django-authlib Documentation

Release 0.16.3

Feinheit AG

Sep 17, 2023

Contents

1 Goals **3**

2 Usage **5**

3 Use of bundled views **7**

4 Admin OAuth2 **9**

5 Little Auth **11**

6 Email Registration **13**

7 Change log **15**

7.1 Next version . . . 15
7.2 0.16 (2023-09-17) . . . 15
7.3 0.15 (2023-07-07) . . . 15
7.4 0.14 (2023-03-21) . . . 16
7.5 0.13 (2022-02-28) . . . 16
7.6 0.12 (2022-01-04) . . . 16
7.7 0.11 (2021-11-22) . . . 16
7.8 0.10 (2020-10-04) . . . 16
7.9 0.9 (2019-02-09) . . . 17
7.10 0.8 (2018-11-17) . . . 17
7.11 0.7 (2018-11-04) . . . 17
7.12 0.6 (2017-12-04) . . . 18
7.13 0.5 (2017-05-17) . . . 18
7.14 0.4 (2017-05-11) . . . 18
7.15 0.3 (2016-12-08) . . . 19
7.16 0.2 (2016-11-22) . . . 19
7.17 0.1 (2016-11-21) . . . 19

authlib is a collection of authentication utilities for implementing passwordless authentication. This is achieved by either sending cryptographically signed links by email, or by fetching the email address from third party providers such as Google, Facebook and Twitter. After all, what's the point in additionally requiring a password for authentication when the password can be easily resetted on most websites when an attacker has access to the email address?

CHAPTER 1

Goals

- Stay small, simple and extensible.
- Offer tools and utilities instead of imposing a framework on you.

CHAPTER 2

Usage

- Install `django-authlib` using pip into your virtualenv.
- Add `authlib.backends.EmailBackend` to `AUTHENTICATION_BAcKENDS`.
- Adding `authlib` to `INSTALLED_APPS` is optional and only useful if you want to use the bundled translation files. There are no required database tables or anything of the sort.
- Have a user model which has a email field named `email` as username. For convenience a base user model and manager are available in the `authlib.base_user` module, `BaseUser` and `BaseUserManager`. The `BaseUserManager` is automatically available as `objects` when you extend the `BaseUser`.
- Use the bundled views or write your own. The bundled views give feedback using `django.contrib.messages`, so you may want to check that those messages are visible to the user.

The Google, Facebook and Twitter OAuth clients require the following settings:

- `GOOGLE_CLIENT_ID`
- `GOOGLE_CLIENT_SECRET`
- `FACEBOOK_CLIENT_ID`
- `FACEBOOK_CLIENT_SECRET`
- `TWITTER_CLIENT_ID`
- `TWITTER_CLIENT_SECRET`

Note that you have to configure the Twitter app to allow email access, this is not enabled by default.

Note: If you want to use OAuth2 providers in development mode (without HTTPS) you could add the following lines to your `settings.py`:

```
if DEBUG:
    # NEVER set this variable in production environments!
    os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"
```

This is required because of the strictness of oauthlib which only wants HTTPS URLs (and rightly so).

CHAPTER 3

Use of bundled views

The following URL patterns are an example for using the bundled views. For now you'll have to dig into the code (it's not much, at the time of writing `django-authlib`'s Python code is less than 500 lines):

```
from django.conf.urls import url
from authlib import views
from authlib.facebook import FacebookOAuth2Client
from authlib.google import GoogleOAuth2Client
from authlib.twitter import TwitterOAuthClient

urlpatterns = [
    url(
        r"^login/$",
        views.login,
        name="login",
    ),
    url(
        r"^oauth/facebook/$",
        views.oauth2,
        {
            "client_class": FacebookOAuth2Client,
        },
        name="accounts_oauth_facebook",
    ),
    url(
        r"^oauth/google/$",
        views.oauth2,
        {
            "client_class": GoogleOAuth2Client,
        },
        name="accounts_oauth_google",
    ),
    url(
        r"^oauth/twitter/$",
        views.oauth2,
```

