More RMI Applications

Network Distributed Programming Mark Cummins (ITB)

This set of notes that covers three different applications that uses RMI technology.

- *Output a number for static calculations*
- *Display the date and time*
- *A chat server*

The student is expected to input the different programs, compile and execute them, thus observing the workings of Java RMI applications. All examples should be tested on a standalone computer first and later on two separate networked computers.

The second application is an application that displays the date and time of the client computer and the remote server. The comments in the code on this Date/Time exercises are considerably detailed which should help with your understanding of the Java RMI processes. All source code can be found in my ITB StudentShare directory in RMI-LABS folder, and on the web at http://markcummins.livedrive.com

Exercise One – Remote Calculator

Step 1: Input **Cal.java** program and compile.
Step 2: Input **CalImpl.java** and compile.
Step 3: Input **CalServer.java** and compile.
Step 4: Input **CalClient.java** and compile.
Step 5: Create the Stub (and Skeleton) classes by typing the following:
rmic CalImpl
Step 6: Execute the application:

Standalone Computer: If running the application on a standalone computer:

- Open three separate MS-DOS windows.
- Run the registry in one window, e.g. **rmiregistry**
- Run the server code in the next window, e.g. **java CalServer**
- Run the client in the last window, e.g. **java CalClient**

The output of this program should be a series of numbers representing a set of calculations.

Networked Computers: do the following:

- Find the IP address of the computer selected to be the server.
- Replace all references to "***localhost***" in the code with the Server's IP address.
- Copy the following programs to the client:
  - CalClient.class,
  - Cal.class
  - CalImpl_Stub.class
- On the server, open two MS-DOS windows and run the registry in one window and the server in the other window.
- On the client computer, open a MS-DOS window and run the client program.

```
public interface Cal
          extends java.rmi.Remote {
    public long add(long a, long b)
        throws java.rmi.RemoteException;

    public long sub(long a, long b)
        throws java.rmi.RemoteException;

    public long mul(long a, long b)
        throws java.rmi.RemoteException;

    public long div(long a, long b)
        throws java.rmi.RemoteException;
}
```

```
public class CalImpl
    extends
      java.rmi.server.UnicastRemoteObject
    implements Cal {

    // Implementations must have an
    //explicit constructor
    // in order to declare the
    //RemoteException exception
    public CalImpl()
        throws java.rmi.RemoteException {
        super();
    }

    public long add(long a, long b)
        throws java.rmi.RemoteException {
        return a + b;
    }

    public long sub(long a, long b)
        throws java.rmi.RemoteException {
        return a - b;
    }

    public long mul(long a, long b)
        throws java.rmi.RemoteException {
        return a * b;
    }

    public long div(long a, long b)
        throws java.rmi.RemoteException {
        return a / b;
    }
}
```

```
import java.rmi.Naming;

public class CalServer {

  public CalServer() {
    try {
      Cal c = new CalImpl();

      System.out.println("List of bindings in remote registry");
      String[] names = Naming.list("rmi://localhost");
      for (int i=0; i<names.length; i++)
      {
     System.out.println(names[i]);
      }

      System.out.println("DateServer starting...");

      Naming.rebind("rmi://localhost:1099/CalServer", c);

    } catch (Exception e) {
      System.out.println("Trouble: " + e);
    }
  }

  public static void main(String args[]) {
    new CalServer();
  }
}
```

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class CalClient {

    public static void main(String[] args) {
        try {
           Cal c = (Cal)Naming.lookup("rmi://localhost/CalServer");
            System.out.println( c.sub(4, 3) );
            System.out.println( c.add(4, 5) );
            System.out.println( c.mul(3, 6) );
            System.out.println( c.div(9, 3) );
        }
        catch (MalformedURLException murle) {
            System.out.println();
            System.out.println("MalformedURLException");
            System.out.println(murle);
        }
        catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
            System.out.println(re);
        }
        catch (NotBoundException nbe) {
            System.out.println();
            System.out.println("NotBoundException");
            System.out.println(nbe);
        }
        catch (java.lang.ArithmeticException ae) {
            System.out.println();
            System.out.println("java.lang.ArithmeticException");
            System.out.println(ae);
        }
    }
}
```

Exercise Two – Remote Date and Time

Create a new folder called Date-Test can follow these steps:

Step 1: Input **RemoteDate.java** program and compile.

Step 2: Input **RemoteDateImpl.java** and compile.

Step 3: Input **DateServer.java** and compile.

Step 4: Input **DateClient.java** and compile.

Step 5: Create the Stub and Skeleton classes by typing the following:

