thecut-authorship Documentation

Release 1.1

The Cut Creative

May 08, 2017

Contents

Contents:

CHAPTER 1

Welcome to thecut-authorship

A set of Django mixins to easily record authorship information for your models.

Features

- Base model allows easy recording of authorship information.
- Integration with Django's class-based views and forms.
- Integration with Django's admin.

Documentation

The full documentation is at https://thecut-authorship.readthedocs.org.

Quickstart

Install `thecut-authorship` using the installation instructions found in the project documentation.

Build a model based on `thecut.authorship.models.Authorship` to record authorship information on it:

```
from thecut.authorship.models import Authorship

class MyModel(Authorship):

    pass
```

This adds `created_by`, `created_at`, `updated_by`, and `updated_at` to your model.

Pass a user into calls to `.save()` to record which user changed the object:

```
example = MyModel()
example.save(user=request.user)
```

If you need to update model data and there's no direct link to a website user, generate and use a site-wide 'generic' user.:

```
from thecut.authorship.models import get_website_user
example = MyModel()
example.save(user=get_website_user())
```

If you wish to automatically record authorship information for changes made in the Django admin, use `thecut.authorship.admin.AuthorshipMixin`.:

```
from .models import MyModel
from django.contrib import admin
from thecut.authorship.admin import AuthorshipMixin

@admin.register(MyModel)
class MyModelAdmin(AuthorshipMixin, admin.ModelAdmin):

    pass
```

If you wish to integrate with `django.forms.ModelForm`, use `thecut.authorship.forms.AuthorshipMixin` and `thecut.authorship.views.AuthorshipMixin`.

In your `forms.py`:

```
from .models import MyModel
from django import forms
from thecut.authorship.forms import AuthorshipMixin

class MyModelForm(forms.ModelForm):

    class Meta(object):
        model = MyModel
```

In your `views.py`:

```
from .forms import MyModelForm
from .models import MyModel
from django.views.generic import CreateView
from thecut.authorship.views import AuthorshipMixin

class MyModelCreateView(AuthorshipMixin, CreateView):

    form_class = MyModelForm
```

`MyModelCreateView` will now automatically pass `request.user` through to `MyModelForm`, which will pass it through to the model's *save()* method.

Credits

See `AUTHORS.rst`.

CHAPTER 2

Installation instructions

1. Install via pip / pypi:

```
$ pip install thecut-authorship
```

2. Add to your project's `INSTALLED_APPS` setting:

```
INSTALLED_APPS = [
    # ...
    'thecut.authorship'
    # ...
]
```

3. Sync your project's migrations:

```
$ python manage.py migrate authorship
```

CHAPTER 3

Usage

Creating a model

Subclass *thecut.authorship.models.Authorship* to create the necessary fields:

```
class MyModel(Authorship):

    pass
```

Saving authorship information

thecut.authorship.models.Authorship.save() will populate the authorship fields when necessary:

```
m = MyCoolModel()
m.save(user=request.user)
```

This will update the model's *thecut.authorship.models.Authorship.updated_by* and *thecut.authorship.models.Authorship.updated_at* fields automatically, as well as *thecut.authorship.models.Authorship.created_by* and *thecut.authorship.models.Authorship.created_at* if the object is new.

If the model creation / update doesn't directly relate to a user, use the site-wide generic authorship user. This can be retrieved with *thecut.authorship.utils.get_website_user()*:

```
m = MyCoolModel()
m.save(user=get_website_user())
```

Tip: If you specify `update_fields` in your call to *thecut.authorship.models.Authorship.save()*, the list will automatically be updated to ensure that authorship information is saved.

Integrating with `django.contrib.admin`

You can automatically update authorship information when a model is altered in the Django admin interface by using *thecut.authorship.admin.AuthorshipMixin*:

```
class MyAdmin(AuthorshipMixin, admin.ModelAdmin):

    pass
```

Hint: This also applies to child inlines if they refer to subclasses of *thecut.authorship.models.Authorship*.

Integrating with class-based views and `ModelForms`

Use *thecut.authorship.views.AuthorshipMixin* on your `django.views.generic.edit.ModelFormMixin`-based views (`django.views.generic.edit.CreateView`, `django.views.generic.edit.UpdateView`.etc):

```
class MyModelCreateView(AuthorshipMixin, CreateView):

    form_class = MyModelForm
```

Then, use *thecut.authorship.forms.AuthorshipMixin* on your `django.forms.ModelForm`-based forms:

```
class MyModelForm(AuthorshipMixin, ModelForm):

    class Meta(object):
        model = MyModel
```

Together, these mixins will—upon a successful form submission—appropriately record `request.user` on the target object.

