RMI and CORBA/IDL

Vittayasak Rujivorakul

Lecture 5

RMI and CORBA/IDL

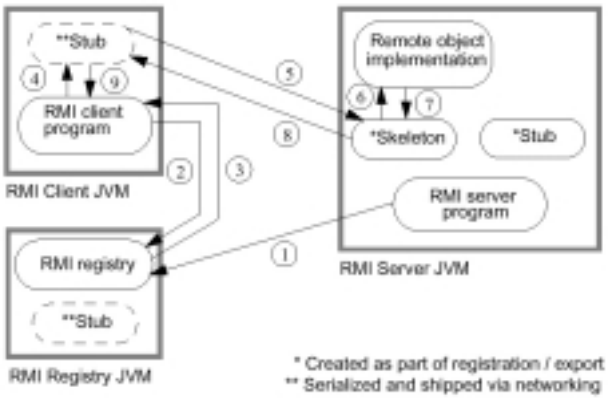
Vittayasak Rujivorakul

Lecture 5

RMI

- Remote Method Invocation
- อนุญาตให้ access object ข้ามเครื่องและข้าม JVMS
- RMI solution ประกอบด้วย 3 ส่วน คือ
  - Server
  - Client
  - RMI Registry
- เราเรียกว่า remote object ซึ่งจะกำหนดรูปแบบการเรียกใช้โดย remote interface

RMI Client, Server and Registry Interaction



ขั้นตอนการทำงาน

1. โปรแกรม RMI Server ทำการสร้าง instance ของ remote object และทำ serialized ส่งผ่าน stub ไปยัง RMI registry
2. โปรแกรม RMI Client เรียกใช้งาน object จาก registry
3. RMI registry ทำการ return สำเนา serialized ของ stub ให้กับ RMI client RMI Client ทำการ deserialized ของ stub เพื่อทำการสร้าง instance
4. RMI Client เรียกใช้งาน remote object's methods ผ่านทาง stub instance
5. Stub ของ RMI client ทำการติดต่อกับ skeleton ของ RMI Server
6. Skeleton invokes method (โหลดสู่ memory) ของ remote object
7. Remote object ส่งผลลัพธ์ของการทำงาน (คำนาม) ให้กับ skeleton

ขั้นตอนการทำงาน (ต่อ)

8. Skeleton ส่งค่าผลลัพธ์ที่ได้ กลับไปยัง RMI client stub
9. RMI client stub ส่งผลลัพธ์ให้กับ RMI client program (Java 1.2 ตัว skeleton จะอยู่ใน Remote object)

Implementing a Basic RMI Server

1. สร้าง remote interface โดยการ extend จาก java.rmi.Remote
2. Implementation remote interfaced และ extend java.rmi.server.Unicast Remote object
3. ทำการ Instantiate remote object
4. Register remote object stub กับ RMI registry

Implementing a Basic RMI Client

1. ใช้ RMI Naming services ในการ lookup และสร้าง remote object stub จาก RMI registry
2. Invoke remote objects methods ผ่าน stub

Installing and Invoking the Server and RMI Registry

1. วาง interface class ไว้ที่ client และ server
2. วาง remote object ไว้ที่ฝั่ง server
3. สร้าง stubs และ skeletons โดยใช้ rmic
4. วาง stubs ที่ฝั่ง client และวาง skeletons ที่ server
5. Start RMI registry โดยเรียก rmiregistry
6. Start โปรแกรม ฝั่ง server
7. Start โปรแกรม ฝั่ง client

RMI and Servlet

- Servlet สามารถเป็นได้ทั้ง RMI client และ RMI server เช่น
  - Servlet ที่เป็น RMI client ที่ access remote database object
  - การใช้ applet ในการ invoke (ตอบรับ) servlet

