|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 05** | **DESIGNING AN ENTERPRISE RMI APPLICATION –SECURE FILE TRANSFER** | | |
| Date of Exercise |  | Date of Output Verification |  |

**Aim**

To develop and deploy the enterprise RMI application for secure file transfer using J2EE programming.

**Description**

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

RMI uses stub and skeleton object for communication with the remote object.

A remote object is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

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:

It initiates a connection with remote Virtual Machine (JVM),

It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),

It waits for the result

It reads (unmarshals) the return value or exception, and

It finally, returns the value to the caller.

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:

It reads the parameter for the remote method

It invokes the method on the actual remote object, and

It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons. stub and skeleton in RMI

Understanding requirements for the distributed applications

If any application performs these tasks, it can be distributed application.

.

The application need to locate the remote method

It need to provide the communication with the remote objects, and

The application need to load the class definitions for the objects.

The RMI application have all these features, so it is called the distributed application.

**Procedure**

* Create the remote interface
* Provide the implementation of the remote interface
* Compile the implementation class and create the stub and skeleton objects using the rmic tool
* Start the registry service by rmiregistry tool
* Create and start the remote application
* Create and start the client application

**Program**

CeaserCipher.java

import java.util.Scanner;

public class CeaserCipher {

public int op, sno, j;

public Scanner in = new Scanner(System.in);

public CeaserCipher() {

setkey();

}

public String encode(String text) {

System.out.println("\n---------Encoding Request Received---------");

System.out.println("Input: " + text);

if (op == 1) {

return left(text);

} else if (op == 2) {

return right(text);

}

return "";

}

public String decode(String text) {

System.out.println("\n---------Decoding Request Received---------");

System.out.println("Input: " + text);

if (op == 2) {

return left(text);

} else if (op == 1) {

return right(text);

}

return "";

}

public void setkey() {

System.out.println("---------Set Key--------");

int i = 0;

do {

if (i != 0) {

System.out.println("Invalid Option!, Try Again");

}

System.out.println("\nSelect Shift Operation: \n1.Left 2.Right");

op = in.nextInt();

in.nextLine();

i++;

} while (op != 1 && op != 2);

System.out.print("\nEnter Shift Number: ");

sno = in.nextInt();

in.nextLine();

}

public String left(String text) {

String output = "";

for (char i : text.toCharArray()) {

j = i;

if (Character.isLowerCase(i)) {

j = i - sno;

if (j < 'a') {

j += 26;

}

}

if (Character.isUpperCase(i)) {

j = i - sno;

if (j < 'A') {

j += 26;

}

}

output = output + (char) j;

}

System.out.println("Output: " + output + "\n");

return output;

}

public String right(String text) {

String output = "";

for (char i : text.toCharArray()) {

if (Character.isLowerCase(i)) {

output = output + (char) ((i - 'a' + sno) % 26 + 'a');

} else if (Character.isUpperCase(i)) {

output = output + (char) ((i - 'A' + sno) % 26 + 'A');

} else {

output = output + i;

}

}

System.out.println("Output: " + output + "\n");

return output;

}

}

RMIClientInterface.java

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface RMIServerInterface extends Remote{

public String showAvailableFiles() throws RemoteException;

public String getFileData(String index) throws RemoteException;

}

RMIServerMain.java

import java.rmi.AlreadyBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.\*;

import java.rmi.server.UnicastRemoteObject;

import java.util.HashMap;

import java.util.Scanner;

public class RMIServerMain extends UnicastRemoteObject implements RMIServerInterface {

public static int op, sno, j;

public static Scanner in = new Scanner(System.in);

public static HashMap hm = new HashMap();

public static CeaserCipher cc = new CeaserCipher();

public RMIServerMain() throws RemoteException {

}

public static void main(String[] args) {

try {

hm.put("1", "William Scott developed this");

hm.put("2", "hello world program");

hm.put("3", "Java is awesome");

hm.put("4", "Sample Text paragraph");

hm.put("5", "This is just a sample paragraph");

Registry r = LocateRegistry.createRegistry(22004);

RMIServerMain s = new RMIServerMain();

r.bind("server", s);

System.out.println("\nServer Bind Complete!...");

while (true) {

System.out.println("\n" + hm);

System.out.println("\n1. Add Data\n2. Remove Data\n3. Change Encryption Key\n");

switch (in.nextLine()) {

case "1":

System.out.println("\nEnter Key: ");

String key = in.nextLine();

System.out.println("Enetr Value: ");

String value = in.nextLine();

hm.put(key, value);

break;

case "2":

System.out.println("\nEnter Key: ");

