



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

---

<b>Name : Ajay Shitkar</b>
<b>Roll No : 58</b>
Experiment No. 02
Implement Client/Server using RPC/RMI
Date of Performance: 23/02/24
Date of Submission: 01/03/24



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

**Aim:** To implement Client/Server using RPC/RMI

**Objective:** Develop a program to implement Client/Server using RPC/RMI

### Theory:

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

##### Stub

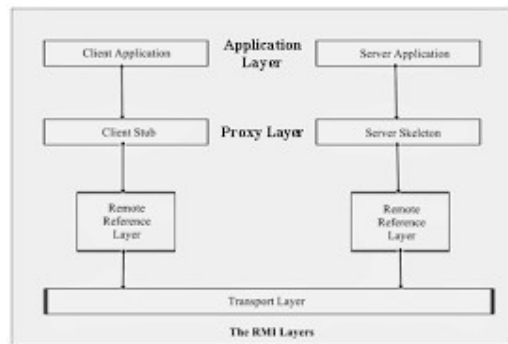
The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

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.

##### 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.



### Steps to write the RMI program

6 steps to write the RMI program.

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

### **Code and output:**

SampleClient.java:-

```
import    java.rmi.*;
import
java.rmi.server.*;
public    class
SampleClient
{ public static void main(String[] args)
{
    // set the security manager for the client
    System.setSecurityManager(new RMISecurityManager());
    //get the remote object from the registry
    try
    {
        System.out.println("Security Manager loaded");
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

```
String url = "//localhost/SAMPLE-SERVER";
SampleServer remoteObject = (SampleServer)Naming.lookup(url);
System.out.println("Got remote object");
System.out.println(" 1 + 2 = " + remoteObject.sum(1,2) );
}

catch (RemoteException exc) {
    System.out.println("Error in lookup: " + exc.toString()); }
catch (java.net.MalformedURLException exc) {
    System.out.println("Malformed URL: " + exc.toString()); }
catch (java.rmi.NotBoundException exc) {
    System.out.println("NotBound: " + exc.toString());
}
}
}
```

SampleServerImpl.java:-

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

public class SampleServerImpl extends UnicastRemoteObject implements SampleServer
{
    SampleServerImpl() throws RemoteException
    {

super();
    }

    public int sum(int a,int b) throws RemoteException
    {
        return
        (a+b); }

    public static void main(String args[])
    {
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

```
try
{
    System.setSecurityManager(new    RMISecurityManager());
    //set the security manager

    //create a local instance of the object
    SampleServerImpl Server = new SampleServerImpl();

    //put the local instance in the registry
    Naming.rebind("SAMPLE-SERVER"  ,  Server);  System.out.println("Server
waiting.....");
}
catch (java.net.MalformedURLException me)    {
    System.out.println("Malformed URL: " + me.toString());  }
catch (RemoteException re) {
    System.out.println("Remote exception: " + re.toString()); }
}

}
```

### SampleServer.java

```
import java.rmi.*;

public interface SampleServer extends Remote
{   public   int   sum(int   a,int   b)   throws
    RemoteException;
}

policy.all:grant      {      permission
java.security.AllPermission;
};
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

```
Administrator: Command Prompt - java -Djava.security.policy=policy.all SampleServerImpl
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd Javaa

C:\Windows\System32\Javaa>javac SampleServerImpl.java
Note: SampleServerImpl.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Windows\System32\Javaa>rmic SampleServerImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

C:\Windows\System32\Javaa>java -Djava.security.policy=policy.all SampleServerImpl
Server waiting.....
```



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd Desktop
The system cannot find the path specified.

C:\Windows\System32>cd Javaa
The system cannot find the path specified.

C:\Windows\System32>cd Javaa
C:\Windows\System32\Javaa>javac SampleServer.java
C:\Windows\System32\Javaa>start rmiregistry
C:\Windows\System32\Javaa>
```

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd Javaa
C:\Windows\System32\Javaa>javac SampleClient.java
Note: SampleClient.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Windows\System32\Javaa>java -Djava.security.policy=policy.all SampleClient
Security Manager loaded
Got remote object
1 + 2 = 3
C:\Windows\System32\Javaa>
```

**Conclusion:** Implementing Client/Server architecture using Remote Procedure Call (RPC) or Remote Method Invocation (RMI) offers a robust framework for distributed computing. RPC/RMI facilitates seamless communication between client and server, abstracting the complexities of network programming. It promotes modularity, scalability, and maintainability by allowing developers to encapsulate functionality into remote methods. While RPC/RMI introduces overhead due to network communication, its advantages in code organization and ease of development outweigh these concerns in many scenarios. Overall, RPC/RMI serves as a foundational technology for building efficient and reliable distributed systems.