google-auth-oauthlib Documentation

Release 0.4.1

Google, Inc.

Nov 04, 2019

Contents

CHAPTER 1

google_auth_oauthlib

1.1 google_auth_oauthlib package

oauthlib integration for Google Auth

This library provides oauthlib integration with google-auth.

get_user_credentials(*scopes*, *client_id*, *client_secret*)

Gets credentials associated with your Google user account.

This function authenticates using your user credentials by going through the OAuth 2.0 flow. You'll open a browser window to authenticate to your Google account. The permissions it requests correspond to the scopes you've provided.

To obtain the `client_id` and `client_secret`, create an **OAuth client ID** with application type **Other** from the Credentials page on the Google Developer's Console. Learn more with the Authenticating as an end user guide.

Parameters

- **scopes** (`Sequence [ str ]`) – A list of scopes to use when authenticating to Google APIs. See the list of OAuth 2.0 scopes for Google APIs.
- **client_id** (*str*) – A string that identifies your application to Google APIs. Find this value in the Credentials page on the Google Developer's Console.
- **client_secret** (*str*) – A string that verifies your application to Google APIs. Find this value in the Credentials page on the Google Developer's Console.

Returns The OAuth 2.0 credentials for the user.

Return type google.oauth2.credentials.Credentials

Examples

Get credentials for your user account and use them to run a query with BigQuery:

```
import google_auth_oauthlib

# TODO: Create a client ID for your project.
client_id = "YOUR-CLIENT-ID.apps.googleusercontent.com"
client_secret = "abc_ThIsIsAsEcReT"

# TODO: Choose the needed scopes for your applications.
scopes = ["https://www.googleapis.com/auth/cloud-platform"]

credentials = google_auth_oauthlib.get_user_credentials(
    scopes, client_id, client_secret
)

# 1. Open the link.
# 2. Authorize the application to have access to your account.
# 3. Copy and paste the authorization code to the prompt.

# Use the credentials to construct a client for Google APIs.
from google.cloud import bigquery

bigquery_client = bigquery.Client(
    credentials=credentials, project="your-project-id"
)
print(list(bigquery_client.query("SELECT 1").result()))
```

1.1.1 Subpackages

google_auth_oauthlib.tool package

1.1.2 Submodules

google_auth_oauthlib.flow module

OAuth 2.0 Authorization Flow

This module provides integration with requests-oauthlib for running the OAuth 2.0 Authorization Flow and acquiring user credentials.

Here's an example of using *Flow* with the installed application authorization flow:

```
from google_auth_oauthlib.flow import Flow

# Create the flow using the client secrets file from the Google API
# Console.
flow = Flow.from_client_secrets_file(
    'path/to/client_secrets.json',
    scopes=['profile', 'email'],
    redirect_uri='urn:ietf:wg:oauth:2.0:oob')

# Tell the user to go to the authorization URL.
auth_url, _ = flow.authorization_url(prompt='consent')

print('Please go to this URL: {}'.format(auth_url))

# The user will get an authorization code. This code is used to get the
```

(continues on next page)

(continued from previous page)

```
# access token.
code = input('Enter the authorization code: ')
flow.fetch_token(code=code)

# You can use flow.credentials, or you can just get a requests session
# using flow.authorized_session.
session = flow.authorized_session()
print(session.get('https://www.googleapis.com/userinfo/v2/me').json())
```

This particular flow can be handled entirely by using *InstalledAppFlow*.

class Flow(*oauth2session*, *client_type*, *client_config*, *redirect_uri=None*, *code_verifier=None*, *autogenerate_code_verifier=False*)

Bases: object

OAuth 2.0 Authorization Flow

This class uses a requests_oauthlib.OAuth2Session instance at *oauth2session* to perform all of the OAuth 2.0 logic. This class just provides convenience methods and sane defaults for doing Google's particular flavors of OAuth 2.0.

Typically you'll construct an instance of this flow using *from_client_secrets_file()* and a client secrets file obtained from the Google API Console.

Parameters

- **oauth2session** (*requests_oauthlib.OAuth2Session*) – The OAuth 2.0 session from requests-oauthlib.
- **client_type** (*str*) – The client type, either web or installed.
- **client_config** (Mapping [str, Any]) – The client configuration in the Google client secrets format.
- **redirect_uri** (*str*) – The OAuth 2.0 redirect URI if known at flow creation time. Otherwise, it will need to be set using *redirect_uri*.
- **code_verifier** (*str*) – random string of 43-128 chars used to verify the key exchange.using PKCE.
- **autogenerate_code_verifier** (*bool*) – If true, auto-generate a code_verifier.

client_type = None

The client type, either 'web' or 'installed'

Type str

client_config = None

The OAuth 2.0 client configuration.

