NAME

User::Identity::Archive::Plain − simple, plain text archiver

INHERITANCE

```
 User::Identity::Archive::Plain
   is a User::Identity::Archive
   is a User::Identity::Item
```

SYNOPSIS

```
 use User::Identity::Archive::Plain;
 my $friends = User::Identity::Archive::Plain−>new('friends');
 $friends−>from(\*FH);
 $friends−>from('.friends');
```

DESCRIPTION

This archiver, which extends User::Identity::Archive, uses a very simple plain text file to store the information of users. The syntax is described in the DETAILS section, below.

Extends "DESCRIPTION" in User::Identity::Archive.

OVERLOADED

Extends "OVERLOADED" in User::Identity::Archive.

METHODS

Extends "METHODS" in User::Identity::Archive.

Constructors

Extends "Constructors" in User::Identity::Archive.

User::Identity::Archive::Plain−>**new**([$name], `%options`)

```
 −Option        −−Defined in            −−Default
  abbreviations                           []
  description    User::Identity::Item     undef
  from           User::Identity::Archive  undef
  name           User::Identity::Item     <required>
  only                                    []
  parent         User::Identity::Item     undef
  tabstop                                 8
```

abbreviations => HASH|ARRAY
  Adds a set of abbreviations for collections to the syntax of the plain text archiver. See section "Simplified class names" for a list of predefined names.

description => STRING
from => FILEHANDLE|FILENAME
name => STRING
only => ARRAY|ABBREV
  Lists the only information (as (list of) abbreviations) which should be read. Other information is removed before even checking whether it is a valid abbreviation or not.

parent => OBJECT
tabstop => INTEGER
  Sets the default tab-stop width.

Attributes

Extends "Attributes" in User::Identity::Archive.

`$obj`−>**abbreviation**(`$name`, [$class])

Returns the class which is capable of storing information which is grouped as `$name`. With `$class` argument, you add (or overrule) the definitions of an abbreviation. The`$class` is automatically loaded.

If `$class` is `undef`, then the abbreviation is deleted. The class name which is deleted is returned.

`$obj->`**abbreviations**()
: Returns a sorted list of all names which are known as abbreviations.

`$obj->`**defaultTabStop**([$integer])
: Returns the width of a tab, optionally after setting it. This must be the same as set in your editor.

`$obj->`**description**()
: Inherited, see "Attributes" in User::Identity::Item

`$obj->`**name**([$newname])
: Inherited, see "Attributes" in User::Identity::Item

Collections

Extends "Collections" in User::Identity::Archive.

`$obj->`**add**($collection, `$role`)
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**addCollection**(`$object` | <[$type], `%options`>)
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**collection**($name)
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**parent**([$parent])
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**removeCollection**($object|$name)
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**type**()
User::Identity::Archive::Plain->**type**()
: Inherited, see "Collections" in User::Identity::Item

`$obj->`**user**()
: Inherited, see "Collections" in User::Identity::Item

Searching

Extends "Searching" in User::Identity::Archive.

`$obj->`**find**($collection, `$role`)
: Inherited, see "Searching" in User::Identity::Item

Access to the archive

Extends "Access to the archive" in User::Identity::Archive.

`$obj->`**from**(<$fh|$filename|ARRAY>, `%options`)
: Read the plain text information from the specified `$fh`, `$filename`, STRING, or ARRAY of lines.

```
 -Option --Default
  tabstop  <default from object>
  verbose  0
```

tabstop => INTEGER
verbose => INTEGER

DETAILS

The Plain Archiver Format

Simplified class names

It is too much work to specify full class named on each spot where you want to create a new object with data. Therefore, abbreviations are introduced. Use new(abbreviations) or **abbreviations()** to add extra abbreviations or to overrule some predefined.

Predefined names:

user	User::Identity

email	Mail::Identity
location	User::Identity::Location
system	User::Identity::System
list	User::Identity::Collection::Emails

It would have been nicer to refer to a *person* in stead of a *user*, however that would add to the confusion with the name-space.

Indentation says all

The syntax is as simple as possible. An extra indentation on a line means that the variable or class is a collection within the class on the line before.

```
user markov
  location home
     country NL
  email home
     address  mark@overmeer.net
     location home
  email work
     address  solutions@overmeer.bet

email tux
   address tux@fish.net
```

The above defines two items: one User::Identity named `markov`, and an e−mail address `tux`. The user has two collections: one contains a single location, and one stores two e−mail addresses.

To add to the confusion: the `location` is defined as field in `email` and as collection. The difference is easily detected: if there are indented fields following the line it is a collection. Mistakes will in most cases result in an error message.

Long lines

If you want to continue on the next line, because your content is too large, then add a backslash to the end, like this:

```
email home
   description This is my home address,     \
               But I sometimes use this for \
               work as well
   address tux@fish.aq
```

Continuations do not play the game of indentation, so what you also can do is:

```
email home
   description               \
This is my home address,     \
But I sometimes use this for \
work as well
   address tux@fish.aq
```

The fields `comment` and `address` must be correctly indented. The line terminations are lost, which is useful for most fields. However, if you need them, you have to check the description of the applicable field.

Comments

You may add comments and white spaces. Comments start with a `'#'` as first non-blank character on the line. Comments are**not allo wed** on the same line as real data, as some languages (like Perl) permit.

You can insert comments and blank lines on all places where you need them:

```
user markov
```

```
   # my home address
   email home

      # useless comment statement
      address tux@fish.aq
      location #mind_the_hash
```

is equivalent to:

```
user markov
   email home
      address tux@fish.aq
      location #mind_the_hash
```

References

Often you will have the need to add the same information to two items, for instance, multiple people share the same address. In this case, you can create a reference. However, this is only permitted for whole items: you can refer to someone's location, but not to the person's street.

To create a reference to an item of someone else, use

```
user markov
   location home = user(cleo).location(home)
   location work
      organization   MARKOV Solutions
```

Configuration parameters

You can add some configuration lines as well. On the moment, the only one defined is

```
tabstop = 4
```

which can be used to change the meaning of tabs in the file. The default setting is 8, but some people prefer 4 (or other values).

DIAGNOSTICS

Error: `$object` is not a collection.
: The first argument is an object, but not of a class which extends User::Identity::Collection.

Error: Cannot load collection module for `$type` ($class).
: Either the specified `$type` does not exist, or that module named `$class` returns compilation errors. If the type as specified in the warning is not the name of a package, you specified a nickname which was not defined. Maybe you forgot the 'require' the package which defines the nickname.

Warning: Cannot read archive from `$source`

Error: Creation of a collection via `$class` failed.
: The `$class` did compile, but it was not possible to create an object of that class using the options you specified.

Error: Don't know what type of collection you want to add.
: If you add a collection, it must either by a collection object or a list of options which can be used to create a collection object. In the latter case, the type of collection must be specified.

Warning: No collection `$name`
: The collection with `$name` does not exist and can not be created.

SEE ALSO

This module is part of User-Identity distribution version 1.01, built on February 11, 2022. Website: *http://perl.overmeer.net/CPAN/*

LICENSE

See *http://dev.perl.org/licenses/*