String k = in.nextLine();

hm.remove(k);

break;

case "3":

cc.setkey();

break;

default:

System.out.println("\nInvalid Option! Try Again....");

break;

}

}

} catch (RemoteException | AlreadyBoundException e) {

System.out.println(e);

}

}

@Override

public String showAvailableFiles() throws RemoteException {

System.out.println("Request for files received");

System.out.println("Files list sent\n");

return hm.keySet().toString();

}

@Override

public String getFileData(String index) throws RemoteException {

System.out.println("Request for file received with index: " + index);

System.out.println("Sending file data for encryption\n");

return cc.encode((String) hm.get(index));

}

}

RMIClientMain.java

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.Scanner;

public class RMIClientMain extends UnicastRemoteObject {

public RMIClientMain() throws RemoteException {

}

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

try {

Scanner in = new Scanner(System.in);

CeaserCipher cc = new CeaserCipher();

Registry r1 = LocateRegistry.getRegistry(22004);

RMIServerInterface rob = (RMIServerInterface) r1.lookup("server");

System.out.println("\nSucessfully Connected to Server");

while (true) {

System.out.println("\n1. Show Indexes of Files\n2. Get File Using Index\n3. Change Encryption Key\n4. Exit\n");

switch (in.nextLine()) {

case "1":

System.out.println("\nFiles Available:");

System.out.println(rob.showAvailableFiles());

break;

case "2":

System.out.println("\nEnter File Index Number: ");

String temp = rob.getFileData(in.nextLine());

System.out.println("\nReceived Encrypted Message: " + temp);

System.out.println("Sent for Decryption");

temp = cc.decode(temp);

System.out.println("Decrpted Message: " + temp);

break;

case "3":

cc.setkey();

break;

case "4":

System.exit(0);

break;

default:

System.out.println("\nInvalid Option! Try Again...");

break;

}

}

} catch (RemoteException | NotBoundException e) {

System.out.println(e);

}

}

}

**Output**

RMIServerMain

run:

---------Set Key--------

Select Shift Operation:

1.Left 2.Right

1

Enter Shift Number: 2

Server Bind Complete!...

{1=William Scott developed this, 2=hello world program, 3=Java is awesome, 4=Sample Text paragraph, 5=This is just a sample paragraph}

1. Add Data

2. Remove Data

3. Change Encryption Key

Request for files received

Files list sent

Request for file received with index: 1

Sending file data for encryption

---------Encoding Request Received---------

Input: William Scott developed this

Output: Ugjjgyk Qamrr bctcjmncb rfgq

Request for file received with index: 1

Sending file data for encryption

2

Enter Key:

1

{2=hello world program, 3=Java is awesome, 4=Sample Text paragraph, 5=This is just a sample paragraph}

1. Add Data

2. Remove Data

3. Change Encryption Key

1

Enter Key:

new

Enetr Value:

william scott is awesome

{new=william scott is awesome, 2=hello world program, 3=Java is awesome, 4=Sample Text paragraph, 5=This is just a sample paragraph}

1. Add Data

2. Remove Data

3. Change Encryption Key

Request for files received

Files list sent

Request for file received with index: new

Sending file data for encryption

RMIClientMain

run:

---------Set Key--------

Select Shift Operation:

1.Left 2.Right

1

Enter Shift Number: 3

Sucessfully Connected to Server

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

1

Files Available:

[1, 2, 3, 4, 5]

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

2

Enter File Index Number:

1

Received Encrypted Message: Ugjjgyk Qamrr bctcjmncb rfgq

Sent for Decryption

---------Decoding Request Received---------

Input: Ugjjgyk Qamrr bctcjmncb rfgq

Output: Xjmmjbn Tdpuu efwfmpqfe uijt

Decrpted Message: Xjmmjbn Tdpuu efwfmpqfe uijt

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

3

---------Set Key--------

Select Shift Operation:

1.Left 2.Right

1

Enter Shift Number: 2

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

2

Enter File Index Number:

1

Received Encrypted Message: Ugjjgyk Qamrr bctcjmncb rfgq

Sent for Decryption

Decrpted Message: William Scott developed this

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

1

Files Available:

[new, 2, 3, 4, 5]

1. Show Indexes of Files

2. Get File Using Index

3. Change Encryption Key

4. Exit

2

Enter File Index Number:

new

Received Encrypted Message: ugjjgyk qamrr gq yucqmkc

Sent for Decryption

Decrpted Message: william scott is awesome

**Result**

Thus, the enterprise RMI application for secure file transfer was successfully developed and deployed

[Signature of the Staff In-charge]

[Name of the Staff In – charge]