PAS Plugin MemberPropertyToGroup Documentation

Release 1.0a1

kitconcept

Mar 05, 2018

Contents

1 Features **3**

2 Documentation **5**

3 Installation **7**

4 Contribute **9**

5 Support **11**

6 Development **13**

7 Credits **15**

8 License **17**

8.1 Features 17

8.2 Extensibility 24

8.3 Indices and tables 25

Plone PAS plugin to create virtual groups based on member properties.

The main use case are organisations that have an existing LDAP infrastructure that organises groups through member properties instead of LDAP groups.

If you have a vanilla Plone site without LDAP, you most likely do not need this plugin.

CHAPTER 1

Features

- Create virtual groups based on member properties
- Create multiple virtual groups based member properties
- Create virtual group based on a member properties prefix
- Grant local permissions based on virtual member properties groups

CHAPTER 2

Documentation

The full documentation for integrators and developers can be found in the "docs" folder. It is also available online at http://paspluginsmemberpropertytogroup.readthedocs.org.

CHAPTER 3

Installation

Install pas.plugins.memberpropertytogroup by adding it to your buildout:

```
[buildout]

 ...

 eggs =
     pas.plugins.memberpropertytogroup
```

and then run "bin/buildout".

If you are on Plone 3, you need to include the plone.app.registry KGS (know good set) and add the [plone3] extras to fetch the additonal dependencies that are not part of Plone 3:

```
[buildout]
extends =
    http://dist.plone.org/release/3.3.6/versions.cfg
    http://good-py.appspot.com/release/plone.app.registry/1.0b2?plone=3.3.6

...

eggs =
    pas.plugins.memberpropertytogroup [plone3]
```

You can find a working example of a Plone 3 buildout here: https://github.com/kitconcept/pas.plugins.memberpropertytogroup/blob/master/plone-3.3.x.cfg

CHAPTER 4

Contribute

- Source code at Github
- Issue tracker at Github or same issues on Kanban board at Waffle.io
- Documentation at ReadTheDocs:

CHAPTER 5

Support

If you are having issues, please let us know.

CHAPTER 6

Development

Plone 3 There must be an `python2.4` binary available in system path, then:

```
$ bootstrap-3.3.x.sh
```

Plone 4 There must be an `python` binary available in system path pointing to Python 2.7 , then:

```
$ bootstrap-4.3.x.sh
```

Plone 5 There must be an `python` binary available in system path pointing to Python 2.7 , then:

```
$ bootstrap-5.0.x.sh
```

CHAPTER 7

Credits

The development of this plugin has been kindly sponsored by Bonn University.

Developed by kitconcept.

Icon by FamFamFam

CHAPTER 8

License

The project is licensed under the GPLv2.

Contents:

