Lab Five
SMTP

CSE 283

Description:

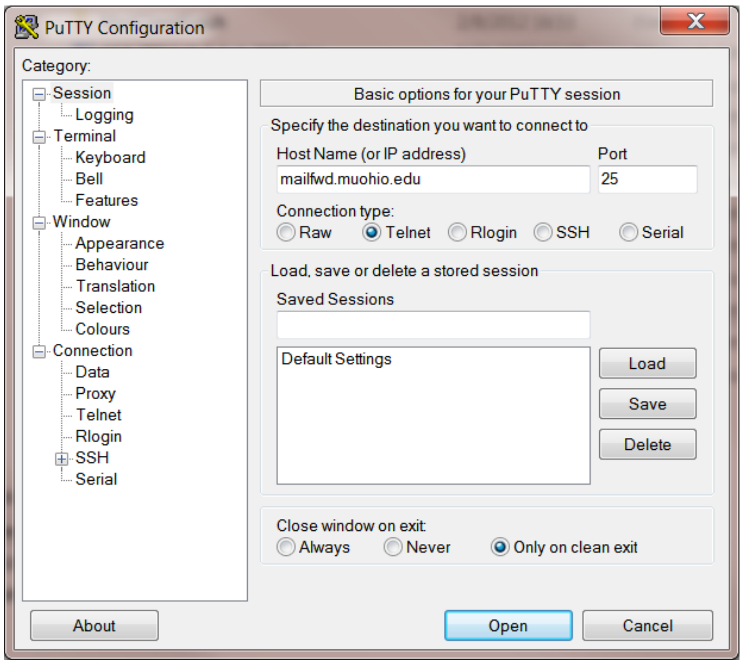
In this lab you will send an email message by communicating with the mail server using the SMTP application layer protocol. You start by executing the client side of the SMTP by typing the messages at the command line. The SMTP messages will be sent via the telnet and must follow the specific order and format as defined by the SMTP protocol. Pay very close attention to when you hit enter (send carriage return and line feed characters) and how and when the server respods to you as you send your message.

Once you have successfully sent email using the command line. You will complete a program that opens a connection to the server for you and sends an email message.

Note: To do this lab at home, you will most likely have to go into your firewall settings and open port 25.

Sending Mail the Old Fashioned Way

1. Click on `PuTTY` to start it up and begin a Telnet session. Set the Host Name to the name of Miami University's SMTP server, `mailfwd.muohio.edu` text field. Edit the Port text field to `25`. Click the Telnet radio button. Then select Open.



Host Name (or IP address): mailfwd.muohio.edu
Port: 25
Connection type: Raw [✓ Telnet] Rlogin SSH Serial
Close window on exit: Always Never [✓ Only on clean exit]

2. You can use the telnet program to send text through a TCP socket. If the text happens to be the commands of a particular protocol and the receiver on the other end understands the protocol, you can communicate with it. Since you have just connected to an SMTP mail server, it understands the SMTP protocol.

3. Greet the mail server by typing the following and then hitting enter. Remember the enter is sent as two characters, a carriage return and line feed (\r\n).

   ```
   helo mailfwd.muohio.edu
   ```

4. Tell the server who the mail is coming from by typing the following and then hitting enter (again carriage return and line feed).

   mail from: <*yourUniqueId*@miamiOh.edu>

5. Tell the server who you are sending mail to by typing the following and then hitting enter (another \r\n). (Either send the mail to yourself or your neighbor)

   rcpt to: <*uniqueId*@miamiOh.edu>

6. Tell the mail server you are ready to type the message by typing the following and then hitting enter (yet another \r\n).

   ```
   data
   ```

   From this point on, the server will not respond to you each time you hit enter. It is looking for a special character sequence (\r\n **.** \r\n). When it receives the sequence, it will know you are done typing your message and send it. Otherwise, it will assume you are continuing to type your message.

