Django-Openhumans Documentation

Tarannum Khan

Jul 23, 2020

Contents:

`django-open-humans` is a self-contained application for the Django web framework.

This app includes the models & views needed to do social logins through Open Humans as well as capabilities to message members and access/delete/upload files for them. It manages the *Oauth2* procedure and token refreshes.

CHAPTER 1

Introduction

This application implements a *OAuth2* login flow that creates regular *Django* `User` objects with an associated `OpenHumansMember` object to interface with Open Humans. It furthermore offers a variety of methods to interact with those members and their data.

This project is it's own small *Django* application that you can re-use in your own larger *Django* project. Under the hood it is using the Python library open-humans-api to interface with the APIs of *Open Humans*.

1.1 Features

- Allow users to login your app with an Open Humans account
- Handles all of the *OAuth2* workflow for you
- Manage your project's files on Open Humans
  - Upload files
  - Delete files
  - Access existing files
- Message users
- Provide a hook to automatically delete data from members that de-authorize you on Open Humans

1.2 Example use cases

This library has been used in a variety of projects that interface with Open Humans. Here are some examples:

- Personal API
- Fitbit Intraday Access
- Google Fit Integration

1.3 Getting in touch

Our code is on GitHub and we are always happy about code & documentation contributions as well as bug reports! To get in touch more directly you can join us on Slack!

CHAPTER 2

Getting Started

This guide assumes you are using `Django` as your web framework, want to start a new project and are using pipenv to manage your dependencies.

The guide is written to work for most UNIX systems.

2.1 Installing `django-open-humans`

Let's start by installing `pipenv` if you haven't done so before:

```
pip install pipenv
```

Now we can install `Django` along with `django-open-humans` in a virtual environment:

```
mkdir our_project
cd our_project
pipenv install django
pipenv install django-open-humans
pipenv shell
```

This installs `django` and `django-open-humans` in our new project `our_project` and starts the new virtual environment. With this we start a new django project:

```
django-admin startproject our_project .
```

2.2 Setting up `django-open-humans`

2.2.1 Set up `settings.py`

Now that we have installed everything we can get started to add `django-open-humans` to the `INSTALLED_APPS` of our Django project:

Listing 1: our_project/our_project/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'openhumans'
]
```

Before we can run make and run the migrations we need to add some environment variables that `django-open-humans` needs to properly work. At a minimum we need to set up the `OPENHUMANS_APP_BASE_URL`, `OPENHUMANS_CLIENT_ID` and `OPENHUMANS_CLIENT_SECRET`. To do so we add a few lines to our `settings.py`:

Listing 2: our_project/our_project/settings.py

```
import os

OPENHUMANS_APP_BASE_URL = os.getenv('OPENHUMANS_APP_BASE_URL', 'http://localhost:5000
↪')
OPENHUMANS_CLIENT_ID = os.getenv('OPENHUMANS_CLIENT_ID', 'your_client_id')
OPENHUMANS_CLIENT_SECRET = os.getenv('OPENHUMANS_CLIENT_SECRET', 'your_client_secret')
```

This code reads the `OPENHUMANS_APP_BASE_URL`, `OPENHUMANS_CLIENT_ID` and `OPENHUMANS_CLIENT_SECRET` from the correspondingly named environment variables you should set using e.g. `export OPENHUMANS_CLIENT_SECRET='my_client_secret'`.

To get your own `client_id` and `client_secret` you can head to Open Humans and make your own OAuth2 data request project.

2.2.2 Set up `urls.py`

To add django-open-humans URLs to your project, update your root `urls.py` configuration file (i.e. the file specified by `settings.ROOT_URLCONF`) as follows:

Listing 3: our_project/our_project/urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('main.urls')),
    path('admin/', admin.site.urls),
]

urlpatterns += [
    path('openhumans/', include('openhumans.urls')),
]
```

2.2.3 Run database migrations

Now we can migrate our tables. Those migrations will create the `User` and `OpenHumansMember` tables for us:

```
./manage.py migrate
```

And that's all to get the basic configuration and integration into your Django project done.

2.3 Setting up your Open Humans project

For the login with *Open Humans* to work you need to correctly configure the `REDIRECT_URL` of the OAuth2 process on Open Humans. The URL path that `django-open-humans` creates for redirects is

```
/openhumans/complete
```

This means if you want to develop locally and your `OPENHUMANS_APP_BASE_URL` is `http://localhost:5000`, your *Redirect URL* should be `http://localhost:5000/openhumans/complete`.

