NAME

Mail::Box::Manage::User − manage the folders of a user

INHERITANCE

```
Mail::Box::Manage::User
  is a Mail::Box::Manager
  is a Mail::Reporter

Mail::Box::Manage::User is extended by
  Mail::Server::IMAP4::User
```

SYNOPSIS

```
use Mail::Box::Manage::User;
use User::Identity;

my $id      = User::Identity->new(...);
my $user    = Mail::Box::Manage::User->new
  ( identity  => $id
  , folderdir => "$ENV{HOME}/Mail"
  , inbox     => $ENV{MAIL}
  );

my $inbox   = $user->open($user->inbox);
my $top     = $user->topfolder;
```

DESCRIPTION

Where the Mail::Box::Manager takes care of some set of open folder, this extension will add knowledge about some related person. At the same time, it will try to cache some information about that person's folder files.

Extends ''DESCRIPTION'' in Mail::Box::Manager.

METHODS

Extends ''METHODS'' in Mail::Box::Manager.

Constructors

Extends ''Constructors'' in Mail::Box::Manager.

Mail::Box::Manage::User->**new**($args)

Use new(default_folder_type) to explicitly state which kind of folders you use.

-Option	--Defined in	--Default
autodetect	Mail::Box::Manager	undef
collection_type		Mail::Box::Collection
default_folder_type	Mail::Box::Manager	'mbox'
delimiter		"/"
folder_id_type		Mail::Box::Identity
folder_types	Mail::Box::Manager	<all standard types>
folderdir	Mail::Box::Manager	['.']
folderdirs	Mail::Box::Manager	<synonym for C<folderdir>>
identity		<required>
inbox		undef
log	Mail::Reporter	'WARNINGS'
topfolder_name		'='
trace	Mail::Reporter	'WARNINGS'

autodetect => TYPE|ARRAY−OF−TYPES

collection_type => CLASS

Subfolders grouped together.

default_folder_type => NAME|CLASS
delimiter => STRING
The separator used in folder names. This doesn't need to be the same as your directory system is using.

folder_id_type => CLASS|OBJECT
folder_types => NEW-TYPE | ARRAY-OF-NEW-TYPES
folderdir => DIRECTORY
folderdirs => [DIRECTORIES]
identity => OBJECT
The main difference between the Mail::Box::Manager and this class, is the concept of some person (or virtual person) who's files are being administered by this object. The OBJECT is an User::Identity.

The smallest identity that will do: `my $id = User::Identity->new('myname')`

inbox => NAME
The name of the user's inbox.

log => LEVEL
topfolder_name => STRING
trace => LEVEL

Attributes

Extends ''Attributes'' in Mail::Box::Manager.

`$obj->`**defaultFolderType**()
Inherited, see ''Attributes'' in Mail::Box::Manager

`$obj->`**folderTypes**()
Inherited, see ''Attributes'' in Mail::Box::Manager

`$obj->`**folderdir**()
Inherited, see ''Attributes'' in Mail::Box::Manager

`$obj->`**identity**()
Returns a User::Identity object.

`$obj->`**inbox**([$name])
(Set and) get the `$name` of the mailbox which is considered the folder for incoming mail. In many protocols, this folder is handled separately. For instance in IMAP this is the only case-insensitive folder name.

`$obj->`**registerType**($type, `$class`, `%options`)
Inherited, see ''Attributes'' in Mail::Box::Manager

Manage open folders

Extends ''Manage open folders'' in Mail::Box::Manager.

`$obj->`**close**($folder, `%options`)
Inherited, see ''Manage open folders'' in Mail::Box::Manager

`$obj->`**closeAllFolders**(, `%options`)
Inherited, see ''Manage open folders'' in Mail::Box::Manager

`$obj->`**isOpenFolder**($folder)
Inherited, see ''Manage open folders'' in Mail::Box::Manager

`$obj->`**open**([$foldername], `%options`)
Inherited, see ''Manage open folders'' in Mail::Box::Manager

`$obj->`**openFolders**()
Inherited, see ''Manage open folders'' in Mail::Box::Manager

Manage existing folders

Extends ‘‘Manage existing folders’’ in Mail::Box::Manager.

Manage folders

`$obj->`**create**($name, `%options`)

Creates a new folder with the specified name. An folder's administrative structure (Mail::Box::Identity) is returned, but the folder is not opened.

In the accidental case that the folder already exists, a warning will be issued, and an empty list/undef returned.

The `%options` are passed to **Mail::Box::create()** of your default folder type, except for the options intended for this method itself.

```
-Option       --Default
 create_real    <true>
 create_supers  <false>
 deleted        <false>
 id_options     []
```

create_real => BOOLEAN

When this option is false, the pysical folder will not be created, but only the administration is updated.

create_supers => BOOLEAN

When you create a folder where upper hierarchy level are missing, they will be created as well.

deleted => BOOLEAN

The folder starts as deleted.

id_options => ARRAY

Values passed to the instantiated Mail::Box::Identity. That object is very picky about the initiation values it accepts.

`$obj->`**delete**($name)

Remove all signs from the folder on the file-system. Messages still in the folder will be removed. This method returns a true value when the folder has been removed or not found, so ‘‘false’’ means failure.

It is also possible to delete a folder using `$folder->delete`, which will call this method here. OPTIONS, which are used for some other folder types, will be ignored here: the user's index contains the required details.

```
-Option   --Defined in        --Default
 recursive  Mail::Box::Manager  <folder's default>
```

recursive => BOOLEAN

example: how to delete a folder

```
print "no xyz (anymore)\n" if $user->delete('xyz');
```

`$obj->`**folder**($name)

Returns the folder description, a Mail::Box::Identity.

