Django-members-roles Documentation

Release

Cowhite Software Pvt Ltd

May 17, 2018

Contents:

CHAPTER 1

Overview

django-members-roles provides a system to invite members to a group(an Organization for example), add roles to the members of the group and then add permissions to urls based on the role of the member.

1.1 Requirements

- Python 2.7, 3.3, 3.4, 3.5
- Django >= 1.11, < 1.12

CHAPTER 2

Installation

Install latest version from pypi:

```
pip install django-members-roles
```

Required settings in settings.py:

```
# Add django_members_roles to INSTALLED_APPS
INSTALLED_APPS = [
    #...
    django_members_roles,
    #...
]
```

Add to urls.py. Note that the exact namespacec of django-members-roles is mandatory:

```
url(r'^django_members_roles/', include(
    'django_members_roles.urls', namespace='django-members-roles')),
```

Middleware or decorator: You can add middleware or decorator for deciding whether a user can access a page or not. Add middleware if you want to check permissions for all urls of the project. Add decorator if you want to check permissions only for some of the urls.

Adding middleware:

```
MIDDLEWARE = [
    # default other middlewares
    # ....
    'django_members_roles.middleware.url_permission_middleware',
]
```

Adding decorator:

```
from .decorators import has_url_permission_decorator
url(r'^$', has_url_permission_decorator(login_required(ExampleClassBasedView.as_
↪view()))),
    name="example-url"),

url(r'^$', has_url_permission_decorator(login_required(example_function_based_view)),
    name="example-url"),
```

Optional settings in settings.py(more details in configuration):

```
DJANGO_MEMBERS_ROLES_CONFIRMATION_REQUIRED # Default is True
DJANGO_MEMBERS_ROLES_TEST_CASE_MODEL_NAME # Default is "group"
DJANGO_MEMBERS_ROLES_TEST_CASE_APP_LABEL # Default is "auth"
DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID # Default is "content_type_id"
DJANGO_MEMBERS_ROLES_QUERY_PARAM_OBJECT_ID # Default is "object_id"
DJANGO_MEMBERS_ROLES_INVITATION_METHOD # Default is "cron"
```

CHAPTER 3

Usage

After doing initial setup mentioned in installation page, you need to know how to use this app.

3.1 What this app does ?

- Allows a group(an organization or a college for example) to invite members to its group.
- Members need to accept the invitation to be part of the group.
- Allows customization of access to pages in the project based on roles.

3.2 Things to be done by the developer:

- Whenever a new group(an organization for example) is created, then an admin need to be added to that group as the first member with role 'admin'. For doing that, you need to call the function 'create_admin_role' with parameters 'content_object', 'user' after the group is created(you can do it either by overriding save method of that group or by catching a post_save signal):

```
class Organization(models.Model): # Organization is a kind of group
    my_field_for_user_who_created_this_org = models.ForeignKey(User)

    #...some fields...

    def save(self, *args, **kwargs):
        new_instance = False
        if self.id:
            new_instance = True
        super(Organization, self).save(*args, **kwargs)
        if new_instance:
            create_admin_role(self, self.my_field_for_user_who_created_this_org)
```

Or, you can do it using post_save signal:

```
from django.db.models.signals import post_save
from django.dispatch import receiver

@receiver(post_save, sender=Organization, dispatch_uid="create_admin_role_for_
↪organization")
def create_admin_role_for_organization(sender, instance, **kwargs):
    if kwargs['created']:
        create_admin_role(instance, instance.my_field_for_user_who_created_this_org)
```

- In the group page(or whereever you want), place the link to manage staff. This is the link where we have done all the invitation system, role adding etc. Dont forget the namespace.:

```
{% url 'django-members-roles:manage-members' content_type_id object_id %}
```

- This app adds the staff/members of the group to the model GenericMember that we have added. But if you want to have additional fields to the members, you can add a one to one model like this:

```
from django_members_roles.models import GenericMember

class MyOrganizationMembership(models.Model):
    generic_member = models.OneToOneField(GenericMember)
    additional_field = models.TextField()
```

- For every url that you want to check whether the current user has the permissions, you need to send the query parameters to that url like this:

```
<a href="{% url 'abc' %}?content_type_id=12&object_id=1">Some link</a>

Where the strings content_type_id and object_id can be changed by changing the
↪settings DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID and DJANGO_MEMBERS_
↪ROLES_QUERY_PARAM_OBJECT_ID.

For example, if you set them as "cti" and "oi" in settings

DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID = "cti"
DJANGO_MEMBERS_ROLES_QUERY_PARAM_OBJECT_ID = "oi"

then, you can send the url like

<a href="{% url 'abc' %}?cti=12&oi=1">Some link</a>

Here content_type_id is the content type id of the group(for example
↪Organization) and object_id is the id of the instance of that Organization.

Only this way, the middleware(or the decorator) that we developed will check the
↪current user for permissions.
```

3.3 How to use ?

3.4 Invitation System Usage:

After visiting the url 'django-members-roles:manage-members', you will have options to see the members, add new member, see the roles and add new role. You can click "add member" to add a new member. And click "Member List" to see the list of members. You can invite multiple people at once.