8.1 Features

8.1.1 Create virtual groups based on member properties

As administrator I can create a group based on member properties

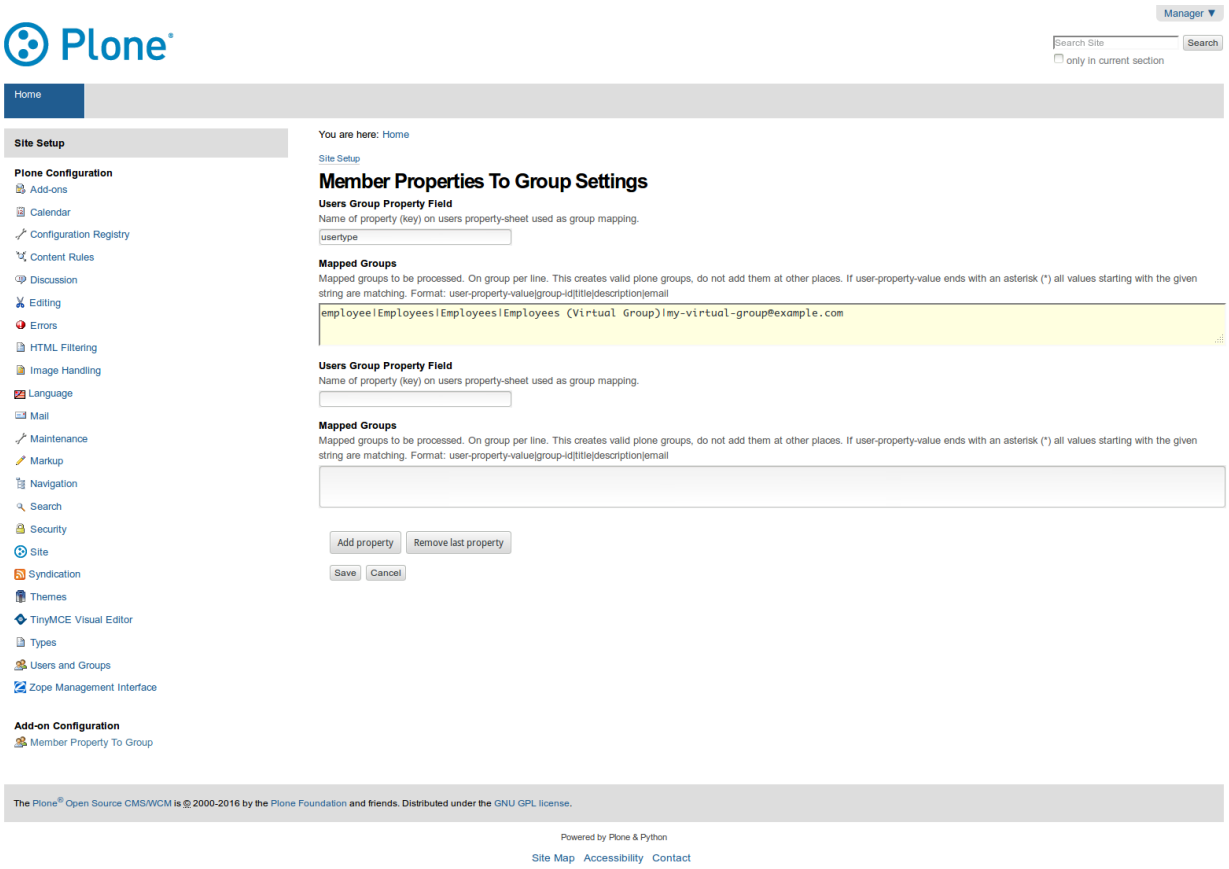
```
Scenario: As administrator I can create a group based on member properties
  Given a user with the property 'usertype' = 'employee'
    and a logged-in manager
   When I create a virtual group 'Employees' with the property 'usertype' = 'employee'
   Then the user is member of the group 'Employees'
```