7. Enter a “to” header line by typing the following and then hitting enter (The mail server does not send a response to this carriage return and line feed).

   to: *uniqueId*@muohio.edu

8. Enter a “from” header line by typing the following and then hitting enter. This would be your “reply to” email address. (Again, the mail server does not send a response to the carriage return and line feed).

   from: *uniqueId*@muohio.edu

9. Enter a “subject” header line by typing the following and then hitting enter. (Still, the mail server does not send a response to the carriage return and line feed).

   subject: *your subject*

10. Type the text that will be contained in your message. After you are done typing, hit enter. Type a single period at the beginning of the line and then hit enter again. The server will finally respond at this point. It treated everything you typed since sending the Data command as one command.

11. End the session and close Putty by typing

    ```
    quit
    ```

12. Check to see if the mail was sent and received.

Starting Eclipse

1. Copy the MailClientStart from the Lab 5 directory to a location in your work area where you want your project stored.

2. Change the name of the directory to your unique ID followed by *Lab5*. For example *bachmaerLab5*.

3. Launch Eclipse.

4. Select *File -> New -> Project…*

5. Select *Java* and *Java Project* and then *Next >>*

6. Give the project a name. The name should be your unique ID followed by *Lab5*.

7. Deselect *default* and browse to the location of the new directory in your work area and select it. Give the project directory the same name as the project.

8. Select *Finish*.

Your Task for Today

Make Java code do exactly what you just did at the command line. You will use three classes and one interface to do this.

- An instantiation of the MailInterface class creates a GUI interface for the user and allows the user to enter the body of an email message and the information necessary to send it.

- An instantiation of the MessageBuilder class takes the information from a MailInterface class object and uses the information to create the commands and data needed to send the message. In order to receive events from MailInterface objects, it implements the MailInterfaceListener interface.

Note that there is more than one main method in the classes you have been given, the driver for the project is the main defined in the MessageBuilder class. This is often done so that each class can be tested individually. The message class object instantiates the MailInterface and SMTPConnection objects and contains the main method you will want to run for this lab.

- The MessageBuilder class object uses an object of the SMTPConnection class to create a connection with a mail server and send the message.

You will have to complete the sendCommand method of the SMTPConnection class and write nearly all of the MessageBuilder class. You are not required to make any changes to the MailInterface class. However, you may want to set some default values in the MailInterface class in order to make your testing go faster.

Run the main in the main in the MessageBuilder class. You should see interface presented by the MailInterface class. Information appropriate to sending an email message is entered in the text fields and area and sent by pressing the send button. The "outgoing mail server" field should contain the name of the mail server your application needs to contact on port 25 to send the message. Usually this server is associated with the organization that is providing the network connection the application will be using. On campus the SMTP server is `mailfwd.muohio.edu`. The "from" email address should also be associated with the server you are using to send the message. (I have to use `bachmaer@miamiOh.edu` when logging into `mailfwd.muohio.edu`). All of this is for security reasons. (Also, so you don't get sick of typing your email address over and over again as you develop you program, you might want to edit parameters passed to the *fromField* and *toField* TextFields in the MailInterface class so that your email address is passed instead of an empty String.) The "to" field of course contains the email address of the recipient of the message. The "subject" filed contains the subject of the message. The "Message" area contains the text that is to be sent.

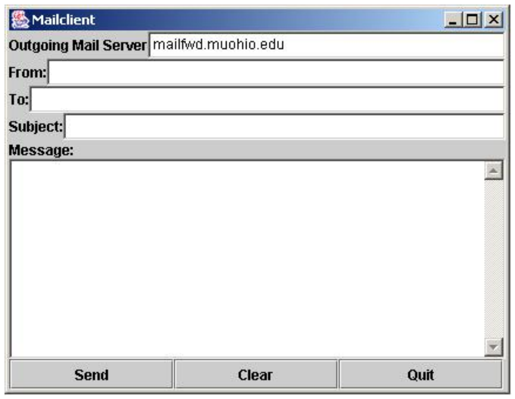


Figure 1: GUI created by the MailInterface Class

The MessageBuilder class includes a constructor and four methods. The constructor instantiates the MailInterface object and passes a reference to itself to the MailInterface constructor. The isValidAddress method is complete and can be used to check if an email address is syntactically valid. The close method is called when the GUI is closed. It is responsible for closing any <u>open</u> streams and sockets as well as freeing any other resources that have been allocated. You must complete this method. The send method of the MessageBuilder class is called each time the send button on the interface is pressed. The method is responsible for seeing that several tasks are completed. These include:

