SENG – 615: Client / Server Technology (Lab.)

LAB - 05

1. Remote Method Invocation (Calling Remote methods)

Remote method Invocation is java based technology used to call remote methods. To use RMI technology we have to create an interface which will extend the Remote Interface given by RMI API in Java.
We have to create prototypes of those method in interface which we want to call remotely, then the implementation of those protypes will be provided.
After the creation of methods, we will have to generate the stub using RMIC command and then the RMI registery will be executed in which we have to register object which contains the remote methods.
The server and client will then be executed to call the remote methods.

Setting up a server process requires five steps:

1. ***Create a Remote Interface***
   Write an interface containg the remote method protypes
   . For example:

```
interface MyRemote extends Remote
```

2. ***Create remote method prototypes in interface***
   Protypes of all those method will be created in interface which we want to remotly For example:

```
String echoMessage(String str) throws RemoteException;
```

3. ***Implement the interface methods***
   The body of the method will be provided by implementing the remote interface. For example:

```
class MyRemoteImpl implements MyRemote
```

4. ***Provide the body of the remote methods***
   The body of the remote methods will be given in class which implements the interface For Example:

```
public String echoMessage(String msg) throws RemoteException{
System.out.println("Message Received from Client " + msg);
return msg;
}
```

5. **Create stub using RMIC command**

   Run command rmic located into the java.bin folder

   rmic `MyRemoteImpl`

6. **Run the RMI registery**

   To register the objects into the registery it is necessary to that RMI registry should be running, use following command located in java.bin folder

   rmiregistry

7. ***Write the Server the class to register the object into registery***

   Server class will register the object of implementing class into the registery to be able to call remotely. For example:

```
Naming.rebind("rmi://localhost/MyObject",mr);
```

8. ***Write client code to loopup the object into RMI registry and then call the remote method***

   To call the remote methods client will have to lookup the registery and searchout the object registered into the registery. For Example:

   MyRemote ob = (MyRemote)Naming.lookup("rmi://localhost/MyObject");

   String str = ob.echoMessage("Hello RMI Test");

Here is an example program that works as RMI echo client / Server

1. **Remote Interface**

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

interface MyRemote extends Remote{
String echoMessage(String str) throws RemoteException;}
```

2. **Remote Interface Implementation**

```
import java.rmi.*;

class MyRemoteImpl implements MyRemote{
public String echoMessage(String msg) throws RemoteException{
System.out.println("Message Received from Client " + msg);

return msg;}}
```

3. **EchoServer Class**

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

class EchoServer{

public static void main(String[]args)
throws RemoteException, java.net.MalformedURLException{

MyRemoteImpl mr = new MyRemoteImpl();
UnicastRemoteObject.exportObject(mr);
Naming.rebind("rmi://localhost/MyObject",mr);
System.out.println("RMI Server Started"); } }
```

4. **EchoServer Class**

```
import java.rmi.*;

public class EchoClient{
public static void main(String[]args)  throws RemoteException, NotBoundException,
java.net.MalformedURLException{

MyRemote ob = (MyRemote)Naming.lookup("rmi://localhost/MyObject");
String str = ob.echoMessage("Hello RMI Test");
System.out.println("Server Replied: "+str); } }
```