Django Complete User Auth: Custom User model + Social Auth with Google

Feb 13, 2018

slack
Bring your team together with Slack, the collaboration hub for work.
ADS VIA CARBON

In this tutorial we'll learn how to properly start a new Django project with a custom user model and add Gmail social authentication via django-allauth. I believe these features are **must haves** in any new Django project.

- A custom user model is highly recommended in the official docs and allows for future customization.
- Adding social authentication via third-party services like Gmail, Facebook, and other services is also common. By using `django-allauth` from the beginning, we can add social auth as needed.

I want to express my appreciation upfront to the work of Raymond Penners on `django-allauth` as well as the open-source cookiecutter-django and demo-allauth-bootstrap projects.

The outline of our work is as follows:

- start a basic Django project
- add a pages app for the homepage
- implement a custom user model
- signup, login, logout
- install django-allauth
- get Google credentials
- update templates

Basic Django

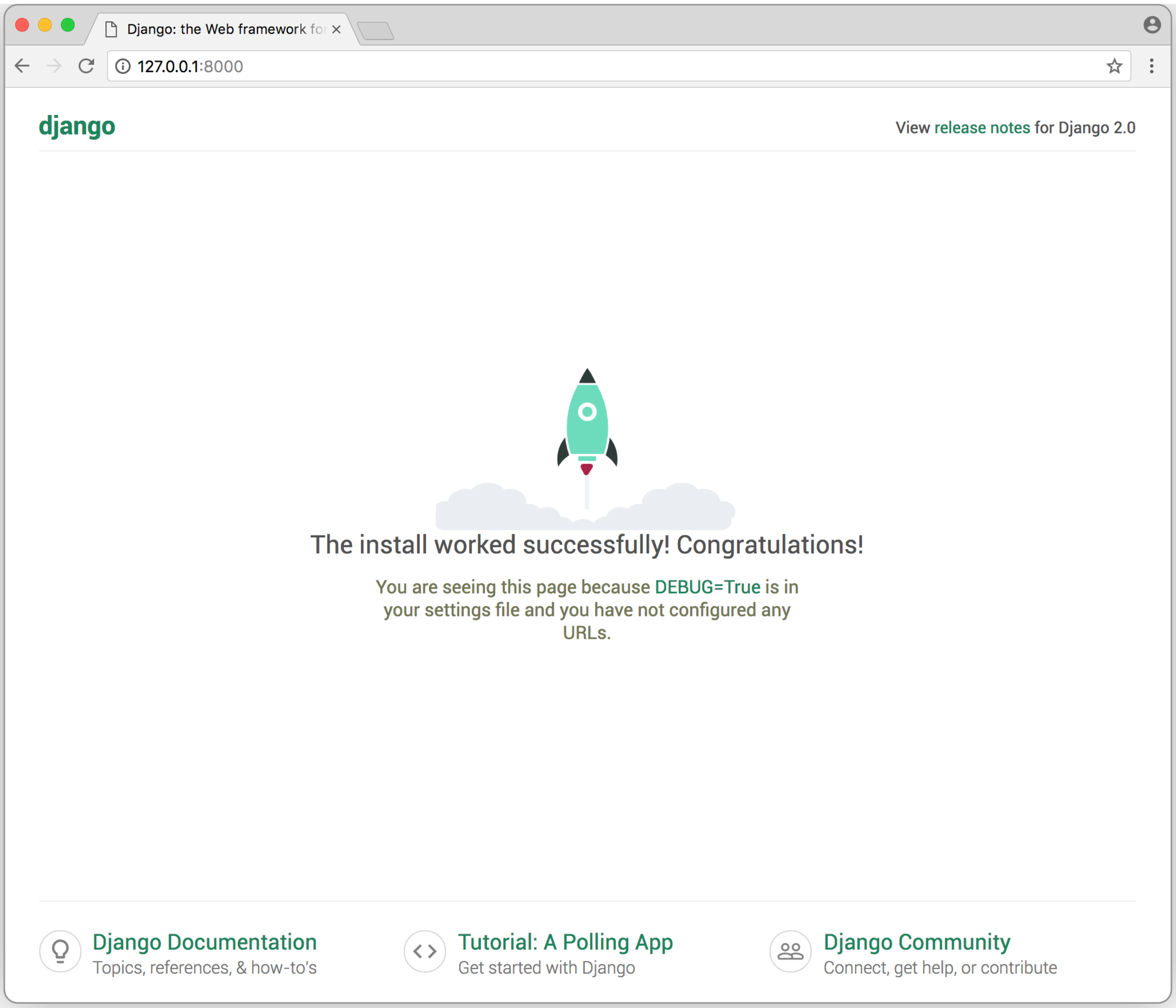
This tutorial assumes your computer is already configured to use `Pipenv` for virtual environments and has Python 3. If you need help, please review Chapter 1: Initial Setup of Django for Beginners which has a complete overview.

On the command line create a new directory `demo` for our code on the Desktop, then install and activate Django in a new virtual environment. Create a new project called `demo_project` that we install in the current directory (don't forget to include the period `.` in the command!), and start the server to confirm installation was successful.

```
$ cd ~/Desktop
$ mkdir demo && cd demo
$ pipenv install django
$ pipenv shell
(demo) $ django-admin startproject demo_project .
(demo) $ python manage.py runserver
```

Note that we very intentionally **did not** run `migrate` yet to create a new database. We must wait until the custom user model is configured before doing so for the first time. More on this shortly.

In a web browser navigate to http://127.0.0.1:8000/ and you should see the Django welcome page.


