Event Based Two Way Communication

From Rxtx

Below is a simple program that shows how to open a connection to a serial device and then interact with it (receiving data and sending data). One thing to note is that the package gnu.io is used instead of javax.comm, though other than the change in package name the API follows the Java Communication API (http://java.sun.com/products/javacomm/) . To find the names of the available ports, see the Discovering comm ports example.

This varies from the other 'Two Way Communication' example in that this uses an event to trigger the reading. One advantage of this approach is that you are not having to poll the device to see if data is available.

Note:

Make sure that you call the notifyOnDataAvailable() method of the SerialPort with a boolean true parameter. Based on my experience with RXTX, just registering to the SerialPort as a SerialPortEventListener is not enough -- the event will not be propagated unless you do this.

Note2:

When all is done be sure to unregister the listener (method removeEventListener()). otherwise its possible that your program hangs during exiting causing the serial port to be blocked.

```
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

/**
 * This version of the TwoWaySerialComm example makes use of the
 * SerialPortEventListener to avoid polling.
 *
 */
public class TwoWaySerialComm
{
    public TwoWaySerialComm()
    {
        super();
    }

    void connect ( String portName ) throws Exception
    {
        CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() )
        {
            System.out.println("Error: Port is currently in use");
        }
        else
        {
            CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);

            if ( commPort instanceof SerialPort )
            {
                SerialPort serialPort = (SerialPort) commPort;
                serialPort.setSerialPortParams(57600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);

                InputStream in = serialPort.getInputStream();
                OutputStream out = serialPort.getOutputStream();

                (new Thread(new SerialWriter(out))).start();
```

```
                serialPort.addEventListener(new SerialReader(in));
                serialPort.notifyOnDataAvailable(true);

            }
            else
            {
                System.out.println("Error: Only serial ports are handled by this example.");
            }
        }
    }

    /**
     * Handles the input coming from the serial port. A new line character
     * is treated as the end of a block in this example.
     */
    public static class SerialReader implements SerialPortEventListener
    {
        private InputStream in;
        private byte[] buffer = new byte[1024];

        public SerialReader ( InputStream in )
        {
            this.in = in;
        }

        public void serialEvent(SerialPortEvent arg0) {
            int data;

            try
            {
                int len = 0;
                while ( ( data = in.read()) > -1 )
                {
                    if ( data == '\n' ) {
                        break;
                    }
                    buffer[len++] = (byte) data;
                }
                System.out.print(new String(buffer,0,len));
            }
            catch ( IOException e )
            {
                e.printStackTrace();
                System.exit(-1);
            }
        }

    }

    /** */
    public static class SerialWriter implements Runnable
    {
        OutputStream out;

        public SerialWriter ( OutputStream out )
        {
            this.out = out;
        }

        public void run ()
        {
            try
            {
                int c = 0;
                while ( ( c = System.in.read()) > -1 )
                {
                    this.out.write(c);
                }
            }
            catch ( IOException e )
            {
                e.printStackTrace();
                System.exit(-1);
            }
        }
    }
```

```
    public static void main ( String[] args )
    {
        try
        {
            (new TwoWaySerialComm()).connect("COM3");
        }
        catch ( Exception e )
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }


}
```

Retrieved from "http://rxtx.qbang.org/wiki/index.php/Event_Based_Two_Way_Communication"

- This page was last modified on 2 April 2010, at 22:33.
- Content is available under GNU Free Documentation License 1.2.