Given

- a user with the property ‘usertype’ = ‘employee’
- a logged-in manager

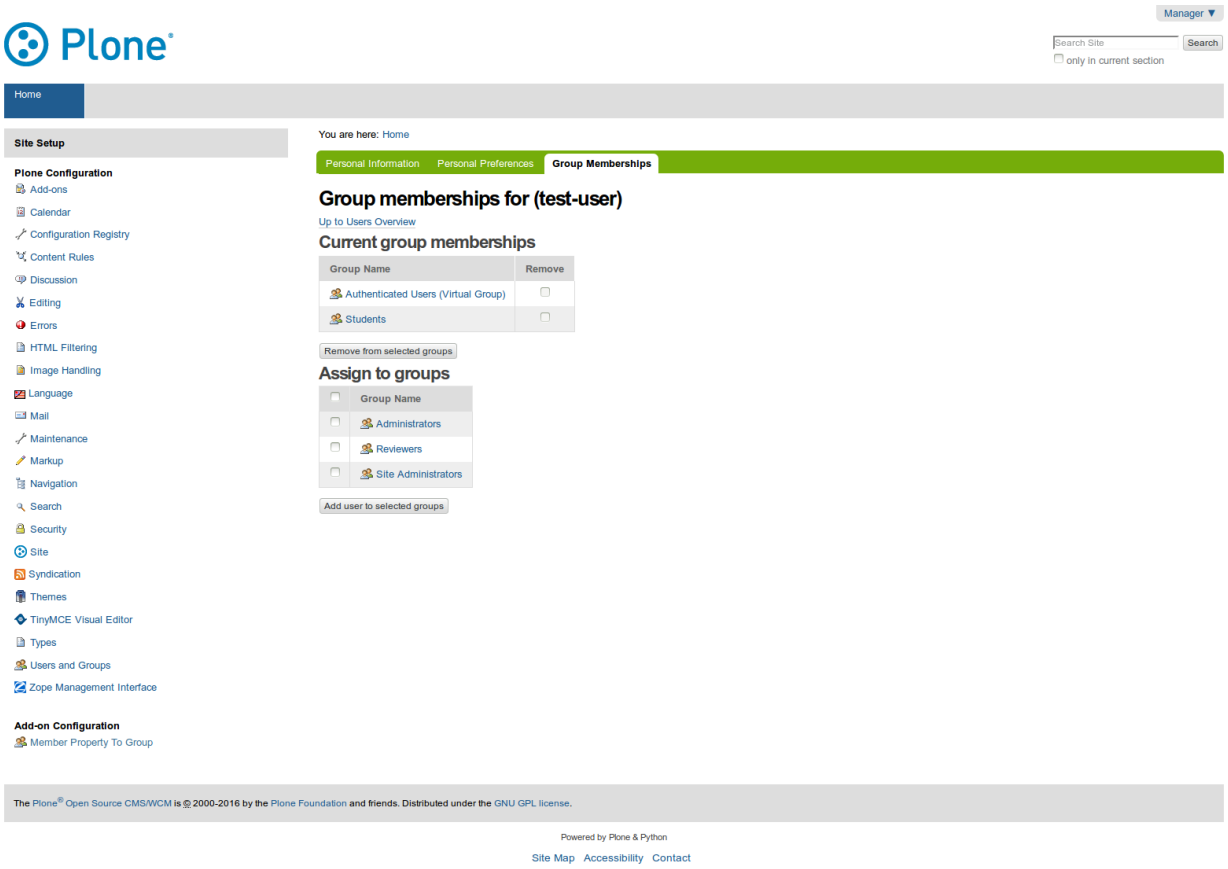
When

- I create a virtual group ‘Employees’ with the property ‘usertype’ = ‘employee’.



Then

- the user is member of the group ‘Employees’



8.1.2 Create multiple virtual groups based member properties

As administrator I can create a group based on multiple member properties

```
Scenario: As administrator I can create a group based on multiple member properties
  # Pass Execution  Not implemented yet
  Given a user 'John Doe' with the property 'usertype' = 'employee'
    and a user 'Jane Doe' with the property 'city' = 'bonn'
    and a logged-in manager
   When I create a virtual group 'Employees' with the property 'usertype' = 'employee'
    and I add another virtual group 'Locals' with the property 'city' = 'bonn' in␣
↪slot 1
   Then the user 'John Doe' is member of the group 'Employees'
    and the user 'Jane Doe' is member of the group 'Locals'
```

