django-freeradius Documentation

Release 0.1 alpha

Fiorella De Luca

Jul 27, 2017

Contents:

Django-freeradius is part of the OpenWISP project.

CHAPTER 1

Abstract Models

Firstly, need to add basic models TimeStampedEditableModel is an abstract base class model, that provides self-updating created and modified fields. Write base class and put abstract=True in the Meta class. This model will then not be used to create any database table. Instead, when it is used as a base class for other models, it's fields will be added to those of the child class.

Example of TimeStampedEditableModel code:

```
#django_freeradius/base/models.py

from model_utils.fields import AutoCreatedField, AutoLastModifiedField

class TimeStampedEditableModel(models.Model):
    """
    An abstract base class model that provides self-updating
    ``created`` and ``modified`` fields.
    """
    created = AutoCreatedField(_('created'), editable=True)
    modified = AutoLastModifiedField(_('modified'), editable=True)

    class Meta:
        abstract = True
```

Include the TimeStampedEditableModel to the AbstractModel

Example:

```
#django_freeradius/base/models.py

class AbstractRadiusGroup(TimeStampedEditableModel):
    id = models.UUIDField(primary_key=True, db_column='id')
    group_name = models.CharField(
            verbose_name=_('groupname'), max_length=255, unique=True, db_column=
↪'groupname', db_index=True)
```

```
    priority = models.IntegerField(verbose_name=_('priority'), default=1)
    creation_date = models.DateField(verbose_name=_('creation date'), null=True, db_
↪column='created_at')
    modification_date = models.DateField(
        verbose_name=_('modification date'), null=True, db_column='updated_at')
    notes = models.CharField(
        verbose_name=_('notes'), max_length=64, blank=True, null=True)

    class Meta:
        db_table = 'radiusgroup'
        verbose_name = _('radiusgroup')
        verbose_name_plural = _('radiusgroups')
        abstract = True

    def __str__(self):
        return self.group_name
```

Introduce a ModelAdmin for TimeStampedEditableAdmin

Example of code:

```
#django_freeradius/base/admin.py

from django.contrib.admin import ModelAdmin


class TimeStampedEditableAdmin(ModelAdmin):
    """
    ModelAdmin for TimeStampedEditableModel
    """

    def get_readonly_fields(self, request, obj=None):
        readonly_fields = super(TimeStampedEditableAdmin, self).get_readonly_
↪fields(request, obj)
        return readonly_fields + ('created', 'modified')


class AbstractRadiusGroupAdmin(TimeStampedEditableAdmin):
    pass
```

Creating a Reusable App

First, You have to install *swapper*. If you are publishing your reusable app as a Python package, be sure to add *swapper* to your project's dependencies.You may also want to take a look at the *Swapper Guide <https://github.com/wq/django-swappable-models>*

Install swapper:

```
pip install swapper
```

In your reusable models use import swapper and add to Meta class swappable = swapper.swappable_setting('reusable_app', 'model'):

```
#django_freeradius/models.py

import swapper

from .base.models import (AbstractNas, AbstractRadiusAccounting,
                          AbstractRadiusCheck, AbstractRadiusGroup,
                          AbstractRadiusGroupCheck, AbstractRadiusGroupReply,
                          AbstractRadiusGroupUsers,
                          AbstractRadiusPostAuthentication,
                          AbstractRadiusReply, AbstractRadiusUserGroup)


class RadiusGroup(AbstractRadiusGroup):

    class Meta(AbstractRadiusGroup.Meta):
        abstract = False
        swappable = swapper.swappable_setting('django_freeradius', 'RadiusGroup')
```

