WEB MAIL

BY SIKE HUANG AND SHANBO LI

1. INTRODUCTION

1.1 WHAT IS THIS ABOUT?

iMail (Web Mail) is a web server application. It sits on a TCP port number and waits for HTTP requests. Users can fill an email form from any browser, which includes From, To, Subject, SMTP Server (Optional), Message. Users can also decide when the email should be sent out.

1.2 FEATURES

- SMTP Server is optional
- Schedule sending time
- All fields have validation
- Result notification via email
- All mails' status can be seen from a list
- Support character set ISO-8859-15

2. PROBLEM

- Web Server, Schedule and Status List
    - Response HTTP request
    - Schedule emails send time
    - Store mail status and change state when finish sending email
    - Automatically send notification to sender's email address
- SMTP Client and Character Set
    - Find MX by DNS lookup of receiver address
    - Conversation with SMTP server
    - ISO-8859-15 character encoding in subject
    - MIME support in message body
    - Error handling

3. Solution

3.1 Web Server, Schedule and Status List

Response HTTP request

GET and POST request will be treated by different handler. GET is used for showing email form and the status page. POST will be handled by Schedule method which will start a timer and send the email in time.

Schedule emails send time

In package *time* a Singleton Class *SendMail* is used to Schedule sending email by invoking *SendMailTask* class.

Store mail status and change state when finish sending email

In package *time* a Class named *MailList* is used to store all email history. Each email's initial state is *PENDING*. Since *MailList* class extends Observer, it can observe *SmtpClient*'s state changing and modify the mail state when *SmtpClient* finishes sending email.

Automatically send notification to sender's email address

When the state of *SmtpClient* changes, it means that an email finished sending. And *MailList* can get the result of it by observing the *SmtpClient* class. *MailList* will call *SmtpClient*'s *simpleSend* method to send a notification to the sender.

3.2 SMTP Client and Character Set

Find MX by DNS lookup of receiver address

MX records are retrieved by Java Naming and Directory Interface. First and foremost, domain name is parsed out by splitting receiver's address with "@" sign, then JNDI contacts with DNS and returns a list of mail exchange servers for that domain.

Conversation with SMTP server

A TCP session is set up to certain SMTP server, and the server is either specified by user, or picked from the first entry in the list of mail exchange servers obtained before. After that, an input stream as well as an output stream is initialized, the input stream is used to issues SMTP instructions, while the output stream gives the responses from the server, and each and every response is checked in order to make error handling feasible.

ISO-8859-15 character encoding in subject

Subject is interpreted as encoded-word [rfc2231], more precisely as "=?iso-8859-15?q?encoded-text?=", encoded-text is quoted printable.

MIME support in message body

Three MME headers are inserted, “mime-version: 1.0”, “content-type: text/plain; charset=ISO-8859-15” and “content-transfer-encoding: quoted-printable”, the content of message is put to MIME body, and the content is quoted-printable encoded as well. Class *assignment1.util.UrlDecoder* transfers url-encoded string to quoted-printable one. For example, “<CR><LF>” in the message is posted as “%0D%0A” and replaced by “=0D=0A” to be sent.

Error handling

As said in “Conversation with SMTP server”, the *SmtpClient* checks the response from the server, if there is any unexpected return code, a *SmtpException* will be thrown and given to the status page.

4. Discussion and conclusions

4.1 Possible application areas

This application is relatively simple, and we consider it mainly as an academic try-out. It is suitable to send western European text message.

4.2 Possible extensions and modifications

- The whole solution can be separated to 2 layers. One is the web server and mail list history. Another is SmtpClient. Each of them can be distributed alone and integrate with any other framework which has the similar interface.
- The solution use a URL decoder and ISO-8859-15 decoder which can be changed to any other decoder so our solution can support almost all popular character set.
- An attachment function could be added to the solution.
- Use a database to store mail history