django-facebook-graph Documentation

Release 0.1

FEINHEIT GmbH

Mar 29, 2017

Contents

Contents:

CHAPTER 1

Installation

Add `'facebook'` to your `INSTALLED_APPS`

This will create Django classes for the main Facebook models. The classes synchronize to Facebook only one-way.

Add the middlewares

The `SignedRequestMiddleware` is the main middleware that stores the signed request in a special session object and allows your app to access it. Most of the framework expects this middleware to be installed to function correctly.

The `AppRequestMiddleware` adds some tools to help dealing with app requests:

```
MIDDLEWARE_CLASSES = (
    <other middlewares>,
    'facebook.middleware.SignedRequestMiddleware',
    'facebook.middleware.AppRequestMiddleware',
)
```

Add the URLs

The basic URLconf entry adds the channel URL, the deauthorize view and some debug tools:

```
url(r'^facebook/', include('facebook.urls')),
```

The registration backend URL activates login functionality through Facebook Connect:

```
url(r'^accounts/', include('facebook.backends.registration.urls')),
```

The App Settings Dict

This dict stores all the details that facebook provides. You should have an entry for every app in your project. It is recommended to use different app (and therefore a different version of this dict) for local development:

```
FACEBOOK_APPS = {
    'name' : {
            'ID': '??????????',
            'SECRET': '??????????',
            'CANVAS-PAGE': 'https://apps.facebook.com/yourapp',
            'CANVAS-URL': '',
            'SECURE-CANVAS-URL': '',
            'REDIRECT-URL': '',
            'DOMAIN' : 'localhost.local:8000',
    }
}
```

The Facebook Javascript SDK

For any client side Facebook integration you need the Javascript SDK.

Add the fb namespace to the <html> tag:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:fb="https://www.facebook.com/2008/
↪fbml">
```

Add this to the header section of your base template:

```
{% load fb_tags %}
<script type="text/javascript">
    FACEBOOK_APP_ID = '{% fb_app_id %}';
    FACEBOOK_REDIRECT_URL = '{% fb_redirect_url %}';
    FACEBOOK_CHANNEL_URL = '{% url channel %}';
</script>
<script type="text/javascript" src="{{ STATIC_URL }}facebook/fb_utils.js"></script>
```

Or this if you use the FeinCMS page extension to discern between different Facebook applications in one installation:

```
{% load fb_tags %}
<script type="text/javascript">
    FACEBOOK_APP_ID = '{% fb_app_id feincms_page.facebook_application %}';
    FACEBOOK_REDIRECT_URL = '{% fb_redirect_url feincms_page.facebook_application %}';
    FACEBOOK_CHANNEL_URL = '{% url channel %}';
</script>
<script type="text/javascript" src="{{ STATIC_URL }}facebook/fb_utils.js"></script>
```

Add this to the bottom of your base template in the scripts section:

```
<div id="fb-root"></div>
<script type="text/javascript">
(function() {
    var e = document.createElement('script'); e.async = true;
    e.src = document.location.protocol +
    '//connect.facebook.net/de_DE/all.js';
    document.getElementById('fb-root').appendChild(e);
```

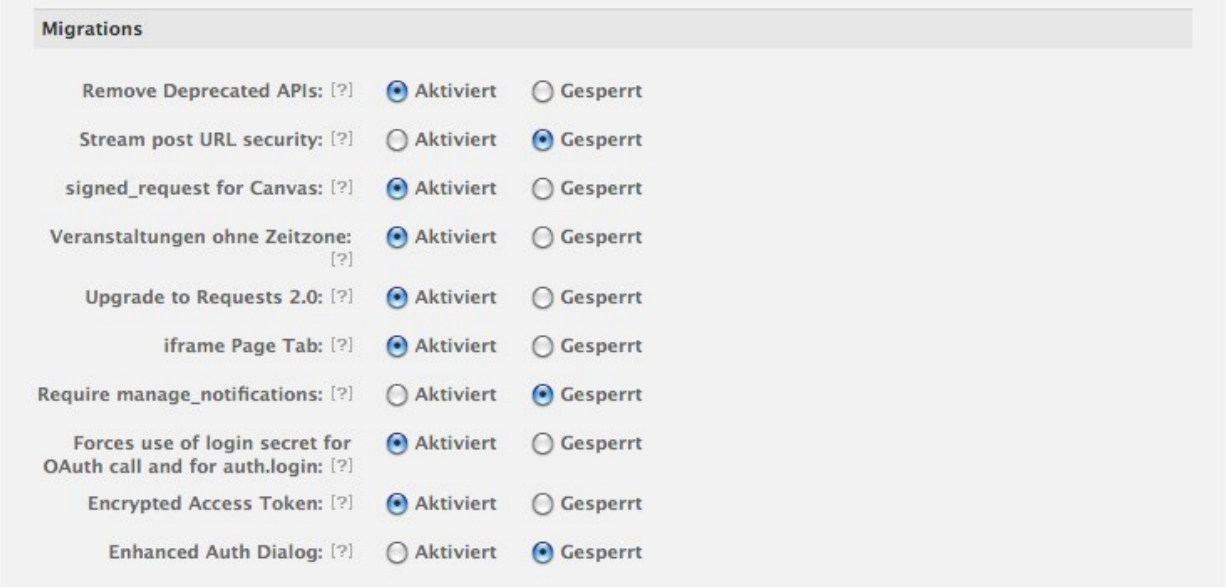
```
}());
</script>
```

The Facebook script is loaded asynchronously. Therefore you have to use the FQ, a simple script queue for inline javascript code that needs the Facebook object. The FQ is run when the SDK has been loaded and the user login status determined. Adding code which is run as soon as the Facebook API is ready is simple:

```
FQ.add(function() {
    // your code here
});
```

Create a Facebook App

Create a new Facebook app on https://developers.facebook.com/apps. You need to have a verified Facebook account. If you don't, Facebook will ask you to verify your account. `django-facebook-graph` uses OAuth 2.0. Activate it in your app settings:



Local Facebook development

If you want to develop locally, follow these steps:

- Create a separate app and set http://localhost.local:8000/ as site URL.
- Map localhost.local to 127.0.0.1 in your `/etc/hosts` file (`/private/etc/hosts` on OS X)

Now you can open your app on Facebook and it will load the data from your runserver. On Firefox you can even chose 'Open Frame in new tab' for quicker page reloads, once you've opened the page in Facebook and the cookie is set.

For Facebook connect, make sure you use the URL localhost.local:8000 and not localhost:8000. This will not work. Facebook enforces the Site URL.

Facebook Connect support for your website

The Facebook Connect support consists of two parts: A backend for django-registration which creates users and an authentication backend which is responsible for the actual login on a Django website.

Setting the authentication backend

We want to handle logins with the default backend first and fall back to the Facebook authentication backend if the default backend couldn't handle the login request:

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'facebook.backends.authentication.AuthenticationBackend',
)
```

Currently `django-facebook-graph` only supports Facebook Connect with the Login Button. The Registration Widget is not supported.

CHAPTER 2

Getting started with django-facebook-graph

You need to create a Facebook Application on Facebook Developers for nearly every functionality of `django-facebook-graph`.

> https://developers.facebook.com/apps

For detailed installation instructions check out the *Installation* section.

Facebook Connect support for your website

Currently the framework supports user login with the facebook login button. It's fairly plug and play. Make sure you have added the authentication backend and login URL as described *Installation* instructions.

Adding the Facebook login to your website

Make sure your `<html>` tag contains the necessary XML namespace information and these FMBL tag to the place where you want the login button to appear:

```
<fb:login-button scope="email" onlogin="window.location.href='{% url auth_login %}?
↪next=/'"></fb:login-button>
```

Checkout the facebook documentation on the login button: http://developers.facebook.com/docs/reference/plugins/login/

Using the Graph API

You can generate a Graph instance with the following command:

```
from facebook.utils import get_graph
graph = get_graph(request)
```

To make a Graph request to Facebook, simply use `graph.request()`. I.e. to get a certain message object:

```
fb_message = graph.request('%s' % post_id)
```

You can also create facebook user objects like so:

```
from facebook.models import User
user = User(id=graph.me['id'])
user.get_from_facebook(graph=graph, save=True)
```

`django-facebook-graph` stores as much data as possible in the session to minimize requests to Facebook. You can access the session class directly to get informations about the current user do this:

```
from facebook.utils import get_session
fb = get_session()
signed_request = fb.signed_request
```

About the Access Token

Facebook distinguishes between the app access token and the user access token. A user access token is needed for requests that need a user's permission. It's generally more powerful than an app access token. You should generally get the user access token when you pass the request argument to the `get_graph` function.

However, some operations require the app access token, like deleting app requests or saving user score. You can implicitly get the app access token by just calling `graph = get_graph()` without providing the request object, or explicitly by calling `get_static_graph()`.