Migrations

Swapper can also be used in Django 1.7+ migration scripts to facilitate dependency ordering and foreign key references. To use this feature in your library, generate a migration script with makemigrations and make the following changes:

```
#django_freeradius/migrations

import swapper

class Migration(migrations.Migration):

    initial = True

 dependencies = [
     swapper.dependency('django_freeradius', 'RadiusReply'),
     swapper.dependency('django_freeradius', 'RadiusCheck'),
 ]

 operations = [
     migrations.CreateModel(
         name='Nas',
         fields=[
             ('id', models.AutoField(auto_created=True, primary_key=True,␣
↪serialize=False, verbose_name='ID')),
             ('created', model_utils.fields.AutoCreatedField(default=django.utils.
↪timezone.now, editable=False, verbose_name='created')),
             ('modified', model_utils.fields.AutoLastModifiedField(default=django.
↪utils.timezone.now, editable=False, verbose_name='modified')),
             ('nas_name', models.CharField(db_column='nasname', db_index=True, help_
↪text='NAS Name (or IP address)', max_length=128, unique=True, verbose_name='nas name
↪')),
             ('short_name', models.CharField(db_column='shortname', max_length=32,␣
↪verbose_name='short name')),
             ('type', models.CharField(max_length=30, verbose_name='type')),
             ('secret', models.CharField(help_text='Shared Secret', max_length=60,␣
↪verbose_name='secret')),
```

```
            ('ports', models.IntegerField(blank=True, null=True, verbose_name='ports
↪')),
            ('community', models.CharField(blank=True, max_length=50, null=True,␣
↪verbose_name='community')),
            ('description', models.CharField(max_length=200, null=True, verbose_name=
↪'description')),
            ('server', models.CharField(max_length=64, null=True, verbose_name=
↪'server')),
        ],
        options={
            'db_table': 'nas',
            'swappable': swapper.swappable_setting('django_freeradius', 'Nas'),
            'verbose_name': 'nas',
            'abstract': False,
            'verbose_name_plural': 'nas',
        },
    ),
```

Extends Models

The user of your app can override one or both models in their own app.

Example:

```
#sample_radius/models.py

from django.db import models
from django.utils.translation import ugettext_lazy as _

from django_freeradius.models import (AbstractNas, AbstractRadiusAccounting,
                                      AbstractRadiusCheck, AbstractRadiusGroup,
                                      AbstractRadiusGroupCheck,␣
↪AbstractRadiusGroupReply,
                                      AbstractRadiusGroupUsers,
                                      AbstractRadiusPostAuthentication,
                                      AbstractRadiusReply, AbstractRadiusUserGroup)


class RadiusGroup(AbstractRadiusGroup):
    details = models.CharField(
            verbose_name=_('details'), max_length=64, blank=True, null=True)


class RadiusCheck(AbstractRadiusCheck):
    details = models.CharField(
            verbose_name=_('details'), max_length=64, blank=True, null=True)
```

Add swapper.load_model() to sample_radius/admin.py. Example:

```
from django.contrib import admin

import swapper
from django_freeradius.admin import (AbstractNasAdmin,
                                     AbstractRadiusAccountingAdmin,
                                     AbstractRadiusCheckAdmin,
                                     AbstractRadiusGroupAdmin,
```

```
                                    AbstractRadiusGroupCheckAdmin,
                                    AbstractRadiusGroupReplyAdmin,
                                    AbstractRadiusGroupUsersAdmin,
                                    AbstractRadiusPostAuthenticationAdmin,
                                    AbstractRadiusReplyAdmin,
                                    AbstractRadiusUserGroupAdmin)

RadiusGroupReply = swapper.load_model("django_freeradius", "RadiusGroupReply")
RadiusGroupCheck = swapper.load_model("django_freeradius", "RadiusGroupCheck")
RadiusGroupUsers = swapper.load_model("django_freeradius", "RadiusGroupUsers")
RadiusUserGroup = swapper.load_model("django_freeradius", "RadiusUserGroup")
RadiusReply = swapper.load_model("django_freeradius", "RadiusReply")
RadiusCheck = swapper.load_model("django_freeradius", "RadiusCheck")
RadiusPostAuthentication = swapper.load_model("django_freeradius",
↪"RadiusPostAuthentication")
Nas = swapper.load_model("django_freeradius", "Nas")
RadiusAccounting = swapper.load_model("django_freeradius", "RadiusAccounting")
RadiusGroup = swapper.load_model("django_freeradius", "RadiusGroup")


@admin.register(RadiusGroup)
class RadiusGroupAdmin(AbstractRadiusGroupAdmin):
    model = RadiusGroup
```

