QDL Mail

Introduction

This module allows you to send *simple* mail from inside QDL. This is not intended to be a tool for making mail programs. It supports messages with different types (so you can sent an HTML message, e.g.) but does not support attachments. It is not a mail server and you do need a valid one with an account. This supports templates as well, so you simply create a basic email with variables in it that are replaced, allowing customization. You can load it using the standard jload call, *viz*,

```
   jload('mail')
mail
   send(message.)
```

Setting up email

Thanks to the fact that the standard Java libraries are used everywhere, the assumption is that the jar for the mail classes must be provided at runtime – due to licensing it cannot be part of the QDL distribution. The current version is 1.6.7. To enable QDL mail, you need to

1. Download the jar:
   https://repo1.maven.org/maven2/com/sun/mail/jakarta.mail/1.6.7/jakarta.mail-1.6.7.jar
2. Put it some place useful, such as $QDL_HOME/lib
3. Add the following to your invocation of QDL
   java **-cp $QDL_HOME/lib/jakarta.mail-1.6.7.jar** … *rest of invocation*

When QDL starts, you should be able to just use mail.

The configuration

In order to use QDL Mail, you need to set a configuration. This is a stem (well, what else?) and has the following elements

Key	Type	Description
bcc	stem	addresses for blind carbon copy
cc	stem	address for carbon copy
content_type	string	mime content type of the message. Default is plain text
debug	boolean	low level debug enable. Use with discretion.
from	string	the address of the sender.
host	string	the address of the email server
password	string	the password for your mail server account
port	int	the port used by the mail server
start_tls	boolean	Use TLS (rather than SSL)
reply_to	string	The reply-to address if it is different from the *from* address
to	stem	address of recipients

use_ssl	boolean	Use SSL (rather than TLS)
usernmae	string	the username to use one the server.

Notes.

- Again, you ***must*** have an valid account on some email server. QDL Mail lets you log in and send an email from your account. Do be sure your email provider is happy with this since many times sending high volumes of mail are flagged as spam and the account is suspended.
- Messages are sent in the background, so it is possible that there may be failures. Consult the logs if there is a problem. If we sent it in the foreground, the workspace might hang for quite some time.
- **to**, **cc** and **bcc** stems may be general stems. All the string elements will be used as addresses.
- If both **use_ssl** and **start_tls** are true, TLS will be used.
- If you do not supply a port, the default for the protocol will be used.

Example

```
cfg. :={
          'to' : ['bob@bigstate.edu','all-staff@bigstate.edu']
        'from' : 'support@bigstate.edu',
        'host' : 'smtp.bigstate.edu',
     'use_ssl' : true
   'start_tls' : true,
    'password' : 'mairzy doats'
};
```

Mail messages

You may either send a string or a list of strings. Note that you really do have to use a list, since there is no canonical ordering of general keys and the lines of your message could well be in random order. In both cases, the very first line is pulled off and used as the subject. The next two messages would be treated as identical

```
message. :=  ['Test message subject',
              'This is a test message',
              'This is the second line'];

message := 'Test message subject\nThis is a test message\nThis is the second line';
```

The message would have a subject of `Test message subject` and the body would be

This is a test message
This is the second line

Messages and templates

A very common use pattern is to have a basic message that includes variables which will be replaced at runtime. We use *templates* in which the variables are enclosed in ${}. You supply a stem of variables names (as keys) and their values. Let's say you have two files called subject.txt:

```
System monitor error notification for ${host}
```

and body.txt:

```
The ${system} encountered a problem attempting to contact the server at

${host}

The message from the error reads: '${message}'

Please contact ${admin_address} for more information.
```

In this case, you might have a template stem that looks like

```
template. :={'host' : 'http://math.bigstate.edu/fts',
             'system' : 'file transfer',
             'message' : 'host unresolvable',
             'admin_address' : 'support@math.bigstate.edu'}
```

So to send the message, you can read the file as, say, stems:

```
   send(file_read('subject.txt',1)~file_read('body.txt',1), template.)
```

The resulting message would be

```
System monitor error notification for  http://math.bigstate.edu/fts
The file transfer system encountered a problem attempting to contact the server at
http://math.bigstate.edu/fts
The message from the error reads:'host unresolvable'
Please contact  support@math.bigstate.edu for more information
```

Function reference

cfg

Description

This call allows you to query the configuration or set it.

Usage

cfg()

cfg(configuration.)

Arguments

none - returns the current configuration.

configuration. - the new configuration.

Output

If niladic, the current configuration. If monadic, it returns the previous configuration.

Examples

Se the section above for the actual structure of this stem.

send

Description

Send an email message from an account.

Usage

`send(message | message.)` - send a message.

`send(message | message., template.)` - send a message using the template. for substitution.

Arguments

message is a string. message. is a list of strings. The first line of either is used as the subject.

Output

This returns **true** stating that the background thread was started.

Examples

See above.