NAME

Mail::Box−Overview − objects used by Mail::Box

DESCRIPTION

Introduction

The MailBox package is a suite of classes for accessing and managing email folders in a folder-independent manner. This package is an alternative to the `Mail::Folder` and `MIME::*` packages. It abstracts the details of messages, message storage, and message threads, while providing better performance than older mail packages. It is meant to provide an object-oriented toolset for all kinds of e−mail applications, under which Mail User-Agents (MUA) and mail filtering programs.

This package is modular −−parts of it can be used independently of the rest. For example, the Mail::Box::Manager can automatically determine that a folder is in Mbox format and return an object of the Mail::Box::Mbox class, or the user program can bypass the manager and create Mail::Box::Mbox objects directly. Similarly, if the user program is only manipulating a single message, a Mail::Message.

The Mail::Box package has special features to help MUA's access folder data quickly in random order. You will not really benefit (neither slower) if you need the full folder sequentially.

You may want to have a look at the sample scripts in the `scripts` directory.

Distributions

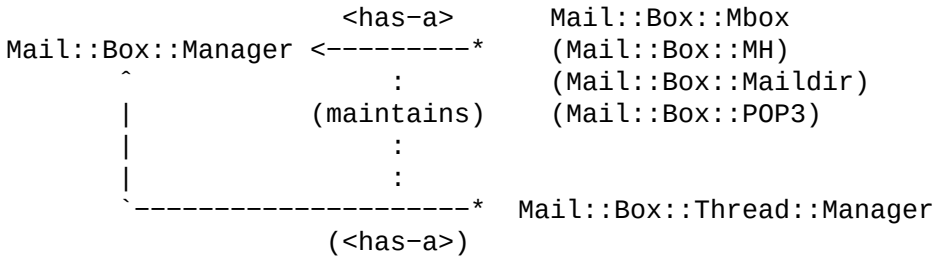
Up to MailBox v2, all `Mail::*` modules were released as a single distribution. From v3, there are a few separate distributions in an attempt to reduce the dependencies:

- Mail::Message
- Mail::Transfer
- Mail::Box
- Mail::Box::IMAP4
- Mail::Box::POP3
- Mail::Box::Parser::C

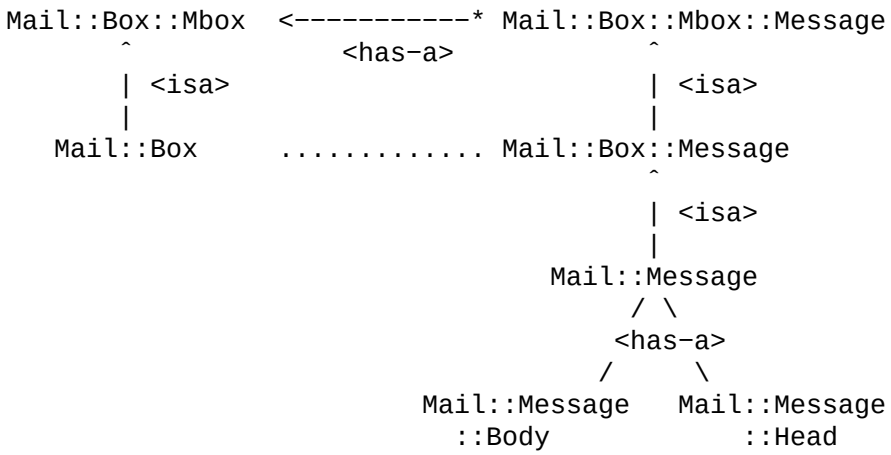
The names of the classes are not always ideal: the 'Mail' namespace on CPAN is quite full.

The class relations

Mail::Box::Manager objects play a central role in any program which is built with MailBox. Each program will create one manager, and then open folders via that manager. Besides folders, the manager can also be used to discover message threads: sequences of messages with their follow-ups.

