repoze.sendmail Documentation

Release 4.1

Agendaless Consulting, Inc.

Apr 21, 2017

Contents

Contents:

repoze.sendmail

repoze.sendmail allows coupling the sending of email messages with a transaction, using the Zope transaction manager. This allows messages to only be sent out when and if a transaction is committed, preventing users from receiving notifications about events which may not have completed successfully. Messages may be sent directly or stored in a queue for later sending. The queued mail approach is the more common and recommended path. For convenience, the package includes a console application which can flush the queue, sending the messages that it finds.

repoze.sendmail is a fork of zope.sendmail. Functionality that was specific to running in a Zope context has been removed, making this version more generally useful to users of other frameworks.

Note that repoze.sendmail works only under Python 2.6+ and Python 3.2+.

Delivering Mail Messages from Application Code

Messages are sent by means of a *Delivery* object. repoze.sendmail include Two delivery implementations:

- repoze.sendmail.delivery.QueuedMailDelivery
- repoze.sendmail.delivery.DirectMailDelivery

A delivery implements the interface defined by repoze.sendmail.interfaces.IDelivery. That interface defines a single *send* method:

```python
def send(fromaddr, toaddrs, message):
    """ Sends message on transaction commit. """
```

- *fromaddr* is the address of the sender of the message.
- *toaddrs* is a list of email addresses for recipients of the message.
- *message* must be an instance of email.message.Message and is the actual message which will be sent.

Delivery via a Mail Queue

To create a queued delivery:

```python
from email.message import Message
from repoze.sendmail.delivery import QueuedMailDelivery

message = Message()
message['From'] = 'Chris <chris@example.com>'
message['To'] = 'Paul <paul@example.com>, Tres <tres@example.com>'
message['Subject'] = "repoze.sendmail is a useful package"
message.set_payload("The subject line says it all.")

delivery = QueuedMailDelivery('path/to/queue')
delivery.send('chris@example.com', ['paul@example.com', 'tres@example.com'],
              message)
```

The message will be added to the maildir queue in 'path/to/queue' when and if the current transaction is committed successsfully.

`repoze.sendmail` includes a console app utility for sending queued messages:

```
$ bin/qp path/to/queue
```

This will attempt to use an SMTP server at localhost to send any messages found in the queue. To see all options available:

```
$ bin/qp --help
```

The QueueProcessor used by the console utility can also be called from Python:

```python
qp = QueueProcessor(mailer, queue_path, ignore_transient=True)
qp.send_messages()
```

The *ignore_transient* parameter, when True, will cause the queue processor to ignore transient errors (any error code not between 500 and 599). This is useful when monitoring systems are used, to prevent filling the error reports with temporary errors.

Direct SMTP Delivery

Direct delivery (using the SMTP protocol) can also be used:

```python
from repoze.sendmail.delivery import DirectMailDelivery
from repoze.sendmail.mailer import SMTPMailer

mailer = SMTPMailer()  # Uses localhost, port 25 be default.
delivery = DirectMailDelivery(mailer)
delivery.send('chris@example.com', ['paul@example.com', 'tres@example.com'],
              message)
```

Delivery via the `sendmail` Command

If you are on a Unix/BSD machine and prefer to use the standard unix *sendmail* interface (which is likely provided by exim, postfix or qmail) via a binary at '/usr/sbin/sendmail' you can simply opt to use the following classes :

```
mailer = SendmailMailer()
delivery = DirectMailDelivery(mailer)
```

you may also customize this delivery with the location of another binary:

```
mailer = SendmailMailer(sendmail_app='/usr/local/bin/sendmail')
```

Transaction Integration

`repoze.sendmail` hooks into the Zope transaction manager and only sends messages on transaction commit. If you are using a framework which does not use transactions by default, you will need to begin and commit a transaction of your own in order for mail to be sent:

```python
import transaction
transaction.manager.begin()
try:
    my_code_here()
    transaction.manager.commit()
except e:
    transaction.manager.abort()
    raise e
```

Change history

4.5 (unreleased)

- TBD

4.4.1 (2017-04-21)

- Moved documentation to RTD.

4.4 (2017-04-21)

- Drop support for Python 3.3.

- Add support for Python 3.6.

- Fix parsing of `debug_smtp` from queue processor config file: it must be a boolean, rather than a string, when passed to the stdlib. (issue #40).

4.3 (2016-12-08)

- Drop support for Python 2.6 and 3.2.

- Add support for Python 3.4 and 3.5.

