FastHTML

Source > OAuth

OAuth

This provides the basic scaffolding for handling OAuth. It is not yet thoroughly tested. See the docs page for an explanation of how to use this.

source

GoogleAppClient

```
GoogleAppClient (client_id, client_secret, redirect_uri=None,
                 redirect_uris=None, code=None, scope=None, **kwargs)
```

A `WebApplicationClient` for Google oauth2

source

GitHubAppClient

```
GitHubAppClient (client_id, client_secret, redirect_uri, code=None,
                 scope=None, **kwargs)
```

A `WebApplicationClient` for GitHub oauth2

source

HuggingFaceClient

```
HuggingFaceClient (client_id, client_secret, redirect_uri=None,
                   redirect_uris=None, code=None, scope=None, state=None,
                   **kwargs)
```

A `WebApplicationClient` for HuggingFace oauth2

source

DiscordAppClient

```
DiscordAppClient (client_id, client_secret, redirect_uri, is_user=False,
                  perms=0, scope=None, **kwargs)
```

A `WebApplicationClient` for Discord oauth2

source

WebApplicationClient.login_link

```
WebApplicationClient.login_link (scope=None)
```

Get a login link for this client

Generating a login link that sends the user to the OAuth provider is done with `client.login_link()`:

---

source

WebApplicationClient.login_link_with_state

```
WebApplicationClient.login_link_with_state (scope=None, state=None)
```

Get a login link for this client

It can sometimes be useful to pass state to the OAuth provider, so that when the user returns you can pick up where they left off. This can be done by using the `login_link_with_state` function with a `state` parameter:

TODO: do all providers support this the same way? This is only tested for HF atm.

```
client = HuggingFaceClient("YOUR_CLIENT_ID","YOUR_CLIENT_SECRET",redirect_uri)
print(client.login_link_with_state(state="test_state"))
```

```
https://huggingface.co/oauth/authorize?
response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=http%3A%2F%2Flocalhost%3A8000%
2Fredirect&scope=openid+profile&state=test_state
```

---

source

_AppClient.parse_response

```
_AppClient.parse_response (code)
```

Get the token from the oauth2 server response

---

source

_AppClient.get_info

```
_AppClient.get_info ()
```

Get the info for authenticated user

---

source

_AppClient.retr_info

```
_AppClient.retr_info (code)
```

Combines `parse_response` and `get_info`

---

source

_AppClient.retr_id

```
_AppClient.retr_id (code)
```

Call `retr_info` and then return id/subscriber value

After logging in via the provider, the user will be redirected back to the supplied redirect URL. The request to this URL will contain a `code` parameter, which is used to get an access token and fetch the user's profile information. See the explainanation here for a worked example. You can either:

- use client.retr_info(code) to get all the profile information, or
- use client.retr_id(code) to get just the user's ID.

After either of these calls, you can also access the access token (used to revoke access, for example) with `client.token["access_token"]`.

Report an issue