Type Mapping [str, Any]

oauth2session = None

The OAuth 2.0 session.

Type requests_oauthlib.OAuth2Session

classmethod from_client_config(*client_config*, *scopes*, ***kwargs*)

Creates a requests_oauthlib.OAuth2Session from client configuration loaded from a Google-format client secrets file.

Parameters

- **client_config** (Mapping [str, Any]) – The client configuration in the Google client secrets format.
- **scopes** (Sequence [str]) – The list of scopes to request during the flow.
- **kwargs** – Any additional parameters passed to requests_oauthlib.OAuth2Session

Returns The constructed Flow instance.

Return type *Flow*

Raises ValueError – If the client configuration is not in the correct format.

classmethod from_client_secrets_file(*client_secrets_file*, *scopes*, ***kwargs*)
Creates a *Flow* instance from a Google client secrets file.

Parameters

- **client_secrets_file** (*str*) – The path to the client secrets .json file.
- **scopes** (Sequence [str]) – The list of scopes to request during the flow.
- **kwargs** – Any additional parameters passed to requests_oauthlib.OAuth2Session

Returns The constructed Flow instance.

Return type *Flow*

redirect_uri
The OAuth 2.0 redirect URI. Pass-through to `self.oauth2session.redirect_uri`.

authorization_url(***kwargs*)
Generates an authorization URL.

This is the first step in the OAuth 2.0 Authorization Flow. The user's browser should be redirected to the returned URL.

This method calls requests_oauthlib.OAuth2Session.authorization_url() and specifies the client configuration's authorization URI (usually Google's authorization server) and specifies that "offline" access is desired. This is required in order to obtain a refresh token.

Parameters **kwargs** – Additional arguments passed through to requests_oauthlib.OAuth2Session.authorization_url()

Returns

The generated authorization URL and state. The user must visit the URL to complete the flow. The state is used when completing the flow to verify that the request originated from your application. If your application is using a different *Flow* instance to obtain the token, you will need to specify the `state` when constructing the *Flow*.

Return type Tuple [str, str]

fetch_token(***kwargs*)
Completes the Authorization Flow and obtains an access token.

This is the final step in the OAuth 2.0 Authorization Flow. This is called after the user consents.

This method calls requests_oauthlib.OAuth2Session.fetch_token() and specifies the client configuration's token URI (usually Google's token server).

Parameters **kwargs** – Arguments passed through to requests_oauthlib.OAuth2Session.fetch_token(). At least one of `code` or `authorization_response` must be specified.

Returns

The obtained tokens. Typically, you will not use return value of this function and instead and use *credentials()* to obtain a Credentials instance.

Return type Mapping [str, str]

credentials

Returns credentials from the OAuth 2.0 session.

fetch_token() must be called before accessing this. This method constructs a google.oauth2.credentials.Credentials class using the session's token and the client config.

Returns The constructed credentials.

Return type google.oauth2.credentials.Credentials

Raises ValueError – If there is no access token in the session.

authorized_session()

Returns a requests.Session authorized with credentials.

fetch_token() must be called before this method. This method constructs a google.auth.transport.requests.AuthorizedSession class using this flow's *credentials*.

Returns

The constructed session.

Return type google.auth.transport.requests.AuthorizedSession

class InstalledAppFlow(*oauth2session*, *client_type*, *client_config*, *redirect_uri=None*, *code_verifier=None*, *autogenerate_code_verifier=False*)

Bases: *google_auth_oauthlib.flow.Flow*

Authorization flow helper for installed applications.

This *Flow* subclass makes it easier to perform the Installed Application Authorization Flow. This flow is useful for local development or applications that are installed on a desktop operating system.

This flow has two strategies: The console strategy provided by *run_console()* and the local server strategy provided by *run_local_server()*.

Example:

```
from google_auth_oauthlib.flow import InstalledAppFlow

flow = InstalledAppFlow.from_client_secrets_file(
    'client_secrets.json',
    scopes=['profile', 'email'])

flow.run_local_server()

session = flow.authorized_session()

profile_info = session.get(
    'https://www.googleapis.com/userinfo/v2/me').json()

print(profile_info)
# {'name': '...',  'email': '...', ...}
```

Note that these aren't the only two ways to accomplish the installed application flow, they are just the most common ways. You can use the *Flow* class to perform the same flow with different methods of presenting the authorization URL to the user or obtaining the authorization response, such as using an embedded web view.

