django-socialprofile Documentation

Release .05

Tim White

July 25, 2016

Contents

Django Social Profile gives you an out-of-the-box way to let users create an account in your application using Google, Twitter, or Facebook authentication.

Users can edit their profile, view other users' profiles, and add multiple types of auth to the same profile.

Django Social Profile relies on omab's excellent python-socialauth to do the actual authentication with the backend providers. If you are just looking for the authentication piece, as opposed to the UI for customers to use, that module will be all you need. If you are willing to spend a bit of time with the UI, you can integrate any of the backends that python-socialauth provides (which is extensive).

Table of Contents

CHAPTER 1

Features

This module is meant to be as quick to integrate as possible, and thus extensive customization will likely benefit from a fork. That said, a number of options are available.

The idea is to let you have a working system for letting users create profiles with social auth, edit them, delete them, and merge them, out of the box.

All the underlying bits to make this work come with python-socialauth, this project just pulls them together with a UI.

CHAPTER 2

Dependencies

Dependencies that **must** be meet to use the application:

- python-social-auth:
- python-openid
- oauth2
- You will need API Keys from Google, Facebook, and Twitter.

CHAPTER 3

Installation

From pypi:

```
$ pip install django-socialprofile
```

or:

```
$ easy_install django-socialprofile
```

or clone from github:

```
$ git clone git://github.com/cyface/django-socialprofile.git
```

and add django-socialprofile to the `PYTHONPATH`:

```
$ export PYTHONPATH=$PYTHONPATH:$(pwd)/django-socialprofile/
```

or:

```
$ cd django-socialprofile
$ sudo python setup.py install
```

CHAPTER 4

Demo App

The socialprofile_demo app is included to quickly let you see how to get a working installation going.

The demo is built as a mobile app using jQueryMobile loaded from the jQuery CDN.

Take a look at the `requirements.txt` file in the `socialprofile_demo` directory for a quick way to use pip to install all the needed dependencies:

```
$ pip install -r requirements.txt
```

The `settings.py`, and `settings_local_template.py` files have a working configuration you can crib from. You'll need to copy `settings_local_template.py` as `settings_local.py` and fill in your API keys.

The templates in the `socialprofile/templates` and `socialprofile_demo/templates` directories give you a good idea of the kinds of things you will need to do if you want to provide a custom interface.

CHAPTER 5

Configuration

Configuration is minimal for socialprofile itself, more config is needed for `python-socialauth`. A quick guide to a basic setup is below, take a look at the demo app for more details.

5.1 Add INSTALLED_APPS

Add social_auth and socialprofile to installed applications:

```
INSTALLED_APPS = (
    ...
    'social.apps.django_app.default',
    'socialprofile',
)
```

5.2 Add urls to urls.py

In your urls.py, you need to pull in the socialprofile urls:

```
# Social Profiles
url(r'^socialprofile/', include('socialprofile.urls')),
```

The `python-socialauth` urls get pulled in by socialprofile as `/socialprofile/socialauth/`.

5.3 Configure python-socialauth

All of the configuration for `python-socialauth` applies to this module, although the supplied templates only cover Google, Facebook, and Twitter. `python-socialauth` can handle a huge number of backends, you can customize as needed.

- Setup your backends:

```
    # Python Socialauth Settings
    AUTHENTICATION_BACKENDS = (
            'django.contrib.auth.backends.ModelBackend',  # Leave Enabled for Admin Access
        'social.backends.twitter.TwitterOAuth',
        'social.backends.facebook.Facebook2OAuth2',
        'social.backends.google.GoogleOAuth2',
    )
```

- Set up what page to go to post-authentication:

```
SOCIAL_AUTH_LOGIN_REDIRECT_URL = '/secure/'
SOCIAL_AUTH_NEW_USER_REDIRECT_URL = '/secure/'
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = '/secure/'
```

- Set up the redirects for forcing auth on the way to other pages:

```
# Core Authentication Settings
LOGIN_URL          = '/socialprofile/select/'
LOGIN_REDIRECT_URL = '/secure/'
LOGIN_ERROR_URL    = '/socialprofile/select/'
```

5.4 Register for your API Keys

5.4.1 Google

https://code.google.com/apis/console/

Set the return URL to http://localhost:8000/socialprofile/socialauth/complete/google-oauth2/ for development when you set up the API key.

