

How to Send Email Using Java

by Joel Thompson



Joel Thompson

In this article I'll first provide an introduction to the developer's corner of the NOCOUG Journal and then offer a "how to" on emailing through Java (with JavaMail and with a Java call to /bin/mail).

This is the first article of the developer's corner for NOCOUG; I am honored to be the author of this section, and I hope that you find these articles clear, interesting, and useful. If you have any questions or need clarification about the topics that I present, please feel free to email me at joel@rhinosystems.com. For details of my background, please see the short biography at the bottom of this article.

I will generally focus on software development around Oracle, Java, and SOA technologies; however, I may from time to time, and depending on your feedback, offer articles on C/C++, .NET, Perl, PHP, Ruby, Shell scripting, and more. Perhaps an informal survey would be in order here, and as such, feel free to email me with the top three technologies that you'd like me to discuss.

Before I begin with this article's topic, I'd like to share a bit of my philosophy on software development. I have worked with Oracle databases and software development since 1989. I think Oracle is an excellent platform for software development and database technology; however, I also embrace open-source technologies whenever possible—primarily because of the "free" aspect of using open source. It seems that there are a lot of businesses these days that are following an open-source business model, meaning that they develop a product, open source it, and then charge for their "enterprise edition" or add-ons to the base technology. I don't speak for Oracle, but a similar approach is apparently how Oracle is going. While they don't open source their database technologies, they are giving away the Oracle database XE version as a starter system. This enables businesses to use Oracle technologies for free, knowing that an upgrade path to "enterprise edition" is available if they need it. Oracle is also giving away JDeveloper IDE, which is another big plus for developers and businesses, and an excellent strategy for Oracle.

With JDeveloper you can develop Java and Java Enterprise applications that are completely based on open standards and integrated with open-source technologies. There is a proviso, however, that Oracle's JDeveloper is heavily centered around ADF, which is Oracle's proprietary technology for simplifying

software development. The real issue with ADF is that you have to deploy ADF libraries and pay for it (if you don't already own a license to Oracle's Application Server). Also, keep in mind that you can deploy ADF applications to other application servers beside Oracle, such as JBoss. I have worked with JDeveloper using ADF; it is a very efficient and timesaving piece of technology, and well worth the cost. However, some of the businesses I consult for would like to develop without ADF, so I am intimately familiar with both environments. I will tend to write about applications that are more open-source-standards based.

Now, on to the topic at hand: creating and sending an email with Java using JavaMail and, alternatively, calling /bin/mail from your Java program.

Prerequisites

First, in order to send mail, you must have an SMTP server that you are going to send the mail through. This server can be running on your machine (typically a Unix box) or some other server. Sendmail and Postfix are two examples of SMTP server software. Also, it is worth noting that SMTP typically runs on port 25; be aware that some ISPs will block port 25 communication from your server (at their routers). If they are blocking port 25 you will get an error message like "no route to server." You should also check your machine's firewall configuration and make sure you can send outbound traffic on port 25. You can test this from your machine by typing "telnet smtp.server.com 25" from the Unix command line. If you get connected, you have clear passage; if not, you have to run an SMTP server on your machine.

Sending Email Using JavaMail

Let's get started with the code that is used to create a simple email and send it via JavaMail. We will authenticate with a username/password to establish a session with the SMTP server and send a message with this session.

You need to have the JavaMail libraries in your CLASSPATH; these can be found in the J2EE library in JDeveloper. (Right-click your project and select Properties and then Libraries, and search for the J2EE listing.). You can also find the JavaMail libraries in other environments like the Java J2EE reference implementation or jboss/client lib directories.

Listing 1: The code to send an email message from Java using JavaMail

```

package com.rhinosystemsinc.tools;

/**
 * Sample prepared by Joel A. Thompson
 * At http://www.rhinosystemsinc.com
 * All rights reserved.
 */

import java.util.Date;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

// File: com/rhinosystemsinc/tools/SendJavaMail.java

public class SendJavaMail {

    public static void main(String[] args) {
        SendJavaMail sm = new SendJavaMail();
        sm.send();
    }

    /**
     * This is a simple inner class that extends
     * javax.mail.Authenticator
     * used for gathering a username and password.
     */

    class ThePasswordAuthenticator extends Authenticator {
        String user;
        String pw;

        public ThePasswordAuthenticator(String username,
                                       String password) {
            super();
            this.user = username;
            this.pw = password;
        }

        public PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(user, pw);
        }
    }

    public void send() {
        String mailer = "myprogram";
        //comma separated list of recipients.
        String to = "bob@foobar.com";
        String cc = "bill@ibm.com,mary@oracle.com";
        String bcc = null;
        String from = "joel@rhinosystemsinc.com";
        String subject = "Testing mailing from Java ";
        String body = "This is the body of the test message.\r\n"+
            "Sent on: " + new Date() + "\r\n"+
            "Regards, \r\n" +
            "Joel" ;

        String smtpserver = "smtp.att.yahoo.com";
        String smtpport = "25";

        //my username for smtp.att.yahoo.com was a fully
        //qualified email, like joel@att.com.
        String user = "yourSMTPusername";
        String password = "yourSMTPpasswd";

        Properties props = new Properties();

        //default to "localhost"
        props.put("mail.smtp.host", smtpserver);

        //default to "25"
        props.put("mail.smtp.port", smtpport);

        //uncomment to turn on debugging output which can be useful.
        //props.put("mail.debug", "true");

        //javax.mail.Session
        Session session = null;