(continues on next page)

(continued from previous page)

```
        {
            "client_class": TwitterOAuthClient,
        },
        name="accounts_oauth_twitter",
    ),
    url(
        r"^email/$",
        views.email_registration,
        name="email_registration",
    ),
    url(
        r"^email/(?P<code>[^/]+)/$",
        views.email_registration,
        name="email_registration_confirm",
    ),
    url(
        r"^logout/$",
        views.logout,
        name="logout",
    ),
]
```

CHAPTER 4

Admin OAuth2

The `authlib.admin_oauth` app allows using Google OAuth2 to allow all users with the same email domain to authenticate for Django's administration interface. You have to use authlib's authentication backend (`EmailBackend`) for this.

Installation is as follows:

- Follow the steps in the "Usage" section above.
- Add `authlib.admin_oauth` to your `INSTALLED_APPS` before `django.contrib.admin`, so that our login template is picked up.
- Add `GOOGLE_CLIENT_ID` and `GOOGLE_CLIENT_SECRET` to your settings as described above.
- Add a `ADMIN_OAUTH_PATTERNS` setting. The first item is the domain, the second the email address of a staff account. If no matching staff account exists, authentication fails:

```
ADMIN_OAUTH_PATTERNS = [
    (r"@example\.com$", "admin@example.com"),
]
```

- Add an entry to your URLconf:

```
urlpatterns = [
    url(r"", include("authlib.admin_oauth.urls")),
    # ...
]
```

- Add `https://yourdomain.com/admin/__oauth__/` as a valid redirect URI in your Google developers console.

Please note that the `authlib.admin_oauth.urls` module assumes that the admin site is registered at `/admin/`. If this is not the case you can integrate the view yourself under a different URL.

It is also allowed to use a callable instead of the email address in the `ADMIN_OAUTH_PATTERNS` setting; the callable is passed the result of matching the regex. If a resulting email address does not exist, authentication (of course) fails:

```
ADMIN_OAUTH_PATTERNS = [
    (r"^.*@example\.org$", lambda match: match[0]),
]
```

If a pattern succeeds but no matching user with staff access is found processing continues with the next pattern. This means that you can authenticate users with their individual accounts (if they have one) and fall back to an account for everyone having a Google email address on your domain:

```
ADMIN_OAUTH_PATTERNS = [
    (r"^.*@example\.org$", lambda match: match[0]),
    (r"@example\.com$", "admin@example.com"),
]
```

You could also remove the fallback line; in this case users can only authenticate if they have a personal staff account.

CHAPTER 5

Little Auth

The `authlib.little_auth` app contains a basic user model with email as username that can be used if you do not want to write your own user model but still profit from authlib's authentication support.

Usage is as follows:

- Add `authlib.little_auth` to your `INSTALLED_APPS`
- Set `AUTH_USER_MODEL = "little_auth.User"`
- Optionally also follow any of the steps above.

CHAPTER 6

Email Registration

For email registration to work, two templates are needed:

- `registration/email_registration_email.txt`
- `registration/email_registration.html`

A starting point would be:

`email_registration_email.txt`:

```
Subject (1st line)

Body (3rd line onwards)
{{ url }}
...
```

`email_registration.html`:

```
{% if messages %}
<ul class="messages">
    {% for message in messages %}
    <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>
        {% if message.level == DEFAULT_MESSAGE_LEVELS.ERROR %}Important: {% endif %}
        {{ message }}
    </li>
    {% endfor %}
</ul>
{% endif %}

{% if form.errors and not form.non_field_errors %}
<p class="errornote">
    {% if form.errors.items|length == 1 %}
    {% translate "Please correct the error below." %}
    {% else %}
    {% translate "Please correct the errors below." %}
    {% endif %}
```

(continues on next page)

(continued from previous page)

```
</p>
{% endif %}

{% if form.non_field_errors %}
{% for error in form.non_field_errors %}
<p class="errornote">
    {{ error }}
</p>
{% endfor %}
{% endif %}

<form action='{% url "email_registration" %}' method="post" >
    {% csrf_token %}
    <table>
        {{ form }}
    </table>
    <input type="submit" value="login">
</form>
```

The above template is inspired from:

- Messages Django documentation
- Django login template

More details are documented in the relevant module.