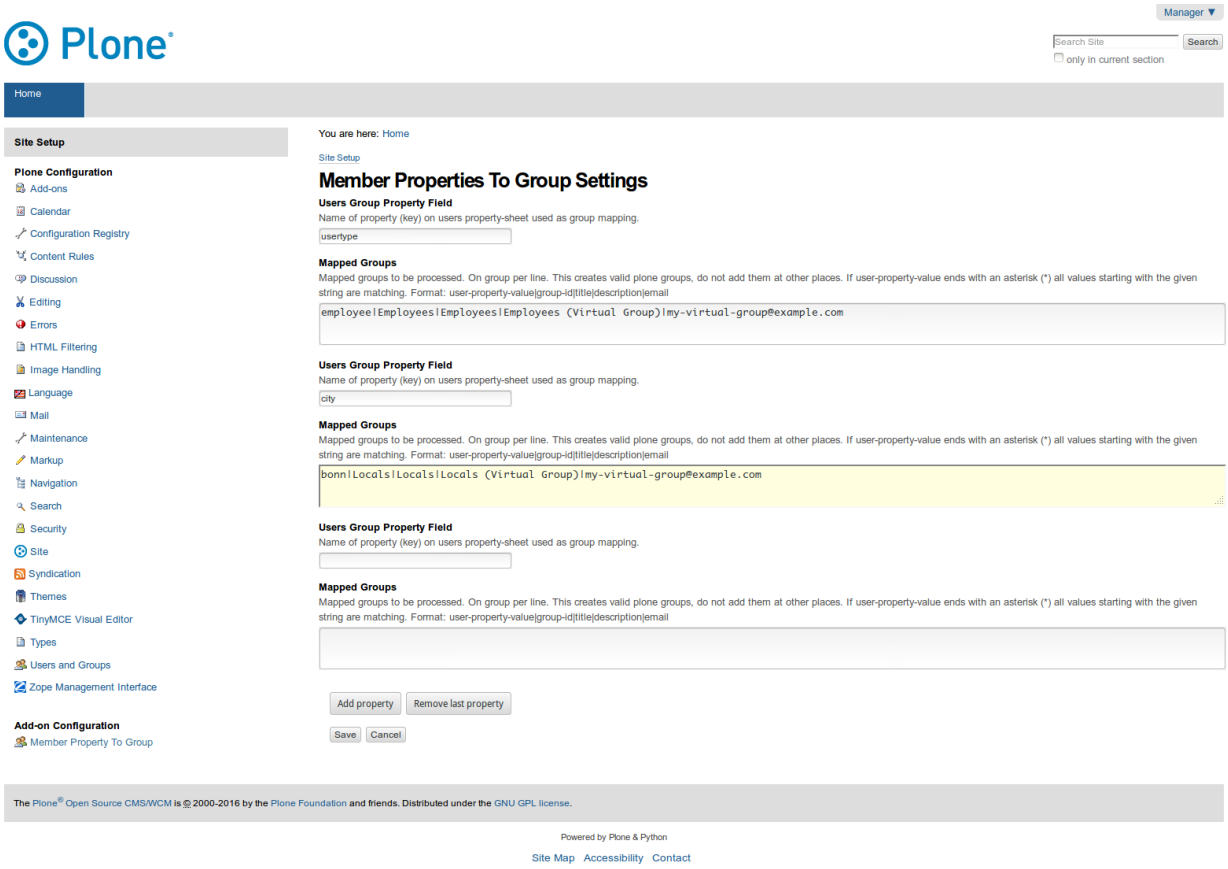
Given

- a user ‘John Doe’ with the property ‘usertype’ = ‘employee’
- a user ‘Jane Doe’ with the property ‘city’ = ‘bonn’