Similarly, there is a deauthorization hook that you can setup on *Open Humans* which will automatically inform you when people have de-authorized your application. `django-open-humans` accepts deauthorization requests at `/openhumans/deauth`.

CHAPTER 3

Interacting with Members

For each user that has logged in through their *Open Humans* account you will get an object of the `OpenHumansMember` class, allowing you to interact with them through the Open Humans API & platform. Below are some examples of how you can use this app to interact with your Open Humans members.

More information can be found in the *model.py section*.

3.1 Accessing an `OpenHumansMember` object

Each `OpenHumansMember` object is associated with a regular *Django* `User` object. Through the Django shell this looks like this:

Listing 1: ./manage.py shell

```
In [1]: from django.contrib.auth import get_user_model

In [2]: User = get_user_model()

In [3]: single_user = User.objects.all()[0] # get a single user

In [4]: print(single_user.openhumansmember)
<OpenHumansMember(oh_id='12341337')>
```

Due to this you can also easily access each `OpenHumansMember` inside your views from the `request` object:

Listing 2: views.py

```
def my_view(request):
  open_humans_member = request.user.openhumansmember
```

3.2 Accessing files for an `OpenHumansMember` object

The `OpenHumansMember` objects have a class method to list and access the files for them:

Listing 3: ./manage.py shell

```
In [1]: from openhumans.models import OpenHumansMember

In [2]: oh_member = OpenHumansMember.objects.all()[0]

In [3]: print(oh_member.list_files())
[{'id': 1234, 'basename': 'my_file.json', 'created': '2018-11-23T17:28:49.114250Z',
↪'download_url': 'https://example.com/my_file.json', 'metadata': {'tags': ['json',
↪'data', 'foo'], 'description': 'an example file'}, 'source': 'direct-sharing-1337'}]
```

The `OpenHumansMember.list_files()` function returns a list of dictionaries, containing metadata for the files available as well as the download link for each file.

3.3 Deleting files for an `OpenHumansMember` object

The `OpenHumansMember` objects have two class methods to delete individual files or multiple files: `OpenHumansMember.delete_all_files()` and `OpenHumansMember.delete_single_file()`.

Warning: If you use the filename to decide which files to delete with the ‘‘delete_single_file‘‘ function it will delete all files with a given name if there is more than one!

Listing 4: ./manage.py shell

```
In [1]: from openhumans.models import OpenHumansMember

In [2]: oh_member = OpenHumansMember.objects.all()[0]

### Delete a single file by file-ID (see accessing files)

In [3]: oh_member.delete_single_file(file_id=1234)

### Delete one or multiple files by file-name.

In [4]: oh_member.delete_single_file(file_basename='my_file.json')

### Delete all files of a member

In [5]: oh_member.delete_all_files()
```

3.4 Uploading files for an `OpenHumansMember` object

To upload files you can use the `OpenHumansMember.upload()` function. For this you will have to provide an open stream (text or binary), a file name that should be used and some meta data.

Listing 5: views.py

```
def upload_file(request):
  oh_member = request.user.openhumansmember

  with open('example_file.json', 'r') as f:
    oh_member.upload(
      f,
      'example.json',
      metadata = {'description': 'foo', 'tags': ['test', 'example']}
    )
```

CHAPTER 4

Configuration Details

The *Setting up django-open-humans* section explains a minimal configuration of `django-open-humans`.

This page lists the existing environment variables you can use to configure your installation.

4.1 Environment variables

4.1.1 `OPENHUMANS_APP_BASE_URL` (required)

The `OPENHUMANS_APP_BASE_URL` describes the base URL of where your app lives. It is needed to correctly setup the `redirect_uri` for the OAuth2 handshakes.

4.1.2 `OPENHUMANS_CLIENT_ID` (required)

You need to set the `OPENHUMANS_CLIENT_ID` to the `CLIENT_ID` you get from Open Humans to enable the OAuth2 flow.

4.1.3 `OPENHUMANS_CLIENT_SECRET` (required)

You need to set the `OPENHUMANS_CLIENT_SECRET` to the `CLIENT_SECRET` you get from Open Humans to enable the OAuth2 flow.

4.1.4 `OPENHUMANS_OH_BASE_URL`

By setting `OPENHUMANS_OH_BASE_URL` you can change the base URL used for all API calls to Open Humans. The default is `https://www.openhumans.org` and there is no reason to change it, unless you run your own Open Humans fork at a different URL.