CHAPTER 7

Change log

7.1 Next version

- Changed the roles implementation to allow using arbitrary names for the role field.
- Stopped crashing when encountering an unknown role – doing nothing in `has_perm` is an acceptable fallback.

7.2 0.16 (2023-09-17)

- Fixed `pyproject.toml` so that data files are actually included.
- Dropped compatibility with Python 3.8.
- Added utilities for role-based permissions. The idea is to allow a less manual way to specify permissions for groups of users, e.g. content managers which should automatically have access to all models in a list of apps without having to manually update the list of permissions in the Django administration interface.

7.3 0.15 (2023-07-07)

- Added Python 3.11.
- Switched to hatchling and ruff.
- Added the option to create admin users during admin OAuth if one doesn't exist already. The `ADMIN_OAUTH_CREATE_USER_CALLBACK` setting should be set to the Python path of a callable receiving the request and the email address; this callable can (but doesn't have to) create a new user for the email address if one doesn't exist already. The default is to not create any users. Adding `ADMIN_OAUTH_CREATE_USER_CALLBACK = "authlib.admin_oauth.views.create_superuser"` makes creation of new superuser accounts automatic.

7.4 0.14 (2023-03-21)

- Added Django 4.1 and 4.2 to the CI matrix.
- Made the bundled OAuth2 views pass the exception message to `messages.error` to ease debugging a bit.
- Changed the confirmation code used by `authlib.email` to be base64 encoded. This avoids problems where some email clients would mangle the link because of the included email address. Older codes are still accepted for the moment.
- Added a note regarding `OAUTHLIB_INSECURE_TRANSPORT` to the README.

7.5 0.13 (2022-02-28)

- Added a `default_auto_field` to the `little_auth` appconfig.

7.6 0.12 (2022-01-04)

- Added pre-commit.
- Dropped Python < 3.8, Django < 3.2.
- Added docs for how to integrate the email registration functionality.

7.7 0.11 (2021-11-22)

- Switched to a declarative setup.
- Switched from Travis CI to GitHub actions.
- Added Python 3.10, Django 4.0 to the CI.
- Avoided the additional request to Google endpoints since the access token already contains identity information in the `id_token` field.

7.8 0.10 (2020-10-04)

- Modified `authlib.admin_oauth` to persist the users' email address and pass it to Google as a `login_hint` so that website managers do not have to repeatedly select the account over and over.
- Allowed specifying arbitrary query parameters for Google's authorization URL.
- Fixed an `authlib.admin_oauth` crash when fetching user data fails.
- Replaced `ugettext*` with `gettext*`.
- Replaced `url()` with `re_path()`.
- Fixed a crash when creating `little_auth` users with invalid email addresses.
- Stopped carrying over login hints from one user to the other in the Google OAuth client...
- **BACKWARDS INCOMPATIBLE** Dropped the request argument from `authlib.email.get_confirmation_code`, it wasn't used, ever.

7.9 0.9 (2019-02-09)

- Dropped support for Python 2.
- Fixed a few problems around inactive users where authlib would either handle them incorrectly or reveal that inactive users exist.
- Added many unittests, raised the code coverage to 100% (except for the uncovered Facebook and Twitter OAuth clients). Switched to mocking requests and responses instead of simply replacing the `GoogleOAuth2Client` for testing.
- Moved the `BaseUser` and `BaseUserManager` to `authlib.base_user` for consistency with `django.contrib.auth.base_user`.
- Dropped the useless `OAuthClient` base class.
- Removed compatibility code for Django<1.11 when verifying whether a redirection URL is safe.
- Changed the `retrieve_next` implementations to only consider HTTPS URLs as safe when processing HTTPS requests.
- Changed the admin OAuth functionality to also use the cookies code from `authlib.views` for redirecting users after authentication.
- Fixed a possible crash in the Twitter OAuth flow when the token from the authentication redirect cannot be determined anymore.
- Fixed a crash in the OAuth2 view if fetching user data fails.

