NAME

User::Identity::Item − general base class for User::Identity

INHERITANCE

```
User::Identity::Item is extended by
  Mail::Identity
  User::Identity
  User::Identity::Archive
  User::Identity::Collection
  User::Identity::Location
  User::Identity::System
```

SYNOPSIS

DESCRIPTION

The `User::Identity::Item` base class is extended into useful modules: it has no use by its own.

METHODS

Constructors

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

```
-Option     --Default
 description  undef
 name         <required>
 parent       undef
```

description => STRING
: Free format description on the collected item.

name => STRING
: A simple name for this item. Try to give a useful name in the context of the item time. Each time when you lookup items, you need to specify this name, so it should be unique and not to hard to handle in your program. For instance, when a person is addressed, you usually will give him/her this a nickname.

parent => OBJECT
: The encapsulating object: the object which collects this one.

Attributes

`$obj`−>**description**()
: Free format description on this item. Please do not add any significance to the content of this field: if you are in need for an extra attribute, please contact the author of the module to implement it, or extend the object to suit your needs.

`$obj`−>**name**([$newname])
: The name of this item. Names are unique within a collection... a second object with the same name within any collection will destroy the already existing object with that name.

  Changing the name of an item is quite dangerous. You probably want to call **User::Identity::Collection::renameRole()** instead.

Collections

`$obj`−>**add**($collection, `$role`)
: The `$role` is added to the `$collection`. The`$collection` is the name of a collection, which will be created automatically with **addCollection()** if needed. The `$collection` can also be specified as existing collection object.

  The `$role` is anything what is acceptable to **User::Identity::Collection::addRole()** of the collection at hand, and is returned. `$role` typically is a list of parameters for one role, or a reference to an array containing these values.

  example:

```
my $ui   = User::Identity->new(...);
my $home = $ui->add(location => [home => street => '27 Roadstreet', ...] );
my $work = $ui->add(location => work, tel => '+31-2231-342-13', ... );

my $travel = User::Identity::Location->new(travel => ...);
$ui->add(location => $travel);

my $system = User::Identity::Collection::System->new(...);
$ui->add($system => 'localhost');
```

$obj->**addCollection**($object | <[$type], %options>)

Add a new collection of roles to an item. This can be achieved in two ways: either create an User::Identity::Collection `$object` yourself and then pass that to this method, or supply all the `%options` needed to create such an object and it will be created for you. The object which is added is returned, and can be used for many methods directly.

For `%options`, see the specific type of collection. Additional options are listed below.

```
-Option--Default
 type    <required>
```

type => STRING|CLASS

The nickname of a collection class or the CLASS name itself of the object to be created. Required if an object has to be created. Predefined type nicknames are `email`, `system`, and `location`.

example:

```
my $me   = User::Identity->new(...);
my $locs = User::Identity::Collection::Locations->new();
$me->addCollection($locs);

my $email = $me->addCollection(type => 'email');
my $email = $me->addCollection('email');
```

$obj->**collection**($name)

In scalar context the collection object with the `$name` is returned. In list context, all the roles within the collection are returned.

example:

```
my @roles = $me->collection('email');        # list of collected items
my @roles = $me->collection('email')->roles; # same of collected items
my $coll  = $me->collection('email');        # a User::Identity::Collection
```

$obj->**parent**([$parent])

Returns the parent of an Item (the enclosing item). This may return `undef` if the object is stand-alone.

$obj->**removeCollection**($object|$name)

$obj->**type**()

User::Identity::Item->**type**()

Returns a nice symbolic name for the type.

$obj->**user**()

Go from this object to its parent, to its parent, and so on, until a User::Identity is found or the top of the object tree has been reached.

example:

```
print $email->user->fullName;
```

Searching

`$obj->`**find**($collection, `$role`)

Returns the object with the specified `$role` within the named collection. The collection can be specified as name or object.

example:

```
my $role  = $me->find(location => 'work');       # one location
my $role  = $me->collection('location')->find('work'); # same

my $email = $me->addCollection('email');
$me->find($email => 'work');
$email->find('work');   # same
```

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.

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.

Error: Each item requires a name

You have to specify a name for each item. These names need to be unique within one collection, but feel free to give the same name to an e-mail address and a location.

Warning: No collection `$name`

The collection with `$name` does not exist and can not be created.

Warning: Unknown option `$name` for a `$class`

One used option is not defined. Check the manual page of the class to see which options are accepted.

Warning: Unknown options `@names` for a `$class`

More than one option is not defined.

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