The `spotipy` Package

The `spotipy` package provides an interface into the Spotify API.

References

The Spotify API:

- https://developer.spotify.com/documentation/web-api/
- https://developer.spotify.com/documentation/general/guides/authorization-guide
- https://developer.spotify.com/documentation/general/guides/scopes/

The `spotipy` Package:

- https://github.com/plamere/spotipy
- https://github.com/plamere/spotipy#quick-start
- https://spotipy.readthedocs.io/en/latest/
- https://spotipy.readthedocs.io/en/latest/#client-credentials-flow
- https://spotipy.readthedocs.io/en/latest/#authorization-code-flow

Installation

First install the package, if necessary:

```
pip install spotipy
```

Setup

Create a Spotify API Client application, note its credentials, then store them in environment variables called `SPOTIPY_CLIENT_ID` and `SPOTIPY_CLIENT_SECRET`, respectively.

If your app doesn't need to make authenticated requests on behalf of the user, this setup should be sufficient. Feel free to move on to the "Basic Usage" example below.

User Authentication

However, if your app does need to make authenticated requests on behalf of the user, there are a few more setup steps.

First, in your client application's developer settings, you'll need to specify a redirect URL. If you're not sure which URL to designate, feel free to choose your GitHub repository URL, or a local URL like `http://localhost:5000/auth/spotify/callback`. Store the value in an environment variable called `SPOTIPY_REDIRECT_URL`.

Note your Spotify username, and store in an environment varible called `SPOTIFY_USERNAME`.

After setting the redirect URL and username, setup the code provided in the "Authenticated Usage" section below, and invoke the provided `get_token()` function. This process will download a file called `.cache-USERNAME` to the root directory of the repo. It looks like this:

```
{
    "access_token": "_____",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_token": "______",
    "scope": "playlist-read-private",
    "expires_at": 1554651631
}
```

Since this file contains secret credentials, ignore it from version control by adding an entry like `.cache-*` to your repository's ".gitignore" file.

Subsequent requests (see the provided `get_playlists()` function below) will use the auth token from this credentials file to avoid additional user logins.

Usage

Basic Usage

Example request which doesn't require authentication (i.e. a song search):

```
# adapted from: https://github.com/s2t2/my-spotify-app-
py/blob/master/list_songs.py

from dotenv import load_dotenv
import os

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

load_dotenv() # load environment variables

print("CLIENT ID:", os.environ.get("SPOTIPY_CLIENT_ID")) # env var used
implicitly by the spotipy package
print("CLIENT SECRET:", os.environ.get("SPOTIPY_CLIENT_SECRET"))  # env var
used implicitly by the spotipy package

# FYI, this client configuration approach expects / implicitly uses env vars
named SPOTIPY_CLIENT_ID and SPOTIPY_CLIENT_SECRET
client_credentials_manager = SpotifyClientCredentials()
```

```
client =
spotipy.Spotify(client_credentials_manager=client_credentials_manager)

response = client.search(q="Springsteen on Broadway", limit=20)

for i, track in enumerate(response['tracks']['items']):
    print(' ', i, track['name'])
```

Authenticated Usage

Example request which requires authentication (i.e. listing a user's playlists):

```
# adapted from: https://github.com/s2t2/playlist-service-
py/blob/4db80cd3f8f2ed018f64bc2a97629d2af105acc3/app/spotify_service.py

import os
from dotenv import load_dotenv

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import spotipy.util as util

load_dotenv() # load environment variables

print("CLIENT ID:", os.environ.get("SPOTIPY_CLIENT_ID")) # env var used
implicitly by the spotipy package
print("CLIENT SECRET:", os.environ.get("SPOTIPY_CLIENT_SECRET")) # env var
used implicitly by the spotipy package
print("REDIRECT URL:", os.environ.get("SPOTIPY_REDIRECT_URI")) # env var used
implicitly by the spotipy package
USERNAME = os.environ.get("SPOTIFY_USERNAME", "OOPS")

# requires user interaction
def get_token():
    AUTH_SCOPE = "playlist-read-private"
    token = util.prompt_for_user_token(USERNAME, AUTH_SCOPE)
    return token

# requires user auth
def get_playlists():
    token = get_token()
    client = spotipy.Spotify(auth=token)

    response = client.current_user_playlists()

    for i, playlist in enumerate(response['items']):
        print(f"{i + 1 + response['offset']} {playlist['uri']}
{playlist['name']}")
```

```python
if __name__ == "__main__":

    get_playlists()
```