Django Google Sheets Import

Release 0.0.1

Alexander Helmboldt

Nov 08, 2021

CONTENTS:

The `django-gsheets-import` Python package is a Django application to facilitate data import from Google Sheets within Django's admin framework. It extends the great django-import-export package, which already provides import and export capabilities for all local file formats supported by tablib. Exporting data from Django's admin to Google Sheets is currently not supported by `django-gsheets-import`, but planned for a future release.

CHAPTER

ONE

GETTING STARTED

1.1 Installation and configuration

The package and its dependencies can be installed from PyPI via `pip install -U django-gsheets-import`. To use the package in your Django project, just add `import_export` and `gsheets_import` to the list of **installed apps** in your `settings.py`, i.e.

```
## in settings.py
INSTALLED_APPS = [
    ...,
    'import_export',
    'gsheets_import',
    ...
]
```

In order for `django-gsheets-import` to work properly, it needs to be associated with an underlying **Google Cloud Project** (GCP). How to properly set up an appropriate GCP using the Google Cloud Console is described in more detail in the corresponding *section*. At this point, let us just note that all of the services needed are available in Google Cloud's Free Tier, so that there is no need to set up a billing account. Assuming that a suitable GCP already exists, go to the Google Cloud Console and navigate to `APIs & Services > Credentials`. From there, copy an **API key**, as well as the desired **OAuth Client ID** and add them to your `settings.py`. The required **project number** can be found under `IAM & Admin > Settings` and must also be added to `settings.py`, i.e.

```
## in settings.py
GSHEETS_IMPORT_API_KEY = '<Your API developers key>'
GSHEETS_IMPORT_CLIENT_ID = '<Your OAuth Client ID>'
GSHEETS_IMPORT_APP_ID = '<Your project number>'
```

The package is now ready to be used with your Django project.

1.2 Features and usage

The `django-gsheets-import` package presented here strongly relies on the functionality provided by the `django-import-export` package. It extends that package by the option to allow the user to import data from their Google Sheets via the Django admin. The usage of `django-gsheets-import` is very similar to that of `django-import-export`, which is nicely documented here. It might also be instructive to have a look at the example Django project that ships with `django-gsheets-import` (see *here* for more details).

In short, integrating the Google Sheets import feature offered by `django-gsheets-import` into your Django project's admin site is a two-step process:

- Define a **resource** which determines how the fields of a given model translate to their import (and export) representations.
- Define the **admin interface** of the considered model as a subclass of `ImportGoogleModelAdmin` or any of the other classes provided by the package's `admin` submodule, namely `ImportGoogleMixin`, `ImportGoogleExportModelAdmin`, and `ImportGoogleExportMixin`.

CHAPTER

TWO

SETTING UP A GOOGLE CLOUD PROJECT

In order to facilitate the interaction of the `django-gsheets-import` package with the Google Cloud, one generally proceeds in three steps, specifically:

1. Create a **Google Cloud Project**.
2. Enable the required Google **APIs**, namely the Sheets API as well as the Picker API.
3. Create and download the required **keys and identifiers** related to those APIs and the user authentication workflow.

In the following, we will give step-by-step instructions to complete all of these tasks.

2.1 Create a new Google Cloud Project

- Go to the Google Cloud Console at https://console.cloud.google.com and sign in with the relevant Google account (e.g. a simple Gmail account).
- In the top navigation bar, click on `Select a project` (if you haven't created a project before), or on the name of the currently selected project or organization.
- Click on `NEW PROJECT` in the following dialog box.
- Choose a project name, an organization, and a corresponding location. Confirm your choices by a click on the `CREATE` button.
- Note that the project does *not* need to be linked to a billing account, see `Main Menu > Billing`.

2.2 Enable APIs

- Navigate to `Main Menu > APIs & Services > Library` and select the API you want to enable.
- For the `django-gsheets-import` package to work properly, you need to enable the **Google Sheets API** as well as the **Google Picker API**.
- After selecting an API from the library and clicking on the `ENABLE` button, you are redirected to an overview page for this API. You can later come back to this page by going to `Main Menu > APIs & Services > Dashboard` and then selecting the API of interest from the table on the bottom.
- The aforementioned table lists all of the APIs that are currently enabled for your project. This includes several APIs that are enabled by default (cf. here in the official documentation), but are not needed for our purposes. It may be wise to disable all of the APIs that are not explicitly needed. At least the `django-gsheets-import` package still works with all but the Sheets and Picker APIs disabled.