ตัวอย่าง การสร้าง RMI Servlet

Java IDL

- Interface Definition Language
- ยอมให้มีการใช้ Remote Object
- ใช้งาน Object Request Broker (ORB)
- บางครั้ง อาจจะ implement ผ่าน servlet
- ไม่เป็น plat form independent
- ไม่สามารถย้าย (serialize) object

//Database.java

```
import java.net.*;
import java.util.*;
import java.rmi.*;

public interface Database extends Remote
{
	public void addClient(String ip_addr, Calendar date) throws RemoteException;
	public Vector getClientVisits(String ip_addr) throws RemoteException;
	public long getNumberOfVisitors() throws RemoteException;
	public long getNumberOfClientVisits(String ip_addr) throws RemoteException;
	public void close() throws RemoteException;
}
```

//DatabaseImpl.java

```
import java.sql.*;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
import java.net.*;
import java.util.Date;

public class DatabaseImpl extends UnicastRemoteObject implements Database
{
	private Connection connection;
	private Statement statement;
    private Date thedate;

	public DatabaseImpl(String driverName, String url) throws StartupException, RemoteException
	{
		try
		{
			//let's setup our connection to the database using JDBC
			setupDBConnection(driverName, url);
		}
		catch(Exception e)
		{
			throw new StartupException(new String("Had a problem connecting to db: "+url));
		}
	}

	/*************************************************************
	* setupDBConnection is used to:
	*  1)  install the JDBC driver for mSQL
	*  2)  establish a connection to the DB
	*  3)  create a Statement allowing us to interact with the DB
	*************************************************************/
	private void setupDBConnection(String driverName, String url) throws Exception
	{
		System.err.println("Loading class");
		try {
	  new imaginary.sql.iMsqlDriver();
		} catch (Exception e) {
		  System.err.println("Did not load Driver");
		}
		System.err.println("Loaded class");

		try {
		  connection = DriverManager.getConnection(url);
		} catch (Exception e) {
		  System.err.println("Did not get a connection");
		}
		System.err.println("Got connection");

		try {
```

```
                statement = connection.createStatement();
                } catch (Exception e) {
                  System.err.println("Could not create statement");
                }
                System.err.println("Created statement" + statement);
        }

        /**************************************************************
        * addClient adds a record to the Client table of the DB
        * it converts the String to a String representation and
        * the Calendar to a String (because of the Mini-SQL limitations)
        * normally, our business logic or rules would be placed in here
        **************************************************************/
        public void addClient(String ip_addr, Calendar date) throws RemoteException
        {

          thedate = date.getTime();
          System.out.println("DatabaseImpl addClient called with " + ip_addr + " and " + thedate);
                String date_str = thedate.toString();

                try
                {
        System.err.println("Here it is again: " + statement);
int numOfRows = statement.executeUpdate("INSERT INTO Client VALUES ('"+ip_addr+"','"+date_str+"')");
                }
                catch(SQLException sqe)
                {
                  sqe.printStackTrace(System.err);
                        throw new RemoteException("Could not add record containing "+ip_addr+" : "+date_str);
                }
                catch(Throwable t)
                  {
                    t.printStackTrace(System.err);
                  }
        }

        /*****************************************************************
        *       getClientVisits returns a Vector of Strings, each of which
        * represents a visit
        *****************************************************************/
        public Vector getClientVisits(String ip_addr) throws RemoteException
        {
                try
                {
                        //query the db for all records containing the ip_addr
                        ResultSet results = statement.executeQuery("SELECT * FROM Client WHERE ip_addr='"+ip_addr+"'");

                        //create our Vector to hold the records
                        Vector dates = new Vector(10,5);
                        //go thru the ResultSet converting each record into
                        //an element of the Vector
                        while(results.next())
                        {
                                dates.addElement(results.getString(2));
                        }
                        //return all of the records
                        return dates;
                }
                catch(SQLException sqle)
                {
                        throw new RemoteException(new String("Could not get records for: "+ip_addr));
                }
        }

        /*****************************************************************
        * getNumberOfVisitors calculates the total number of visitors in
        * the database
```

```
	**********************************************************************/
	public long getNumberOfVisitors() throws RemoteException
	{
		//number of visitors
		long numOfVisitors = 0L;

		try
		{
			//try to get all of the visitors in the db
			ResultSet results = statement.executeQuery("SELECT * FROM Client");

			//step thru our ResultSet
			while(results.next())
			{
				numOfVisitors++;
			}

			return numOfVisitors;
		}
		catch(SQLException sqle)
		{
			throw new RemoteException(new String("Could not calculate the number of visitors"));
		}
	}

	/**********************************************************************
	*	getNumberOfClientVisits calculates the number of times a client
	* with the given String interacted with our service
	**********************************************************************/
	public long getNumberOfClientVisits(String ip_addr) throws RemoteException
	{
		//number of visits
		long numOfVisits = 0L;

		try
		{
			//lets query the db, finding the records which contain this ip_addr
			ResultSet result = statement.executeQuery("SELECT * FROM Client WHERE ip_addr='"+ip_addr+"'");
			//step thru our ResultSet to count the number of records
			while(result.next())
			{
				numOfVisits++;
			}

			return numOfVisits;
		}
		catch(SQLException sqle)
		{
			throw new RemoteException(new String("Could not interact with DB"));
		}
	}

	public void close() throws RemoteException
	{
		try
		{
			connection.close();
		}
		catch(Exception e)
		{
			throw new RemoteException("Could not close Database");
		}
	}
}
```

