django-oauthost Documentation

Release 1.1.1

Igor ’idle sign’ Starikov

Feb 04, 2022

Contents

Reusable application for Django to protect your apps with OAuth 2.0.

It allows to guard your applications views with OAuth 2.0 in quite a trivial way.

CHAPTER 1

Requirements

1. Python 3.6+
2. Django 1.8+
3. Django Auth contrib enabled
4. Django Admin site contrib enabled (for simple oauthost data manipulation).

CHAPTER 2

Table of Contents

2.1 Quick start

Warning: OAuth 2 requires secure connections, so oauthost will check for https *if your project is not in debug mode*, and will refuse to function if check fails.

- Do not use Django's cookie-based session engine with `oauthost`, it may cause security issues.
- Do not use OAuth1 clients.
- Verify `MIDDLEWARE_CLASSES` setting has

  django.contrib.sessions.middleware.SessionMiddleware and *django.middleware.csrf.CsrfViewMiddleware*
- Verify `TEMPLATE_CONTEXT_PROCESSORS` has *django.core.context_processors.request*

  Note: For Django 1.8+: *django.template.context_processors.request* should be defined in `TEMPLATES/OPTIONS/context_processors`.
- Add `oauthost` into `INSTALLED_APPS`

2.1.1 Step by step

0. Initialize DB tables for `oauthost`. Run from command line:

   ```
   python manage.py migrate
   ```
1. Attach *oauthost.urls* to project URLs file (e.g. *urls.py*)

```
from oauthost.urls import urlpatterns as oauthost_urlpatterns

urlpatterns = ...  # Your actual urlpatterns are ommited.

urlpatterns += oauthost_urlpatterns
```

Now authorization endpoint is available at *{ BASE_URL }auth/*.

And token endpoint is available at *{ BASE_URL }token/*.

2. Decorate application views which require OAuth 2 authorization with *@oauth_required* (let's suppose those are views from *polls* application):

```
from oauthost.decorators import oauth_required


@oauth_required(scope='my_polls:my_stats')
def stats(request, poll_id):
    """Scope associated with this view is `my_polls:my_stats`."""

@oauth_required(scope_auto=True)
def results(request, poll_id):
    """Scope for this view would be evaluated to `polls:results`."""
```

3. Use Django Admin site contrib package to manipulate `oauthost` data (e.g. register clients).

   3.1. Register *scopes* for your Django application.

   Scope identifiers examples: *polls:index*, *polls:detail*, *polls:results*.

   Note: You can use `syncscopes` management command which automatically creates scopes for *oauth_required* decorated views available in application(s), which names are passed to the command:

   ```
   python manage.py syncscopes polls
   ```

   3.2. Register a **client** which could be granted with access to your resources.

   Note: Just right there on client registration page you can set up redirection endpoints, register authorization codes and issue tokens. Latter two should normally be issued to a client itself as described in paragraph no 4.

   Or use API:

   ```
   from oauthost.toolbox import register_client


   # Define some scopes to restrict our client to.
   my_scopes = ['polls:vote', 'polls:stats']

   # `user` might be `request.user` if in a view.
   register_client('My OAuth Client', '1234', 'http://myapp.
   ↪com/', user, scopes_list=my_scopes)
   ```

2.1.2 Tokens and protected resources

4. Access authorization and/or token endpoints (see no 1 above) from within the client (registered in no 3.2) to gain credentials (namely an *access token*) to access protected views.

4.1. First your client needs to get an access token and there are several ways to get it.

> **Note:** In the examples below we use client with ID 1234, which has one redirection endpoint (e.g. *http://myapp.com/*).

4.1.1. Grant token through authorization code.

1. Request for authorization code with GET HTTP method:

```
{BASE_URL}auth/?client_id=1234&response_type=code
```

2. Grab *code* param value from URL your client is redirected to (e.g. *http://myapp.com/*).

3. Exchange authorization code for access token using POST HTTP method:

```
{BASE_URL}token/ grant_type=authorization_code&code={code_from_no_
↪2}&redirect_uri=http://myapp.com/&client_id=1234
```

4. Get *access_token* param value from JSON document returned by server.

4.1.2. Grant token implicitly.

1. Request for authorization code with GET HTTP method:

```
{BASE_URL}auth/?client_id=1234&response_type=token
```

2. Get *access_token* param value from JSON document returned by server.

4.2. Second your client should supply token from no 4.1 (or no 3.2) to server when accessing any protected views of your application. Currently there are three ways to do it. Let's suppose our access token is 987654.

4.2.1. Recommended way is to pass token in HTTP Authorization Bearer header:

```
GET /polls HTTP/1.1
Host: myapp.com
Authorization: Bearer 987654
```

4.2.2. You can also use POST HTTP method (*access_token* param is checked):

```
POST /polls HTTP/1.1
Host: myapp.com
Content-Type: application/x-www-form-urlencoded

access_token=987654
```

4.2.3. Finally you can use GET HTTP method (*access_token* param is checked):

```
GET /polls?access_token=987654 HTTP/1.1
Host: myapp.com
```

2.2 Helpers

2.2.1 Piston Authentication class

Piston is a mini-framework for Django for creating RESTful APIs - http://bitbucket.org/jespern/django-piston/wiki/Home

Oauthost comes with an authentication class for Piston resources.

Piston resource view creation example:

```
from piston.resource import Resource
from oauthost.toolbox import PistonAuthHelper

my_resource_view = Resource(MyResourceHandler, authentication=PistonAuthHelper('my_
↪resource:my_scope'))
```

See Piston documentation for more information on authentication customizations.

2.3 Things to read

OAuth 2.0 Authorization Protocol - http://tools.ietf.org/html/rfc6749

OAuth 2.0 Bearer Tokens - http://tools.ietf.org/html/rfc6750

All different flavors:

- Yandex - http://api.yandex.ru/oauth/doc/dg/concepts/About.xml
- GitHub - https://developer.github.com/v3/oauth/
- Google - https://developers.google.com/accounts/docs/OAuth2
- Facebook - https://developers.facebook.com/docs/facebook-login/access-tokens

CHAPTER 3

Get involved into django-oauthost

Submit issues. If you spotted something weird in application behavior or want to propose a feature you can do that at https://github.com/idlesign/django-oauthost/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/django-oauthost, write your code, whether it should be a bugfix or a feature implementation, and make a pull request right from the forked project page.

Translate. If want to translate the application into your native language use Transifex: https://www.transifex.net/projects/p/django-oauthost/.

Spread the word. If you have some tips and tricks or any other words in mind that you think might be of interest for the others — publish it.

CHAPTER 4

Also

If the application is not what you want for OAuth 2.0, you might be interested in considering other choices at http://djangopackages.com/grids/g/oauth-servers/