django-permissionsx Documentation

Release 2.0.0

Robert Pogorzelski

Nov 04, 2017

Contents

- **Version:**
  - 2.0.0
- **Python package:**
  - http://pypi.python.org/pypi/django-permissionsx/
- **Source code:**
  - http://github.com/robpogorzelski/django-permissionsx/
- **Bug tracker:**
  - http://github.com/robpogorzelski/django-permissionsx/issues/
- **Example project:**
  - http://github.com/robpogorzelski/django-permissionsx-example/

CHAPTER 1

Introduction

django-permissionsx is an alternative to Django permissions system. The main difference is that this package does not store anything in the database. Instead, it evaluates permissions dynamically based on the context data (request attributes, template context etc.)

Example:

```
def user_has_something(context):
    return context('request.user.has_something_method')

def is_owner_and_has_something(context):
    return is_owner(context) and user_has_something(context)
```

CHAPTER 2

Contents

2.1 Quick Start

2.1.1 Compatibility note

This package in version 2.x is intended to work with Python 3.6+ and Django 1.11+.

2.1.2 1. Install *django-permissionsx* package:

```
pip install django-permissionsx
```

2.1.3 2. Don't forget to add *permissionsx* to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.staticfiles',
    [...]
    'permissionsx',
```

2.1.4 3. Define permissions in a module of your choice:

```
def is_owner(context):
    return context('request.user') == context('view.get_object.user')

def is_staff(context):
    return context('request.user.is_staff')
```

```
def is_staff_or_owner(context):
    return is_owner(context) or is_staff(context)
```

2.1.5 4. Use permissions in your views, e.g.:

```
from permissionsx.contrib.django import views
from example.profiles.permissions import is_staff_or_owner

class ObjectDetailView(views.PermissionsDetailView):

    model = Item
    permissions = is_staff_or_owner
```

2.1.6 5. Apply permissions to your templates if you need:

```
{% load permissionsx_tags %}

{% permissions 'example.profiles.permissions.is_staff_or_owner' as is_staff_or_owner
↪%}
<ul id="utility-navigation">
    {% if is_staff_or_owner %}
        <a href="#">Edit article</a>
    {% endif %}
</ul>
```

2.1.7 6. If you'd like to display a default message each time a user is denied access, simply:

```
from permissionsx.contrib.django import views

class ObjectDetailView(PermissionsDetailView):

    [...]
    permissions_response_class = views.AccessDeniedView
```

2.1.8 7. You're done!

If a given object has a *ForeignKey* field named *user* that points to the user requesting the view (*request.user*), the permissions module will either let execute one of the HTTP methods or simply redirect to the login screen (or log the user out, depends on the configuration).

Browse through *permissionsx.tests.permissions* for more ideas on how to use this package or check out the example for a good starting point. Also, drop me a line if you have questions (or there is something I could improve on my side).

2.2 Settings

2.2.1 PERMISSIONSX_REDIRECT_URL_NAME

Defaults to *login*.

If a user was not granted access, resolve the URL name and redirect to *PERMISSIONSX_REDIRECT_URL_NAME*.

2.2.2 PERMISSIONSX_LOGOUT_IF_DENIED

Defaults to *False*.

If a user was not granted access, log the user out before redirecting to *PERMISSIONSX_REDIRECT_URL_NAME*.

2.3 Compatibility

2.3.1 Python 3.6

Package	django-permissionsx 2.0.0
Django 1.11.x	
django-tastypie 0.14.0	
django-debug-toolbar 1.8	

2.4 Changelog

2.4.1 2.0.0

- **The changes are drastic and backwards incompatible:**
  - First of all, I got rid off Django ORM-like syntax for defining permissions and replaced it with a pure and simple Python. Easier to follow (and debug).
  - The code was largely refactored, simplified and optimized (mostly thanks to the above).
  - Focused on Python 3.6+, the future is already here.
  - Removed if_false/if_true overrides.

2.4.2 1.4.0

- Updated package to work with Django 1.1x and Python 3.5+.

2.4.3 1.3.4

- Adding redirects inside of a wrapping P() is now possible, e.g.