```
rmic  RemoteDateImpl
```

Step 5: Execute the application:

Standalone Computer:
If running the application on a standalone computer do the following:

- Open three separate MS-DOS windows.
- Run the registry in one window, e.g. rmiregistry
- Run the server code in the next window, e.g. java DateServer
- Run the client in the last window, e.g. java DateClient

The output of the program should be the date and time on the client computer followed by the date and time on the server computer.

Networked Computer:
The code is written to run on a standalone computer but if you are running the application on two networked computers, do the following:

- Find the IP address of the computer selected to be the server.
- Replace all references to “localhost” in the code with the Server’s IP address.
- Copy the following programs to the client:
  - DateClient.class,
  - RemoteDate.class
  - RemoteDateImpl_Stub.class
- On the server, open two MS-DOS windows and run the registry in one window and the server in the other window.
- On the client computer, open a MS-DOS window and run the client program.

```
// Filename:  RemoteDate.java
import java.rmi.*;
import java.util.Date;

/** A statement of what the client & server must agree upon. */
public interface RemoteDate extends java.rmi.Remote {

      /** The method used to get the current date on the remote */
      public Date getRemoteDate() throws java.rmi.RemoteException;

      /** The name used in the RMI registry service. */
      public final static String LOOKUPNAME =
"rmi://localhost/RemoteDate";
}
```

```
// Filename:  RemoteDateImpl.java
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;

public class RemoteDateImpl extends UnicastRemoteObject implements
RemoteDate
{
      /** Construct the object that implements the remote server.
       * Called from main, after it has the SecurityManager in
place.
       */
      public RemoteDateImpl() throws RemoteException {
            super();   // sets up networking
      }
      /** The remote method that "does all the work". This won't get
       * called until the client starts up.
       */
      public Date getRemoteDate() throws RemoteException {
            return new Date();
      }
}
```

```
//Filename:  DateServer.java
import java.rmi.*;

public class DateServer {
     public static void main(String[] args) {
          // You may want a SecurityManager for downloading of
classes:
          // System.setSecurityManager(new
RMISecurityManager());
          try {
               // Create an instance of the server object
               RemoteDateImpl im = new RemoteDateImpl();
```

```
                System.out.println("DateServer starting...");
                // Locate it in the RMI registry.
                Naming.rebind(RemoteDate.LOOKUPNAME, im);

                System.out.println("DateServer ready.");
           } catch (Exception e) {
                System.err.println(e);
                System.exit(1);
           }
      }
}
```

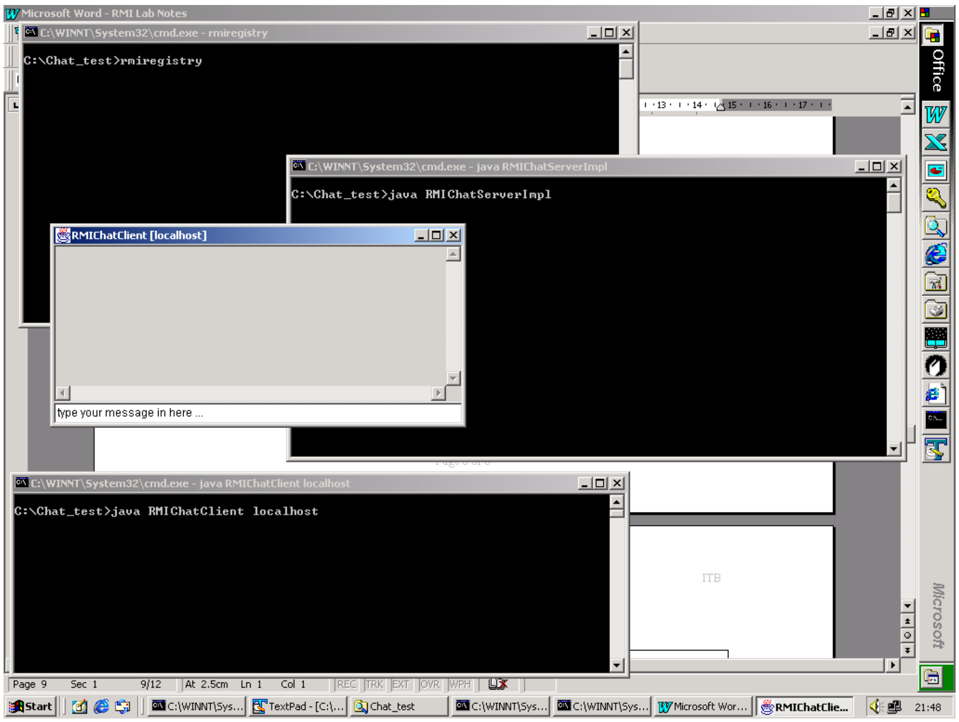
```
//Filename:  DateClient.java

import java.rmi.*;
import java.util.*;
import java.text.DateFormat;


