django-linked-accounts Documentation

Release 2.0

Andrii Kurinnyi, Bryan Landers

January 26, 2012

CONTENTS

Contents:

CHAPTER

ONE

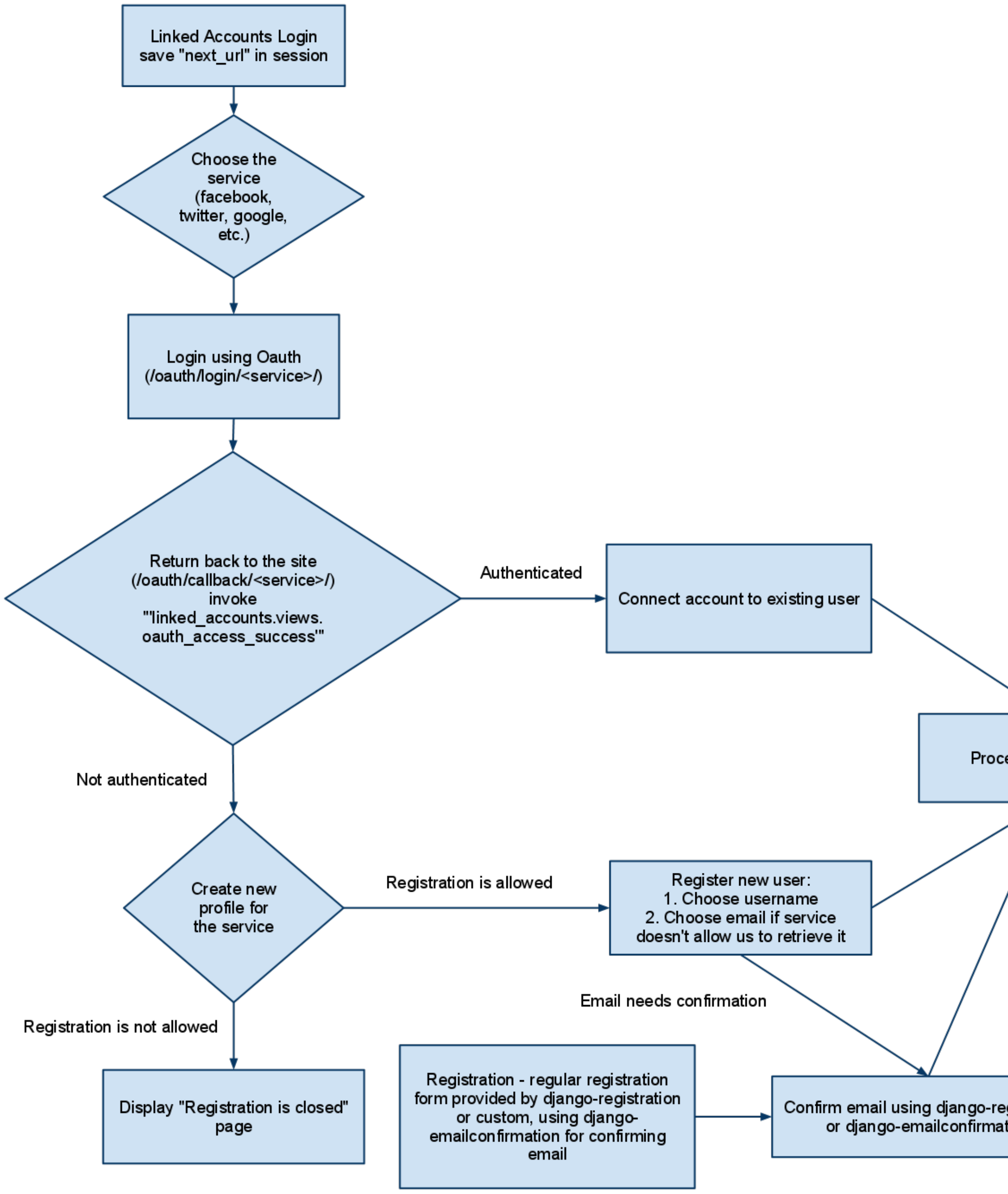
OVERVIEW

Django Linked Accounts handles linking Django auth.User accounts with OAuth-compatible third-party service accounts.

