fcm-django Documentation

Release latest

Oct 21, 2020

Contents

Django app for Firebase Cloud Messaging. Used as an unified platform for sending push notifications to mobile devices (android / ios). Developed with the help of https://github.com/olucurious/PyFCM

FCMDevice model fields

- registration_id (required - is FCM token)
- name (optional)
- active (default: true)
- user (optional)
- device_id (optional - can be used to uniquely identify devices)
- type ('android', 'web', 'ios')

Functionality:

- all necessary migrations
- model admins for django admin
- admin actions for testing single and bulk notification sending
- automatic device pruning: devices to which notifications fail to send are marked as inactive
- Django rest framework viewsets

CHAPTER 1

Setup

You can install the library directly from pypi using pip:

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

Edit your settings.py file:

```
INSTALLED_APPS = (
        ...
        "fcm_django"
)

FCM_DJANGO_SETTINGS = {
        "FCM_SERVER_KEY": "[your api key]"
}
```

Native Django migrations are in use. `manage.py migrate` will install and migrate all models.

CHAPTER 2

Messages

You can read more about different types of messages here.

In short, there are two types: notifications and data messages.

Notification:

```
{
    "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "notification" : {
      "body" : "great match!",
      "title" : "Portugal vs. Denmark",
      "icon" : "myicon"
    }
}
```

Data message:

```
{
   "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
   "data" : {
     "Nick" : "Mario",
     "body" : "great match!",
     "Room" : "PortugalVSDenmark"
   },
}
```

As in the following example, you can send either a notification, a data message, or both.

CHAPTER 3

Sending messages

```
from fcm_django.models import FCMDevice

device = FCMDevice.objects.all().first()

device.send_message("Title", "Message")
device.send_message(data={"test": "test"})
device.send_message(title="Title", body="Message", icon=..., data={"test": "test"})
```

CHAPTER 4

Sending messages in bulk

```
from fcm_django.models import FCMDevice

devices = FCMDevice.objects.all()

devices.send_message(title="Title", body="Message")
devices.send_message(title="Title", body="Message", data={"test": "test"})
devices.send_message(data={"test": "test"})
```

CHAPTER 5

Django REST Framework (DRF) support

Viewsets come in two different varieties:

- `FCMDeviceViewSet`
  - Permissions as specified in settings (`AllowAny` by default, which is not recommended)
  - A device may be registered without associating it with a user
  - Will not allow duplicate registration_id's
- `FCMDeviceAuthorizedViewSet`
  - Permissions are `IsAuthenticated` and custom permission `IsOwner`, which will only allow the `request.user` to get and update devices that belong to that user
  - Requires a user to be authenticated, so all devices will be associated with a user
  - Will allow duplicate registration_id's for different users, so you are responsible for cleanup (if that is generally perceived as undesired behaviour or if the package itself should be doing the cleanup, open an issue or email me)

Routes can be added one of two ways:

- **Routers_** (include all views)

http://www.django-rest-framework.org/tutorial/6-viewsets-and-routers#using-routers

```
from fcm_django.api.rest_framework import FCMDeviceAuthorizedViewSet

from rest_framework.routers import DefaultRouter

router = DefaultRouter()

router.register(r'devices', FCMDeviceAuthorizedViewSet)

urlpatterns = patterns('',
        # URLs will show up at <api_root>/devices
        url(r'^', include(router.urls)),
```

(continues on next page)

(continued from previous page)

```
        # ...
)
```

- Using **as_view_** (specify which views to include)

http://www.django-rest-framework.org/tutorial/6-viewsets-and-routers#binding-viewsets-to-urls-explicitly

```
from fcm_django.api.rest_framework import FCMDeviceAuthorizedViewSet

urlpatterns = patterns('',
        # Only allow creation of devices by authenticated users
        url(r'^devices?$', FCMDeviceAuthorizedViewSet.as_view({'post': 'create'}),␣
↪name='create_fcm_device'),
        # ...
)
```

CHAPTER 6

Python 3 support

`fcm-django` is fully compatible with Python 3.4 & 3.5

CHAPTER 7

Acknowledgments

https://github.com/jleclanche/django-push-notifications

CHAPTER 8

Need help, have any questions, suggestions?

Submit an issue/PR or email me at mojca.rojko@gmail.com