- Instantiating an SMTPConnection object if one has not already be created that can be used to send the message.
- Retrieving the data contained in the text fields and areas of the GUI interface using assessor methods of the MailInterface class.
- Checking the data entered by the user (valid email addresses, etc.)
- Using the data to create the necessary commands to send the message.
- Using the SMTPConnection object to send the commands.
- Checking the response of the server to each command.
- Insuring that the sendMail method returns true if the message was sent successfully and false otherwise.

The version of the send method you have been given retrieves the contents of the text fields and the text area in the MailInterface. It currently always returns false. You must write all code needed to accomplish the above actions. Break things up and add whatever methods you feel are necessary to the Message class. Your message class must create and send following commands and check for the respective reply codes.

• mail from: <*email address*>	reply code 250
• rcpt to: <*email address*>	reply code 250
• data	reply code 354

Each of these commands must be followed by a carriage return and a line feed. Note that the carriage return and line feed characters are already defined in a public static data member of the SMTPConnection class called CRLF.

The body (data) of the message can be sent as one long string and must include the following header lines:

- from: *email address*
- to: *email address*
- subject: *message subject line*

Each header line is followed by a carriage return and a line feed. The message text follows the subject line and must be terminated by a carriage return and a line feed, a period, and then another carriage return and a line feed. The reply code is 250. All of this text should be sent using the sendCommand method of the SMTPConnection class. Any problems with transmitting the message should be handled by printing an error message and having the send method of the Message class return false. If the message is successfully transmitted, a true should be returned to the MailInterface.

The SMTPConnection class includes a constructor and two methods. The constructor must be passed a String containing the address of the smtp mail server. It creates a Socket for TCP communication with the server. After the socket is created it retrieves the associated input and output streams and uses them to read and check the initial reply from the server as well as send a HELO command and checks the reply. If there is any problem with any of these operations, the constructor throws an exception. The parseReply method can be used to parse out an int representation of the code contained in a reply from the server. The SMTPConnection class constructor gives you example code on how to send and receive text using a TCP connection. You should look at this method when you are implementing your sendCommand method.)

1. You must implement the sendCommand method of the SMTPConnection class. It must take in an argument of type String that contains a command and an int that contains the desired return code from the server. It sends the command to the server. If the return code sent by the server equals the expected return code it returns true. Otherwise it returns false. Again, the SMTPConnection class constructor contains example code for sending commands, reading replies and parsing those replies. The sendCommand method can add a carriage return and a line feed to each received command string or this can be done in the Message class where the messages are put together. It is up to you, but you must pay attention to getting the carriage return and a line feeds correct.

2. You must complete the sendMail and close methods of the MessageBuilder class.

3. Test your client by sending email to yourself.

4. Once you have the program working, use it to send me an email message (`Eric.Bachmann@miamiOh.edu`). Let me know how life is treating you and how the class is going.

Final Notes and Turnin:

Make sure your project is in a folder with a name consisting of your unique ID followed by *Lab5*.

Use the standard window compression tool (right click on the folder and select ‘Send to -> Compressed (zip) folder’ to zip up your project (no 7zip, rars, etc.). The .zip folder should have the same name as your project.

Submit your lab through the niikha website for the class. If there is something unusual about your submission such as a bug you were unable to fix or the assignment was not completed. Please enter a comment in niikha as you upload you submission.

Instructions for Getting telnet to Run on Windows 7

1. Click Start.
2. Click Control Panel.
3. Now, go to Programs or Programs and Features.
4. Click Turn Windows features on or off.
5. In the Windows Features dialog box, tick Telnet Client check box.
6. Click OK.