Mass Emailing Techniques

...This is not about spam!

Pascal Robert
MacTI

Why?

- Most of us have to send mass email (newsletter, special offers, reminders, etc.)
- Sending emails can take some time
- Have to avoid to be tagged as spam
- ...And we have to manage bounces

Basic tips

- Put the real name of the recipient in the headers.

```
ERMailDelivery.setFromAddress(from, name)

ERMailDelivery.setToAddress(from, name)
```

- Sending HTML email? Don’t forget to include the plain text version.

```
ERMailDeliveryHTML.setHiddenPlainTextContent()
```

- Your CSS should inline, as in :

```
<div style="font-size: 12px; ">My text</div>
```

Basic tips

- Send your email to various services (Gmail, Hotmail, anti-spam appliance) and check the headers for spam rating.

  Example : (25%) BODY: contains URL text "unsubscribe", (10%) URL: contains "unsubscribe" text

- Check your logs for "blocked using" messages.
- Avoid spam terms ("unsubcribe" in a link, "open in a browser") in the message.

Basic tips

- Try to put a (at least) one second delay between each sent email.
- Don’t sort your recipients list by domain.
- Remove bad addresses for your lists.
- Summary : your email should look like it’s coming from a regular email client.

First way to mass nirvana: ERJavaMail

- ERJavaMail have a queue system...
- And with a WOLongResponsePage...
- You can send email in a separate thread!

First way to mass nirvana

```
public Object performAction() {
try {
    ERMailDeliveryPlainText plainText = new ERMailDeliveryPlainText();
    for (int i = 1; i < 10; i++) {
        plainText.newMail();
        plainText.setFromAddress("probert@macti.ca", "Pascal Robert");
        plainText.setToAddress("probert@macti.ca", "Pascal Robert");
        plainText.setTextContent("This is a test.");
        plainText.setSubject("This is nirvana! (and a good spammy subject!)");
        plainText.sendMail(false);
    }
    } catch (AddressException e) {
        e.printStackTrace();
    } catch (MessagingException e) {
        e.printStackTrace();
    }
    return "OK";
}
```

ERJavaMail : Queue Properties

- Two properties to control the size of the queue :

  er.javamail.senderQueue.size

  er.javamail.milliSecondsWaitIfSenderOverflowed

- Tune the value of those properties by looking at your MTA logs (blacklisting and bounces).
- I suggest a queue size of 5, and 5000 ms for the next wave.

ERJavaMail : SMTP properties

- If you need to use SMTP authentification :

er.javamail.smtpAuth = true

er.javamail.smtpUser = user@domain

er.javamail.smtpPassword = password

mail.smtp.starttls.enable = true

mail.smtp.starttls.required = false

ERJavaMail : Stats

- ERMailSender store basic statistics about the queue : memory usage, number of mail sent, number of errors and number of messages currently in the queue.
- Available with ERMailSender.sharedMailSender().stats()

ERJavaMail : Errors

- Not only ERJavaMail can track statistics and queue your emails, it also track errors.
- Errors = errors that your MTA is capable of detecting without sending the email to a remote SMTP server.
- Works with NSNotificationCenter

```
NSNotificationCenter.defaultCenter().addObserver
(..., ERMailSender.InvalidEmailNotification, ...)
```

- It won't track down errors coming from the recipient's SMTP/MTA server.

DEMO!

ERJavaMail : Bounces

- To handle bounces, you will have to check the email that the sender you use receive.
    - You should use an account that is used only to send emails from your WO apps.
- You can use the IMAP and POP3 from JavaMail to fetch the bounces and analyse them.
- ... Or ask someone to fix your list manually :-)

ERJavaMail : IMAP sample code

```
public boolean openIMAPConnection() {
       javax.mail.Session sessionEmail = javax.mail.Session.getDefaultInstance
(props,null);

        sessionEmail.setDebug(false);
       imapStore = (IMAPStore)sessionEmail.getStore("imap");
       imapStore.connect("imapserver.com", "user", "password");
        inbox = (IMAPFolder)imapStore.getFolder("Retours");
       inbox.open(Folder.READ_WRITE);
       return true;
}

public boolean closeIMAPConnection(boolean purgeDeletedEmails) {
    inbox.close(purgeDeletedEmails);
    imapStore.close();
    return true;
}

public boolean corrigeErreur(InfoErreur info) {
   if (info != null) {
       MimeMessage message = (MimeMessage)inbox.getMessageByUID(info.uidCourriel
());
       message.setFlag(Flags.Flag.DELETED,true);
       listeErreurs.removeObject(info);
       return true;
   }
   return false;
 }
```

