IOException e)
            {

                e.printStackTrace();
            }
            try
            {
                in.close();
            }
            catch (IOException e)
            {

                e.printStackTrace();
            }

        }
        catch (FileNotFoundException e)
        {

            e.printStackTrace();
        }

        return mydata;

    }


    public String[] listFiles(String serverpath) throws
       ↪ RemoteException
    {
        File serverpathdir = new File(serverpath);
        return serverpathdir.list();

    }

    public boolean createDirectory(String serverpath)
       ↪ throws RemoteException
    {
        File serverpathdir = new File(serverpath);
```

```
        return serverpathdir.mkdir();

    }

    public boolean removeDirectoryOrFile(String
       ↪ serverpath) throws RemoteException
    {
        File serverpathdir = new File(serverpath);
        return serverpathdir.delete();

    }


}
```

```
import java.io.*;
import java.rmi.registry.*;

public class RMIServer1 implements Serializable
{
    static int portnumber;
    //String remoteObject ="remoteObject";
    static String start = "start";

    public static void main(String[] args)
    {

        try
        {
            if(start.equals(args[0]))
            {
                portnumber = Integer.parseInt(args[1]);

            }
            Registry reg = LocateRegistry.
               ↪ createRegistry(portnumber);    //
               ↪ Creates and exports a Registry
               ↪ instance on the local host that
               ↪ accepts requests
            //on the specified port.
```

```
            RmiImplementation imp =  new
                ↪ RmiImplementation("F://ServerStorage"
                ↪ );
            reg.bind("remoteObject", imp);
            System.out.println("Server is ready.");
            System.out.println(portnumber);
        }
        catch(Exception e)
        {
            System.out.println("Server failed: " + e);
        }
    }
}
```

```
import java.io.*;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;


public class RMIClient1 implements Serializable
{

    public static void main(String[] args) throws
       ↪ IOException
    {

        String environment;
        String hostname;
        int portnumber;
        String clientpath;
        String serverpath;
        String upload = "upload";
        String download = "download";
        String dir= "dir";
        String mkdir= "mkdir";
        String rmdir= "rmdir";
```

```
        String rm= "rm";
        String shutdown= "shutdown";


        try
        {

            environment = System.getenv("SERVER_PORT");
            System.out.println(environment);

            hostname = environment.split(":")[0];

            portnumber = Integer.parseInt(environment.
               ↪ split(":")[1]);
            System.out.println("seeking connection on:"
               ↪  + environment);

            Registry myreg = LocateRegistry.getRegistry
               ↪ (hostname, portnumber);
            RmiInterface inter = (RmiInterface)myreg.
               ↪ lookup("remoteObject");

            //to upload a file
            if(upload.equals(args[0]))
            {
                clientpath= args[1];
                serverpath = args[2];

                File clientpathfile = new File(
                   ↪ clientpath);
                byte [] mydata=new byte[(int)
                   ↪ clientpathfile.length()];
//                for (byte b : mydata) {
//                    System.out.println((int)b);
//                }
                try
                {
                    FileInputStream in = new
                       ↪ FileInputStream(
                       ↪ clientpathfile);
```

```
            System.out.println("uploading to
               ↪ server...");
            int smtg = in.read(mydata,0,mydata.
               ↪ length);
            System.out.println(smtg);
            inter.uploadFileToServer(mydata,
               ↪ serverpath, (int)
               ↪ clientpathfile.length());

            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    //to download a file
    if(download.equals(args[0]))
    {
        serverpath = args[1];
        clientpath= args[2];

        byte [] mydata = inter.
           ↪ downloadFileFromServer(serverpath
           ↪ );
        System.out.println("downloading...");
        File clientpathfile = new File(
           ↪ clientpath);
        FileOutputStream out=new
           ↪ FileOutputStream(clientpathfile);
        out.write(mydata);
            out.flush();
        out.close();
    }

    //to list all the files in a directory
    if(dir.equals(args[0]))
    {
        serverpath = args[1];
        String[] filelist = inter.listFiles(
           ↪ serverpath);
        for (String i: filelist)
```

```
            {
                System.out.println(i);
            }
        }

        //to create a new directory
        if(mkdir.equals(args[0]))
        {
            serverpath = args[1];
            boolean bool = inter.createDirectory(
               ↪ serverpath);
            System.out.println("directory created
               ↪ :" + bool);
        }

        //to remove/delete a directory
        if(rmdir.equals(args[0]) || rm.equals(args
           ↪ [0]))
        {
            serverpath = args[1];
            boolean bool = inter.
               ↪ removeDirectoryOrFile(serverpath)
               ↪ ;
            System.out.println("directory deleted
               ↪ :" + bool);
        }
        //to shutdown the client
        if(shutdown.equals(args[0]))
        {
            System.exit(0);
            System.out.println("Client has shutdown
               ↪ . Close the console");
        }

    }
    catch(Exception e)
    {
        e.printStackTrace();
        System.out.println("error with connection
           ↪ or command. Check your hostname or
```

```
                    ↪ command");
            }
        }

}
```