Django: the Web framework fo
127.0.0.1:8000
django
View release notes for Django 2.0
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Pages app

Now it's time for our homepage. I like to create a dedicated app for static pages like home, about, terms, privacy, etc. First stop the server with `Control+c` and then run `startapp` to make a new app `pages.`

```
(demo) $ python manage.py startapp pages
```

Add the new app to our `INSTALLED_APPS` setting and also update the templates `DIRS` setting so we can use a project-level `templates` directory in the project.

```
# demo_project/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',

    'pages',  # new
]

TEMPLATES = [
    ...
    'DIRS': [os.path.join(BASE_DIR, 'templates')],
    ...
]
```

Now make our new templates directory from the command line, add a `home.html` template, and also add a `urls.py` file we'll need in the new `pages` app.

```
(demo) $ mkdir templates
(demo) $ touch templates/home.html
(demo) $ touch pages/urls.py
```

The homepage will be deliberately basic at this point: just a string of text.

```
<!-- templates/home.html -->
<h1>Homepage</h1>
```

We also need to update the project-level `urls.py` file to "include" the `pages` app and then add the code for our `pages/urls.py` file, too.

```
# demo_project/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('pages.urls')),
    path('admin/', admin.site.urls),
]
```

```
# pages/urls.py
from django.urls import path

from .views import HomePageView

urlpatterns = [
    path('', views.HomePageView.as_view(), name='home'),
]
```

We've referenced a view called `HomePageView` but have yet to create it. Let's do that now. We can use Django's class-based generic view TemplateView here.

```
# pages/views.py
from django.views.generic import TemplateView


class HomePageView(TemplateView):
    template_name = 'home.html'
```

Now run the server again.

```
(demo) $ python manage.py runserver
```

Refresh the browser at http://127.0.0.1:8000/ and our new homepage appears!

Custom User Model

Now we'll configure a custom user model that simply extends the existing `User` model. That means it's exactly the same but we have the ability to add fields to it in the future as needed–date of birth, location, gender, etc. If we did not use a custom user model upfront, it would be very difficult to make these changes. Hence why **all new projects should use a custom user model.**

There are four steps required to implement the new custom user model.

First, create a new app called `users`. You probably need to stop the server with `Control+c` in order to run this command.

```
(demo) $ python manage.py startapp users
```

Add the new app to our `INSTALLED_APPS` setting **and** add a new setting for `AUTH_USER_MODEL`. This tells Django that instead of using the default `User` model look in our app `users` for the model called `CustomUser` and use that instead *everywhere in our project*.

```
# demo_project/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',

    'users', # new
    'pages',
]

AUTH_USER_MODEL = 'users.CustomUser' # new
```

Second we need to create our new `CustomUser` model.

```
# users/models.py
from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser(AbstractUser):
    pass
    # add additional fields in here
```

Third, various forms in Django interact with the user especially around creating and changing users. We'll update both the default `UserCreationForm` and `UserChangeForm` to point to our new `CustomUser` model.

Create a `forms.py` file in the `users` app.

```
(demo) $ touch users/forms.py
```

Then add the following code.

```
# users/forms.py
from django import forms
from django.contrib.auth.forms import UserCreationForm, UserChange
from .models import CustomUser

class CustomUserCreationForm(UserCreationForm):

    class Meta:
        model = CustomUser
        fields = ('username', 'email')

class CustomUserChangeForm(UserChangeForm):

    class Meta:
        model = CustomUser
        fields = UserChangeForm.Meta.fields
```

The fourth and final step is to update `admin.py`. The admin app gives us the power to create, modify, and delete users. Therefore we need to update it, too, to use `CustomUser` throughout.

Here's the code.

```
# users/admin.py
from django.contrib import admin
from django.contrib.auth import get_user_model
from django.contrib.auth.admin import UserAdmin

from .forms import CustomUserCreationForm, CustomUserChangeForm
from .models import CustomUser

class CustomUserAdmin(UserAdmin):
    model = CustomUser
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm

admin.site.register(CustomUser, CustomUserAdmin)
```

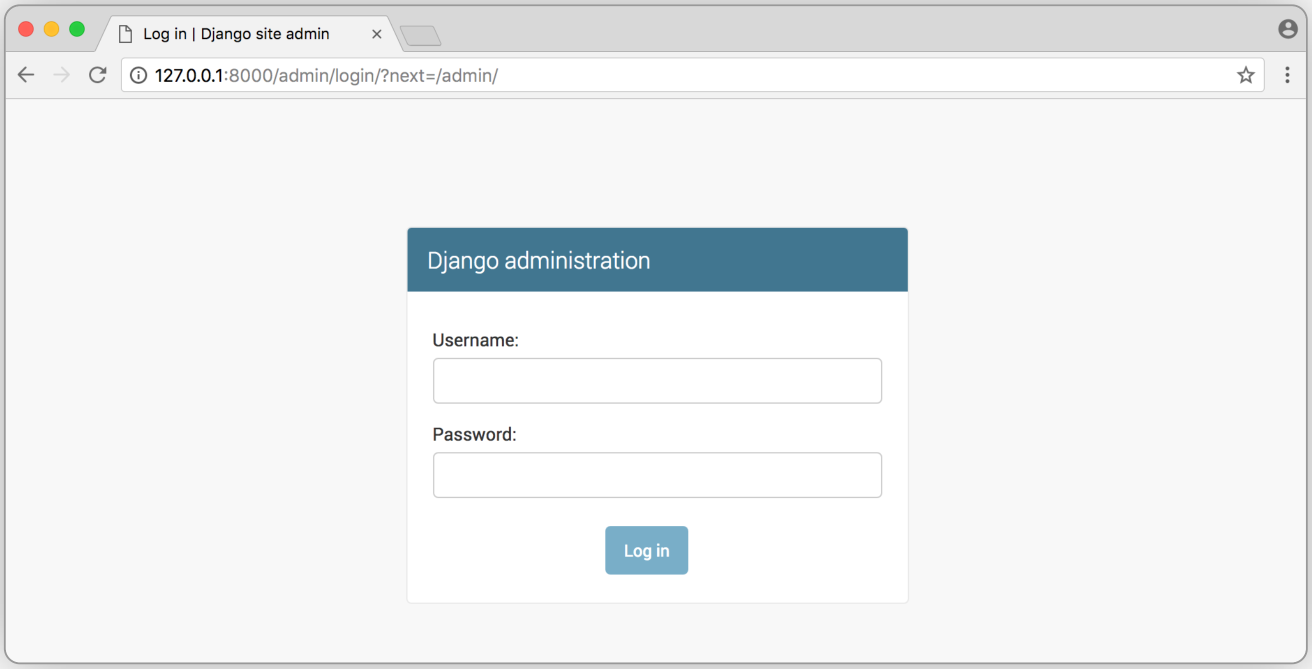
And we're done. **Now** we can create our new database for the first time.

```
(demo) $ python manage.py makemigrations users
(demo) $ python manage.py migrate users
```

