UCLA PIC20C Winter 2002 -02/07/2002

Overview of Sun's JavaMail API

and Internet application development featuring J2EE (servlets, JDBC), MySQL…

Denis Bourdon - dbourdon@dbourdon.com
UCLA Wireless Adaptive Mobility lab
Telecom Grenoble, France

direct SMTP dialogue

> telnet smtp.ucla.edu 25

220 caracal.noc.ucla.edu ESMTP Sendmail 8.9.1a/8.9.1; Fri, 1 Feb 2002 14:33:55 -0800 (PST)
HELO dbourdon.com
250 caracal.noc.ucla.edu Hello [4.40.157.54], pleased to meet you
MAIL FROM: <dbourdon@ucla.edu>
250 <dbourdon@dbourdon.com>... Sender ok
RCPT TO: <dbourdon@cs.ucla.edu>
250 <dbourdon@cs.ucla.edu>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hello world!
How are you going dude?
.
250 OAA10035 Message accepted for delivery
QUIT
221 caracal.noc.ucla.edu closing connection

Message Source

ReturnPath: <dbourdon@ucla.edu>
Delivered-To: online.fr -denbourdon@free.fr
Received: (qmail15896 invoked from network); 1 Feb 2002 22:41:34 -0000
Received: from naza0.gandi.net (80.67.173.201)
by mrelay2-2.free.fr with SMTP; 1 Feb 2002 22:41:34 -0000
Received: from cougar.noc.ucla.edu (cougar.noc.ucla.edu [169.232.10.18])
by naza0.gandi.net (Postfix) with ESMTP id 902A338375
for <dbourdon@dbourdon.com>; Fri, 1 Feb 2002 23:41:33 +0100 (CET)
Received: from caracal.noc.ucla.edu by cougar.noc.ucla.edu
(Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10)
with ESMTP id <0GQV0016YLL9DY@cougar.noc.ucla.edu> for dbourdon@dbourdon.com ;
Fri, 1 Feb 2002 14:39:10 -0800 (PST)
Received: from dbourdon.com ([4.40.157.54])
by caracal.noc.ucla.edu (8.9.1a/8.9.1) with SMTP id OAA10035 for
<dbourdon@ucla.edu>; Fri, 01 Feb 2002 14:37:50 -0800 (PST)
Date: Fri, 01 Feb 2002 14:37:50 -0800 (PST)
From: dbourdon@ucla.edu
To: dbourdon@ucla.edu
Message-id: <200202012237.OAA10035@caracal.noc.ucla.edu>
MIME-version: 1.0
Content -type: TEXT/PLAIN; CHARSET=US-ASCII

Hello world!
How are you going dude?

Motivations

- How to program an automated sender?
  ⇒ UNIX socket programming (😐)
- What if I have attachments?
  ⇒ Which encoding? Base64? Do I have to write the encoder/decoder? (😐)
- Error handling?

LET'S JUMP TO JAVA!!!

Overview of JavaMail

- The JavaMail API is a set of abstract classes that model a mailsystem (POP3, SMTP protocols)
  ⇒ here focus on SMTP (**outgoing** mail)
- Needs the JavaBeans Activation Framework 1.0.1
  "It enables to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and to instantiate the appropriate bean to perform said operations."

Downloads

- JavaMail download:
  Sun's java website:
  http://java.sun.com/products/javamail
  or as a part of Sun's J2EE 1.3
- JAF download:
  http://java.sun.com/products/javabeans/glasgow/jaf.html

First application

"Retrieve your password"

Servlet

- Servlet engine: JSDK WebServer v2.1

  http://java.sun.com/products/servlet
- doGet() displays a simple form where the user enters his username
- doPost() looks to the database; if found, sends an email to the user with hispassword

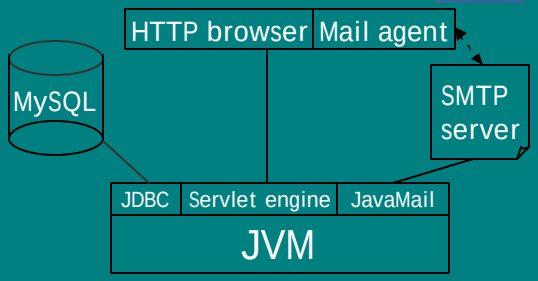
Database access

- JDBC access to MySQL with the most common (and stable) driver: MM Mysql

  http://mmmysql.sourceforge.net/
- SQL detail of the table

```
CREATE TABLE registration (
  username varchar(64) NOT NULL,
  password varchar(64) NOT NULL,
  email varchar(64) NOT NULL,
  PRIMARY KEY  (username)
);
```

Multi-pier diagram



Setting the CLASSPATH

```
%demo_javamail%\mm.mysql-2.0.4-bin.jar;
%demo_javamail%\mail.jar;
%demo_javamail%\activation.jar;
%demo_javamail%\jsdk2.1\src;
```

⇒ a "rich" J2EE application

Servlet

```
public class DemoJavaMail extends HttpServlet {

 public void init( ServletConfig config ) throws ServletException {
    // usual routine for servlets
    super.init(config);
  }

 protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws IOException {
    PrintWriter out = response.getWriter(); // handler for the HTML output
    out.print("<html>\n <body>\n");
    out.print("<form action=\"" + request.getServletPath() + "\"
    method=\"post\">\n");
    out.print("<input name=\"username\">\n");
    out.print("<input type=\"submit\">\n");
    out.print("</body>\n</html>\n");
    out.flush();
  }
```

Servlet (cn.)

```
protected void doPost(HttpServletRequest request, HttpServletResponse
   response) throws IOException {
    PrintWriter out = response.getWriter(); // handler for the HTML output
    String username = request.getParameter("username");
    // work…
  }
```

⇒ more about servlets soon in class!