Warning: You must use *thecut.authorship.views.AuthorshipMixin* on the view *and* *thecut.authorship.forms.AuthorshipMixin* on the form for this to work.

Integrating with Django REST Framework

Use *thecut.authorship.api.views.AuthorshipMixin* on your `CreateModelMixin` / `UpdateModelMixin`-based API views.

CHAPTER 4

API Reference

Models

class `thecut.authorship.models.`**`Authorship`**(**args*, ***kwargs*)

Abstract model to track when an instance was created/updated and by whom.

`created_at` = **None**

`datetime` for when this object was first created.

`created_by`

User who first created this object (required).

`save`(*user=None*, ***kwargs*)

A custom `Model.save()` method that appropriately populates authorship fields.

`updated_at` = **None**

`datetime` for when this object was last saved.

`updated_by`

User who last saved this object (required).

View mixins

class `thecut.authorship.views.`**`AuthorshipMixin`**

Adds the request's `User` instance to the form kwargs.

API view mixins

class `thecut.authorship.api.views.`**`AuthorshipMixin`**

ModelForm mixins

class thecut.authorship.forms.**AuthorshipMixin**(*user*, **args*, ***kwargs*)
Mixin for a ModelForm which sets created_by and updated_by fields for the instance when saved.

Requires that a User instance be passed in to the constructor. Views which utilise AuthorshipViewMixin handle this already.

Django admin mixins

class thecut.authorship.admin.**AuthorshipMixin**
Mixin for a model admin to set created/updated by on save.

Utilities

class thecut.authorship.utils.**get_website_user**
Get a generic 'website' user.

Can be used to specify the required user when there is no direct link to a real user.

Settings

thecut.authorship.settings.**AUTH_USER_MODEL = 'auth.User'**
The user model that *thecut.authorship.utils.get_website_user()* will query against. Defaults to settings.AUTH_USER_MODEL.

thecut.authorship.settings.**WEBSITE_USER = {u'username': u'website'}**
A dictionary that *thecut.authorship.utils.get_website_user()* will pass to get_or_create() in order to return the generic website user.

CHAPTER 5

Testing

Running unit tests

Using your system's Python / Django

You can perform basic testing against your system's Python / Django.

1. Install the test suite requirements:

```
$ pip install -r requirements-test.txt
```

2. Ensure a version of Django is installed:

```
$ pip install Django
```

3. Run the test runner:

```
$ python runtests.py
```

Using a virtualenv

You can use `virtualenv` to test without polluting your system's Python environment.

1. Install `virtualenv`:

```
$ pip install virtualenv
```

2. Create and activate a `virtualenv`:

```
$ cd thecut-authorship
$ virtualenv .
$ source bin/activate
(thecut-authorship) $
```

3. Follow 'Using your system's Python / Django' above.

Using tox

You can use tox to automatically test the application on a number of different Python and Django versions.

1. Install `tox`:

```
$ pip install -r requirements-test.txt
```

2. Run `tox`:

```
(thecut-authorship) $ tox --recreate
```

Tox assumes that a number of different Python versions are available on your system. If you do not have all required versions of Python installed on your system, running the tests will fail. See `tox.ini` for a list of Python versions that are used during testing.

Test coverage

The included `tox` configuration automatically detects test code coverage with `coverage`:

```
$ coverage report
```

Available tests

Forms

class thecut.authorship.tests.test_forms.**TestAuthorshipMixin**(*methodName='runTest'*)

test_requires_an_extra_argument_on_creating_an_instance()
Ensure that *thecut.authorship.forms.AuthorshipMixin*-based forms cannot be instantiated without passing in a user.

test_sets_user_attribute()
Ensure that *thecut.authorship.forms.AuthorshipMixin*-based forms properly set `thecut.authorship.forms.AuthorshipMixin.user` when one is passed on instantiation.

class thecut.authorship.tests.test_forms.**TestAuthorshipMixinSave**(*methodName='runTest'*)

test_calls_super_class_save_method(**args*, ***keywargs*)
Ensure that `thecut.authorship.forms.AuthorshipMixin.save()` calls the superclass's save method..