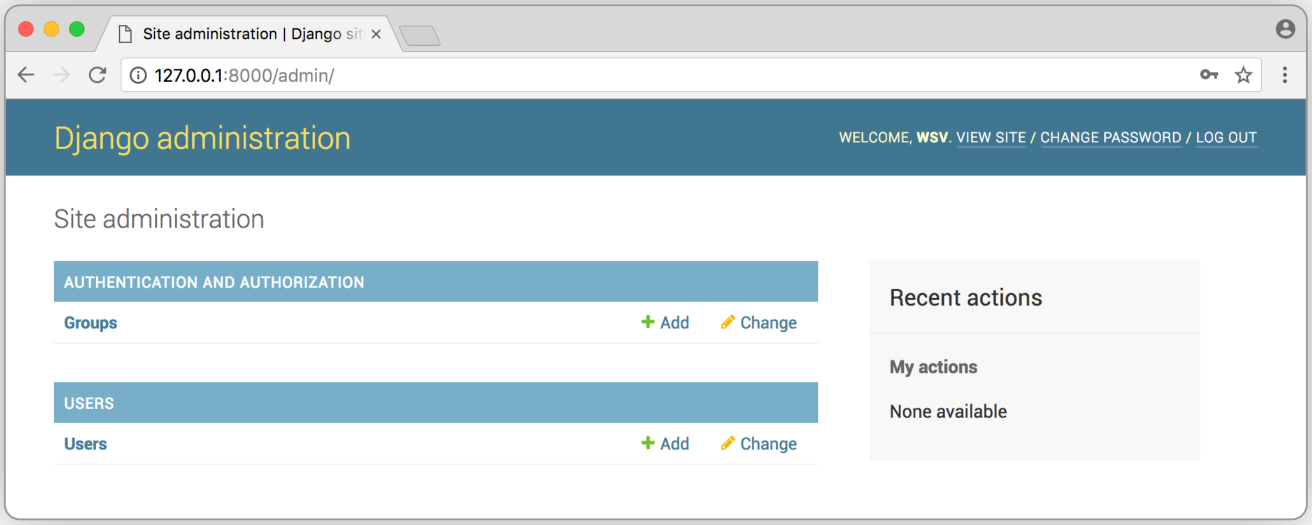
To confirm everything worked let's create a `superuser` account and login to the admin.

```
(demo) $ python manage.py createsuperuser
```

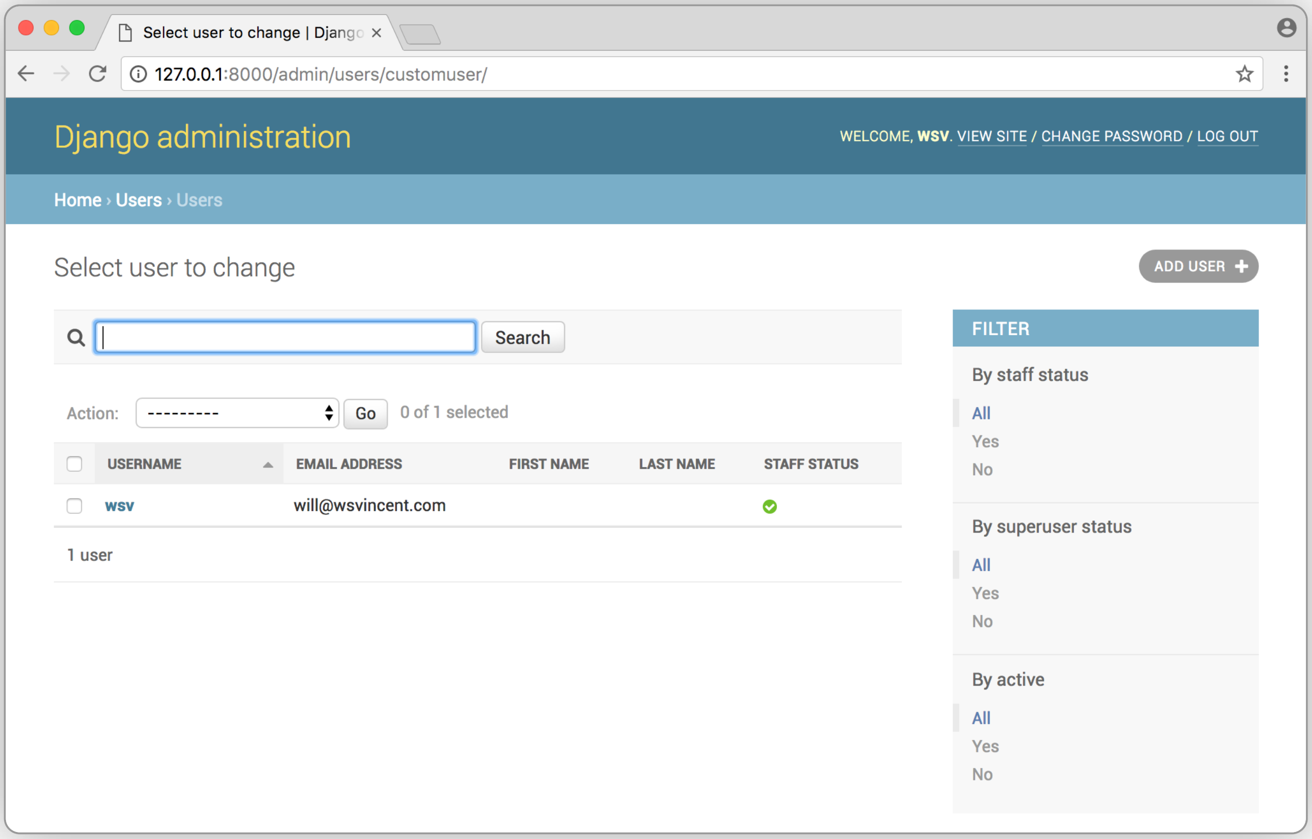
Start up the server with `python manage.py runserver` and then navigate to http://127.0.0.1:8000/admin.