- a logged-in manager

When

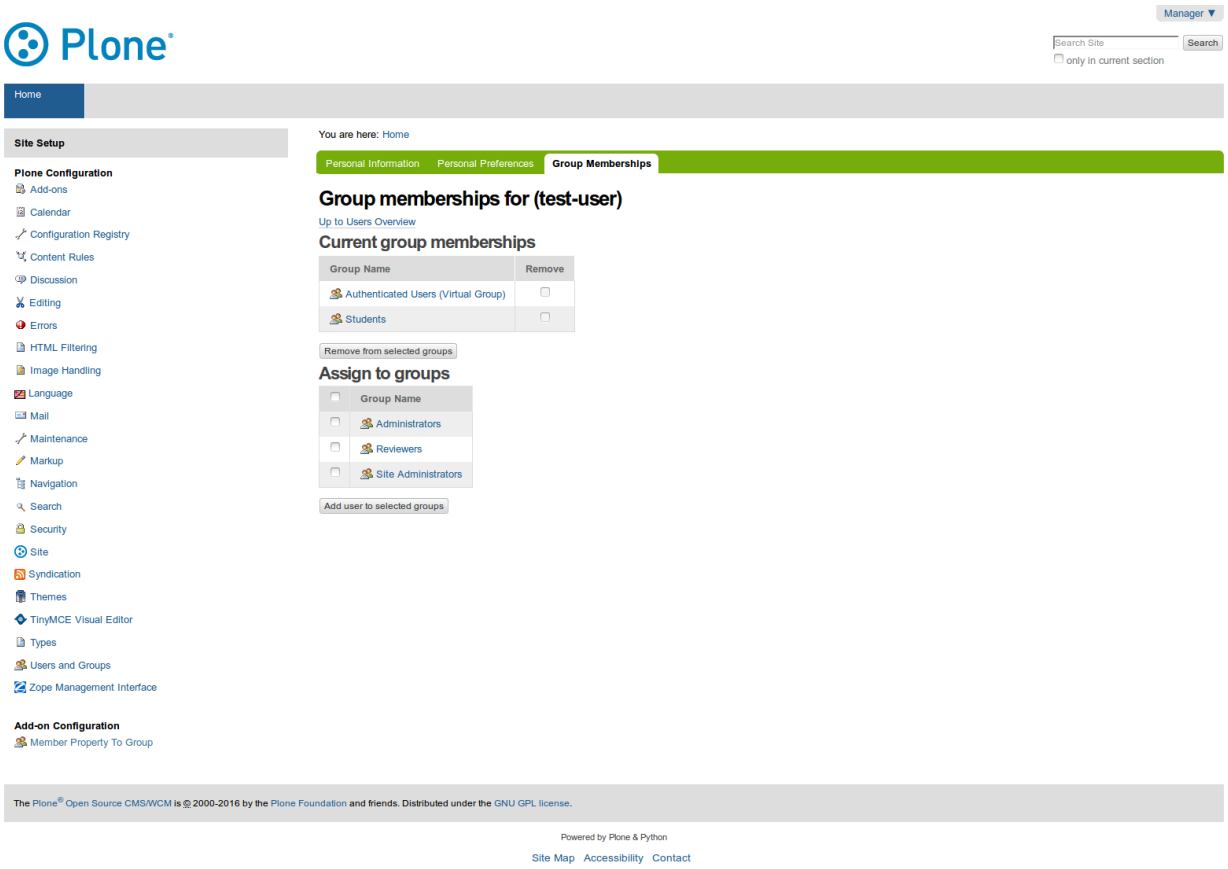
- I create a virtual group ‘Employees’ with the property ‘usertype’ = ‘employee’