1.1 Supported Services

- Facebook
- Twitter
- Google
- Yahoo

1.2 Flow


Linked Accounts Login
save "next_url" in session
Choose the service (facebook, twitter, google, etc.)
Login using Oauth (/oauth/login/<service>/)
Return back to the site (/oauth/callback/<service>/) invoke "'linked_accounts.views. oauth_access_success'"
Authenticated
Connect account to existing user
Proce
Not authenticated
Create new profile for the service
Registration is allowed
Register new user:
1. Choose username
2. Choose email if service doesn't allow us to retrieve it
Email needs confirmation
Registration is not allowed
Display "Registration is closed" page
Registration - regular registration form provided by django-registration or custom, using django-emailconfirmation for confirming email
Confirm email using django-re
or django-emailconfirmat

This app implements following flow:

CHAPTER

TWO

INSTALLATION

Installing and integrating Django Linked Accounts requires the following steps:

1. Install Dependencies
2. Install Django App
3. Include URLs
4. Add Authentication Backend
5. Obtain Service Keys
6. Add Settings

2.1 Install Dependencies

Django Linked Accounts depends on django-oauth-flow.

You can run the following commands through `pip install` to install the necessary dependencies:

```
-e git://github.com/zen4ever/django-oauth-flow.git#egg=django-oauth-flow
httplib2>=0.6.0
oauth2>=1.5.167
```

Or use supplied `requirements.txt` file:

```
git clone git://github.com/zen4ever/django-linked-accounts.git
cd django-linked-accounts/
pip install -r requirements.txt
```

You can install the python package with the following:

```
pip install -e git://github.com/zen4ever/django-linked-accounts.git
```

2.2 Install Django App

Add `linked_accounts` to your `INSTALLED_APPS` (`settings.py`):

```
INSTALLED_APPS = (
    # ...
    "linked_accounts",
)
```

2.3 Include URLs

Include URLs for `linked_accounts` in your URLconf (`urls.py`):

```
urlpatterns = patterns("",
    # ...
    url(r"^linked_accounts/", include("linked_accounts.urls"))
)
```

2.4 Add Authentication Backend

Add the custom authentication backend to your `AUTHENTICATION_BACKENDS` (`settings.py`) to allow OAuth authentication with Django Linked Accounts.

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'linked_accounts.backends.LinkedAccountsBackend',
)
```

2.5 Obtain Service Keys

You will need to obtain a `KEY` and `SECRET` with each third-party service you want to support. Yes, it's tedious. But, hey, you only have to do it once for your app and you can piggyback on established social graphs and make things easier for your users by importing information already entered elsewhere.

Here are some handy reference links to get you started:

- Facebook: https://developers.facebook.com/apps (click the "Create New App" button...)
- Twitter: https://dev.twitter.com/apps/new
- Google: https://code.google.com/apis/console/
- Yahoo: https://developer.apps.yahoo.com/wsregapp
- LinkedIn: https://www.linkedin.com/secure/developer

Most of these services require you to register an authentication callback URL to redirect to after a user authorizes your app. Here's an example callback URL for Twitter:

```
http://yourdomain.com/linked_accounts/complete/twitter/
```

2.6 Add Settings

Lastly, add OAuth settings in your `settings.py` for each service you want to integrate with using the `KEY` and `SECRET` values you obtained in the previous installation step.

Use the following code as a reference and include only the services you want to support:

```
OAUTH_FLOW_SETTINGS = {
    'facebook': {
        'KEY': '',
        'SECRET': '',
        'SCOPE': ['email'],
    },
    'twitter': {
        'KEY': '',
        'SECRET': '',
    },
    'google': {
        'KEY': '',
        'SECRET': '',
        'SCOPE': ['https://www.googleapis.com/auth/userinfo.profile']
    },
    'yahoo': {
        'KEY': '',
        'SECRET': '',
    }
}
```

CHAPTER

THREE

GETTING STARTED

Welcome to the documentation for Django Linked Accounts. Here you'll find everything you need to get started using this reusable app in your Django projects. Good luck and thanks for reading!

3.1 Settings

The following are optional settings variables that can be included in your project's `settings.py` to override certain defaults.

3.1.1 `LINKED_ACCOUNTS_ALLOW_REGISTRATION`

Default: `True`

Set this to `False` to prohibit auth.User creation after OAuth login. This is useful if you want to only allow your users to link third-party accounts to their existing auth.User account, but not allow sign up via third-party service.

3.1.2 `LINKED_ACCOUNTS_ALLOW_LOGIN`

Default: `True`

Set this to `False` to prohibit users from logging in via OAuth. This is useful if you want to only retrieve data from a third-party service that requires OAuth authentication.

3.1.3 `LINKED_ACCOUNTS_NEXT_KEY`

Default: `oauth_next`

This setting can be used to override the default session variable key used to store `next_url` between redirects to and from OAuth services. You probably won't need to change this setting unless you use the `oauth_next` session key for something else in your project.

3.1.4 `LINKED_ACCOUNTS_ID_SESSION`

Default: `_linked_account_id`

Modify this setting to change the session key that stores the `LinkedAccount` id temporarily during sign up via third-party service.

3.1.5 `LINKED_ACCOUNTS_HANDLERS`

Default:

