django-la-facebook Documentation

Release 0.1.beta

Daniel Greenfeld and contributors

December 18, 2016

Contents

Contents:

CHAPTER 1

Usage

Get `django-la-facebook` into your python path:

```
pip install django-la-facebook
```

Add `la_facebook` to your INSTALLED_APPS in settings.py:

```
INSTALLED_APPS = (
    ...
    'la_facebook',
    ...
    )
```

Add `la_facebook` to your root urlconf (urls.py):

```
urlpatterns = patterns('',
    ...,
    url(r"^la_facebook/", include("la_facebook.urls")),
    ...,
)
```

1.1 Settings

In order to authenticate your site's users with Facebook, you need a unique identifier for Facebook to associate your site with. Facebook considers your site an "app" and so you must acquire an `FACEBOOK_APP_ID` and `FACEBOOK_APP_SECRET` from the Facebook Developer app.

You will need to enter your sites domain into the Facebook developer app, and the site will restrict authentication requests to that domain. For initial testing and experiments, you will want to enter "http://localhost/" for the website URL. For later testing, you will probably want create a staging or testing subdomain. The domain set here must match the domain entered into Django's sites framework.

See the documentation for Settings about how to enter these values in your Django settings file.

1.2 Using the default authentication flow

By default, the la_facebook application:

- provides a method of creating a django.contrib.auth user model for an authenticated facebook user

- Updates fields in the model pointed to by the AUTH_PROFILE_MODULE setting that match available fields in Facebook's user data.
- Creates and manages an association object that stores the user's associated facebook id, authentication token (which is used to access the information in the users facebook profile as authorized by the user), and the expiration date of that token.

These steps are handled by directing a user to the la_facebook login view. By default this view, like Django's login url LOGIN_URL, will use a next querystring parameter to redirect the browser to a page after user authenticates with Facebook.

If an error occurs during authentication, or the user denies to authenticate, the browser is redirected to a template located at la_facebook/fb_error.html (see the provided template for some information about what context variables may be provided in the case of an error).

If you wish a more customized behavior for Facebook authentication, see the Callbacks documentation.

1.3 Templates

In your login page, or as part of your login form, you should include a link to the Facebook login view, which will then redirect the user to facebook to login and authenticate to your site. A simple example might look like this:

```
<p><a href="{% url la_facebook_login %}?next={{ next }}">Login with FaceBook</a></p>
```

You can used the following image provided by facebook as a graphical link:

```
http://static.ak.fbcdn.net/images/fbconnect/login-buttons/connect_light_medium_long.gif
```

Other than the error template mentioned above, no particular templates are used or customized.

1.4 Changing the display layout of the login page

By default the login flow redirects you to facebook for login in your main browser window, and the layout of this page is suited to be a complete page. Facebook also supports an alternate "popup" display style, which is better suited for popup windows. This can reduce the feeling that the user is leaving your site. You must pass the display:popup option to the login view in your own project urls.py, and you must handle the creation and destruction of the actual popup window yourself.

1.5 Template Tags

TODO

CHAPTER 2

Settings

This document describes the settings needed by la_facebook.

FACEBOOK_ACCESS_SETTINGS this is the only setting required. It is a dictionary of app specific settings as follows:

FACEBOOK_APP_ID - this is a key that uniquely identifies your Facebook app, in the common case, the Facebook app will be your project or site.

FACEBOOK_APP_SECRET - This secret is used in generating the authenticated token. Both the key and the secret are available through the Facebook's Developer app.

CALLBACK (optional) - this is a dotted module path string (similar to using a string for a view) that points to a subclass of la_facebook.callbacks.default. The default value is "la_facebook.callbacks.default.default_facebook_callback"

PROVIDER_SCOPE (optional) - a list, of strings, of permissions to ask for. The list of these is here

LOG_LEVEL (optional) - A string value containing one of standard python logging levels of DEBUG, INFO, WARNING, ERROR or CRITICAL. Defaults to "ERROR", which should be relatively quiet.