- Add `ignore_transient` parameter to `QueueProcessor`, to prevent raising temporary errors in some situations. (PR #37)

- Reset 'tpc_phase' to zero during 'tpc_abort' / 'tpc_finish'. (issue #30)

4.2 (2014-02-17)

- Add "savepoint" support to transactional mail integration. (PR #24/28)
- Mail Delivery utilities can now be passed a transaction manager (falling back to the ''transaction.get()'' default), making it easier to override. (PR #27)

4.1 (2013-06-26)

- Replace 'utf_8' encoding name with preferred spelling ('utf-8').
- Replace 'latin_1' encoding name with preferred spelling ('iso-8859-1')
- Include the time of the error when logging errors from the queue processor.
- response.MIMEPart now correctly sets the charset of the email payload if it's one of the content_type parameters of the Message or Attachment.
- The SMTPMailer now accepts an "ssl" argument, which, if passed, will cause the SMTP factory to return an SMTP_SSL connection instead of a plain old SMTP connection.
- The SMTPMailer now uses a 10-second timeout by default, used when an SMTP connection is made but the server does not respond in enough time.

4.0 (2013-04-23)

- Add support for bulding docs and testing doctest snippets under `tox`.
- Add `setup.py docs` alias (installs Sphinx).
- Converted docs to Sphinx.

4.0b2 (2013-03-28)

- Issue #13: fixed handling of headers with with multibyte unicode characters at the point where the header is split into multiple lines.
- Pull #15 - Extended repoze.sendmail with configurable *usr/sbin/sendmail* binary support

4.0b1 (2013-01-09)

- Dropped support for Jython until a Jython-2.7-compatible version of `zope.interface` becomes available.
- Dropped support for Python 2.5.
- Added suupport for Python 3.3.
- Improved test for SSL feature under Python 3.x.
- Added new tests for proper encoding of binary attachments.
- Cauterized resource leak warnings under Python 3.2.

3.2 (2012-05-03)

- Issue #7: fixed handling of to/from addresses with non-ascii characters when using queued mail delivery.

- Suppressed duplicate usage message output from `qp`.

3.1 (2012-03-26)

- Fixed `qp` queue processor mail delivery under Python 3.0.

- Added 'setup.py dev' alias (runs `setup.py develop` plus installs `nose` and `coverage`).

3.0 (2012-03-20)

- Fixed *Message-Id* handling (see http://bugs.repoze.org/issue177).

- Provided improved support for encoding messages to bytes. It should now be possible to represent your messages in *email.message.Message* objects just with unicode (excepting bytes for binary attachments) and the mailer will handler it as appropriate.

- Added tests for cPython 2.5, 2.6, 2.7, 3.2, jython 2.5 and pypy 1.8 compatibility.

2.3 (2011-05-17)

- Queued delivery now creates a copy of the passed in messsage before adding the 'X-Actually-{To,From}' headers. This avoids rudely mutating the message being sent in ways that might not be expected by the sender. (LP #780000)

2.2 (2010-09-14)

- Made debug output for SMTP mailer optional. (Thanks to Igor Stroh for patch.)

2.1 (2010-07-28)

- Silently ignore redundant calls to abort transaction. (LP #580164)

2.0 (2010-03-10)

Represents major refactoring with a number of backwards incompatible changes. The focus of the changes is on simplifying and updating the internals, removing usage of deprecated APIs, removing unused functionality and using the *email* module from the standard library wherever possible. A few changes have been made solely to reduce internal complexity.

- Public facing APIs no longer accept messages passed as strings. Messages must be instances of email.message.Message.

- Deprecated APIs have been replaced with newer 'email' module throughout.

- Functions that return message ids no longer strip containing less than and greater than characters.

- Events were removed entirely. There was nothing in the code actually performing a notify anyway. Removes dependency on zope.event.

- Normalized directory structure. (Got rid of 'src' directory.)

- Got rid of functions to send queued mail from a thread or a daemon process. These are more appropriately handled in the calling code.

- Removed vocabulary. It was a fossil from its days as zope.sendmail and was not used by anything.

- Got rid of the zcml directives. These were written in such a way that you would end up putting deployment configuration in your zcml, which is a fundamentally broken pattern. Users of the ZCA may still register utilities aginst the IMailDelivery and IMailer interfaces. This is the recommended way to use repoze.sendmail with the Zope Component Architecture.

- Removed all interfaces that did not correspond to a rational plug point. This leaves only IMailDelivery and IMailer.

- Removed dependency on zope.i18nmessageid

- No longer works under Python 2.4 (Python 2.5 required).

1.2 (2010-02-11)

- Maildir storage for queue can now handle unicode passed in for message or to/from addresses.

1.1 (2009-02-24)

- Added logging to queue processor console app.

- Added ini config parsing to queue processor console app.

1.0 (2009-02-24)

- Initial release

- Copy of zope.sendmail with dependency on security removed.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search