- I add another virtual group ‘Locals’ with the property ‘city’ = ‘bonn’ in slot 1



Then

- the user ‘John Doe’ is member of the group ‘Employees’
- the user ‘Jane Doe’ is member of the group ‘Locals’



8.1.3 Create virtual group based on a member properties prefix

As administrator I can create a group based on member properties prefixes

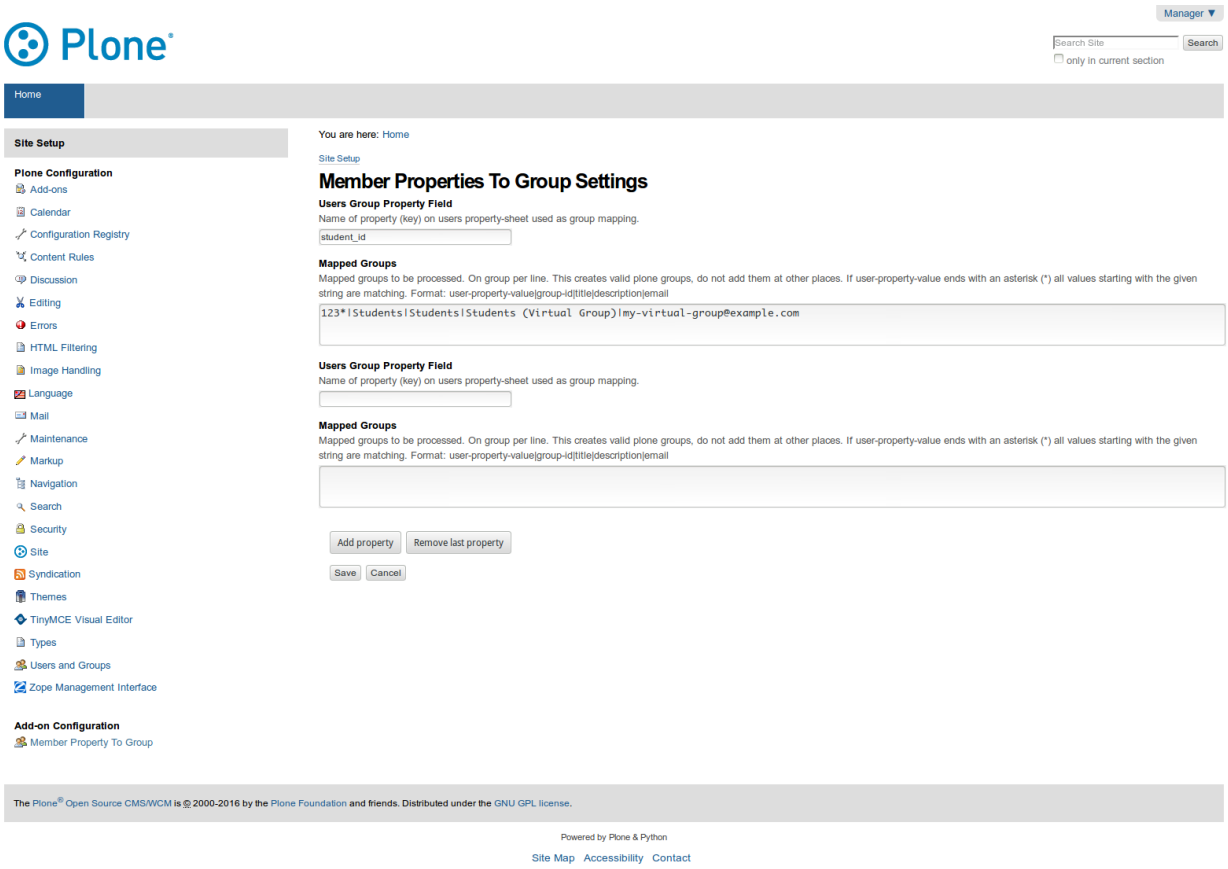
```
Scenario: As administrator I can create a group based on member properties prefixes
  Given a user with the property 'student_id' = '1234567'
    and a logged-in manager
   When I create a virtual group 'Students' with the property 'student_id' = '123*'
   Then the user is member of the group 'Students'
```