Sending posts onto a Facebook wall

Ensure your app has sufficient permissions

This snippet can be used to ask for the `publish_stream` extended permission:

```
function get_publish_perms(callback_fn) {
    FB.login(function(response) {
        if (response.session) {
            if (response.perms) {
                // fb.perms.push(response.perms);
                if (response.perms.indexOf('publish_stream') != -1) {
                    callback_fn();
                }
            } else {
                alert('No permission.');
            }
        } else {
            // user is not logged in
        }
    }, {perms:'publish_stream'});
}
```

To determine whether a permission is already provided you culd use the following snippet:

```
FB.getLoginStatus(function(response) {
    fb.loginStatus = response;
    fb.perms = $.parseJSON(response.perms).extended;
}, true);
```

The second parameter, `true` causes a reload of the login status. This adds the permissions to the response too, which is very helpful for us.

The permissions are also loaded into the fb object on page load. So you could also just try:

```
if (fb.perms.indexOf('publish_stream') != -1) {
    post_to_wall();
} else {
    FB.login(function(response){
              },
              {perms: 'publish_stream' }
    );
}
```

The logical consequence if the if-statement fails would be to make a call to `FB.login()` to show a login window. The problem here is that most browsers block the popup if it doesn't follow an immediate user action. It is therefore recommended to attach the above function to a click event on a button.

Actually create a Facebook wall post

Now that everything else is taken care of actually creating the wall post is easy:

```
from facebook.utils import get_graph
def my_view(request, ...):
    graph = get_graph(request)
    graph.put_wall_post('Hello World!', {
        'name': 'Link name',
        'link': 'http://www.example.com/at/this/location/',
        })
```

It might still be a good idea to enclose the `put_wall_post` call in `try..except` clause.

Keep in mind that if too many users remove a wallpost that had been created through the Graph API your app will get classified as spam.

CHAPTER 3

Django-facebook-graph reference

Client side (Javascript) reference

The `FB` object

This is the object generated by the Facebook Javascript SDK. It's available once the SDK is loaded and provides all the methods to interact with Facebook. Check out the Facebook documentation at http://developers.facebook.com/docs/reference/javascript/.

The `fb` object

This is a helper object generated by our facebook app. If the user is logged in it stores some useful informations such as user info and permissions. Feel free to add your own attributes. By default it provides the following attributes if the user is logged in:

- **`auth`**
  - `accessToken`: The user access token.
  - `expiresIn`: Seconds until the access token expires.
  - `signedRequest`: The signed request. SHA-1 encrypted.
  - `userID`: The current user's Facebook ID.
- `get_perms(callback_fn)` This calls the callback function with a list containing the user's permissions. The reason to use this instead of `FB.api` is that the permissions are cached.
- `status` The status of the User. One of `connected`, `notConnected` or `unknown`.

CHAPTER 4

Use Cases for django-facebook-graph

Facebook Connect with Django

You can automatically login a user that has already been authenticated. Add this to your `login.html` template:

```
{% block extra-js %}
{% if not request.user.is_authenticated %}
<script type="text/javascript">
FQ.add(function(){
    if(fb.status == 'connected') {
        window.location = '{% url fb_connect %}?next={% url myview %}';
    }
});
</script>
{% endif %}
{% endblock %}
```

For views that require a logged in user you could either check the status on the server with `graph.me`, or better, on the client side with the following piece of code:

```
{% if request.user.is_authenticated %}
FQ.add(function(){
    if (fb.status == 'not_authorized') {
        window.location = '{% url fb_logout %}';
    }
});
{% endif %}
```

The advantage of checking the status in the browser is that the response time is usually shorter.

App Tabs and Facebook login

It is not easy to do a deeplink into an app tab. Facebook doesn't really support it. The only workaround is using the *Redirect2AppDataMiddleware* and calling facebook/redirect with the url urlencoded as app_data parameter.

That's how it looks:

```
'REDIRECT-URL': 'http://apps.facebook.com/<MY_APP_NAMESPACE>/facebook/redirect/?
↪next=http%3A%2F%2Fwww.facebook.com%2F<FB_PAGE>%3Fsk%3Dapp_<APP_ID>%26app_data%3D%2
↪<DEEPLINK_URL>%2F',
```

Make sure you have the *canvas url* parameters in the developer app set to the root or wherever facebook should fetch the redirect from.

Login while keeping the App requests

On some browsers you have to make sure that all protocols match. I.e. if your iframe is loaded via https you cannot redirect to a http url. The problem here is that if Facebook itself is on a http url the redirect will be to the http url as well.

```
<fb:login-button scope="email" onlogin="login_redirect();"></fb:login-button>

function login_redirect(){
    window.location = ('https:' == location.protocol ? 'https://' : 'http://')+
↪'yourdomain.com{% url join_team %}{% if request.GET.request_ids %}?request_ids={{␣
↪request.GET.request_ids }}{% endif %}';
}
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search