Log in | Django site admin
127.0.0.1:8000/admin/login/?next=/admin/
Django administration
Username:
Password:
Log in

Log in with your superuser credentials.



Click on “Users” and you should see your superuser account.


Select user to change | Django
127.0.0.1:8000/admin/users/customuser/
Django administration
WELCOME, WSV. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Users › Users
Select user to change
ADD USER
Search
Action: --------- Go 0 of 1 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
wsv	will@wsvincent.com			✓

1 user

FILTER
By staff status
All
Yes
No
By superuser status
All
Yes
No
By active
All
Yes
No

Signup, login, logout

Our goal in this section is to update the homepage so it has working links for sign up, login, and logout functionality. We have yet to update our project-level `urls.py` for the new `users` app so let’s do that now. We also want to add Django’s built-in `auth` module.

```
# demo_project/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('pages.urls')),
    path('users/', include('users.urls')), # new
    path('users/', include('django.contrib.auth.urls')), # new
    path('admin/', admin.site.urls),
]
```

When a user logs in or logs out Django needs us to specify where to redirect them. Let’s send everyone to the homepage which has a named url of `home`. In the `settings.py` file add the following two lines at the bottom.

```
# demo_project/settings.py
LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'
```

Now update our existing homepage so that if a user is logged in it will show a personalized greeting and logout link, otherwise it will have signup and log in links.

```
<!-- templates/home.html -->
<h1>Django Login Mega-Tutorial</h1>
{% if user.is_authenticated %}
<p>Hi {{ user.username }}
<p><a href="{% url 'logout' %}">Log out</a></p>
{% else %}
<p><a href="{% url 'signup' %}">Sign Up</a></p>
<p><a href="{% url 'login' %}">Log In </a></p>
{% endif %}
```

For our login template, we need to create it at `templates/registration/login.html` which is where Django looks by default.

```
(demo) $ mkdir templates/registration
(demo) $ touch templates/registration/login.html
```

Then update it as follows:

```
<!-- templates/registration/login.html -->
<h2>Login</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Login</button>
</form>
```

Now for our sign up template. Since this does not come built-in to Django we need to create and configure our template, views, urls by hand. let's start with the template.

```
(demo) $ touch templates/signup.html
```

The signup form uses a `csrf_token` for security.

```
<!-- templates/signup.html -->
<h2>Sign up</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign up</button>
</form>
```

Create a `users/urls.py` file.

```
(demo) $ touch users/urls.py
```

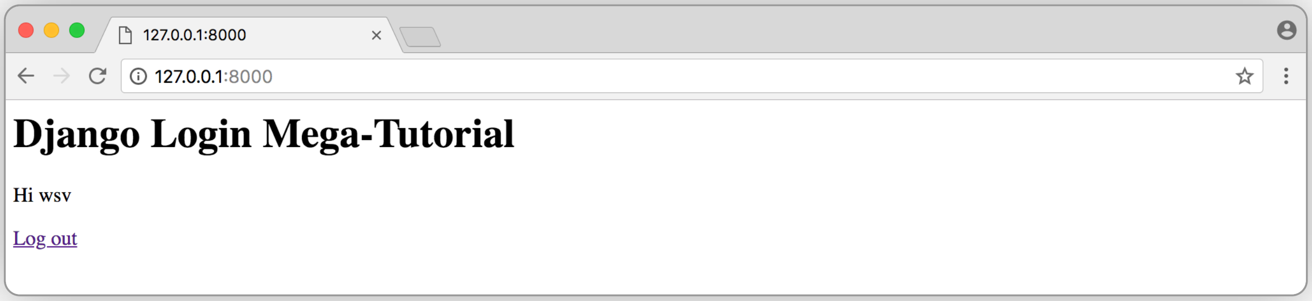
It will reference just our sign up view for now.

```
# users/urls.py
from django.urls import path
from .views import SignUpView

urlpatterns = [
    path('signup/', SignUpView.as_view(), name='signup'),
]
```

The view will extend `CreateView` and specify our `CustomUserCreationForm`, redirects to `login` upon submission, and uses our `signup.html` template.

```
# users/views.py
from django.urls import reverse_lazy
from django.views.generic.edit import CreateView

from .forms import CustomUserCreationForm

class SignUp(CreateView):
    form_class = CustomUserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'signup.html'
```

Now we can try our links out. Start up the server with `python manage.py runserver` and navigate to http://127.0.0.1:8000/. Refresh the page if you don't see the changes right away.