Given

- a user with the property ‘student_id’ = ‘1234567’
- a logged-in manager

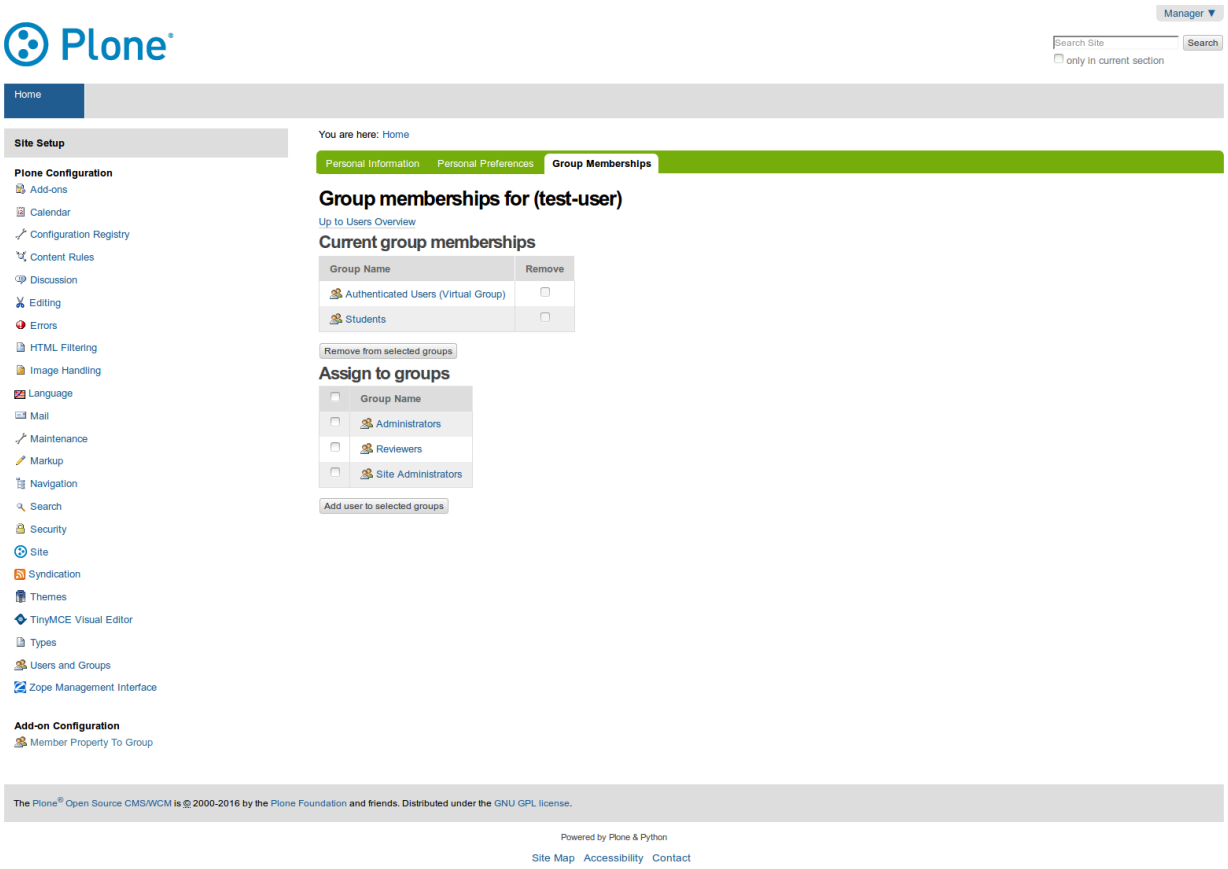
When

- I create a virtual group ‘Students’ with the property ‘student_id’ = ‘123*’



Then

- the user is member of the group ‘Students’



8.1.4 Grant local permissions based on virtual member properties groups

As reviewer I can grant permissions based on member properties groups

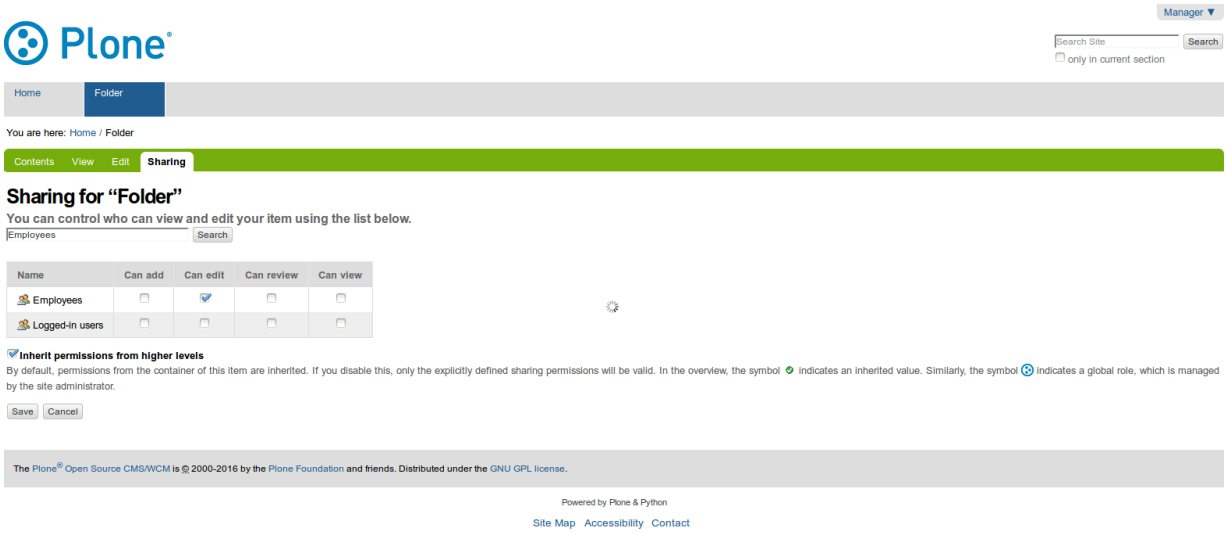
```
Scenario: As reviewer I can grant permissions based on member properties groups
  Given a user with the property 'usertype' = 'employee'
    and a virtual group 'Employees' with the property 'usertype' = 'employee'
    and a logged-in manager
   When I grant the virtual group 'Employees' the 'Edit' permission on a folder
```

Given

- a user with the property ‘usertype’ = ‘employee’
- a virtual group ‘Employees’ with the property ‘usertype’ = ‘employee’