5.4.2 Twitter

https://dev.twitter.com/apps/new

Set the callback URL to http://localhost:8000/socialprofile/socialauth/complete/twitter/ for development when you set up the API key.

5.4.3 Facebook

Facebook is a bit of a pain, since you can only have one URL per API key.

https://developers.facebook.com/apps

Set the site URL http://localhost:8000/ for local development.

Facebook also allows you to request additional information beyond authentication. The default setup assumes you are requesting the user's email address. See below for how to note that in the API settings.

See https://developers.facebook.com/docs/authentication/permissions/#extended_perms for details on other permissions you can request.

5.4.4 Add API Keys to Settings

Take the keys from your APIs and add them to your settings:

```
SOCIAL_AUTH_TWITTER_KEY          = ''
SOCIAL_AUTH_TWITTER_SECRET       = ''
SOCIAL_AUTH_FACEBOOK_KEY         = ''
SOCIAL_AUTH_FACEBOOK_SECRET      = ''
SOCIAL_AUTH_FACEBOOK_SCOPE       = ['public_profile', 'email']
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {'fields': 'first_name,last_name,gender,picture,link'}
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY    = ''
```

```
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ''
SOCIAL_AUTH_GOOGLE_OAUTH_SCOPE  = ['https://www.googleapis.com/auth/userinfo.profile',]
```

Note that the extended permissions and such there are typical, you may also want to request the ability to post as that user and so forth.

CHAPTER 6

Django Admin

This project creates a `SocialProfile` object for each User that is registered via one of the social methods. The data on this profile can be viewed and edited in the Django admin by editing the `User` object. The `SocialProfile` data appears at the bottom of the `User` detail in the Django admin.

To do this, a custom `User` admin definition is created in `admin.py`, so if you have other things that override the User admin, you'll want to merge this customization in with that, or provide your own admin definition for `SocialProfile`.

CHAPTER 7

Views and Layers

7.1 Login Modal Layer

The 'socialprofile/select' view provides a login modal that you can use to both force existing users to sign in as well as to enable new users to select how they want to authenticate to the site.

If you have LOGIN_URL set to `/socialprofile/select/`, this will work automatically.

The default template has attributes to make this a nice modal using jQueryMobile, but the HTML is straightforward, and a custom template should be simple to create.

7.2 Self Profile View

The `socialprofile/` view lets a user see their own profile. The default template checks to see if they profile is indeed theirs, and displays an 'edit' button taking them to the `socialprofile/edit/` view.

This view supports a `?returnTo=` parameter to specify a URL path to return to once the user is done. The default template uses this for the `< Return` button.

7.3 Other Profile View

The `socialprofile/view/<username>` view lets a user see any profile. You may want to adjust the template to hide any profile fields that should not be public.

This view supports a `?returnTo=` parameter to specify a URL path to return to once the user is done. The default template uses this for the `< Return` button.

7.4 Profile Edit View

The `socialprofile/edit/` view lets a user edit their own profile. In typical Django fashion, a GET request to this view will display the form, while a POST request to this view will try and save the changes.

This view supports a `?returnTo=` parameter to specify a URL path to return to once the user is done. The default template uses this for the `Cancel` and `Done` button. When the form returns to the Self Profile View, it passes `returnTo`.

7.5 Profile Add Auth Type

A user can add an additional social authentication type to their existing profile. If they originally created their profile using Google auth, then they could add Facebook and Twitter, enabling them to sign in with any of those services and access the same account.

To do this, just have the customer log in with their new auth type, and python-socialauth will do the rest.

7.6 Profile Delete Auth Type

This is a default feature of python-socialauth, and is available using:

```
{% url "social:disconnect" user_social_auth.provider %}
```

... in a template.

7.7 Delete Account

It is important to let customers remove their accounts, and the /socialprofile/delete view prompts them to ensure they really want to delete their account before sending them to /socialprofile/delete/action?confirm=true.

You may want to provide your own function to do this, that perhaps only deactivates their account.