Next nirvana : mailing lists

- Some mailing/distribution list software let you subscribe recipients with a API or with a text file.
- Many advantages : only one email to send, mailing list manager and the MTA take of everything else. And should be more efficient for delivery.
- ... But you can't customize the content per recipient, and bounces handling can be a pain.

Mailing lists : CommuniGate Pro example

```
    private CGProCLI cli;

public void connection(String adminUsername, String adminPassword) {
    cli = new CGProCLI("mycgp.server.com",106,adminUsername,adminPassword);
}

public void createMailingList(String listEmail, String listOwnerEmail,
Hashtable preferences) {
    cli.createList(listEmail,listOwnerEmail);
    if ((preferences != null) && (preferences.size() > 0)) {
        cli.updateList(listEmail,preferences);
    }
}

public void addSubscriber(String listName, String subscriberEmail, String
subscriberRealName) {
    if (subscriberRealName != null) {
        cli.list(listName,"subscribe","\"" + subscriberRealName + "\" <" +
subscriberEmail + ">",true,false);
    } else {
        cli.list(listName,"subscribe",subscriberEmail,true,false);
    }
}
```

DEMO!

Third option : campaign monitor systems

- More recent campaign monitor systems (MailChimp, CakeMail, ...) have APIs to create lists and campaigns from external systems.
- You only need to manage lists and campaigns, marketing can use the tools provided by the provider to send the messages.
- ... But it can cost a couple of $.

Third option : campaign monitor systems

- MailChimp offer both an XML-RPC or a JSON/HTTP API.
- Content can be created with their Web interface, or with the API (as a HTML/text string or by providing a URL).
- The API will check for invalid (badly formatted) email addresses when you subscribe people and send you a list of errors.

Third option : campaign monitor systems

- MailChimp allow offers web hooks. This allow MailChimp to contact a URL that you specify to get the list of people who unsubscribed or bounces (soft or hard).
- They also have HTML code to create subscribing forms that you can embed in your pages.
- Inbox Inspector (extra $) let check your content and layout with multiple mail clients and anti-spam tools.

MailChimp subscribe

```
    private IMailChimpServices    mcServices   = null;
private String apiKey = "xxxxxxxxxxxxxxxxxxxx";
private String listId = "a212de2a40";
public NSMutableArray<String> subscribersErrors;

mcServices = MailChimpServiceFactory.getMailChimpServices();

for (Iterator<? extends Subscribers> itrSubscribers = subscribers.iterator(); itrSubscribers.hasNext();) {
    Subscribers subscriber = itrSubscribers.next();
    if (subscriber.email() != null) {
        HashMap<String,String> addresses = new HashMap<String,String>();
        addresses.put("EMAIL",subscriber.email());
        addresses.put("EMAIL_TYPE","HTML");
        addresses.put("OPTINIP", "192.168.0.1");
        addresses.put("LNAME", subscriber.lastName());
        addresses.put("FNAME", subscriber.firstName());
        emails.add(addresses);
    }
}

    HashMap<String, Object> result = mcServices.listBatchSubscribe(apiKey, listId, emails, false, true, false);
    NSLog.out.appendln(result.get("error_count"));
    NSLog.out.appendln(result.get("success_count"));
    Object[] errors = (Object[])result.get("errors");
    if (errors.length > 0) {
        for (int i = 0; i < errors.length; i++) {
            if (errors[i] instanceof HashMap) {
                subscribersErrors.addObject(((HashMap)errors[i]).get("message"));
            }
        }
    }
```

DEMO!

Alternatives

- Apache James. It's a MTA written in Java who offers SMTP, NNTP, POP3 and IMAP services. James Mailets might be a good option to handle bounces.
- Direct MTA submission (with files). CGP and Postix allow this.