Base Language - Feature #3343

implement a 4GL syntax extension and the runtime backing to send emails (including attachments)

09/28/2017 11:46 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Conversion Tools - Bug #3353: lexer num_literal should support hex...	**Closed**

History

#1 - 10/19/2017 12:00 PM - Greg Shah

- Related to Bug #3353: lexer num_literal should support hexadecimal literals added

#2 - 10/19/2017 12:13 PM - Greg Shah

- Status changed from New to WIP

Which Java Library

The following tutorial is useful for seeing common use cases of the JavaMail API:

https://www.javatpoint.com/java-mail-api-tutorial

Official site:

https://javaee.github.io/javamail/

The reference implementation is not difficult to use, but the Apache Commons Email library is better. In addition, the license of commons-email is Apache 2.0 so it is OK to use. The Oracle JavaMail implementation is CDDL and GPL v2, which is not OK (https://javaee.github.io/javamail/JavaMail-License).

Apache Commons Email:

https://commons.apache.org/proper/commons-email/
https://commons.apache.org/proper/commons-email/userguide.html

Client or Server or Both

A traditional interactive UI that sends emails will expect that the code is executed on the FWD client. Consider a scenario where the FWD server is running in AWS and the FWD client is running at a customer site. If the end user's corporate email account (which is on premises and not accessible at in AWS), then executing on the client will be required. This is consistent with the client-centric design of the 4GL.

On the other hand, appserver, batch and other server-side environments may want to send emails from the FWD server directly to avoid having client platform dependencies. This seems a useful scenario as well.

For this reason, I am planning to allow the 4GL developer to chose client or server (default is client) to use when sending an email. This will appear to be a simple logical attribute that can be optionally set, but the runtime will need to remote the processing to the client when needed. The same code (commons-email and the supporting FWD runtime features) will work in both places.

#3 - 10/20/2017 09:06 AM - Greg Shah

- File fakesmtp_gui.png added

For testing purposes, the FakeSMTP project (http://nilhcem.com/FakeSMTP/) is very useful to easily run a dummy SMTP server that can be used to test **insecure transports (this does not support SSL/TLS or STARTTLS)**. Download version 2.0, unzip the jar file and run it like this:

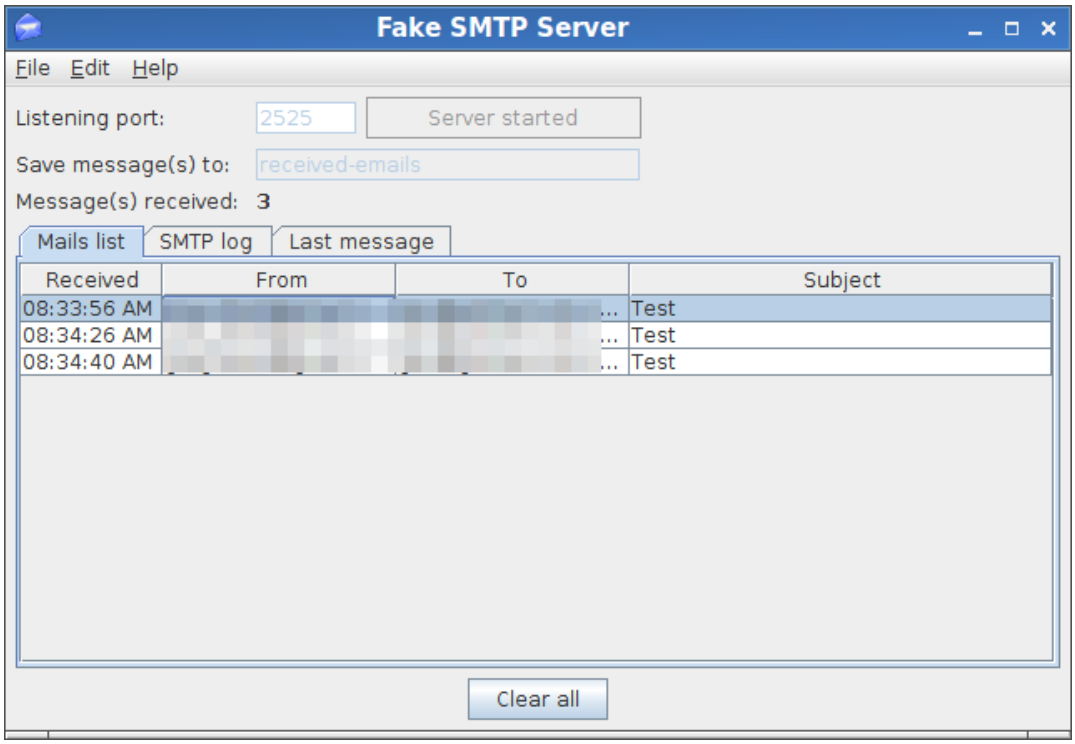
```
java -jar fakeSMTP-2.0.jar -s -p 2525 -m
```

-s starts listening as soon as it is launched.