/* A very simple client for the RemoteDate service. */
public class DateClient {

      /** The local proxy for the service. */
      protected static RemoteDate netConn = null;

      public static void main(String[] args) {
           try {
                DateFormat ClientDate = DateFormat.getInstance();
                String now = ClientDate.format(new Date());
                System.out.println(now); // Todays Date
                netConn =
(RemoteDate)Naming.lookup(RemoteDate.LOOKUPNAME);
                Date today = netConn.getRemoteDate();
                System.out.println(today.toString()); // XX use a
DateFormat...
           } catch (Exception e) {
                System.err.println("RemoteDate exception: " +
e.getMessage());
                e.printStackTrace();
           }
      }
}
```

More RMI Applications

Exercise Three – Remote Chat System

The RMI chat system is taken from the Manning text book titled "*Java Netowrk Programming*". This is a simple RMI based Client/Server chat system and a full explanation of the code is given in chapter 24 of the text book.

Below are two screen shots showing the three DOS screens and the chat system window. You can input text at the bottom of the chat window and seconds later it will appear in the chat display window.



C:\WINNT\System32\cmd.exe - rmiregistry

```
C:\Chat_test>rmiregistry
```

C:\WINNT\System32\cmd.exe - java RMIChatServerImpl

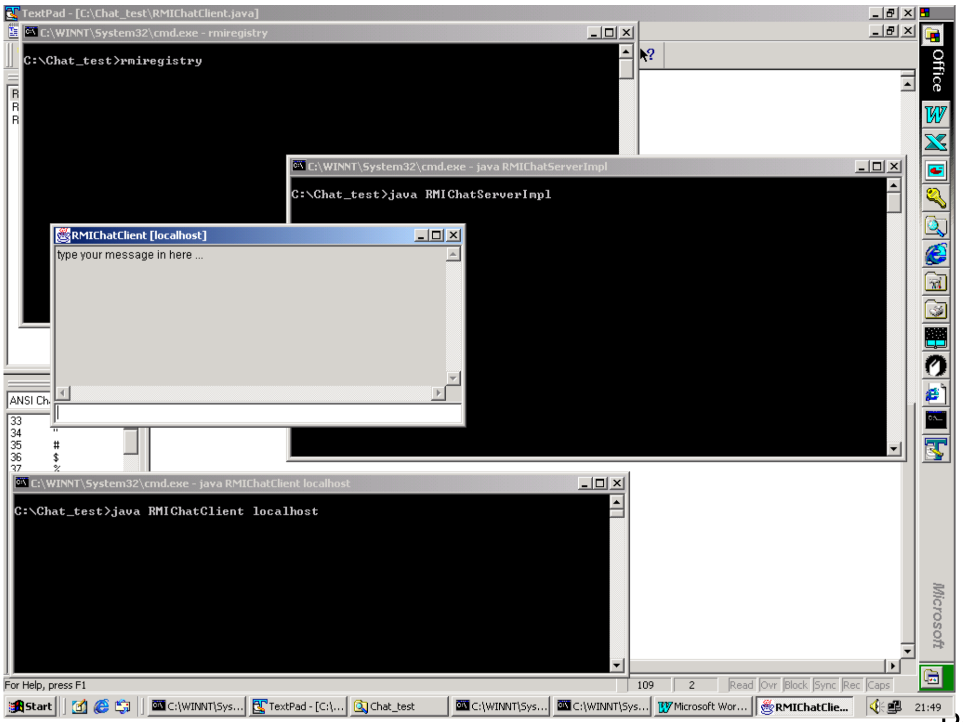
```
C:\Chat_test>java RMIChatServerImpl
```

RMIChatClient [localhost]

type your message in here ...

C:\WINNT\System32\cmd.exe - java RMIChatClient localhost

```
C:\Chat_test>java RMIChatClient localhost
```



C:\WINNT\System32\cmd.exe - rmiregistry

```
C:\Chat_test>rmiregistry
```

C:\WINNT\System32\cmd.exe - java RMIChatServerImpl

```
C:\Chat_test>java RMIChatServerImpl
```

RMIChatClient [localhost]

type your message in here ...

C:\WINNT\System32\cmd.exe - java RMIChatClient localhost

```
C:\Chat_test>java RMIChatClient localhost
```

The source code for the Chat System