`$obj->`**folderCollection**($name)

Returns a pair: the folder collection (Mail::Box::Collection) and the base name of `$name`.

`$obj->`**rename**($oldname, `$newname`, `%options`)

Rename the folder with name `$oldname` to `$newname`. Both names are full pathnames.

```
-Option       --Default
 create_supers  <false>
```

create_supers => BOOLEAN
When you rename a folder to a place where upper hierarchy levels are missing, they will get be defined, but with the deleted flag set.

`$obj->`**topfolder**()
Returns the top folder of the user's mailbox storage.

Move messages to folders
Extends ''Move messages to folders'' in Mail::Box::Manager.

`$obj->`**appendMessage**([$folder|$foldername], `$messages`, `%options`)
Inherited, see ''Move messages to folders'' in Mail::Box::Manager

`$obj->`**copyMessage**([$folder|$foldername], `$messages`, `%options`)
Inherited, see ''Move messages to folders'' in Mail::Box::Manager

`$obj->`**moveMessage**([$folder|$foldername], `$messages`, `%options`)
Inherited, see ''Move messages to folders'' in Mail::Box::Manager

Manage message threads
Extends ''Manage message threads'' in Mail::Box::Manager.

`$obj->`**threads**([$folders], `%options`)
Inherited, see ''Manage message threads'' in Mail::Box::Manager

Internals
Extends ''Internals'' in Mail::Box::Manager.

`$obj->`**decodeFolderURL**($url)
Inherited, see ''Internals'' in Mail::Box::Manager

`$obj->`**toBeThreaded**($folder, `$messages`)
Inherited, see ''Internals'' in Mail::Box::Manager

`$obj->`**toBeUnthreaded**($folder, `$messages`)
Inherited, see ''Internals'' in Mail::Box::Manager

Error handling
Extends ''Error handling'' in Mail::Box::Manager.

`$obj->`**AUTOLOAD**()
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**addReport**($object)
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Mail::Box::Manage::User->**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**errors**()
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**log**([$level, [$strings]])
Mail::Box::Manage::User->**log**([$level, [$strings]])
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**logPriority**($level)
Mail::Box::Manage::User->**logPriority**($level)
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**logSettings**()
Inherited, see ''Error handling'' in Mail::Reporter

`$obj->`**notImplemented**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj->`**report**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj->`**reportAll**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj->`**trace**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj->`**warnings**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

Cleanup

Extends ‘‘Cleanup’’ in Mail::Box::Manager.

`$obj->`**DESTROY**()
Inherited, see ‘‘Cleanup’’ in Mail::Reporter

DETAILS

Extends ‘‘DETAILS’’ in Mail::Box::Manager.

DIAGNOSTICS

Error: Cannot create `$name`: higher levels missing
Unless you set create(create_supers), all higher level folders must exist before this new one can be created.

Error: Cannot rename `$name` to `$new`: higher levels missing
Unless you set create(create_supers), all higher level folders must exist before this new one can be created.

Error: Folder `$name` is already open.
You cannot ask the manager for a folder which is already open. In some older releases (before MailBox 2.049), this was permitted, but then behaviour changed, because many nasty side-effects are to be expected. For instance, an **Mail::Box::update()** on one folder handle would influence the second, probably unexpectedly.

Error: Folder `$name` is not a Mail::Box; cannot add a message.
The folder where the message should be appended to is an object which is not a folder type which extends Mail::Box. Probably, it is not a folder at all.

Warning: Folder does not exist, failed opening `$type` folder `$name`.
The folder does not exist and creating is not permitted (see open(create)) or did not succeed. When you do not have sufficient access rights to the folder (for instance wrong password for POP3), this warning will be produced as well.

The manager tried to open a folder of the specified type. It may help to explicitly state the type of your folder with the `type` option. There will probably be another warning or error message which is related to this report and provides more details about its cause. You may also have a look at new(autodetect) and new(folder_types).

Warning: Folder type `$type` is unknown, using autodetect.
The specified folder type (see open(type), possibly derived from the folder name when specified as url) is not known to the manager. This may mean that you forgot to require the Mail::Box extension which implements this folder type, but probably it is a typo. Usually, the manager is able to figure-out which type to use by itself.

Error: Illegal folder URL ’$url’.
The folder name was specified as URL, but not according to the syntax. See **decodeFolderURL()** for an description of the syntax.

Error: No foldername specified to open.
: `open()` needs a folder name as first argument (before the list of options), or with the `folder` option within the list. If no name was found, the MAIL environment variable is checked. When even that does not result in a usable folder, then this error is produced. The error may be caused by an accidental odd-length option list.

Error: Package `$package` does not implement `$method`.
: Fatal error: the specific package (or one of its superclasses) does not implement this method where it should. This message means that some other related classes do implement this method however the class at hand does not. Probably you should investigate this and probably inform the author of the package.

Error: Unable to remove folder `$dir`

Error: Use **appendMessage()** to add messages which are not in a folder.
: You do not need to copy this message into the folder, because you do not share the message between folders.

Warning: Use **moveMessage()** or **copyMessage()** to move between open folders.
: The message is already part of a folder, and now it should be appended to a different folder. You need to decide between copy or move, which both will clone the message (not the body, because they are immutable).

Warning: Will never create a folder `$name` without having write access.
: You have set open(create), but only want to read the folder. Create is only useful for folders which have write or append access modes (see Mail::Box::new(access)).

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