NAME

Mail::Address – parse mail addresses

SYNOPSIS

```
 use Mail::Address;
 my @addrs = Mail::Address->parse($line);

 foreach $addr (@addrs) {
     print $addr->format,"\n";
 }
```

DESCRIPTION

`Mail::Address` extracts and manipulates email addresses from a message header. It cannot be used to extract addresses from some random text. You can use this module to create RFC822 compliant fields.

Although `Mail::Address` is a very popular subject for books, and is used in many applications, it does a very poor job on the more complex message fields. It does only handle simple address formats (which covers about 95% of what can be found). Problems are with

- no support for address groups, even not with the semi-colon as separator between addresses;
- limited support for escapes in phrases and comments. There are cases where it can get wrong; and
- you have to take care of most escaping when you create an address yourself: `Mail::Address` does not do that for you.

Often requests are made to the maintainers of this code improve this situation, but this is not a good idea, where it will break zillions of existing applications. If you wish for a fully RFC2822 compliant implementation you may take a look at Mail::Message::Field::Full, part of MailBox.

. Example

```
  my $s = Mail::Message::Field::Full->new($from_header);
  # ref $s isa Mail::Message::Field::Addresses;

  my @g = $s->groups;           # all groups, at least one
  # ref $g[0] isa Mail::Message::Field::AddrGroup;
  my $ga = $g[0]->addresses;    # group addresses

  my @a = $s->addresses;        # all addresses
  # ref $a[0] isa Mail::Message::Field::Address;
```

METHODS

Constructors

Mail::Address–>**new**(`$phrase`, `$address`, [`$comment`])

Create a new `Mail::Address` object which represents an address with the elements given. In a message these 3 elements would be seen like:

```
 PHRASE <ADDRESS> (COMMENT)
 ADDRESS (COMMENT)
```

example:

```
 Mail::Address->new("Perl5 Porters", "perl5-porters@africa.nicoh.com");
```

`$obj`–>**parse**($line)

Parse the given line a return a list of extracted `Mail::Address` objects. The line would normally be one taken from a To,Cc or Bcc line in a message

example:

```
 my @addr = Mail::Address->parse($line);
```

Accessors

`$obj`−>**address**()
: Return the address part of the object.

`$obj`−>**comment**()
: Return the comment part of the object

`$obj`−>**format**(@addresses)
: Return a string representing the address in a suitable form to be placed on a `To`, `Cc`, or `Bcc` line of a message. This method is called on the first address to be used; other specified addresses will be appended, separated by commas.

`$obj`−>**phrase**()
: Return the phrase part of the object.

Smart accessors

`$obj`−>**host**()
: Return the address excluding the user id and '@'

`$obj`−>**name**()
: Using the information contained within the object attempt to identify what the person or groups name is.

  Note: This function tries to be smart with the "phrase" of the email address, which is probably a very bad idea. Consider to use **phrase()** itself.

`$obj`−>**user**()
: Return the address excluding the '@' and the mail domain

SEE ALSO

This module is part of the MailTools distribution, *http://perl.overmeer.net/mailtools/*.

AUTHORS

The MailTools bundle was developed by Graham Barr. Later, Mark Overmeer took over maintenance without commitment to further development.

Mail::Cap by Gisle Aas <aas@oslonett.no>. Mail::Field::AddrList by Peter Orbaek <poe@cit.dk>. Mail::Mailer and Mail::Send by Tim Bunce <Tim.Bunce@ig.co.uk>. For other contributors see ChangeLog.

LICENSE