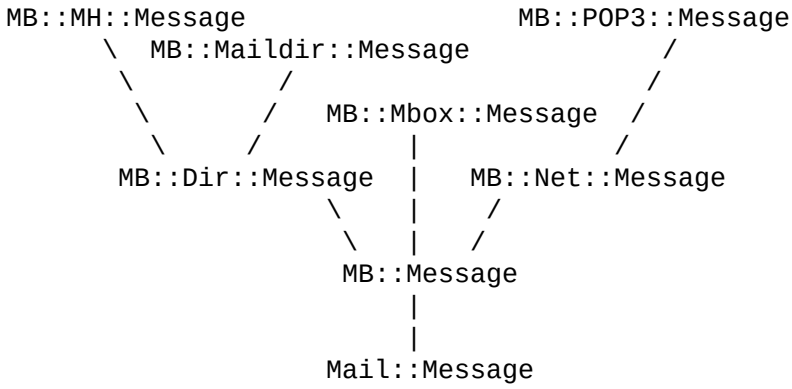


Each folder maintains a list of messages. Much effort is made to hide differences between folder types and kinds of messages. Your program can be used for MBOX, MH, Maildir, and POP3 folders with no change at all (as long as you stick to the rules).



The situation for MH and Maildir folders is a little more complicated, because they have an extra intermediate level of abstraction: Mail::Box::Dir. The POP3 folder has an intermediate Mail::Box::Net.

In the future, when more Mbox-like folder types get implemented, there may be a Mail::Box::File level too. The following is also true for the mail boxes



The Manager

The mailbox manager Mail::Box::Manager encapsulates folder management issues. It maintains a set of open mail folders (mailboxes), and provides methods for opening and closing them, efficiently moving messages between folders, and efficiently appending messages to folders. It contains Mail::Box objects which may be of different types. Most folder types can be detected automatically.

The main manager also manages message-thread detector objects, and informs them when the contents of a folder have changed. This manager class is the only one you instantiate yourself: objects of all other classes will be provided by your folder manager.

You are strongly advised to use this object, but you can often do without it and open a specific folder-type directly.

The Messages

Mail::Message

A base class that defines an interface for manipulating the head and body of a message. There are various header object types (Mail::Message::Head's) and a bunch of body object types (Mail::Message::Body's).

The Mail::Message::Construct package is loaded when more complex tasks have to be performed on messages, like creating replies, bounces, or a forward message. These functionalities are described and implemented in the ::Construct file, but are automatically added to the Mail::Message namespace when used.

Message types which are foreign to MailBox can be used in the MailBox environment: there are some converters implemented via Mail::Message::Convert. Particularly the popular Mail::Internet and

MIME::Entity are supported.

Mail::Box::Message
: An abstract base class which defines an interface for mail messages which are stored in any folder. It inherits from Mail::Message, and adds the basic idea of *location* to a message.

Mail::Message::Body
: This is the base class for all message bodies. It describes what you can do with any kind of body. The body types differ on the way how the keep the body content during the run of your program.

  One special case of the body types is the Mail::Message::Body::Multipart, which contains a set of Mail::Message::Part objects. These are just like normal messages, except that they are contained in an other message. The Mail::Message::Body::Nested body type is comparible, but contains only one message: they are used for `message/rfc822` message encodings.

  When needed, the functionality of the body objects is extended with Mail::Message::Body::Construct and Mail::Message::Body::Encode. The former package implements things like concatenation, the later controls message encoding and decoding. In the current implementation this is limited to transfer encodings (implemented in the Mail::Message::TransferEnc packages). Automatic character and mime recodings are on the wish-list.

Mail::Message::Head
: The header for a single message. Maintains a set of Mail::Message::Field objects, each containing one header line. Fields are the only objects which have no logging and tracing facilities, purely for reasons of performance.

  The header object has three sub-classes: the Mail::Message::Head::Complete version knows all lines for sure, Mail::Message::Head::Subset maintains an unknown subset of lines, and the Mail::Message::Head::Delayed has no lines yet but knows where to get them.