3.5 Project Url:

Go to the admin url '/admin/django_members_roles/projecturl/' and then click the button on the top right 'update project urls' which will take few seconds or minutes to update the list of project urls from your project.

3.6 Url Permission:

Add a new url permission at /admin/django_members_roles/urlpermissionrequired/add/ by selecting respective project url and necessary permissions required for that url. When checking, we will check whether the role belonging to the currently logged in user(for that content type) has all the permissions mentioned for this url. If atleast one permission is not present for that role of the currently logged in user(for that content type) then it will return 403 Permission Denied. All this happens with the help of a decorator or middleware that we developed.

3.7 Role Permission:

This might be little confusing so handle with little care. This model has two fields, one is content_type and the other is permissions. This must be added only in the admin. The developer need to decide the permissions that a group member can add to the roles for a group. As you know, a single project will have lots of permissions for many models. When the admin of the group is adding the permissions for a role, we should show only few options in the dropdown rather than all. So, the developer need to decide what are the permissions that may be required for a content type(or a group) and then add all those permissions to the content type in this role permission model.

3.8 Role:

When adding/editing a role in the interface we developed at 'django-members-roles:manage-members', you can add all the permissions for that role. You can only pick some of the permissions here, not all. The list permissions in the dropdown shown here is dependant on the permissions enabled for a content type(that we added in RolePermission model).

CHAPTER 4

Configuration

Optional settings in settings.py:

DJANGO_MEMBERS_ROLES_CONFIRMATION_REQUIRED

Default: True

Allowed values: True/False

If True:

A person who is invited to a group, needs to accept invitation before being a member of that group.

If False:

A person who is invited to a group, will be a member of the group even without accepting the invitation

DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID and DJANGO_MEMBERS_ROLES_QUERY_PARAM_O

Default for DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID: content_type_id # You can give anything like abc

Default for DJANGO_MEMBERS_ROLES_QUERY_PARAM_OBJECT_ID: is "object_id" # You can give anything like def

For every request that needs permission validation based on this app, it expects two query parameters(Or GET parameters). One if the value based on the setting DJANGO_MEMBERS_ROLES_QUERY_PARAM_CONTENT_TYPE_ID and the other is the value based on the setting DJANGO_MEMBERS_ROLES_QUERY_PARAM_OBJECT_ID.

This app retrieves the content object using the above two settings and then it will see if the currently logged in user is a member of that content object. Then, it will fetch the role from the member and check the permissions added to that role and decides whether the current page can be viewed by the current user or not.

DJANGO_MEMBERS_ROLES_INVITATION_METHOD

Default: cron

Allowed values:

celery # Preferred

cron

direct

Celery:

If the setting value is "celery" then we use celery for background tasks. That is, when someone invites people to a content object(or simply a group like Organization) then the invitation(s) will be sent to the people invited using celery(background process).

Cron:

We have provided a management command that you can add in cron so the invitations will be sent.

Direct:

We dont suggest this but you can use this. Direct means, the invitations to all the people invited will be sent in the request response cycle and not in the background process. This will affect the performance of the application. Please dont set it to "direct".

DJANGO_MEMBERS_ROLES_TEST_CASE_MODEL_NAME and DJANGO_MEMBERS_ROLES_TEST_CASE_APP_LABEL

Note: Mostly not useful to you, so you can ignore this setting :)

Default for DJANGO_MEMBERS_ROLES_TEST_CASE_MODEL_NAME: group Default for DJANGO_MEMBERS_ROLES_TEST_CASE_APP_LABEL: auth

When the test cases are executed, we need a content object(or a model instance) which the tests treat as a group to add members to it. It can be any content object. And the content type and object id belonging to the content object will be based on the values provided for DJANGO_MEMBERS_ROLES_TEST_CASE_MODEL_NAME and DJANGO_MEMBERS_ROLES_TEST_CASE_APP_LABEL.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search