LOG_FILE (optional) - The path to a file that will received appended logging information. By default, logged messages will print to stdout.

Example:

```
FACEBOOK_ACCESS_SETTINGS = {
        "FACEBOOK_APP_ID": FACEBOOK_APP_ID,
        "FACEBOOK_APP_SECRET": FACEBOOK_APP_SECRET,
        # The following keys are optional
        # "CALLBACK": "la_facebook.callbacks.default.default_facebook_callback",
        # "PROVIDER_SCOPE": ['email','read_stream'],
        # "LOG_LEVEL": "DEBUG",
        # "LOG_FILE": "/tmp/la_facebook.log",
}
```

CHAPTER 3

Callbacks

In the context of la_facebook, callbacks are a subclass of la_facebook.callbacks.base.BaseFacebookCallback.

The callback is called after the user authenticates with Facebook, either for the first time, or when returning after authentication expiration. The callback determines what happens next, whether that be a user being created, or the browser being redirected.

la_facebook ships with two callbacks. BaseFacebookCallback defines the key steps that any response to a facebook auth should include, but is largely an abstract class. DefaultFacebookCallback is a working callback that addresses the most common use case for basic user auth. Most projects will probably be fine to just use this default implementation.

3.1 What the default implementation does

When a new user comes to your site, and clicks on a link that calls the la_facebook.views.facebook_login view, they are redirected to facebook where they are authenticated and approve your app. They are then redirected to your site and the default callback is executed. This callback fist checks whether there is already an authenticated Django user in the session. If not the Facebook id of the user is examined and it is determined whether they have previously authenticated to the site by looking for database model that stores an association between a Django user and a Facebook user.

If there exists a prior association, the expiration of that users Facebook authentication token is updated, the Django session expiration is set to match, and the user is logged and redirected.

If no prior association is present, a django user and its association object are created. The default Django username consists of the facebook user's name sluggified and concatenated with their facebook id. Any profile object, as defined in Django's settings, is updated with any matching fields in the users Facebook data.

3.2 Callback Reference

A callback is a view like object is called with instances of a Django request, OAuthAccess, and OAuth20Token.

class **BaseFacebookCallback**
Provides a largely abstract class defining an auth interaction with Facebook.

__call__(*request*, *access*, *token*)
is called by the view and handles the basic check of whether the user is authenticated and dispatches to the other methods as nessasary and then returns the response to the view and thus to the browser (usually a redirect)

fetch_user_data(*request*, *access*, *token*)
Not implemented.

lookup_user(*request*, *access*, *token*)
Not implemented.

redirect_url(*request*, *access*, *token*)
Checks in order: the request GET params, the session, settings for a url to redirect to.

handle_no_user(*request*, *access*, *token*, *user_data*)
Not implemented.

handle_unauthenticated_user(*request*, *access*, *token*, *user_data*)
Not implemented.

identifier_from_data(*data*)
Concatenates a sluggified facebook name and user id

class `la_facebook.callbacks.default`.**DefaultFacebookCallback**
Provides the default implementation.

fetch_user_data(self, request, access, token):
Uses the authorized token and makes an API call to Facebook to retrieve the user graph data.

lookup_user(self, request, access, user_data):
Based on the Facebook user ID in the :param user_data:, will attempt to lookup a an associated Django user. If one is not found, returns None.

persist(self, user, token, user_data):
Creates or updates the user association object, and if available updates the Django user's email from the Facebook user's data.

handle_no_user(self, request, access, token, user_data):
The default implementation simply returns `create_user()`.

login_user(self, request, user):
The default implementation assumes Django's model backend, and will log the user in via that backend's login method.

handle_unauthenticated_user(self, request, user, access, token, user_data):
Given a valid user, the user is first logged in, and then their Facebook data is created or updated through persist. Finally the session's expiration is set to match the expiration of the Facebook auth token.

update_profile_from_graph(self, request, access, token, profile):
Given a profile object, will try to update any fields whose names match the Facebook usergraph object.

create_profile(self, request, access, token, user):
if `AUTH_PROFILE_MODULE` is set, will attempt to create and then update a profile object for the given user.

create_user(self, request, access, token, user_data):
If the user does not already exist, creates a user, a profile if available, and logs the user in.

