django-sanction Documentation

Release 0.3

Demian Brecht

January 14, 2014

Contents

Contents

CHAPTER 1

Overview

`django-sanction` is a Django app for the sanction Python package. As its parent package, the intention of this package is to provide an incredibly easy-to-use Django OAuth 2.0 authorization/authentication app that is easy to grok.

This package makes no assumption as to what persistence mechanism you use. As such, no default custom user models (or abc's) are implemented. Duck-typing is taken advantage of here and a specific API is required of the custom user model. This is defined below under User API. Integrating this package, defining the user model is likely where you'll spend most of your time.

note Since 0.3, this package has been refactored to be compatible with Django 1.5+. **It is not backwards-compatible with earlier Django versions**.

CHAPTER 2

Configuration

First, `django_sanction` must be added to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    'django_sanction',
    # ...
)
```

The URLs defined by `django_sanction` must be added and anchored to a root path. In your applications' urls.py:

```
urlpatterns = patterns('',
    url(r'^o/', include('django_sanction.urls')),
)
```

The above will define the URLs `'login/<provider>'` and `'logout'` under the path `/o`. For example: `http://example.com/o/login/google`.

2.1 Custom User

Whether you're adding custom fields to the `User` implementation or not, you will need to implement a custom user model (introduced in Django 1.5). This is due to the API requirements on the `User` as described in the next section. The provided Django `User` is not abstract, so you cannot easily extend it without monkey-patching or other hackery.

To define a custom user model, use the `AUTH_USER_MODEL` variable in your settings file:

```
AUTH_USER_MODEL = 'example.User'
```

2.2 Providers

Providers are defined as a dict (`SANCTION_PROVIDERS`) with the following fields:

- `auth_endpoint`: The providers' authorization page
- `token_endpoint`: The URL where token exchange takes place
- `resource_endpoint`: Where a `sanction` client can request resources from

- client_id: Your applications' client ID, as generated by the provider
- client_secret: The applications' secret, generated by the provider
- redirect_uri: The expected URL that the provider should return the user to upon authorization.
- scope: An iterable of authorization items required by your application.
- auth_params: Additional key/value pairs to send to the authorization page.

Each entry should be keyed with a unique identifier that will be used throughout the rest of the application:

```
SANCTION_PROVIDERS = {
    'google': {
        'auth_endpoint': 'https://accounts.google.com/o/oauth2/auth',
        'token_endpoint': 'https://accounts.google.com/o/oauth2/token',
        'resource_endpoint': 'https://www.googleapis.com/oauth2/v1',
        'client_id': '421833888173.apps.googleusercontent.com',
        'client_secret': 'VueqKFZyz-aoL4rQFleEIT1j',
        'redirect_uri': 'http://localhost:8080/o/login/google',
        'scope': ('email', 'https://www.googleapis.com/auth/userinfo.profile',),
        'auth_params': {'access_type': 'offline'}
    },
    'facebook': {
        'auth_endpoint': 'https://www.facebook.com/dialog/oauth',
        'token_endpoint': 'https://graph.facebook.com/oauth/access_token',
        'resource_endpoint': 'https://graph.facebook.com',
        'scope': ('email',),
        'parser': lambda data: dict(parse_qsl(data)),
        'client_id': '152107704926343',
        'client_secret': '80c81e4d7d5bc68ecc8cf1da0213382e',
        'redirect_uri': 'http://localhost:8080/o/login/facebook',
    }
}
```

2.3 Authentication Backends

Django has to know about the backend to be used in order to authenticate OAuth 2.0 users. The following example will run the user through django-sanction's authentication backend and then will fall back to the built-in Django backend (should be added to your settings file):

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

2.4 Login and Logged In URLs

While not strictly required, LOGIN_URL and LOGIN_REDIRECT_URL should be defined in your application settings. The former defines where a user may log in from and the latter defines where a user is redirected to upon successful authentication:

```
LOGIN_URL = '/'
LOGIN_REDIRECT_URL = '/profile'
```

CHAPTER 3

User API

As django-sanction does not make any assumptions about how you're going to persist your data, there is not a generic way to provide an interface. As such, we rely on duck typing. The following API *must* be implemented by your user object:

```
class User(object):
    def current_provider(self, request):
        """ Get the current provider

        Returns the current provider used by the current request's user
        """
        pass

    @staticmethod
    def fetch_user(provider, client):
        """ Fetches the user from the OAuth 2.0 provider

        This should return an instance of a ``User`` based on data from the
        provider resource.
        """
        pass

    @staticmethod
    def get_user(user_id)
        """ Returns a user object

        Retrieves an instance of a ``User`` (or ``None``) given the
        application-specific ``user_id``.
        """
        pass
```

Implementing the `User` class will likely be where you'll spend the most of your time while integrating `django_sanction`.

CHAPTER 4

Refreshing Tokens

Depending on how you use the OAuth 2.0-authorized user content, you may or may not have to worry about refreshing tokens. If you're simply using it to generate user accounts with data such as email addresses and name, then you likely won't have to worry about refreshing it. However, if your app interacts with an OAuth 2.0 provider *after* a token has expired (i.e post to a users' Facebook wall), then this is something that you'll have to worry about.

Refreshing tokens is entirely up to the application (there are far too many deviations or simply non-conformance among the providers to provide a general solution). Token refreshes themselves are outside the scope of this document. Please consult the documentation provided by the OAuth 2.0 provider.

note The example app demonstrates how to go about refreshing tokens for Facebook and Google.

CHAPTER 5

Other

5.1 Reference

5.1.1 Authentication backend

5.1.2 URLs

The URLs defined by django-sanction

django_sanction should be initialized in your project's urls.py as such:

```
urlpatterns = patterns('',
    url(r'^o/', include('django_sanction.urls')),
)
```

The prefix `o` can be replaced by any path you would like to use for the sanction auth flow. Two views are registered under this path:

- `[prefix]/logout/`, and
- `[prefix]/login/(\w+)`

  note The parameter for the login flow *must* match a key used in `SANCTION_PROVIDERS` in your project settings file.

5.1.3 Views

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index