```
/*
 * Java Network Programming, Second Edition
 * Merlin Hughes, Michael Shoffner, Derek Hamner
 * Manning Publications Company; ISBN 188477749X
 *
 * http://nitric.com/jnp/
 *
 * Copyright (c) 1997-1999 Merlin Hughes, Michael Shoffner, Derek
Hamner;
 * all rights reserved; see license.txt for details.
 */

import java.rmi.*;

public interface RMIChatServer extends Remote {
  public static final String REGISTRY_NAME = "Chat Server";
  public abstract String[] getMessages (int index) throws
RemoteException;
  public abstract void addMessage (String message) throws
RemoteException;
}
```

```
/*
 * Java Network Programming, Second Edition
 * Merlin Hughes, Michael Shoffner, Derek Hamner
 * Manning Publications Company; ISBN 188477749X
 *
 * http://nitric.com/jnp/
 *
 * Copyright (c) 1997-1999 Merlin Hughes, Michael Shoffner, Derek
Hamner;
 * all rights reserved; see license.txt for details.
 */
import java.rmi.*;
import java.util.*;
import java.rmi.server.*;
import java.rmi.registry.*;

public class RMIChatServerImpl extends UnicastRemoteObject
implements RMIChatServer {
  protected Vector messages;

  public RMIChatServerImpl () throws RemoteException {
    messages = new Vector ();
  }
  public String[] getMessages (int index) {
    int size = messages.size ();
    String[] update = new String[size - index];
    for (int i = 0; i < size - index; ++ i)
      update[i] = (String) messages.elementAt (index + i);
    return update;
  }
  public void addMessage (String message) {
    messages.addElement (message);
  }
  public static void main (String[] args) throws RemoteException {
    RMIChatServerImpl chatServer = new RMIChatServerImpl ();
    Registry registry = LocateRegistry.getRegistry ();
    registry.rebind (REGISTRY_NAME, chatServer);
  }
}
```

```
/*
 * Java Network Programming, Second Edition
 * Merlin Hughes, Michael Shoffner, Derek Hamner
 * Manning Publications Company; ISBN 188477749X
 *
 * http://nitric.com/jnp/
 *
 * Copyright (c) 1997-1999 Merlin Hughes, Michael Shoffner, Derek
Hamner;
 * all rights reserved; see license.txt for details.
 */

import java.awt.*;
import java.rmi.*;
import java.awt.event.*;
import java.rmi.registry.*;

public class RMIChatClient implements Runnable, ActionListener {
  protected static final int UPDATE_DELAY = 10000;

  protected String host;
  protected Frame frame;
  protected TextField input;
  protected TextArea output;

  public RMIChatClient (String host) {
    this.host = host;

    frame = new Frame ("RMIChatClient [" + host + "]");
    frame.add (output = new TextArea (), "Center");
    output.setEditable (false);
    frame.add (input = new TextField (), "South");
    input.addActionListener (this);
    frame.addWindowListener (new WindowAdapter () {
      public void windowOpened (WindowEvent ev) {
        input.requestFocus ();
      }

      public void windowClosing (WindowEvent ev) {
        stop ();
      }
    });
    frame.pack ();
  }
  protected RMIChatServer server;
  protected Thread updater;

  public synchronized void start () throws RemoteException,
NotBoundException {
    if (updater == null) {
      Registry registry = LocateRegistry.getRegistry (host);
      server = (RMIChatServer) registry.lookup
(RMIChatServer.REGISTRY_NAME);
      updater = new Thread (this);
      updater.start ();
      frame.setVisible (true);
    }
  }

  public synchronized void stop () {
    if (updater != null) {
      updater.interrupt ();
      updater = null;
      server = null;
    }
    frame.setVisible (false);
  }

  public void run () {
    try {
      int index = 0;
      while (!Thread.interrupted ()) {
        String[] messages = server.getMessages (index);
        int n = messages.length;
        for (int i = 0; i < n; ++ i)
          output.append (messages[i] + "\n");
        index += n;
        Thread.sleep (UPDATE_DELAY);
      }
    } catch (InterruptedException ignored) {
    } catch (RemoteException ex) {
      input.setVisible (false);
      frame.validate ();
      ex.printStackTrace ();
    }
  }

  public void actionPerformed (ActionEvent ev) {
    try {
      RMIChatServer server = this.server;
      if (server != null)
        server.addMessage (ev.getActionCommand ());
      input.setText ("");
    } catch (RemoteException ex) {
      Thread tmp = updater;
      updater = null;
      if (tmp != null)
        tmp.interrupt ();
      input.setVisible (false);
      frame.validate ();
      ex.printStackTrace ();
    }
  }

  public static void main (String[] args) throws RemoteException,
NotBoundException {
    if (args.length != 1)
      throw new IllegalArgumentException ("Syntax: RMIChatClient
<host>");
    RMIChatClient chatClient = new RMIChatClient (args[0]);
    chatClient.start ();
  }
}
```