CHAPTER 4

Running The Test Project

1. Make sure you have Django 1.2 or greater and oauth2 at 1.5.163 or greater installed
2. Change directory to la_facebook/test_project
3. python manage.py syncdb
4. python manage.py runserver localhost:8000
5. Open a browser and point to that URL
6. Manually do the tests via your browser

4.1 Logging

By default the project will print debug level logging info both to stdout and to a log file located at /tmp/la_facebook.log.

4.2 Test Coverage

The test project supports coverage via django-coverage. To enable it you will need on your python path:

```
coverage==3.4
django-coverage==1.1.1
```

To run coverage at the command-line:

```
python manage.py test_coverage la_facebook
```

CHAPTER 5

Background

This documents provides a quick overview of the events involved in authenticating a user against facebook.

Facebook has had various authentication methods in the past (Facebook Connect), but has currently standardized on using OAuth 2 Protocol.

Facebook's own documentation does a reasonable job of explaining the process, but it is summarized here. Facebook offers two workflows for user authentication, client (javascript) based, and server side. This project aims to provide a Python based, server side option.

There are three parts to Facebook's authentication:

- user authentication (ensures that the user is who they say they are)
- app authorization (ensures that the user knows exactly what data and capabilities they are providing to your site)
- app authentication (ensures that the user is giving their information to your app and not someone else)

Facebook will only authenticate a user in relation to a specific app, there is no "just authorize the user" option. In our case, the "app" that is authenticated is your entire Django project or site, not a specific Django app. For Facebook to associate your site with the authentication, you will need an app ID from Facebook's Developer app, which is placed in your Django settings file.

A user is authenticated to facebook, and your app in one step. The first time this happens, the user will be prompted to approve the level of access you are asking for. By default and at a minimum this includes the basic info that is publicly available about the user on Facebook. (for additional permissions, see the documentation for settings). The permissions approved will be global to all your Django apps in your Django project.

Once your site is authorized by Facebook, an authorization token is stored in a model associated with a Django user (which by default is created if needed). That user that will then be the authenticated user for subsequent requests (request.user).

CHAPTER 6

Goals

Security is HARD. Security is DANGEROUS. Doing authentication via a third-party JavaScript library is STUPID.

Yet authentication via Facebook's JavaScript library is all over the place.

The better way would be to do authentication via Facebook-flavored OAUTH on the backend. With a well documented, testable project complete with working code examples. The working code examples should be in a dirt simple test project. The test project allows a developer to quickly analyze why facebook auth is failing without the complications of working in their entire system stack.

Our goals:

1. Good documentation that will build on readthedocs.org.
2. Proper logging for debug and intrusion analysis
3. Working test projects.
4. Working tests
5. Formal releases on PyPI

CHAPTER 7

Credits

7.1 django-la-facebook team

- Alexandros Bantis
- Daniel Greenfeld
- David Peters
- Jacob Burch
- Preston Holmes

7.2 django-oauth-access team

A lot of the core code of this project was liberally lifted from django-oauth-access. We want to extend all gratitude and thanks to the guys who made that excellent project.

- Brian Rosner
- Patrick Altman

CHAPTER 8

TODO

8.1 Code

1. User name determination rules
2. more examples of callbacks

8.2 Docs

1. Core readme
2. docs/usage.txt (finish)
3. access
4. callbacks
5. exceptions
6. our settings
7. views

CHAPTER 9

External Links

Project homepage:

https://github.com/cartwheelweb/django-la-facebook

PyPI:

http://pypi.python.org/pypi/django-la-facebook

Django Packages:

http://djangopackages.com/packages/p/django-la-facebook/

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

l

Symbols

B

D

F

H

I

L

R