```

```

        if (user != null && password != null) {
            props.put("mail.smtp.auth", "true");
            session =
                Session.getInstance(props,
                    new ThePasswordAuthenticator(user, password));
        } else {
            session = Session.getDefaultInstance(props, null);
        }

        /**
         * using
         * javax.mail.Message & javax.mail.internet.MimeMessage
         * javax.mail.internet.InternetAddress
         */
        Message msg = new MimeMessage(session);

        try {

            msg.setFrom(new InternetAddress(from));

            /**
             * Parse the given comma separated sequence of
             * addresses into an Array of InternetAddress objects
             * Then set as the desired recipient.
             */
            msg.setRecipients(Message.RecipientType.TO,
                InternetAddress.parse(to, false));

            if (cc != null)
                msg.setRecipients(Message.RecipientType.CC,
                    InternetAddress.parse(cc, false));

            if (bcc != null)
                msg.setRecipients(Message.RecipientType.BCC,
                    InternetAddress.parse(bcc, false));

            //subject line.
            msg.setSubject(subject);

            //set the body of the message, also consider setContent()
            //for Multipart content
            msg.setText(body);

            //the name of your program.
            msg.setHeader("X-Mailer", mailer);

            //Date sent from this computer.
            msg.setSentDate(new Date());

            // send the message!
            Transport.send(msg);

            System.out.println("\nMail was sent successfully.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

```

Sending Email Using /bin/mail

The next example of sending email out through your Java program doesn't involve JavaMail, but rather relies on the Unix operating system's command "/bin/mail". In this case the machine's SMTP software needs to be running in order to send out an email.

You can test to see if you have "/bin/mail" installed by simply typing it in the Unix command prompt, as shown here:

```

% /bin/mail -v -s testing joel@rhinosystems.com
some message body
.
cc:
%

```

The "-v" turns on debugging so you can see where the message system is or isn't working.

Listing 2: The code to send an email message from Java using "/bin/mail".

```

package com.rhinosystemsinc.tools;

/**
 * Sample prepared by Joel A. Thompson
 * at http://www.rhinosystemsinc.com
 * All rights reserved.
 */

import java.lang.System;
import java.util.*;
import java.io.*;

public class SendCommandMail {

    public static void main(String[] args) {
        SendCommandMail sm = new SendCommandMail();
        sm.send();
    }

    public void send() {
        //comma separated list of recipients.
        String to = "joel@rhinosystems.com,iggy_fernandez@hotmail.com";
        String subject = "Testing mailing from Java";
        String body = "This is the body of the test message.";

        Process proc = null;
        Runtime rt = Runtime.getRuntime();
        String cmdStr="";
        String osName="";
        Properties props = System.getProperties();

        //the operating systems name as known to java.
        osName = props.getProperty("os.name");

        try {
            //Check to make sure we are not on Windows!
            if (!osName.toUpperCase().contains("WINDOWS")) {
                // assume UNIX
                cmdStr = new String("/bin/mail -s \"\" +
                    subject + \"\" " + to);
            }
            //execute the command, it will run as the current user.
            //and the "from" string will be the user@machine-name.com
            proc = rt.exec(cmdStr);

            InputStream is = proc.getInputStream();

```

```

        InputStreamReader ir = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(ir);

        //set the body of the message.
        OutputStream os = proc.getOutputStream();
        os.write(body.getBytes());
        os.close();

        String line = "";
        //show the output of the command here.
        while ((line = br.readLine()) != null) {
            System.out.println("output:" + line);
        }
        br.close();

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

```

Concluding Remarks

The above two samples of code are pretty simple, and hopefully will be helpful to you in your software development. I think it's best to learn by example. All of my examples are tested; however, I am sure there are some environments in which there may be problems. Please feel free to email me and I'll do my best to help you. ▲

Joel Thompson is president of RHINO Systems Inc., a software consulting firm serving the greater Sacramento and Bay Area. He has programmed in Java/J2EE since 1997; prior to that he worked as senior programmer and development manager at Oracle Corporation, beginning in 1989. He resides in Auburn, Calif., with his wife and three children, and enjoys snowboarding, jogging, tennis, and many other athletic activities.

Copyright © 2008, Joel Thompson

Database Management

Develop	Ensure the highest quality code is developed regardless of user skill set or location
Optimize	Maximize code performance and eliminate time-intensive manual tuning processes
Validate	Performance test your code for scalability before deploying to production

Can Toad® Turn Your Team Into Experts?

Sit back and relax. You have Toad's development best practices on your side.

You could spend your day tracking down bad code.
Or you can take control with Toad® for Oracle.

Promote development best practices in your organization.
Read "Implementing Database Development Best Practices" at
www.quest.com/relax



©2007 Quest Software, Inc. All rights reserved. Quest and Quest Software are trademarks or registered trademarks of Quest Software. All other brand or product names are trademarks or registered trademarks of their respective holders. www.quest.com/NoCOUG.

Simplify Identity Management



Enhance security, provision users quickly,
and effectively protect your resources
with IT Convergence's
Oracle Identity Management solutions.

ITConvergence
EXPERIENCE the power of results

Contact us today: www.itconvergence.com