2.3 Obtain credentials

- The use of the Google Picker API requires the creation of an **API key**.
  - Navigate to `Main Menu > APIs & Services > Credentials` and click on `CREATE CREDENTIALS` at the top.
  - Restrictions for the newly created API key do not have to be added for the package to work, but should still be implemented for security reasons.
    * Under `Application restrictions`, select `HTTP referrers (websites)` and add an appropriate URL under `Website restrictions`. Note that this can be skipped during local development and testing.
    * Under `API restrictions`, choose `Restrict key` and select both the Google Picker API and the Google Sheets API from the drop-down list.
- The implementation of a proper authentication and authorization workflow requires the creation of **OAuth credentials**. Obtaining those is a two-step process: First, we need to configure the OAuth consent screen. Second, we need to create an appropriate OAuth 2.0 client ID.
  1. To configure the **consent screen**, go to `Main Menu > APIs & Services > OAuth consent screen`.
     - As `User Type` you typically want to choose `External`. Click on `CREATE` and fill out the needed information.
     - For `django-gsheets-import` to work properly, you need to add the (non-sensitive) `.../auth/drive.file` scope connected to the Google Sheets API in the next step.
     - Add the email addresses of one or more test users with a valid Google account.
     - To eventually remove the restriction on the number of (test) users, you may want to have your app verified by Google. For more information on the verification process, see here, while details on unverified apps can be found here.
  2. To create an **OAuth 2.0 client ID**, go to `Main Menu > APIs & Services > Credentials` and click on `CREATE CREDENTIALS` at the top.
     - Select `Web application` as `Application type`.
     - Set the `Authorised JavaScript Origin` to `<Domain>`, where `<Domain>` is typically `http://localhost:8000` for local testing with the Django development server, or your deployment domain under which your web application is reachable. You can also add multiple relevant URIs here.
- Accessing the selected Google Sheet while only using the non-sensitive `.../auth/drive.file` scope requires the project's **App ID** to be set. It is automatically created with each Google Cloud Project and can be found as `Project number` on your project's dashboard or under the same name at `Main Menu > IAM & Admin > Settings`.

CHAPTER

THREE

THE DEMO APPLICATION

In order to demonstrate the Google Sheets import feature provided by the `django-gsheets-import` package, the code ships with a small Django project whose admin interface uses the import functionality. It can be found in the `tests/testapp/` subfolder. In the following, we briefly sketch how to run the demo project and use it for testing the import feature.

3.1 Setting up the Django project

Assuming that Python and pip have already been installed globally or in an appropriate virtual environment (recommended), the demo project can be set up by following the steps listed below.

```
## go to the project folder
cd tests/testapp/

## install the required dependencies
pip install -r requirements.txt

## prepare the database
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
python manage.py loaddata authors works

## run the development server
python manage.py runserver
```

3.2 Getting the demo sheet

We have prepared a read-only sample Google Sheet called **GSheets Import Demo**, which is publicly available here. It contains two tables in two subsheets, one appropriate for each model in the demo project. In order to use the sample sheet, click on the above link and sign in with your favorite Google Account (if you haven't done so already). The demo sheet should then automatically be available from that account's Google Drive.

3.3 Setting up a Google Cloud Project

As previously mentioned, the interaction between Django and the user's Google Drive is facilitated by an underlying Google Cloud Project (GCP). For now, you will have to create such a project yourself in order to use it with the demo application. This is possible with every standard Google Account, does not incur any additional costs and should not take more than a couple of minutes. The exact steps to create and set up a GCP for use with `django-gsheets-import` are outlined *here*. Once the GCP is ready, you need to retrieve your project's API key, OAuth client ID, and project number and add them to the demo application's configuration file, specifically

```
## in tests/testapp/settings.py
GSHEETS_IMPORT_API_KEY = '<Your API developers key>'
GSHEETS_IMPORT_CLIENT_ID = '<Your OAuth Client ID>'
GSHEETS_IMPORT_APP_ID = '<Your project number>'
```

3.4 Running the demo app

Testing the import feature using the demo app typically amounts to the following steps:

- Navigate to `http://localhost:8000` in your browser and sign in as the Django project's superuser you created above.
- Both of the models in the project's `literature` app were supplemented by the Google Sheets import functionality. Choose one of the models from the sidebar, which brings you to the admin's changelist view. Here, click on the `IMPORT` button in the top right corner.
- The `Google Sheet` format is already pre-selected, so click on the `Select a file...` button.
- From the pop-up window, select the same Google Account which you used to access the sample sheet above.
- Grant your previously created Google Cloud Project the necessary rights when prompted.
- Select the "GSheets Import Demo" sheet from the Google Picker window.
- Make sure to select the subsheet appropriate for the current model from the corresponding drop-down list.
- Click on the `SUBMIT` button.
- If you like what you see, click on the `CONFIRM IMPORT` button to have the displayed data added to the underlying database.

3.5 Further information on the demo app

Here, we compile a few more details on the demo project and its implementation.

- As mentioned above, the import functionality was added to both of the models in our demo app. Correspondingly, if you have a look at `literature/admin.py`, you will find that both the `AuthorAdmin` class and the `WorkAdmin` class inherit from `ImportGoogleModelAdmin`.
- Otherwise, the main work is in implementing an appropriate import resource class for each model, which is done in `literature/resources.py`. A minimal implementation is used for the `AuthorResource` class, while a bit more customization was performed in writing the `WorkResource` class. Much more on import resources can be found in the documentation of the `django-import-export` package.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search