```
LINKED_ACCOUNTS_HANDLERS = (
    ('facebook', 'linked_accounts.handlers.FacebookHandler'),
    ('twitter', 'linked_accounts.handlers.TwitterHandler'),
    ('google', 'linked_accounts.handlers.GoogleHandler'),
    ('yahoo', 'linked_accounts.handlers.YahooHandler'),
)
```

This setting contains a set of classes responsible for retrieving user profile information for third-party services. It is inherited from `AuthHandler`, but can be overridden by supplying your own classes.

3.1.6 `LINKED_ACCOUNTS_ALWAYS_UPDATE_PROFILE`

Default: `False`

Set this to `True` if you want the data stored in `LinkedAccount.api_response` each time a user successfully logs in via OAuth. Keep the default value to only update this data once during the first OAuth login for each service.

3.1.7 `LINKED_ACCOUNTS_EMAIL_MATCH`

Default: `True`

Will attempt to look for LinkedAccount on login assuming that email is an identifier. Useful when you migrating your authentication system from OpenID, because your OpenID identifier will never match your OAuth identifier.

3.1.8 `LINKED_ACCOUNTS_AUTO_REGISTRATION`

Default: `True`

Instead of redirecting to registration form will automatically create a new user with suitable username.

3.2 Login

Django Linked Accounts contains a sample login view.

It displays the template “linked_accounts/login.html”, which contains login links for each supported service. For example:

```
<a href="{% url linked_accounts_service_login "facebook" %}">Sign in with Facebook</a>
<a href="{% url linked_accounts_service_login "twitter" %}">Sign in with Twitter</a>
```

The login view also saves the GET parameter `next` in a session variable that is used to redirect the user after successful OAuth authentication.

If you are going to use Django Linked Accounts as your main authentication mechanism, set the following in your `settings.py`:

```
LOGIN_URL = "/linked_accounts/login/"
```

Alternatively, you can pass the additional GET parameter `service` to the login view to bypass Django Linked Accounts' login template rendering and redirect you to the django-oauth-access login view, preserving `next` GET parameter in the redirect URL.

If you are planning to use Django Linked Accounts as a supplemental app to `django.contrib.auth`, for example, to link existing third-party accounts to `auth.User` accounts, you can include links in your `linked_accounts/login.html` in addition to your `auth.User` login form like this:

```
<a href="{% url linked_accounts_login %}?service=facebook&amp;next={{ next }}">Sign in with Facebook
<a href="{% url linked_accounts_login %}?service=twitter&amp;next={{ next }}">Sign in with Twitter</
```

3.3 Registration

Django Linked Accounts contains a simple registration view.

When a logged-out user successfully completes OAuth authentication with a third-party service, a new `LinkedAccount` profile is created and the user is redirected to `/linked_accounts/register/` where they can choose a username and enter their email address. If an email address was collected during OAuth authentication, it will be listed as an initial value in the registration form.

Once the registration form is submitted, a new `auth.User` is created and is logged in.

You can prohibit registration via third-party services by setting `LINKED_ACCOUNTS_ALLOW_REGISTRATION` to `False` in your `settings.py`. This will prevent the creation of new users authenticated with third-party services, which might be useful for private betas or similar. Please note that if a valid `auth.User` is already linked to a third-party service in Django Linked Accounts, login via that service will be allowed.

Django Linked Accounts provides a simple `RegistrationForm` which is used to collect each user's email address during registration. However, please note that this app does not handle email confirmation or any other transactional email notifications. If this app does not match the desired flow for your project, you can inherit and override the registration form, view, or even individual methods found in `AuthCallback` in your own custom views.

CHAPTER

FOUR

INDICES AND TABLES

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