django-fitbit Documentation

Release 0.3.0

Issac Kelly, Rebecca Lovewell, Dan Poirier, Percy Perez

Jun 27, 2017

Contents

Contents:

CHAPTER 1

Getting started

1. Add *django-fitbit* to your Django site's requirements, however you prefer, and install it. It's installable from PyPI.
2. Add *fitapp* to your INSTALLED_APPS setting:

```
INSTALLED_APPS += ['fitapp']
```

3. Add the *django-fitbit* URLs to your URLconf:

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

3. Register your site at the Fitbit developer site to get a key and secret.
4. Add settings for *FITAPP_CONSUMER_KEY* and *FITAPP_CONSUMER_SECRET*:

```
FITAPP_CONSUMER_KEY = '9898XH'
FITAPP_CONSUMER_SECRET = 'abcdefg123456'
```

5. If you need to change the defaults, add settings for *FITAPP_LOGIN_REDIRECT*, *FITAPP_LOGOUT_REDIRECT*, and/or *FITAPP_ERROR_TEMPLATE*.
6. To display whether the user has integrated their Fitbit, or change a template behavior, use the *is_integrated_with_fitbit* template filter. Or in a view, call the *fitapp.utils.is_integrated()* function. You can also use the decorator *fitapp.decorators.fitbit_integration_warning()* to display a message to the user when they are not integrated with Fitbit.
7. To send the user through authorization at the Fitbit site for your app to access their data, send them to the *fitapp.views.login()* view.
8. To get step data for a user from a web page, use the AJAX *fitapp.views.get_steps()* view.
9. If you are using sqlite, you will want to create a celery configuration that prevents the fitapp celery tasks from being executed concurrently. If you are using any other database type, you can skip this step.

CHAPTER 2

Settings

FITAPP_CONSUMER_KEY

The OAuth 2.0 client id assigned to your app by Fitbit when you register your app at the Fitbit developer site. You must specify a non-null value for this setting.

FITAPP_CONSUMER_SECRET

The secret that goes with the FITAPP_CONSUMER_KEY. You must specify a non-null value for this setting.

FITAPP_VERIFICATION_CODE

The verification code fitbit assigns to your app for the purpose of verifying subscriber endpoints. This is optional, and is only needed if you plan on subscribing to user data updates. To use this feature, add a subscriber using the Fitbit developer interface. Fitbit will provide you with a verification code to use here. Once you have deployed the code, you can click "Verify" on Fitbit to verify it. We recommend you keep this verification code in place as long as you are using the subscriber so that if any changes are made, reverification happens automatically.

FITAPP_LOGIN_REDIRECT

Default `'/'`

The URL which to redirect the user to after successful Fitbit integration, if no forwarding URL is given in the 'fitapp_next' session variable.

FITAPP_LOGOUT_REDIRECT

Default `'/'`

The URL which to redirect the user to after removal of Fitbit account credentials, if no forwarding URL is given in the 'next' GET parameter.

FITAPP_SUBSCRIBE

Default `False`

When this setting is True, we will subscribe to user data. Fitbit will send notifications when the data changes and we will queue tasks to get the updated data. When requests for fitbit data are made to fitapp, we will always pull the latest data from our own database instead of getting it directly from Fitbit. To use this feature, you will need to setup a celery worker to handle the tasks. Following celery's guide for Django will get you started.

FITAPP_SUBSCRIBER_ID

This setting is only applicable if *FITAPP_SUBSCRIBE* is True. This is the unique ID of the subscriber endpoint that was set up for your Fitbit app on their developer site.

FITAPP_ERROR_TEMPLATE

Default `'fitapp/error.html'`

The template used to report an error integrating the user's Fitbit.

FITAPP_DECORATOR_MESSAGE

Default `'This page requires Fitbit integration.'`

The default message used by the *fitapp.decorators.fitbit_integration_warning()* decorator to inform the user about Fitbit integration. If a callable is provided, it is called with the request as the only parameter to get the final value for the message.

CHAPTER 3

Views, decorators, and templates

There are several views and decorators your site will use to drive Fitbit integration.

fitapp.decorators.**fitbit_integration_warning**(*msg=None*)

Adds a message to inform the user about Fitbit integration if their account is not already integrated with Fitbit.

Parameters **msg** – The message text to use if the user is not integrated. If msg is a callable, it is called with the request as the only parameter. Otherwise, msg is passed in as the message text. If no message is provided, the value in *FITAPP_DECORATOR_MESSAGE* is used.

This decorator does not prevent the user from seeing the view if they are not integrated with Fitbit - it only adds a message to the user's messages. If you would like to change the behavior of your view based on whether the user is integrated, you can check the user's status by using *fitapp.utils.is_integrated()*.

Example:

```
from django.http import HttpResponse
from django.contrib.auth.decorators import login_required
from fitapp.decorators import fitbit_integration_warning

@fitbit_integration_warning(msg="Integrate your account with Fitbit!")
@login_required
def my_view(request):
    return HttpResponse('Visible to authenticated users regardless' +
            'of Fitbit integration status')
```

In this example, the `fitbit_integration_warning` decorator only operates if the user is logged in. The view content is visible to all users who are logged in, regardless of Fitbit integration status.