4.1.5 OPENHUMANS_LOGIN_REDIRECT_URL

Specifies where a user should be redirected to after they have logged in into your app with their Open Humans account. By default `OPENHUMANS_LOGIN_REDIRECT_URL` should link to `/`.

4.1.6 OPENHUMANS_LOGOUT_REDIRECT_URL

Specifies where a user should be redirected to after they have logged out of your app with their Open Humans account. By default `OPENHUMANS_LOGOUT_REDIRECT_URL` should link to `/`.

4.1.7 OPENHUMANS_DEAUTH_ON_DELETE

If an OpenHumansMember object is deleted (e.g. via "delete account"), send a POST to the Open Humans API to "withdraw" (deauthorize) this member of the activity on the Open Humans site.

Default: `True`.

4.1.8 OPENHUMANS_DELETE_ON_ERASURE

If you have set up the deauthorization hook on Open Humans the `OPENHUMANS_DELETE_ON_ERASURE` will specify how an incoming request from this hook will be processed. If set to `True` an `OpenHumansMember` object will be only deleted if the user requested to do so.

By setting this option to `False` member objects will not be deleted, even if they requested it.

Default: `True`.

4.1.9 OPENHUMANS_DELETE_ON_DEAUTH

If you have set up the deauthorization hook on Open Humans the `OPENHUMANS_DELETE_ON_DEAUTH` will specify how an incoming request from this hook will be processed. If set to `True` an `OpenHumansMember` object will always be deleted, even if the member did not request this deletion.

Default: `False`.

4.1.10 OPENHUMANS_WEBHOOK_SECRET

If you have set a "webhook secret" for your activity, set this to that string to verify incoming requests are from Open Humans. If this is set and the expected verification header (`X-Openhumans-Webhooks-Signature`) is not present or invalid, the request will result in a PermissionDenied error. If this is NOT set, this header (if included) will NOT be checked.

Default: `None`.

CHAPTER 5

views.py

class openhumans.views.**DeleteAllFiles**(***kwargs*)

post(*request*)
Delete all project files in Open Humans for this project member.

class openhumans.views.**DeleteFile**(***kwargs*)

post(*request*)
Delete specified file in Open Humans for this project member.

openhumans.views.**complete**(*request*)
Receive user from Open Humans. Store data, start data upload task.

openhumans.views.**deauth**(*request*)
Receive and act on deauthorization notification from Open Humans.

class openhumans.views.**list_files**(***kwargs*)

get(*request*)
List files.

CHAPTER 6

models.py

class openhumans.models.**OpenHumansMember**(**args*, ***kwargs*)

Data storage, auth, etc. related to an Open Humans member account.

exception DoesNotExist

exception MultipleObjectsReturned

classmethod create(*oh_id*, *data*, *user=None*)

Create an Open Humans member, and corresponding User, if not provided.

Parameters

- **oh_id** – This field is the Openhumans id.
- **data** – This field contain data related to access token and refresh token.
- **user** – This field's default value is None.

delete_single_file(*file_id=None*, *file_basename=None*)

Deletes a file. Specify file_id or file_basename but not both.

Parameters

- **file_id** – This field is the file id of the file to be deleted.
- **file_basename** – This field is the file basename of the file to be deleted. It's default value is None.

get_access_token(*client_id='OPENHUMANS_CLIENT_ID'*, *client_secret='OPENHUMANS_CLIENT_SECRET'*)

Return access token. Refresh first if necessary.

Parameters

- **client_id** – This field is the client_id of the project. Project info can be found at https://www.openhumans.org/direct-sharing/projects/manage/
- **client_secret** – This field is the client secret of the project. Project info can be found at https://www.openhumans.org/direct-sharing/projects/manage/

static get_auth_url()
Gets the authentication url.

classmethod get_create_member(*data*, *user=None*)
use the data returned by *ohapi.api.oauth2_token_exchange* and return an oh_member object

list_files()
List files.

message(*subject*, *message*)
Send messages.

Parameters

- **subject** – This field is the subject of the message.
- **message** – This field is the message which is to be send.

classmethod oh_code_to_member(*code*)
Exchange code for token, use this to create and return OpenHumansMember. If a matching OpenHumansMember already exists in db, update and return it.

upload(*stream*, *filename*, *metadata*, *file_identifier=None*)
Upload file to Open Humans.

openhumans.models.**make_unique_username**(*base*)
Ensure a unique username. Probably this never actually gets used.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

Index