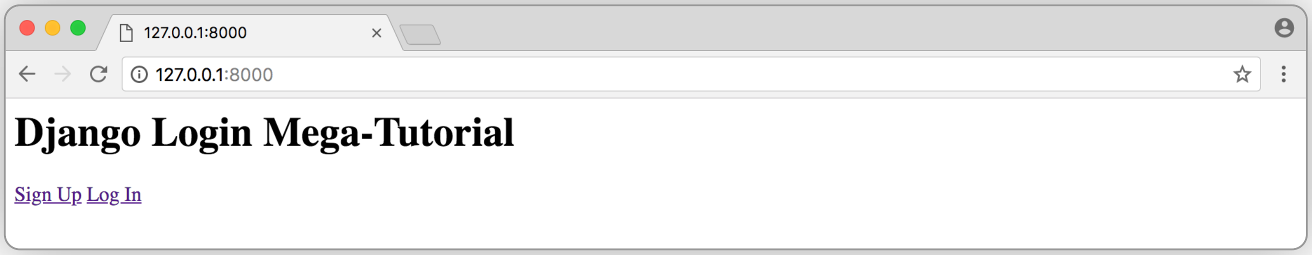


Django Login Mega-Tutorial

Hi wsv

Log out

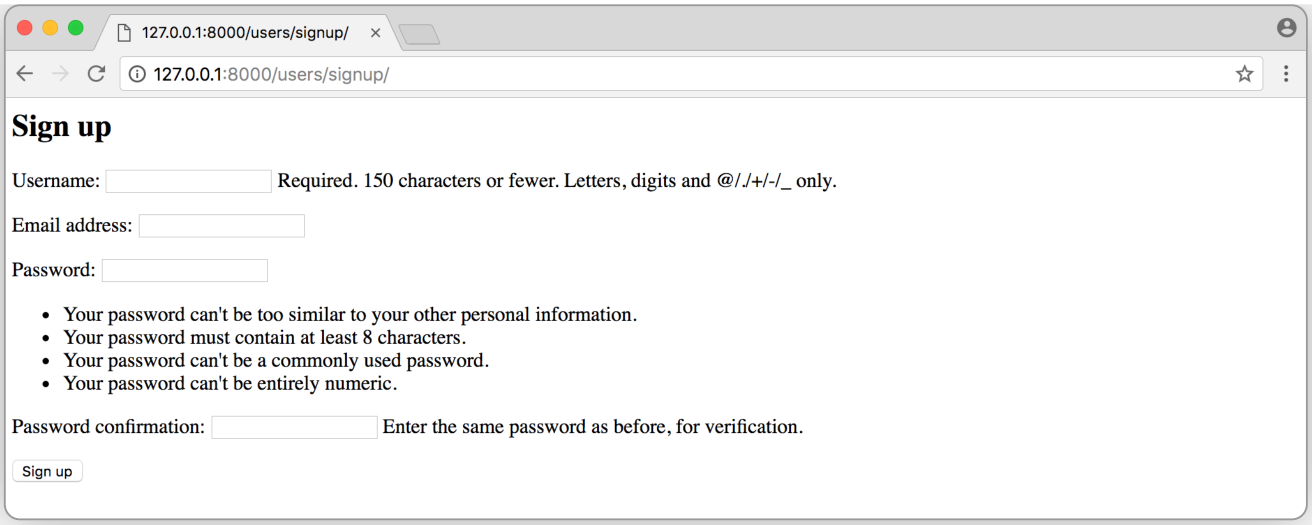
Click on the "logout" link which will redirect you to the logged-out version of the homepage.



Django Login Mega-Tutorial

Sign Up Log In

Try clicking on the "Sign Up" link.


127.0.0.1:8000/users/signup/

Sign up

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address:

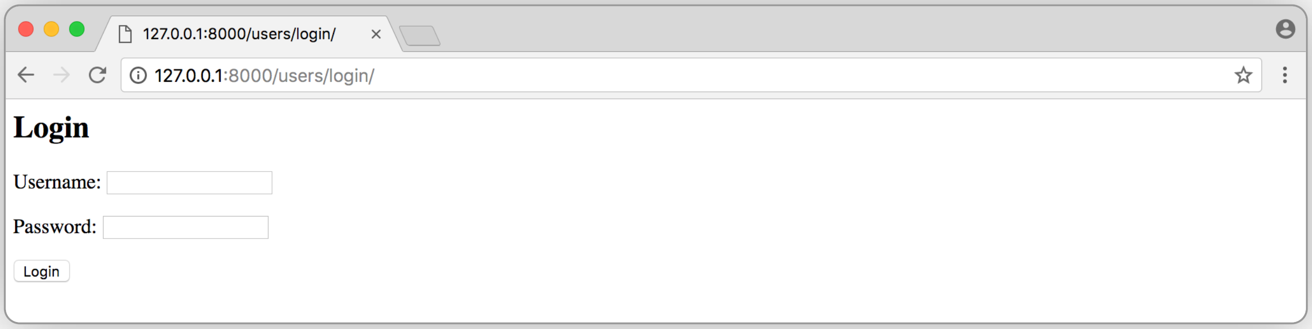
Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Sign up

Create a new account and you'll be redirected to the login page.