  The latter two will automatically get the missing header lines from the mailbox files when needed, and so transform into a `::Complete` header. It is fully transparent to the user of MailBox in which shape the header really is on the moment.

The Folder types

Mail::Box
: A base class that defines a standard interface for mail boxes which is independent of mailbox type. Objects of this class contain a Mail::Box::Locker and a list of Mail::Box::Message objects.

Mail::Box::Dir
: The base class for all folders which use a directory organization: each message is a separate entity (file) grouped in a directory. Each Mail::Box::Dir::Message represents one message, one such entity.

Mail::Box::Net
: The base class for all folders which have the messages outside direct reach of the MailBox library, for instance on a remote system, or in a database.

Mail::Box::Mbox
: This class derives from Mail::Box, and implements its interface for mbox-style folders. It maintains a set of Mail::Box::Mbox::Message objects, which are derived from a Mail::Box::Message.

  Mbox-style folders have one file containing multiple messages per folder. When folders get large, access tends to get slow.

Mail::Box::MH
: This class derives from Mail::Box::Dir, and implements its interface for MH-style folders. It maintains a set of Mail::Box::MH::Message objects, which are derived from a Mail::Box::Dir::Message.

  MH-style folders are represented by a directory, where each message is stored in a separate file. The message files are sequentially numbered. It is fast to open one single message, but hard to get an overview.

Mail::Box::MH::Index
: The base class for MH mailbox indexes which provides methods for reading, writing, and managing message indexes. These indexes are used to speed-up access to directory based folders.

Mail::Box::MH::Labels
: Also for efficiency reasons, a separate file is maintained which contains flags about the messages. This file for instance lists new files. This way, the MH message files do not have to be opened to find that out.

Mail::Box::Maildir
: Like the MH folder type, this class derives from Mail::Box::Dir. It implements its interface for Maildir-style folders. It maintains a set of Mail::Box::Maildir::Message objects, which are derived from a Mail::Box::Dir::Message.

Mail::Box::POP3
: Implements the POP3 protocol based on Mail::Box::Net. The Mail::Transport::POP3 implementation handles the protocol details. In this kind of folders, you can only read and delete messages.

Various Other Classes

Mail::Box::Thread::Manager
: Maintains a set of message-threads over one or more folders. A message-thread is a start message with all the replies on it. And the replies on replies, and so on. This object is used to construct the thread for a set of open folders.

  This object maintains linked lists of Mail::Box::Thread::Node objects. Mail::Message::Dummy's fill-up some holes.

Mail::Box::Locker
: Provides a folder locking interface which is inherited by the Mail::Box class. Currently it supports dot-file locking (`filename.lock`), flock filehandle locking, and locking over NFS. Each is implemented in a separate class. A multi-locker, using a set of lock-methods at the same time is also available.

Mail::Box::Search
: The set of search packages implement various search techniques in an uniformal way. Although implementing your own search algorithm is simple in general, in practice multiparts, encodings, and mime-types complicate things.

Mail::Box::Parser
: The parser reads messages, and transforms them into data-structures such that the content of header and body can be used within the program. The first parser is implemented in pure Perl. A second parser is under development, and will written in C, to gain speed.

Mail::Box::Tie
: Provides hash (Mail::Box::Tie::HASH) or array tied (Mail::Box::Tie::ARRAY) access to any mail folder derived from Mail::Box. This beautifies your code in some applications.

Mail::Transport
: Various ways of sending and receiving messages are implemented. Sending is possible via external programs, like `mail`, `Mailx`, `sendmail`, or autonomously with direct SMTP. Receiving is currently only implemented via POP3.

Mail::Reporter
: A debugging and logging class which is inherited by most of the Mail:: modules. For each object, you can say what log and error reports must be kept or directly presented to the user. This way you can decide to have Mail::Box report about problems, or do it all yourself.

All classes are written to be extensible.

SEE ALSO

This module is part of Mail-Box distribution version 3.009, built on August 18, 2020. Website: *http://perl.overmeer.net/CPAN/*

LICENSE