7.10 0.8 (2018-11-17)

- **BACKWARDS INCOMPATIBLE** Replaced the email registration functionality of referencing users with arbitrary payloads. This allows not only verifying the email address but also additional data which may or may not be related to the user in question. On the other hand the comparison of `last_login` timestamps is gone, which means that links may be reused as long as less than `max_age` seconds have passed. This makes it even more important to keep `max_age` small. The change mostly affects the functions in `authlib.email`.

7.11 0.7 (2018-11-04)

- Fixed a race condition when creating new users by using `get_or_create` instead of some homegrown `exists` and `create` trickery.
- Changed all locations to pass `new_user` as keyword argument to `post_login_response`.
- Changed the `admin/login.html` template in `authlib.admin_oauth` to make the SSO button a bit more prominent. Also, replaced "SSO" with "Google" because that is all that is supported right now.
- Added the possibility to use callables in `ADMIN_OAUTH_PATTERNS` instead of hard-coded staff email addresses.
- Extracted the confirmation code generation from `get_confirmation_url` as `get_confirmation_code`.
- Fixed usage of deprecated Google OAuth2 scopes.
- Added compatibility with Python 2.
- Extracted the post login redirect cookie setting into a new `set_next_cookie` decorator.

- Dropped compatibility shims for Django<1.11.
- Changed the `EmailBackend` to use `_default_manager` instead of assuming that the default manager is called `objects`.
- Fixed an edge case bug where `render_to_mail` would crash when encountering an empty text for the subject and body.
- Enforced keyword-only usage of the views and functions in `authlib.views` where it is appropriate.
- Removed the default messages emitted when creating a new user and when logging out.
- Added a `post_logout_response` callable and argument to `authlib.views.logout` to customize messages and redirects after logging an user out.
- Added a `email_login` callable and argument to the `oauth2` and `email_registration` view to customize the creation, authentication and login of users.
- Changed the `EmailRegistrationForm` to save the request as `self.request`, not `self._request`. Made use of this for moving the email sending to the form class as well, further shortening the view.

7.12 0.6 (2017-12-04)

- Fixed usage of a few deprecated APIs.
- Modified `little_auth.User` to fall back to an obfuscated email address if the full name is empty.
- Made it possible to override the default max age of three hours for magic links sent by email.
- Fixed a problem where the `little_auth` migrations were depending on the latest `django.contrib.auth` migration instead of the first migration without good reason.

7.13 0.5 (2017-05-17)

- Moved from `ADMIN_OAUTH_DOMAINS` to `ADMIN_OAUTH_PATTERNS` to allow regular expression searching.
- Finally started adding tests.
- Added django-authlib documentation to Read the Docs.

7.14 0.4 (2017-05-11)

- Added some documentation to the README.
- Google client: Removed the deprecated profile scope, and switched to online access only (we do not need offline access).
- Added the `authlib.admin_oauth` app for a minimal Google OAuth2 authentication solution for Django's administration interface.
- Added the `authlib.little_auth` app containing a minimal user model with email as username for a quick and dirty `auth.User` replacement.
- Allow overriding the view name used in `authlib.email.get_confirmation_url`.

7.15 0.3 (2016-12-08)

- Fixed the redirect URL generation of the Facebook and Google client.
- Changed the name of the post login redirect cookie from `next` to `authlib-next` to hopefully prevent clashes.
- Authentication providers may also return `None` as email address; handle this case gracefully by showing an error message instead of crashing.
- Pass full URLs, not only paths to the OAuth2 libraries because otherwise, secure redirect URLs aren't recognized as such.

7.16 0.2 (2016-11-22)

- Added views for registration and logging in and out.
- Added a base user model and an authentication backend for authenticating using email addresses only.

7.17 0.1 (2016-11-21)

- Initial release containing helpers for authentication using an email address, either verified by sending a magic link or retrieved from Facebook, Google or Twitter.