Parameters

- **oauth2session** (*requests_oauthlib.OAuth2Session*) – The OAuth 2.0 session from `requests-oauthlib`.
- **client_type** (*str*) – The client type, either `web` or `installed`.
- **client_config** (`Mapping` [`str`, `Any`]) – The client configuration in the Google client secrets format.
- **redirect_uri** (*str*) – The OAuth 2.0 redirect URI if known at flow creation time. Otherwise, it will need to be set using *redirect_uri*.
- **code_verifier** (*str*) – random string of 43-128 chars used to verify the key exchange.using PKCE.
- **autogenerate_code_verifier** (*bool*) – If true, auto-generate a code_verifier.

run_console(*authorization_prompt_message='Please visit this URL to authorize this application: {url}'*, *authorization_code_message='Enter the authorization code: '*, ***kwargs*)

Run the flow using the console strategy.

The console strategy instructs the user to open the authorization URL in their browser. Once the authorization is complete the authorization server will give the user a code. The user then must copy & paste this code into the application. The code is then exchanged for a token.

Parameters

- **authorization_prompt_message** (*str*) – The message to display to tell the user to navigate to the authorization URL.
- **authorization_code_message** (*str*) – The message to display when prompting the user for the authorization code.
- **kwargs** – Additional keyword arguments passed through to *authorization_url()*.

Returns

The OAuth 2.0 credentials for the user.

Return type google.oauth2.credentials.Credentials

run_local_server(*host='localhost'*, *port=8080*, *authorization_prompt_message='Please visit this URL to authorize this application: {url}'*, *success_message='The authentication flow has completed. You may close this window.'*, *open_browser=True*, ***kwargs*)

Run the flow using the server strategy.

The server strategy instructs the user to open the authorization URL in their browser and will attempt to automatically open the URL for them. It will start a local web server to listen for the authorization response. Once authorization is complete the authorization server will redirect the user's browser to the local web server. The web server will get the authorization code from the response and shutdown. The code is then exchanged for a token.

Parameters

- **host** (*str*) – The hostname for the local redirect server. This will be served over http, not https.
- **port** (*int*) – The port for the local redirect server.
- **authorization_prompt_message** (*str*) – The message to display to tell the user to navigate to the authorization URL.

- **success_message** (*str*) – The message to display in the web browser the authorization flow is complete.
- **open_browser** (*bool*) – Whether or not to open the authorization URL in the user's browser.
- **kwargs** – Additional keyword arguments passed through to *authorization_url()*.

Returns

The OAuth 2.0 credentials for the user.

Return type google.oauth2.credentials.Credentials

authorization_url(***kwargs*)

Generates an authorization URL.

This is the first step in the OAuth 2.0 Authorization Flow. The user's browser should be redirected to the returned URL.

This method calls `requests_oauthlib.OAuth2Session.authorization_url()` and specifies the client configuration's authorization URI (usually Google's authorization server) and specifies that "offline" access is desired. This is required in order to obtain a refresh token.

Parameters **kwargs** – Additional arguments passed through to `requests_oauthlib.OAuth2Session.authorization_url()`

Returns

The generated authorization URL and state. The user must visit the URL to complete the flow. The state is used when completing the flow to verify that the request originated from your application. If your application is using a different *Flow* instance to obtain the token, you will need to specify the `state` when constructing the *Flow*.

Return type `Tuple`[`str`, `str`]

authorized_session()

Returns a `requests.Session` authorized with credentials.

fetch_token() must be called before this method. This method constructs a `google.auth.transport.requests.AuthorizedSession` class using this flow's *credentials*.

Returns

The constructed session.

Return type google.auth.transport.requests.AuthorizedSession

credentials

Returns credentials from the OAuth 2.0 session.

fetch_token() must be called before accessing this. This method constructs a `google.oauth2.credentials.Credentials` class using the session's token and the client config.

Returns The constructed credentials.

Return type google.oauth2.credentials.Credentials

Raises `ValueError` – If there is no access token in the session.

fetch_token(***kwargs*)

Completes the Authorization Flow and obtains an access token.

This is the final step in the OAuth 2.0 Authorization Flow. This is called after the user consents.