Database access

```
try {
  Class.forName("org.gjt.mm.mysql.Driver");
} catch(java.lang.ClassNotFoundException e) {
  System.err.println(e.getMessage());
}
try {
  con = DriverManager.getConnection("jdbc:mysql://localhost/demo_javamail",
"root", "");
  stmt = con.createStatement();
  username = request.getParameter("username");
  ResultSet rs = stmt.executeQuery("SELECT email, password FROM registration
WHERE username ='" + username + "'");
  if (rs.next()) {
    // we have such an user and as username is the primary key => one single row
    email = rs.getString("email");
    password = rs.getString("password");
  } else {
    // process that
  }
} catch(SQLException ex) {
    System.err.println("SQLException: \n" + ex.getMessage());
}
```

mail (finally!)

```
String Body = "Your username is \"{0}\" your password is \"{1}\" (without the quotes).";
Body = MessageFormat.format(Body, new Object[] {username, password});

props = new Properties();
props.put("mail.smtp.host", "smtp.ucla.edu");
try {
  mail = new MimeMessage( javax.mail.Session.getDefaultInstance( props, null) );
  mail.setFrom( new InternetAddress("dbourdon@ucla.edu") );
  mail.addRecipient(Message.RecipientType.TO,
        new InternetAddress(username + "<" + email + ">"));
  mail.setSubject("Here is your username and password!");
  mail.setText(Body);
  Transport.send(mail); // send the mail
}
catch(AddressException ex) {
  System.out.println("AddressException: \n" + ex.getMessage());
}
catch(MessagingException ex) {
  System.out.println("MessagingException: \n" + ex.getMessage());
}
```

The Result!

```
Return-Path: <dbourdon@ucla.edu>
Delivered-To: online.fr-denbourdon@free.fr
Received: (qmail 15737 invoked from network); 3 Feb 2002 02:56:08 -0000
Received: from naza0.gandi.net (80.67.173.201)
  by mrelay2-2.free.fr with SMTP; 3 Feb 2002 02:56:08 -0000
Received: from caracal.noc.ucla.edu (caracal.noc.ucla.edu [169.232.10.11])
    by naza0.gandi.net (Postfix) with ESMTP id 75ED43831F
    for <dbourdon@dbourdon.com>; Sun,  3 Feb 2002 03:56:03 +0100 (CET)
Received: from ganon ([4.40.157.54])
    by caracal.noc.ucla.edu (8.9.1a/8.9.1) with ESMTP id SAA29476
    for <dbourdon@dbourdon.com>; Sat, 2 Feb 2002 18:55:47 -0800 (PST)
Date: Sat, 2 Feb 2002 18:55:47 -0800 (PST)
Message-ID: <3153624.1012704965318.JavaMail.javamailuser@localhost>
From: dbourdon@ucla.edu
To: dbourdon <dbourdon@dbourdon.com>
Subject: Here is your username and password!
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit


Your username is "dbourdon" your password is "foo" (without the quotes).
```

Second application

"MailTester"

Objectives

- test several SMTP hosts
- have all "advanced" mail functionalities:
  - TO, CC, BCC multivalued and nominative
  - "reply-to" field
  - attachments
  - HTML body
  - "receipt"

⇒ all JavaMail functionalities, and reusable source code

Data Encryption

- In the GET method, the form is sent using the encryption type (enctype) "multipart/form-data" to transfer attachments

⇒ we need to decrypt it in the POST, using a state-machine to parse the ServletInputStream and store all the data in a Dictionary, i.e. a new Hashtable()

Parsing method

1. find the "Content-Type" boundary in the header sent by the form
2. look for this boundary in the raw data
   - if it's an attribute (from, cc), just add the pair [key, value as a String]
   - if it's a MIME part, add [key, value as a MimeBodyPart]

Putting all together

- The Content of the email is created as a MimeMultipart object which contains indexed BodyPart objects:
  - "body" (i.e. message text) has index 0
  - attachments have indexes 1, 2, …
- class Attachment

  implements DataSource from the JavaBeans Extension Framework, to represent an attachment inside an e-mail.

Result

- JavaMail creates the data "text" from the Message object:

```
From: dbourdon@ucla.edu
To: denbourdon@free.fr
Subject: hello
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="----=_Part_0_2852155.1012714305939"
X-Mailer: MailTester servlet v1.0


------=_Part_0_2852155.1012714305939
Content-Type: text/plain; charset=us-ascii
Content-Transfer -Encoding:  7bit

hello
------=_Part_0_2852155.1012714305939
Content-Type: image/pjpeg;  name=image.jpg
Content-Transfer -Encoding:  base64
Content-Disposition: attachment; filename=image.jpg

xX/YqfaCDUHofuvFNGs0Ugo8bgMrD2EHY6ucji8VBYXV5X5h4QVVixBY8AeIJoKkD+8/biMmL9oI
bDyXtjRilxHyEijZgAyuooSDtyHjXQ3/APoQ2/2vbr9u+1Tr8Ouvd9laU92hXc+NOn+0t8X/AJev
5aj/ABGj16/nr9Ouh16j4evXQ6/l/jodfz6/npOvXw6dD192h/2Ovz/x1//Z
------=_Part_0_2852155.1012714305939--
```