django-gcm Documentation

Release 1.0.9

Adam Bogdal

May 23, 2016

Contents

Django-gcm is a reusable application. It serves as a server for the GCM service and allows you to register devices and send messages to them. Contents:

CHAPTER 1

Quickstart

1. Install package via *pip*:

```
$ pip install django-gcm
```

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

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

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

   To check gcm urls just use the following command:

```
$ python manage.py gcm_urls

GCM urls:
* Register device
    /gcm/v1/device/register/
* Unregister device
    /gcm/v1/device/unregister/
```

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

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

GCM_APIKEY = "<api_key>"
```

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

CHAPTER 2

Sending messages

Using `console`:

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

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

Using `Django orm`:

```
from gcm.models 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.

2.1 Multicast message

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

```
from gcm.models import get_device_model
Device = get_device_model()

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

2.2 Topic messaging

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

```
from gcm.api import GCMMessage

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

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

CHAPTER 3

Signals

gcm.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 `gcm.models.Device` (see *Extending device model*) represents the registered device.

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

gcm.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 `gcm.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)

4.1 Device model

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

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

class MyDevice(AbstractDevice):
    pass
```

4.2 Use your model

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

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

```
GCM_DEVICE_MODEL = 'your_app.models.MyDevice'
```

CHAPTER 5

Api key authentication

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

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

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

```
INSTALLED_APPS = [
      ...
      'gcm',
      'tastypie',
  ]
```

5.1 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 gcm.models import AbstractDevice

class MyDevice(AbstractDevice):

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

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

```
GCM_DEVICE_MODEL = 'your_app.models.MyDevice'
```

5.2 Resource class

In your application, you need to create your own Resource class. It has to inherit from *gcm.resources.DeviceResource*.

```
# your_app/resources.py
from gcm.resources import DeviceResource
from tastypie.authentication import ApiKeyAuthentication
```

```
class AuthResource(DeviceResource):

    class Meta(DeviceResource.Meta):
        authentication = ApiKeyAuthentication()

    def get_queryset(self):
        qs = super(AuthResource, self).get_queryset()
        # to make sure that user can update only his own devices
        return qs.filter(user=self.request.user)

    def form_valid(self, form):
        form.instance.user = self.request.user
        return super(AuthResource, self).form_valid(form)
```

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

```
# your_app/urls.py
from django.conf.urls import include, url
from tastypie.api import Api
from .resources import AuthResource

gcm_api = Api(api_name='v1')
gcm_api.register(AuthResource())


urlpatterns = [
    url(r'^gcm/', include(gcm_api.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')),
]
```

Note: See an example project `gcm/example/apikeyauth_project`

CHAPTER 6

Device registration endpoints

Default `django-gcm` endpoints:

- /gcm/v1/device/register/
- /gcm/v1/device/unregister/

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

6.1 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
http://localhost:8000/gcm/v1/device/register/
```

6.2 Unregister

POST parameters:

`dev_id` Unique device identifier

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

g

D

G