This method calls `requests_oauthlib.OAuth2Session.fetch_token()` and specifies the client configuration's token URI (usually Google's token server).

Parameters **kwargs** – Arguments passed through to `requests_oauthlib.OAuth2Session.fetch_token()`. At least one of `code` or `authorization_response` must be specified.

Returns

The obtained tokens. Typically, you will not use return value of this function and instead and use *`credentials()`* to obtain a `Credentials` instance.

Return type `Mapping` [`str`, `str`]

classmethod from_client_config(*client_config*, *scopes*, ***kwargs*)

Creates a `requests_oauthlib.OAuth2Session` from client configuration loaded from a Google-format client secrets file.

Parameters

- **client_config** (`Mapping` [`str`, `Any`]) – The client configuration in the Google client secrets format.
- **scopes** (`Sequence` [`str`]) – The list of scopes to request during the flow.
- **kwargs** – `Any` additional parameters passed to `requests_oauthlib.OAuth2Session`

Returns The constructed Flow instance.

Return type *Flow*

Raises `ValueError` – If the client configuration is not in the correct format.

classmethod from_client_secrets_file(*client_secrets_file*, *scopes*, ***kwargs*)

Creates a *`Flow`* instance from a Google client secrets file.

Parameters

- **client_secrets_file** (*`str`*) – The path to the client secrets .json file.
- **scopes** (`Sequence` [`str`]) – The list of scopes to request during the flow.
- **kwargs** – `Any` additional parameters passed to `requests_oauthlib.OAuth2Session`

Returns The constructed Flow instance.

Return type *Flow*

redirect_uri

The OAuth 2.0 redirect URI. Pass-through to `self.oauth2session.redirect_uri`.

google_auth_oauthlib.helpers module

Integration helpers.

This module provides helpers for integrating with requests-oauthlib. Typically, you'll want to use the higher-level helpers in *`google_auth_oauthlib.flow`*.

session_from_client_config(*client_config*, *scopes*, ***kwargs*)

Creates a `requests_oauthlib.OAuth2Session` from client configuration loaded from a Google-format client secrets file.

Parameters

- **client_config** (Mapping [str, Any]) – The client configuration in the Google client secrets format.
- **scopes** (Sequence [str]) – The list of scopes to request during the flow.
- **kwargs** – Any additional parameters passed to requests_oauthlib.OAuth2Session

Raises ValueError – If the client configuration is not in the correct format.

Returns

The new oauthlib session and the validated client configuration.

Return type Tuple [requests_oauthlib.OAuth2Session, Mapping [str, Any]]

session_from_client_secrets_file(*client_secrets_file*, *scopes*, ***kwargs*)

Creates a requests_oauthlib.OAuth2Session instance from a Google-format client secrets file.

Parameters

- **client_secrets_file** (*str*) – The path to the client secrets .json file.
- **scopes** (Sequence [str]) – The list of scopes to request during the flow.
- **kwargs** – Any additional parameters passed to requests_oauthlib.OAuth2Session

Returns

The new oauthlib session and the validated client configuration.

Return type Tuple [requests_oauthlib.OAuth2Session, Mapping [str, Any]]

credentials_from_session(*session*, *client_config=None*)

Creates google.oauth2.credentials.Credentials from a requests_oauthlib.OAuth2Session.

fetch_token() must be called on the session before before calling this. This uses the session's auth token and the provided client configuration to create google.oauth2.credentials.Credentials. This allows you to use the credentials from the session with Google API client libraries.

Parameters

- **session** (*requests_oauthlib.OAuth2Session*) – The OAuth 2.0 session.
- **client_config** (Mapping [str, Any]) – The subset of the client configuration to use. For example, if you have a web client you would pass in *client_config['web']*.

Returns The constructed credentials.

Return type google.oauth2.credentials.Credentials

Raises ValueError – If there is no access token in the session.

This library contains experimental oauthlib integration with google-auth.

CHAPTER 2

Installing

google-auth can be installed with pip:

```
$ pip install --upgrade google-auth-oauthlib
```

google-auth-oauthlib is open-source, so you can alternatively grab the source code from GitHub and install from source.

CHAPTER 3

Usage

Consult the *Module Reference* documentation.

CHAPTER 4

License

google-auth-oauthlib is made available under the Apache License, Version 2.0. For more details, see LICENSE

CHAPTER 5

Contributing

We happily welcome contributions, please see our contributing documentation for details.

Python Module Index

g

Index