127.0.0.1:8000/users/login/

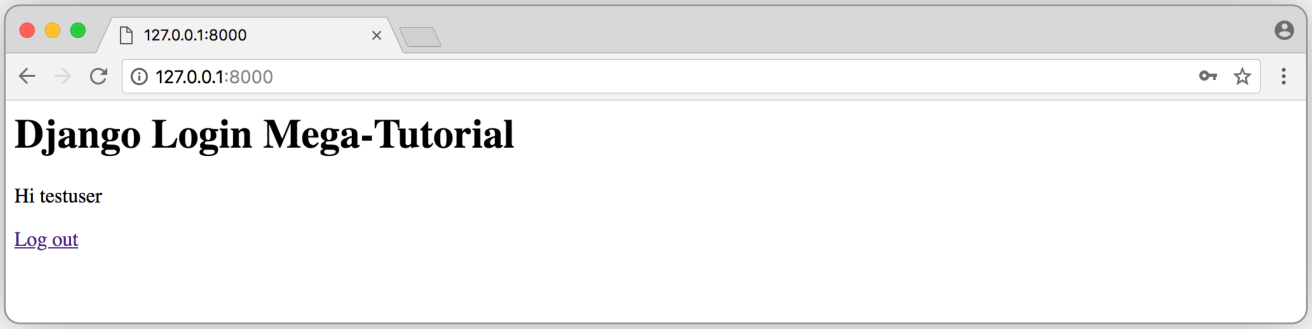
Login

Username:

Password:

Login

Log in and you'll be redirected to the homepage. My new account was called `testuser`.

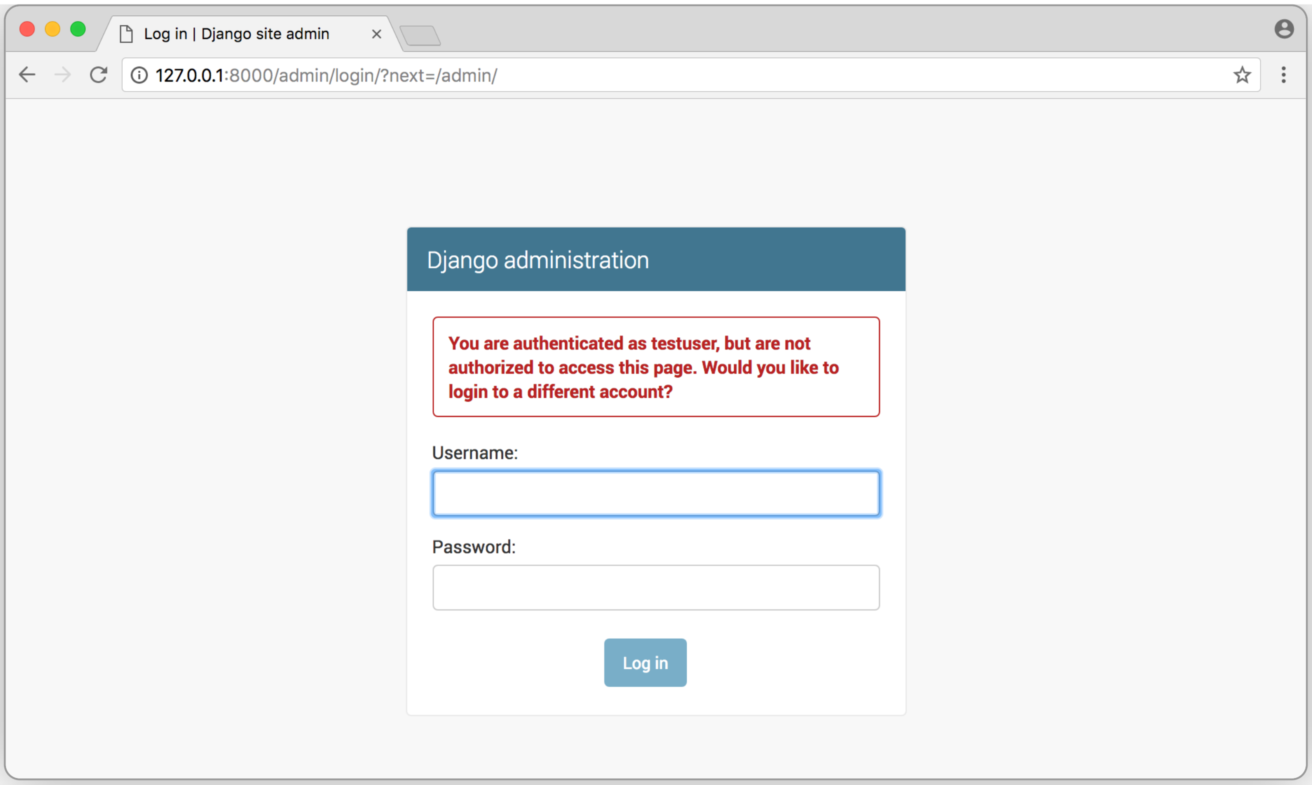

127.0.0.1:8000

Django Login Mega-Tutorial

Hi testuser

Log out

As the final confirmation let's go into the admin to see both of our users. Navigate to http://127.0.0.1:8000/admin/.