Update Settings

Update the settings to trigger the swapper:

```
#django_freeradius/tests/settings.py

if os.environ.get('SAMPLE_APP', False):
        INSTALLED_APPS.append('sample_radius')
        DJANGO_FREERADIUS_RADIUSREPLY_MODEL = "sample_radius.RadiusReply"
        DJANGO_FREERADIUS_RADIUSGROUPREPLY_MODEL = "sample_radius.RadiusGroupReply"
        DJANGO_FREERADIUS_RADIUSCHECK_MODEL = "sample_radius.RadiusCheck"
        DJANGO_FREERADIUS_RADIUSGROUPCHECK_MODEL = "sample_radius.RadiusGroupCheck"
        DJANGO_FREERADIUS_RADIUSACCOUNTING_MODEL = "sample_radius.RadiusAccounting"
        DJANGO_FREERADIUS_NAS_MODEL = "sample_radius.Nas"
        DJANGO_FREERADIUS_RADIUSGROUPUSERS_MODEL = "sample_radius.RadiusGroupUsers"
        DJANGO_FREERADIUS_RADIUSUSERGROUP_MODEL = "sample_radius.RadiusUserGroup"
        DJANGO_FREERADIUS_RADIUSPOSTAUTHENTICATION_MODEL = "sample_radius.
↪RadiusPostAuthentication"
        DJANGO_FREERADIUS_RADIUSGROUP_MODEL = "sample_radius.RadiusGroup"
```

CHAPTER 2

Installation and configuration of Freeradius

We will install freeradius 3.x.

For this it will be much easier if you become the root user.

```
sudo su
```

First, let's add the PPA repository for the Freeradius 3.x stable branch:

```
apt-add-repository ppa:freeradius/stable-3.0
```

```
apt-get update
```

Now you can install freeradius with freeradius-postgres and freeradius-mysql modules:

```
apt-get install freeradius freeradius-mysql freeradius-postgresql
```

Let's go to the file /etc/freeradius/mods-available/sql.

You have to change driver, dialect, server, port, login, password, radius_db.

Example for configuration with postgresql:

```
driver = "rlm_sql_postgresql"

dialect = "postgresql"

# Connection info:

       server = "localhost"
       #port = 3306
       login = "<user>"
       password = "<password>"

# Database table configuration for everything except Oracle

radius_db = "freeradius"
```

Create softlink for modules that you want to add:

```
cd mods-enabled/

ln -s ../mods-available/sql ./

ln -s ../mods-available/redis ./

ln -s ../mods-available/rediswho ./
```

Launch freeradius in debug mode:

```
freeradius -X
```

You may also want to take a look at the *Freeradius documentation <http://freeradius.org/doc/>*

CHAPTER 3

Configure Freeradius to use a RESTful API.

First we need to install the rest module (rml_rest):

```
apt-get install freeradius-rest
```

To enable the module rlm_rest by symlinking, eg:

```
ln -s /etc/freeradius/mods-available/rest /etc/freeradius/mods-enabled/rest
```

rml_rest module configuration

Example:

```
#/etc/freeradius/mods-enabled/rest

connect_uri = "http://127.0.0.1:8000"

authorize {
 uri = "${..connect_uri}/api/authorize/"
 method = 'post'
 body = 'json'
 data = '{"username": "%{User-Name}", "password": "%{User-Password}"}'
 tls = ${..tls}

}

authenticate {
 uri = "${..connect_uri}/api/authorize/"
 method = 'post'
 body = 'json'
 data = '{"username": "%{User-Name}", "password": "%{User-Password}"}'
 tls = ${..tls}
```

```
}
```

Configure the default site:

```
#/etc/freeradius/sites-enabled/default:

authorize {
    rest
    # ... other configuration
}

authenticate {
    rest
    # ... other configuration
}
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search