Django Google Drive Storage Documentation

Release 1.1.4

Gian Luca Dalla Torre

June 19, 2016

Contents

1 Prerequisites 3

2 Installation 5

3 Setup 7

4 Use 9

5 Source and License 11

Django Google Drive Storage is a Django Storage implementation that uses Google Drive as a backend for storing data.

Please take note that with **this implementation you could not save or load data from a user's Drive**. You can use only a Drive **dedicated to a Google Project**. This means that:

- this storage interacts with Google Drive as a Google Project, not a Google User.
- your project can use Google Drive only through Google Drive SDK. Because no user is associated with this Drive, **you cannot use Google Drive User Interface**.
- this storage authenticates with Google using public private keys. See *prerequisites* for how to obtain it.

Having stated that, with this storage you gain a 15GB space hosted on Google Server where you are able to store data using Django models.

CHAPTER 1

Prerequisites

To use this storage, you have to:

- set up a project and application in the Google Developers Console
- obtain the key (service account) for your Google Project associated with Google Drive service

CHAPTER 2

Installation

This storage is hosted on PyPI. It can be easily installed through *pip*:

```
pip install django-googledrive-storage
```

CHAPTER 3

Setup

Once installed, there are a few steps to configure the storage:

- add the module *gdstorage* to your installed apps in your *settings.py* file:

```
INSTALLED_APPS = (
    ...,
    'django.contrib.staticfiles',
    'gdstorage'
)
```

- create a section in your *setting.py* that contains the configuration for this storage:

```
#
# Google Drive Storage Settings
#

GOOGLE_DRIVE_STORAGE_KEY = '<your private key as a string>'
GOOGLE_DRIVE_STORAGE_SERVICE_EMAIL = '<your service mail>'
```

The *GOOGLE_DRIVE_STORAGE_SERVICE_EMAIL* should be the email Google assigned to your project, while the *GOOGLE_DRIVE_STORAGE_KEY* must be the *private key as a string* obtained by Google.

Since Google will provide you an *encrypted p12* file, to obtain the private key you can use the following command:

```
openssl pkcs12 -in <p12_file> -out <key_file> -nocerts -nodes
```

You also have to **provide the password that Google shows you when the p12 certificate has been created** [1].

Note: **Django Google Drive Storage** is using Django Appconf to handle settings, so you can setup *GOOGLE_DRIVE_STORAGE_KEY* as an environment variable outside the Django app.

This will increase security to your environment.

Thanks to Johannes Hoppe for his contribution

- instantiate the storage on you *models.py* file before using into the models:

```
from gdstorage.storage import GoogleDriveStorage

# Define Google Drive Storage
gd_storage = GoogleDriveStorage()
```

[1] Usually the password used by Google is `notasecret` but it will eventually changes in future

CHAPTER 4

Use

Once configured, it can be used as storage space associated with Django:

```
class Map(models.Model):
    id = models.AutoField( primary_key=True)
    map_name = models.CharField(max_length=200)
    map_data = models.FileField(upload_to='/maps', storage=gd_storage)
```

CHAPTER 5

Source and License

Source can be found on GitHub with its included license.