Log in | Django site admin

127.0.0.1:8000/admin/login/?next=/admin/

Django administration

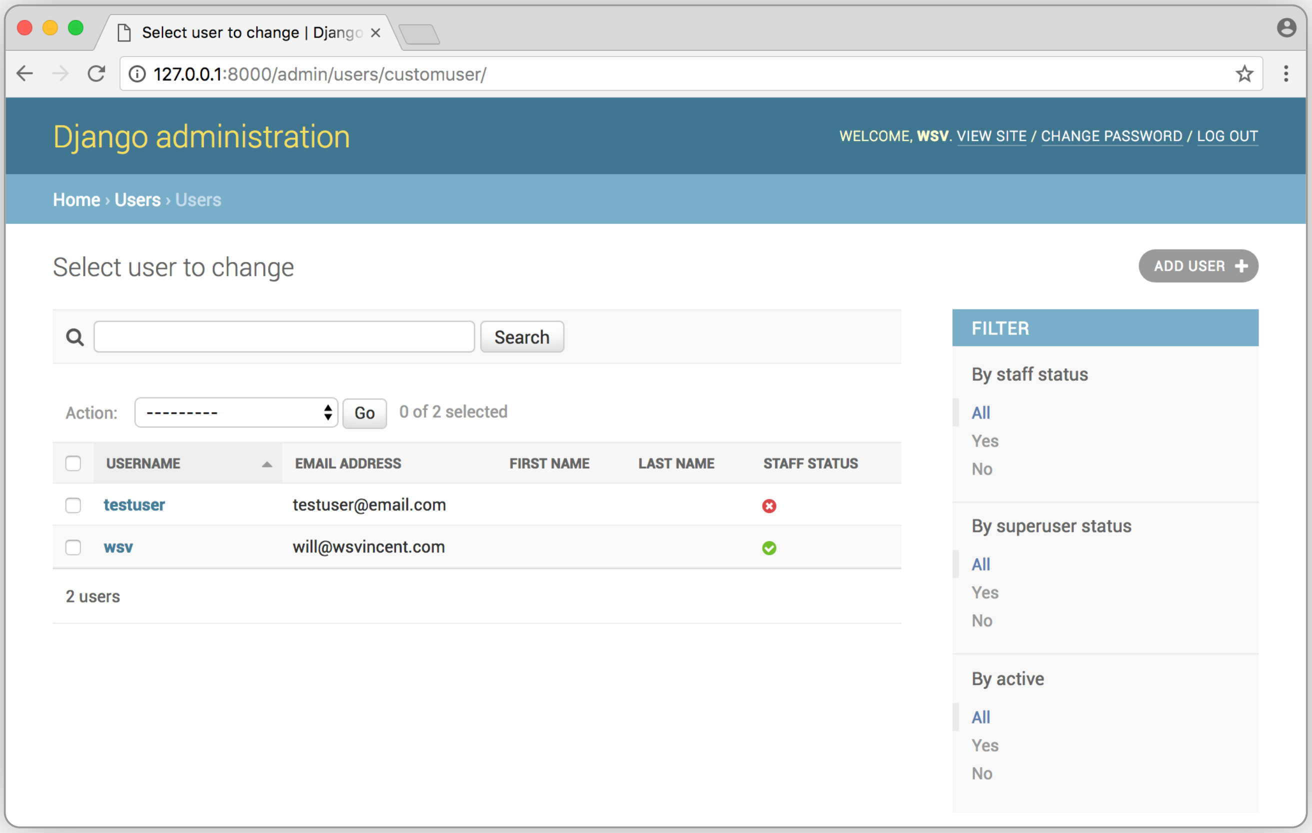
You are authenticated as testuser, but are not authorized to access this page. Would you like to login to a different account?

Username:

Password:

Log in

Oops! We have two users but only one–our superuser–can access the admin. This is a common gotcha. No worries; login with your superuser account here. Then click on "Users".


Select user to change | Django

127.0.0.1:8000/admin/users/customuser/

Django administration

WELCOME, WSV. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Users › Users

Select user to change

ADD USER

Search

Action: --------- Go 0 of 2 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
testuser	testuser@email.com			✗
wsv	will@wsvincent.com			✓

2 users

FILTER

By staff status
All
Yes
No

By superuser status
All
Yes
No

By active
All
Yes
No

So far so good. Now let's integrate `django-allauth` so our users can signup in one click with a third-party service like Gmail.

Django allauth

The first step is to install `django-allauth` using `Pipenv`.

```
(demo) $ pipenv install django-allauth
```

There are a number of changes we need to make in our `settings.py` file. Let's start with `INSTALLED_APPS`: Allauth relies on the built-in `sites` app but we need to add it ourselves as it no longer comes installed by default. We also add three specific `allauth` apps as well as one for Google.

The order matters here: add the `django-allauth` config **below** the Django default apps but **above** our own apps.

```
# demo_project/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites', # new

    'allauth', # new
    'allauth.account', # new
    'allauth.socialaccount', # new
    'allauth.socialaccount.providers.google', # new

    'users',
    'pages',
]
```

Now at the bottom of the file we need to add some more configs. First we set our `AUTHENTICATION_BACKENDS` to use the existing `ModelBackend` so users can log in to the admin site. We also add allauth's specific `AuthenticationBackend`. And since we'll use the `Sites` app for configuring each 3rd party provider in the admin app, we also add a

`SITE_ID`. Finally we want to store the user's email address but not require a username so we add config for `ACCOUNT_EMAIL_REQUIRED` and `ACCOUNT_USERNAME_REQUIRED`.

```
# demo_project/settings.py
AUTHENTICATION_BACKENDS = (
    "django.contrib.auth.backends.ModelBackend",
    "allauth.account.auth_backends.AuthenticationBackend",
)

SITE_ID = 1

ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_USERNAME_REQUIRED = False
```

Now we need to update our project-level `urls.py` file. We'll mimic the Allauth docs and include it at `accounts/`. Note that we also are **removing** the inclusion of the default auth module which was there earlier.