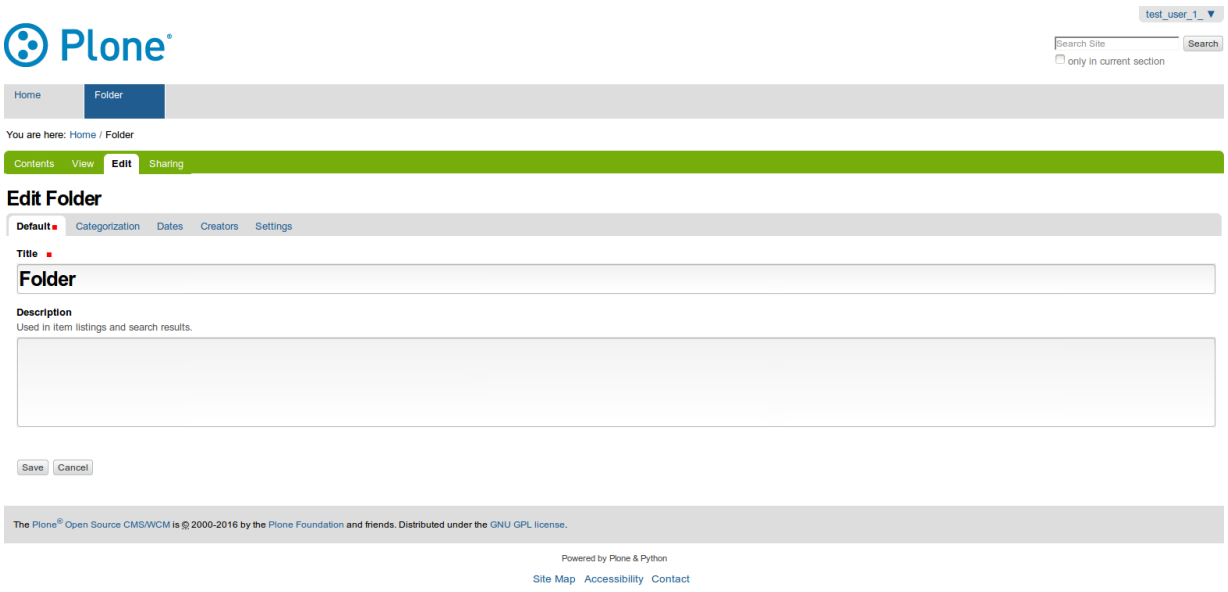
When

- I grant the virtual group the ‘edit’ permission on a folder.



Then

- the user can edit the folder.



8.2 Extensibility

8.2.1 Reason for custom code

`pas.plugins.memberpropertytogroup` has one shortcoming: With this approach it is not possible to list the groups a members in a performant way. One would have to loop over all member instances for each group id, which gets expensive soon if there are many users.

Specific backends - or user providers - offering its specific ways to get around this. I.e for users stored in a SQL database the group may be queried efficiently. The same may apply for LDAP, dependent on how the users are stored.

For other storages or for the default Plone users this does not apply

8.2.2 Solution

The plugins method `getGroupMembers` is responsible to return the members of a given group. There a utility component is queried providing the interface `pas.plugins.memberpropertytogroup.interfaces.IGetGroupMembers`. If there is no utility found an empty tuple is returned. This is the default behavior.

Integrators using this module may provide their own solution by registering a utility for this interface. In case a utility was found, it gets called with the plugin instance and the group id as parameters. The result of the call is then considered as a list of members of the group and returned as is.

8.2.3 Example

Here is a simple but complete example for a specific `IGetGroupMembers` providing function.

In a file `getgroupmembers.py`:

```
from pas.plugins.memberpropertytogroup.interfaces import IGetGroupMembers
from zope.component import provider

@provider(IGetGroupMembers)
def example_group_member_fetcher(plugin, group_id):
    # ... here the real code to get the groups members
    # fake here as example
    group_members = ('foo', 'bar', 'baz')
    return group_members
```

And a line of zcml configuration in `configure.xml`:

```
...
<utility component=".getgroupmembers.example_group_member_fetcher" />
...
```

8.3 Indices and tables

- genindex
- modindex
- search