//RMIServer.java

```
import java.rmi.*;

class RMIServer
{
	private final static String driverName = "imaginary.sql.iMsqlDriver";
	private final static String url = "jdbc:msql://tbt-test3.Central.Sun.COM:1112/ClientLog";

		public static void main(String args[])
		{
			try
			{
				Database database = new DatabaseImpl( driverName, url );
				Naming.rebind("ClientLog", database);
			}
			catch(StartupException se)
			{
				System.err.println("StartupException: "+se.getMessage());
			}
			catch(Exception e)
			{
				System.err.println("An Eror occured during RMI Server initialization.");
				e.printStackTrace();
				System.exit(0);
			}
		}
}
```

//ClientLogServlet.java

```
import java.io.*;
import java.util.*;
import java.rmi.*;
import javax.servlet.*;

public class ClientLogServlet extends GenericServlet
{
		static
		{
			System.err.println("CLIENTLOGSERVLET CLASS LOADED");
		}

		private Database database;
		private ServletContext context;
		private final String rmiParam = "RMI_Registry_Server";
 private final String remoteObjParam = "Remote_Object_Name";

		private String rmiRegistry, remoteObjectName;

		public void init(ServletConfig config) throws ServletException
		{
			super.init(config);
			context = config.getServletContext();
			context.log("init in ClientLogServlet called");

			try
			{
				rmiRegistry = config.getInitParameter(rmiParam);
				System.err.println("rmiRegistry: "+rmiRegistry);
				remoteObjectName = config.getInitParameter(remoteObjParam);
				System.err.println("remoteObjParam: "+remoteObjectName);
				if(rmiRegistry != null && remoteObjectName != null)
				{
					System.err.println("URL: "+"rmi://"+rmiRegistry+"/"+remoteObjectName);
					database = (Database) Naming.lookup("rmi://"+rmiRegistry+"/"+
				remoteObjectName);
```

```
                        System.err.println("found registry");
                }
                else
                {
                        log("InitParams were not specified... using default values");
                        database = (Database) Naming.lookup("rmi://localhost/ClientLog");
                        System.err.println("found default registry");
                }
        }
        catch(Exception e)
        {
                System.err.println("Did NOT work");
                e.printStackTrace(System.err);
                context.log(e,"ERROR GETTING RMI REGISTRY");
                destroy();
        }
}

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
        try
        {
                context.log("service ClientLogServlet called");
                System.err.println("Request class: "+req.getClass());
                System.err.println("Response class: "+res.getClass());
                ServletOutputStream resOut = res.getOutputStream();
                System.err.println("got output stream");
                ObjectOutputStream oos = new ObjectOutputStream(resOut);
                System.err.println("got object output stream");
                String clientInetAddr = req.getRemoteAddr();
                System.err.println("got clientIntAddr: "+clientInetAddr);

                database.addClient(clientInetAddr, Calendar.getInstance());
                System.err.println("added client: ");
                long totalVisitors = database.getNumberOfVisitors();
                System.err.println("got totalVisitors: "+totalVisitors);
                long clientVisits  = database.getNumberOfClientVisits(clientInetAddr);
                System.err.println("got clientVisits: "+clientVisits);
                Vector dates = database.getClientVisits(clientInetAddr);

                oos.writeLong(totalVisitors);
                System.err.println("wrote totalVisitors");
                oos.writeLong(clientVisits);
                System.err.println("wrote clientVisits");
                oos.writeObject(dates);
                System.err.println("wrote dates");
        }
        catch(Exception e)
        {
                System.err.println("Problem in ClientLogServlet: ");
        e.printStackTrace(System.err);
                context.log(e,"ERROR IN CLIENT LOG SERVLET");
        }
}

public void destroy()
{
        try
        {
                database.close();
        }
        catch(Exception e)
        {
                System.exit(0);
        }
}
}
```