test_does_not_set_created_by_if_instance_is_saved(**args*, ***keywargs*)
Ensure that *thecut.authorship.forms.AuthorshipMixin*-based forms do not set `thecut.authorship.models.AuthorshipMixin.created_by` if the target object has already been saved.

test_sets_created_by_if_instance_is_not_saved(**args*, ***keywargs*)
Ensure that *thecut.authorship.forms.AuthorshipMixin*-based forms appropriately set `thecut.authorship.models.AuthorshipMixin.created_by` when a user is provided and the target object has not been saved before.

test_sets_updated_by_to_given_user(**args*, ***keywargs*)
Ensure that *thecut.authorship.forms.AuthorshipMixin*-based forms appropriately set `thecut.authorship.models.AuthorshipMixin.updated_by` when a user is provided.

Models

class `thecut.authorship.tests.test_models.`**TestAuthorshipModel**(*methodName='runTest'*)

test_does_not_change_created_by_when_model_instance_is_saved()
Ensure that *thecut.authorship.models.Authorship.created_by* is not updated for existing models.

test_sets_created_by_when_model_instance_is_first_saved()
Check if `created_by` is correctly set on first save.

test_sets_updated_at_if_update_fields_is_specified()
Ensure that *thecut.authorship.models.Authorship.updated_at* is updated, even when `update_fields` is specified.

test_sets_updated_by_if_update_fields_is_specified()
Ensure that *thecut.authorship.models.Authorship.updated_by* is updated, even when `update_fields` is specified.

test_sets_updated_by_when_model_instance_is_saved()
Ensure that *thecut.authorship.models.Authorship.updated_by* is updated on save.

Utils

class `thecut.authorship.tests.test_utils.`**TestGetWebsiteUser**(*methodName='runTest'*)
Tests for get_website_user().

test_get_website_user_returns_same_user()
Test if the same user is returned over multiple calls.

test_get_website_user_returns_user()
Test if something is returned.

CHAPTER 6

History

1.2 (2017-05-09)

- Added support for Django 1.11 and Python 3.6.
- Drop support for Django 1.9.

1.1 (2016-12-21)

- Added support for Django REST Framework.

1.0 (2016-08-16)

- Removed deprecated APIs.
- Removed compatibility code for unsupported versions of Django.
- Improved test coverage.
- Removed code paths in `Authorship.save()` that could not logically be reached.

0.11 (2016-08-16)

- Rewrote documentation.
- Redesigned testing environment.

0.10.1 (2015-11-19)

- Fixed bug when saving on Django 1.4
- Started using `unittest` from Python standard library. Removes Python < 2.7 support.
- Updated tox configuration to test against newer versions of Django / Python.
- Fixed bug that stopped authorship information being updated when `update_fields` is defined but empty.

0.10 (2015-10-09)

- Test against Django 1.8
- Fixed bug where models were incorrectly detected as 'not new' (for the purpose of setting `created_at` and `created_by`) when a pk is manually specified.

0.9 (2015-08-25)

- Set ''on_delete=models.PROTECT' on authorship fields that relate to users.

0.7.1 (2014-12-11)

- Ensure that `created_at` and `created_by` are updated regardless of the contents of `update_fields`.
- Added Django admin mixin to save authorship information when using inlines.

0.7 (2014-11-24)

- Updated documentation.
- Removed `Makefile`.
- Altered testing environment to support Django 1.7
- Added Django 1.7 `AppConfig`.

0.5.3 (2014-07-09)

- Added unit tests for model and form mixin.
- Improved Python 3 compatibility.
- Updated test environment to test against newer versions of Django.
- Ensure that `updated_at` and `updated_by` are updated regardless of the contents of `update_fields`.

0.5.2 (2014-06-20)

- Added `AuthorshipFactory` for testing.

0.5.1 (2014-03-19)

- Removed `distribute` from the install_requires list.

0.5 (2013-03-15)

- First release.

CHAPTER 7

Credits

- Josh Crompton <josh.crompton@thecut.net.au>
- Elena Williams <elena.williams@thecut.net.au>
- Matt Austin <matt.austin@thecut.net.au>
- Kye Russell <kye.russell@thecut.net.au>
- Mark Lockett <mark.lockett@thecut.net.au>

Python Module Index

t

Index

W