1. Implementation of “Calculator” Service (Add, Sub, Mul, Div) using Java RMI.

# Step 1. Defining the Remote Interface

To create an RMI application, the first step is defining of a remote interface between the client and server objects.

/\* SampleServer.java \*/

import java.rmi.\*;

public interface SampleServer extends Remote

{

public int sum(int a,int b) throws RemoteException;

public int sub(int a,int b) throws RemoteException;

public int mul(int a,int b) throws RemoteException;

public float div(float a,float b) throws RemoteException;

public int mod(int a,int b) throws RemoteException;

public String hello(String str) throws RemoteException;

}

# Step 2. Develop the remote object by implement the remote interface

* The server is a simple unicast remote server.
* Create server by extending java.rmi.server.UnicastRemoteObject.
* The server uses the RMISecurityManager to protect its resources while engaging in remote communication.

/\* SampleServerImpl.java \*/

import java.rmi.\*;

import java.rmi.server.\*;

import java.rmi.registry.\*;

public class SampleServerImpl extends UnicastRemoteObject

implements SampleServer

{

SampleServerImpl() throws RemoteException

{

super();

}

* The server must bind its name to the registry, the client will look up the server name.
* Use java.rmi.Naming class to bind the server name to registry. In this example the name call “SAMPLE-SERVER”.
* In the main method of your server object, the RMI security manager is created and installed.
* Implement the remote mehtods

public static void main(String args[])

{

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());

}

}

/\* SampleServerImpl.java \*/

public int sum(int a,int b) throws RemoteException

{

return a + b;

}

public int sub(int a,int b) throws RemoteException

{

return a - b;

}

public int mul(int a,int b) throws RemoteException

{

return a \* b;

}

public float div(float a,float b) throws RemoteException

{

return a / b;

}

public int mod(int a,int b) throws RemoteException

{

return a % b;

}

public String hello(String str) throws RemoteException

{

return "hello "+str;

}

}

# Step 3. Develop the client program

* In order for the client object to invoke methods on the server, it must first look up the name of server in the registry. You use the java.rmi.Naming class to lookup the server name.
* The server name is specified as URL in the from ( rmi://host:port/name )
* Default RMI port is 1099.
* The name specified in the URL must exactly match the name that the server has bound to the registry. In this example, the name is “SAMPLE-SERVER”
* The remote method invocation is programmed using the remote interface name (remoteObject) as prefix and the remote method name (sum) as suffix.

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");

String url = "//localhost/SAMPLE-SERVER";

String str="15012011038";

SampleServer remoteObject = (SampleServer)Naming.lookup(url);

System.out.println("Got remote object");

System.out.println(remoteObject.hello(str));

System.out.println(" 15 + 20 = " + remoteObject.sum(15,20) );

System.out.println(" 15 - 20 = " + remoteObject.sub(15,20) );

System.out.println(" 15 \* 20 = " + remoteObject.mul(15,20) );

System.out.println(" 15 / 20 = " + remoteObject.div(15,20) );

System.out.println(" 15 % 20 = " + remoteObject.mod(15,20) );

}

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());

}

}

}

# Step 4 & 5. Compile the Java source files & Generate the client stubs and server skeletons

* Assume the program compile and executing at elpis on ~/rmi
* Once the interface is completed, you need to generate stubs and skeleton code. The RMI system provides an RMI compiler (rmic) that takes your generated interface class and procedures stub code on its self.

elpis:~/rmi> set CLASSPATH=”~/rmi”

elpis:~/rmi> javac SampleServer.java

elpis:~/rmi> javac SampleServerImpl.java

elpis:~/rmi> rmic SampleServerImpl

elpis:~/rmi> javac SampleClient.java

# 6. Start the RMI registry

* The RMI applications need install to Registry. And the Registry must start manual by call rmiregisty.
* The rmiregistry us uses port 1099 by default. You can also bind rmiregistry to a different port by indicating the new port number as : rmiregistry <new port>

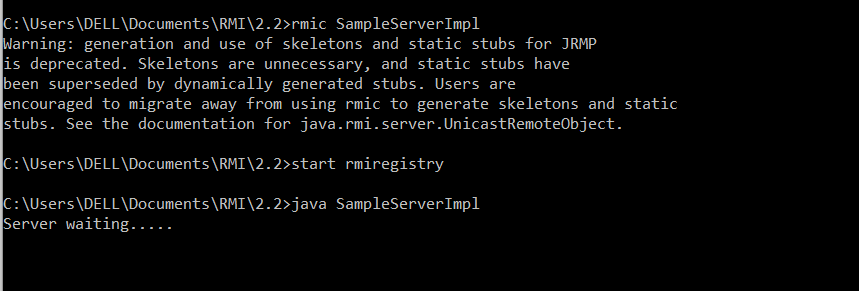
elpis:~/rmi> rmiregistry

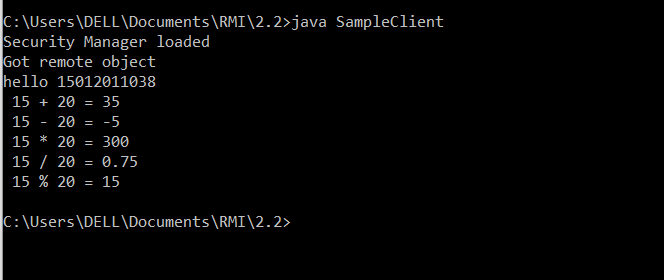
*remark: On Windows, you have to type in from the command line:*

*> start rmiregistry*

# Step 7 & 8. Start the remote server objects & Run the client

* Once the Registry is started, the server can be started and will be able to store itself in the Registry.





1. Implementation of “Hello World” Service using Java RMI.

/\* SampleServer1.java \*/

import java.rmi.\*;

public interface SampleServer1 extends Remote

{

public String hello() throws RemoteException;

}

/\* SampleServerImpl1.java \*/

import java.rmi.\*;

import java.rmi.server.\*;

import java.rmi.registry.\*;

public class SampleServerImpl1 extends UnicastRemoteObject

implements SampleServer1

{

SampleServerImpl1() throws RemoteException

{

super();

}

/\* SampleServerImpl1.java \*/

public static void main(String args[])

{

try

{

// System.setSecurityManager(new RMISecurityManager());

//set the security manager

//create a local instance of the object

SampleServerImpl1 Server = new SampleServerImpl1();

//put the local instance in the registry

Naming.rebind("SAMPLE-SERVER1" , 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()); }

}

/\* SampleServerImpl.java \*/

public String hello() throws RemoteException

{

return "HELLO WORLD";

}

}

import java.rmi.\*;

import java.rmi.server.\*;

public class SampleClient1

{

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");

String url = "//localhost/SAMPLE-SERVER1";

SampleServer1 remoteObject = (SampleServer1)Naming.lookup(url);

System.out.println("Got remote object");

System.out.println(remoteObject.hello());

}

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());

}

}

}

Output

