Django Google Drive Storage Documentation

Release 1.3.5

Gian Luca Dalla Torre

Sep 30, 2018

Contents

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 json private key file (OAuth 2.0 for Server to Server Applications) 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_JSON_KEY_FILE = '<path to your json private key file>'
```

The *GOOGLE_DRIVE_STORAGE_JSON_KEY_FILE* must be the path to *private json key file* obtained by Google.

Note: **Django Google Drive Storage** is using Django Appconf to handle settings, so you can setup *GOOGLE_DRIVE_STORAGE_JSON_KEY_FILE* 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()
```

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

File permissions

Using the storage this way, all files will be saved as publicly available for read (which is the most common use case), but sometimes you could have different reason to use Google Storage.

It is possible to specify a set of file permissions[1] to change how the file could be read or written.

This code block will assign read only capabilities only to the user identified by *foo@mailinator.com*.

```
from gdstorage.storage import GoogleDriveStorage, GoogleDrivePermissionType,␣
↪GoogleDrivePermissionRole, GoogleDriveFilePermission

permission =  GoogleDriveFilePermission(
   GoogleDrivePermissionRole.READER,
   GoogleDrivePermissionType.USER,
   "foo@mailinator.com"
)

gd_storage = GoogleDriveStorage(permissions=(permission, ))

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)
```

Note: Thanks to Anna Sirota for her contribution

[1] A detailed explanation of Google Drive API permission can be found here.

CHAPTER 6

Source and License

Source can be found on GitHub with its included license.