```
class TestView(PermissionsTemplateView):

    permissions = Permissions(
        P(user__is_authenticated=True) &
        P(
            P(user__is_superuser=True) | P(object__owner=Cmp('user')), if_
↪false=AccessDeniedView.as_view()
        )
    )
```

2.4.4 1.3.3

- Minor maintenance release. No changes affecting current installations.

2.4.5 1.3.2

- Minor maintenance release. No changes affecting current installations.
- **Changed policy on compatibility. Each release is now guaranteed to support:**
  - latest stable release and next upcoming of Django;
  - latest Python 2.x and 3.x versions;
  - latest stable versions of interoperable packages (currently *django-debug-toolbar* and *django-tastypie*)

2.4.6 1.3.1

- Minor maintenance release. No changes affecting current installations.

2.4.7 1.3.0

- On a view level *permissions_class* is now *permissions*.
- Renamed *get_permissions* to *get_rules*.
- Renamed *permissions* to *rules*.
- Renamed *check_permissions* to *check*.
- Other internal API changes.
- Object passed to *permissions* must be an instance.
- Added example project at http://github.com/robpogorzelski/django-permissionsx-example.
- Removed *PERMISSIONSX_DEBUG* setting.
- Renamed *PermissionsDebugPanel* to *PermissionsPanel* (following django-debug-toolbar).

2.4.8 1.2.1

- Bugfix release. Merging permissions with a `Permissions` instance with no rules defined was raising *TypeError* exception.

2.4.9 1.2.0

- Removed *set_request_context()* method from `Permissions`. This was adding unjustified complexity. Instead, inheritance and *super()* calls can be used.
- Added new operator: *Cmp()*. This allows to compare permission rules to request object even if they are not currently available in the method scope. Also, this prevents exceptions from non-existing relations (e.g. *request.user.company* while *company* can be null).
- Simplification. Removed dependency on Django patches or middleware tricks. Now if a user is anonymous and permissions are checked, and they fail on specific attributes of the *User* instance (e.g. *get_profile()*), user will be denied access for that specific rule by default.
- Updated Django Debug Toolbar integration.
- Added support for passing permission rules to classes having permissions already defined. This will cause all rules to be merged using AND (&). For example, following is now possible:

`accounts/permissions.py`

```
class ContentEditablePermissions(Permissions):

    def get_permissions(self, request, **kwargs):
        try:
            request.content = Content.objects.get(slug=kwargs.get('slug'))
        except Content.DoesNotExist:
            request.content = None
        return P(user__is_author_of=Arg('content')) | P(content__publisher=Cmp('user.
↪publisher'))
```

`content/views.py`

```
class ContentUpdateView(DjangoViewMixin, UpdateView):

    model = Content
    template_name = 'content/content_edit.html'
    form_class = ContentCreateUpdateForm
    permissions_class = ContentEditablePermissions(
        P(content__can_change_price=True)
    )
```

So the final result would be:

```
request.content.can_change_price() & (request.user.is_author_of(request.content) |␣
↪(request.content.publisher == request.user.publisher))
```

2.4.10 1.1.4

- Fixed Django debug toolbar panel.
- Removed caching (explanation).

2.4.11 1.1.3

- Added in-memory caching (`settings.PERMISSIONSX_CACHING`).
- Added tests for Django Views, settings and overrides.

- Changed the way overrides work. Few things got simplified by the way. Now it is possible to use multiple overrides attached to `P` objects, not the top-level `Permissions`.

2.4.12 1.1.2

- Added support over overriding response behavior on a permission level.
- One-liners for defining permissions.
- `Arg` allows passing request object to permission checking function.
- Package `django-classy-tags` is no longer a requirement.
- Added Sphinx documentation with extended examples.

2.4.13 1.1.0

- New syntax possible for retrieving related objects, e.g. `P(user__get_profile__related_object__is_something=`

2.4.14 1.0.0

- Added support for custom response classes (e.g. for changing redirect URL, adding custom user message).
- Added tests for checking permissions.
- Minor fixes and improvements.

2.4.15 0.0.9

- Added support for Django templates, including per-object checks.
- Renamed class-level `permissions` to `permissions_class`.
- Dropped support for simple permissions defining for the benefit of greater flexibility.
- Renaming and refactoring, again. Good stuff: managed to get rid of middleware and a class. Things got largely simplified in general.
- Requirement: `django-classy-tags`.

2.4.16 0.0.8

- This version is backward **incompatible**.
- Changed syntax to follow QuerySet filtering convention.
- Sadly, tests are gone. Need to write new ones, what will not happen until 1.0.0 release.
- Example project's gone. Will be back at a different URL.
- `PERMISSIONSX_DEFAULT_URL` was renamed to `PERMISSIONSX_REDIRECT_URL`.
- New setting was added: `PERMISSIONSX_LOGOUT_IF_DENIED`.

2.5 Reference

Automatically generated documentation:

2.5.1 permissionsx.base

PermissionsX - Authorization for Django.

copyright Copyright (c) 2013-2017 by Robert Pogorzelski.

license BSD, see LICENSE for more details.

2.5.2 permissionsx.contrib.django

PermissionsX - Authorization for Django.

copyright Copyright (c) 2013-2017 by Robert Pogorzelski.

license BSD, see LICENSE for more details.

2.5.3 permissionsx.contrib.django_debug_toolbar

2.5.4 permissionsx.contrib.tastypie

2.6 Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

P