django-fcm Documentation

Release 0.0.2

Chitrank Dixit

Sep 05, 2017

Contents

Django-fcm is a reusable application. It serves as a server for the FCM (Firebase Cloud Messaging) service and allows you to register devices and send messages to them.

Contents:

CHAPTER 1

Quickstart

1. Install package via *pip*:

```
$ pip install django-fcm
```

2. Add *django-fcm* resources to your URL router:

```
# urls.py
from django.conf.urls import include, url

urlpatterns = [
    url(r'fcm/', include('fcm.urls')),
]
```

   To check fcm urls just use the following command:

```
$ python manage.py fcm_urls

FCM urls:
* Register device
    /fcm/v1/devices/
* Unregister device
    /fcm/v1/devices/{id}/
```

3. Configure *django-dcm* in your `settings.py` file:

```
INSTALLED_APPS = [
    # ...
    'fcm',
]

FCM_APIKEY = "<api_key>"
```

Note: To obtain api key please go to https://console.firebase.google.com/ and grab the key for the server app.

CHAPTER 2

Sending messages

Using `console`:

```
# Get the list of devices
$ python manage.py fcm_messenger --devices
> Devices list:
> (#1) My phone

# python manage.py fcm_messenger --device_id=<device_id> --msg=<message> [--collapse-
↪key <key>]
$ python manage.py fcm_messenger --device_id=1 --msg='my test message'
```

Using `Django orm`:

```
from fcm.utils import get_device_model
Device = get_device_model()

my_phone = Device.objects.get(name='My phone')
my_phone.send_message({'message':'my test message'}, collapse_key='something')
```

`collapse_key` parameter is optional (default message).

If you want to send additional arguments like `delay_while_idle` or other, add them as named variables e.g.:

```
my_phone.send_message({'message':'my test message'}, delay_while_idle=True, time_to_
↪live=5)
```

Note: For more information, see Lifetime of a Message and Sending a downstream message docs.

Multicast message

`django-fcm` supports sending messages to multiple devices at once. E.g.:

```
from fcm.utils import get_device_model
Device = get_device_model()

Device.objects.all().send_message({'message':'my test message'})
```

Topic messaging

`django-fcm` supports sending messages to multiple devices that have opted in to a particular fcm topic:

```
from fcm.api import FCMMessage

FCMMessage().send({'message':'my test message'}, to='/topics/my-topic')
```

Note: For more information, see Send messages to topics.

CHAPTER 3

Signals

fcm.signals.**device_registered**
Sent when a device is registered. Provides the following arguments:

sender The resource class used to register the device.

device An instance of `fcm.models.Device` (see *Extending device model*) represents the registered device.

request The `HttpRequest` in which the device was registered.

fcm.signals.**device_unregistered**
Sent when a device is unregistered. Provides the following arguments:

sender The resource class used to unregister the device.

device An instance of `fcm.models.Device` (see *Extending device model*) represents the unregistered device.

request The `HttpRequest` in which the device was unregistered.

CHAPTER 4

Extending device model

Allows you to store additional data in the device model (e.g. foreign key to the user)

Device model

In your application, you need to create your own *Device* model. This model has to inherit from *fcm.models.AbstractDevice*.

```
# import the AbstractDevice class to inherit from
from fcm.models import AbstractDevice

class MyDevice(AbstractDevice):
    pass
```

Use your model

In the end, you have to inform *django-fcm* where it can find your model.

Add appropriate path to the `settings.py` file:

```
FCM_DEVICE_MODEL = 'your_app.MyDevice'
```

CHAPTER 5

Api key authentication

Allows you to manage access to the FCM api using one of the available `tastypie` authentication methods - *ApiKeyAuthentication*.

Note: I strongly recommend see django-tastypie Authentication docs.

Adding authentication requires *djangorestframework* added to your *INSTALLED_APPS* in the `settings.py` file:

```
INSTALLED_APPS = [
      ...
      'fcm',
      'rest_framework',
  ]
```

Adding user field

You need to extend *Device* model and add user field. (See *Extending device model*)

```
# your_app/models.py
from django.conf import settings
from django.db import models
from fcm.models import AbstractDevice

class MyDevice(AbstractDevice):

    user = models.ForeignKey(settings.AUTH_USER_MODEL)
```

Add appropriate path to the `settings.py` file:

```
FCM_DEVICE_MODEL = 'your_app.models.MyDevice'
```

Serializer class

In your application , you can create the serializer, or customize it according the extra field you have included in your *Device* model.

```
from rest_framework import serializers
from fcm.models import Device


class DeviceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Device
        fields = ('dev_id','reg_id','name','is_active')
```

View class

In your application, you need to create your view either through *ModelViewSet* or can user or override methods as specified in django-rest-framework documentation.

```
from rest_framework import viewsets
from fcm.models import Device
from fcm.serializers import DeviceSerializer


class DeviceViewSet(viewsets.ModelViewSet):
    queryset = Device.objects.all()
    serializer_class = DeviceSerializer
```

You need to hook your viewset class up in your `urls.py` file:

```
# your_app/urls.py
from django.conf.urls import url, include

from rest_framework import routers
from fcm.views import DeviceViewSet

router = routers.DefaultRouter()
router.register(r'devices', DeviceViewSet)

urlpatterns = [
    url(r'^v1/', include(router.urls))

]
```

Include your `urls.py` file in the main URL router:

```
# urls.py
from django.conf.urls import include, url

urlpatterns = [
    url(r'', include('your_app.urls')),
]
```

CHAPTER 6

Device registration endpoints

Default `django-fcm` endpoints:

- /fcm/v1/devices/
- /fcm/v1/devices/{id}/

Note: Command `python manage.py fcm_urls` returns the current endpoints.

Register

POST parameters:

`dev_id` Unique device identifier

`reg_id` Registration token

`name` Optional device name

```
curl -X POST -H "Content-Type: application/json" -d '{"dev_id": "test", "reg_id":"abcd
↪", "name":"test device"}' \
http://localhost:8000/fcm/v1/devices/
```

Unregister

POST parameters:

`dev_id` Unique device identifier

```
curl -X POST -H "Content-Type: application/json" -d '{"dev_id": "test"}' http://
↪localhost:8000/fcm/v1/devices/{id}/
```

Python Module Index

f

Index

D

F