//RMIApplet.java

```
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.*;
import java.util.*;

public class RMIApplet extends Applet
{
 static {
   System.out.println("Loading class");
 }

 public RMIApplet() {
   System.out.println("Constructing applet instance");
 }

        private String servletName;
        private long clientAccess, totalAccess;
        private Vector dates;
        private TextArea report = new TextArea(5, 20);

        public void init()
        {
        add(new Label("RMItApplet"));
          try
                {
                        servletName = getParameter("servlet");
                        System.out.println("Got Servlet Name" + servletName);

                 setSize(500,500);
                        URL documentBase = getDocumentBase();
                        System.out.println("Got Document Base" + documentBase);

                        URL servletAddress = new URL(documentBase, servletName);
                        System.out.println("Got Servlet Address" + servletAddress);

                        InputStream input = servletAddress.openStream();
                        System.out.println("Got InputStream" + input);

                        ObjectInputStream objectInput = new ObjectInputStream(input);
                        System.out.println("Got ObjectInputStream" + objectInput);

                        totalAccess = objectInput.readLong();
                        System.out.println("Performed first Read" + totalAccess);

                        clientAccess = objectInput.readLong();
                        System.out.println("Performed second Read" + clientAccess);

                        dates = (Vector) objectInput.readObject();
                        System.out.println("Read Vector" + dates);

                        report = new TextArea(5,40);
                        add(report);
                        buildReport();
                        validate();
                }
                catch(Exception e)
                {
                        System.err.println("EXCEPTION: "+e.toString());
                }
        }

        private void buildReport()
        {
```

```
                report.setText("Thanks for accessing our web site!");
                report.append("\nThe total number of visitors is: "+totalAccess);
                report.append("\nYou have accessed this site "+clientAccess+" times");
                report.append("\n\nDates accessed:");
                for(int i=0;i<dates.size();i++)
                {
                        report.append("\n"+(String)dates.elementAt(i));
                }

                System.err.println("REPORT: \n\n\n"+report.getText());
        }
}
```

//RMIApplet.html

```
<html>
  <head><title>RMI Servlet Applet</title></head>
  <body>
    <H1>Howdy All</h1>
    <APPLET CODE="RMIApplet.class" CODEBASE="/applets/" width=150 height=100>
    <param name=servlet value=servlet/ClientLogServlet>
  </APPLET>
  </body>
</html>
```

//StartupException.java

```
class StartupException extends Exception
{
        private String message;

        StartupException(String m)
        {
                message = m;
        }

        public String getMessage()
        {
                return message;
        }
}
```