-p is the port number it listens on.

-m puts it into memory-only mode (it does not store the emails in the file system).

It provides a GUI to monitor the results:



#4 - 10/20/2017 09:32 AM - Greg Shah

It is possible to test both SSL and STARTTLS connection security using GMail:

https://support.google.com/a/answer/176600?hl=en

Follow the instructions for "Gmail SMTP server" and make sure you create an "application password" for direct access to your account (this can be deleted after testing is done):

https://support.google.com/accounts/answer/185833?hl=en

Other notes:

- When you use smtp.gmail.com:465, you must use SSL.
- When you use smtp.gmail.com:587, you must use STARTTLS (Google calls it TLS, but it is really STARTTLS).
- Use the full email address as the account name (including the gmail.com portion).

#5 - 10/20/2017 09:36 AM - Greg Shah

I used the internal Golden Code SMTP gateway to test STARTTLS with a self-signed certificate. This case requires the email session to be defined with the property mail.smtp.ssl.trust set to the hostname used for connection. That hostname can be localhost or a FQDN.

#6 - 10/20/2017 10:01 AM - Greg Shah

Useful links that explain the different connection security methods for SMTP:

https://www.fastmail.com/help/technical/ssltlsstarttls.html
https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
https://en.wikipedia.org/wiki/SMTP_Authentication

There are these approaches:

- unencrypted (userid, password and all contents are passed in plain text)
- STARTTLS (communication starts on an unencrypted socket and then is upgraded to TLS)
- SSL/TLS (encrypted socket from the beginning)

STARTTLS can be used on port 25 or 587. 587 is expected to be used for STARTTLS, even if some email clients call it TLS. There are 2 ways to use it: REQUIRED mode (fails if the server doesn't support it) and USED IF IT IS SUPPORTED (will use STARTTLS if the SMTP server supports it but will use insecure mode if the server doesn't support it).

SSL is normally used on port 465. SSL and TLS are effectively the same thing, but some email clients will confuse matters by calling port 587 TLS when it is really STARTTLS.

All three connection types can be used on any port, but they will default to 25 (unencrypted mode and STARTTLS) or 465 (SSL).

The authentication method is orthogonal to the connection security. The most common approaches:

- none (no authentication is used)
- userid/password login

The login processing is a bit weird in that either the userid or the password can be missing, it is not clear to what degree these scenarios work. It will be dependent on the SMTP server implementation, most likely.

#7 - 10/25/2017 12:20 PM - Greg Shah

3353a revision 11192 provides a good first version of the utility code. This allows sending text, html and mixed text and html emails, attachments, embedded images (in the HTML using content-ids) and all the common connection security approaches of SSL, STARTTLS and unencrypted. It is designed to have all the email details defined up front (in an EmailDefinition class and then there is the Emailer.send() method that can be used to send the defined email. Errors will only occur during send since until that time the state is just being stored in the definition (a container class). This design can be run from the server or easily remoted to the client.

#8 - 10/30/2017 04:51 PM - Greg Shah

Revision 11201 has a complete implemention that has been tested and cleaned up. It supports HTML and TEXT, attachments (by file and URL), embedded images in the HTML (by file and URL), TO, CC, BCC, FROM, REPLY-TO and so forth. It has been tested with multiple 4GL programs which have been tested in all connectivity modes. The javadoc is done, though a wiki page will still need to be written.

#9 - 10/30/2017 04:53 PM - Greg Shah

Branch 3353a rebased to trunk revision 11186. 11209 is now the latest revision.

This version is going through conversion regression testing right now. I've already done testing of the functionality and some basic 4GL execution at runtime. ChUI runtime application testing is not needed, since there is no runtime functionality that is affected there that I haven't already tested elsewhere.

#10 - 10/30/2017 09:40 PM - Greg Shah

Conversion and manual testing has passed.

Task branch 3353a was merged to trunk as revision 11187. Branch 3353a has been archived.

#11 - 11/02/2017 05:58 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

The documentation for this functionality can be found in Email Send.

Files

fakesmtp_gui.png	18.2 KB	10/20/2017	Greg Shah