The template(s) for any view with this decorator should display a user's messages.

fitapp.views.**login**(*request*, **args*, ***kwargs*)

Begins the OAuth authentication process by obtaining a Request Token from Fitbit and redirecting the user to the Fitbit site for authorization.

When the user has finished at the Fitbit site, they will be redirected to the *fitapp.views.complete()* view.

If 'next' is provided in the GET data, it is saved in the session so the *fitapp.views.complete()* view can redirect the user to that URL upon successful authentication.

URL name: *fitbit-login*

fitapp.views.**complete**(*request*, **args*, ***kwargs*)

After the user authorizes us, Fitbit sends a callback to this URL to complete authentication.

If there was an error, the user is redirected again to the *error* view.

If the authorization was successful, the credentials are stored for us to use later, and the user is redirected. If 'next_url' is in the request session, the user is redirected to that URL. Otherwise, they are redirected to the URL specified by the setting *FITAPP_LOGIN_REDIRECT*.

If *FITAPP_SUBSCRIBE* is set to True, add a subscription to user data at this time.

URL name: *fitbit-complete*

fitapp.views.**error**(*request*, **args*, ***kwargs*)

The user is redirected to this view if we encounter an error acquiring their Fitbit credentials. It renders the template defined in the setting *FITAPP_ERROR_TEMPLATE*. The default template, located at *fitapp/error.html*, simply informs the user of the error:

```
<html>
    <head>
        <title>Fitbit Authentication Error</title>
    </head>
    <body>
        <h1>Fitbit Authentication Error</h1>

        <p>We encontered an error while attempting to authenticate you
        through Fitbit.</p>
    </body>
</html>
```

URL name: *fitbit-error*

fitapp.views.**logout**(*request*, **args*, ***kwargs*)

Forget this user's Fitbit credentials.

If the request has a *next* parameter, the user is redirected to that URL. Otherwise, they're redirected to the URL defined in the setting *FITAPP_LOGOUT_REDIRECT*.

URL name: *fitbit-logout*

fitapp.views.**get_steps**(*request*, **args*, ***kwargs*)

An AJAX view that retrieves this user's step data from Fitbit.

This view has been deprecated. Use *get_data* instead.

URL name: *fitbit-steps*

CHAPTER 4

Template tags

is_integrated_with_fitbit

fitapp.templatetags.fitbit.**is_integrated_with_fitbit**(*user*)

Returns `True` if we have Oauth info for the user.

For example:

```
{% if request.user|is_integrated_with_fitbit %}
    do something
{% else %}
    do something else
{% endif %}
```

CHAPTER 5

Utilities

is_integrated

fitapp.utils.**is_integrated**(*user*)

Returns `True` if we have Oauth info for the user.

This does not require that the token and secret are valid.

Parameters **user** – A Django User.

CHAPTER 6

Management commands

refresh_tokens

This django management command can be used to refresh user access tokens. Running without arguments will refresh only the tokens that are expired.

Using the `--all` option will refresh all access tokens, whether expired or not.

Using the `--deauth` option tells the command to remove the `UserFitbit` object for any tokens that fail to refresh for whatever reason. This can be handy to prune `UserFitbit` objects that have somehow managed to get an invalid refresh token (an unrecoverable state).

CHAPTER 7

Links to related information

- Django-fitbit home
- Fitbit developer site
- Python-fitbit library

CHAPTER 8

Release history

0.3.0 (2017-01-25)

- More consistently refresh tokens when needed
- Added refresh_tokens command to manually refresh user fitbit tokens

0.2.6 (2016-12-14)

- Add exponential back-off and random jitter to task retries
- Enable some configuration around subscriptions: - FITAPP_SUBSCRIPTIONS: List exactly which subscriptions to retrieve and the order to retrieve them in - FITAPP_HISTORICAL_INIT_DELAY: The initial delay (in seconds) to wait before retrieving any historic data - FITAPP_BETWEEN_DELAY: The delay (in seconds) to wait between retrieving each type of resource

0.2.5 (2016-11-03)

- Add docstrings to all models, help_text to all fields

0.2.4 (2016-05-04)

- More refresh token bugfixes

0.2.2 (2016-03-30)

- Refresh token bugfixes

- Use fitbit==0.2.2

0.2.0 (2016-03-23)

- Integrate with python-fitbit OAuth2 (fitbit==0.2)
- Update documentation to state that *FITAPP_CONSUMER_KEY* should be the OAuth 2.0 client id

0.1.2

- Enable fitbit subscriber verification
- Better error handling in celery tasks

0.1.1

- Fix packaging issue (missing fixture data)

0.1.0

- Support for subscribing to time series data
- Many bug fixes

0.0.1

- Initial release

Django-fitbit is a Django app for integrating a user's Fitbit data into your site.

It handles the details of getting your app authorized to access your user's Fitbit data via the Fitbit web API.

CHAPTER 9

Testing

Please add tests for any changes you submit.

To install all the requirements for running the tests:

```
pip install -r requirements/dev.txt
```

To run the tests for specific python version (ie. py27-1.8.X):

```
tox -e py27-1.8.X
```

If you would like to run specific test you can bypass tox altogether and run:

```
python -m run_tests fitapp.tests.test_integration.TestLoginView.test_unauthenticated
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index