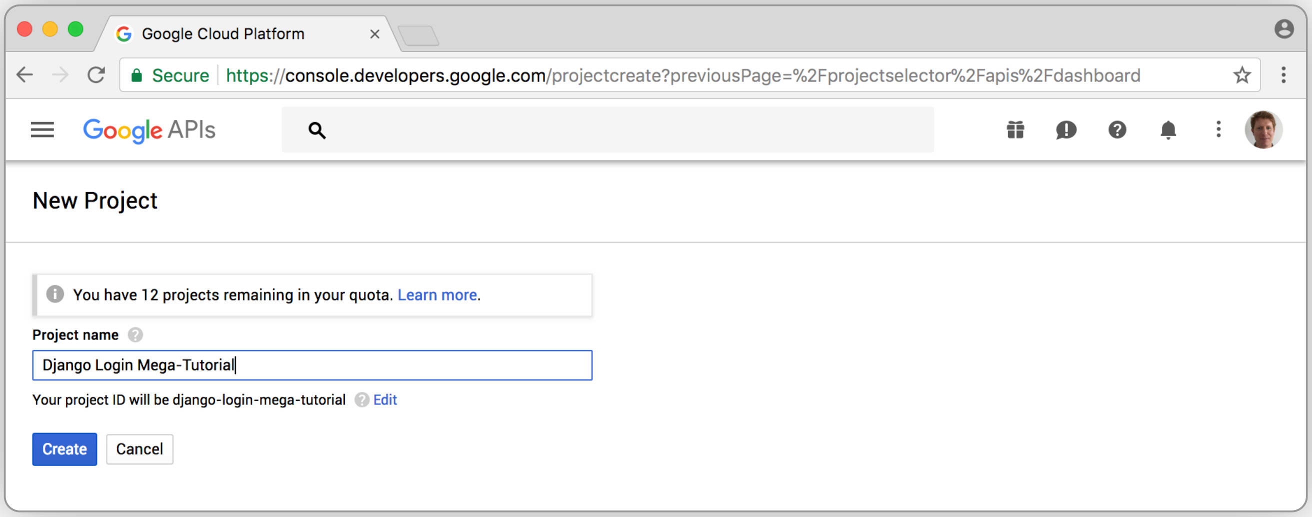
```
# demo_project/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('pages.urls')),

    # Django Admin
    path('admin/', admin.site.urls),

    # User management
    path('users/', include('users.urls')),
    path('accounts/', include('allauth.urls')), # new
]
```

We haven't made any model changes at this point but we have made some dramatic changes to our Installed Apps so it's a good time to migrate and update our database to reflect the changes.

```
(demo) $ python manage.py migrate
```

Google credentials

To allow users to log in with their Gmail credentials we need to register our new website with Google. Go to the Google Developers Console and enter the name of your new project. I've called this one "Django Login Mega-Tutorial." The click on the "Create" button.


Google Cloud Platform
Secure https://console.developers.google.com/projectcreate?previousPage=%2Fprojectselector%2Fapis%2Fdashboard
Google APIs
New Project

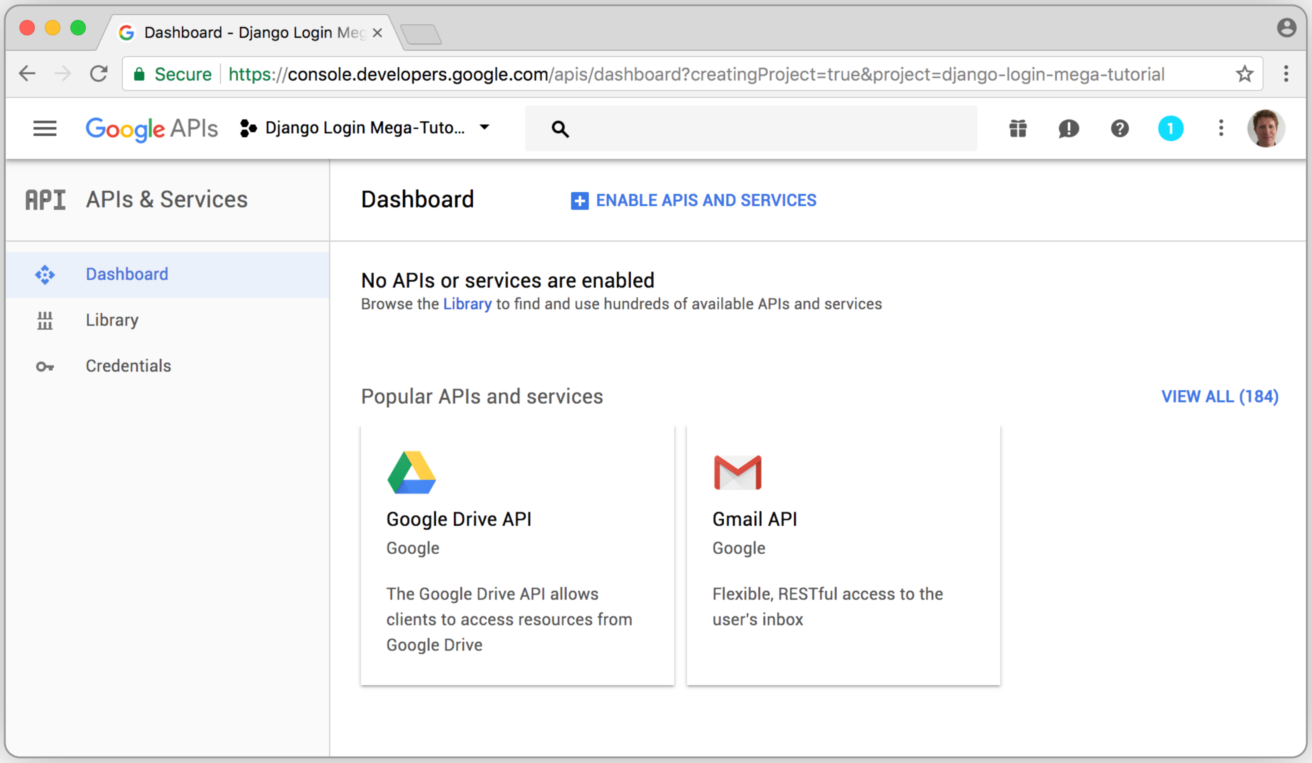
You have 12 projects remaining in your quota. Learn more.

Project name

Django Login Mega-Tutorial

Your project ID will be django-login-mega-tutorial Edit

You'll be sent to the Google "API & Services" page. We want to use Gmail so click on it.


Dashboard - Django Login Meg
Secure https://console.developers.google.com/apis/dashboard?creatingProject=true&project=django-login-mega-tutorial
Google APIs
Django Login Mega-Tuto...
API APIs & Services
Dashboard
Library
Credentials
ENABLE APIS AND SERVICES
Dashboard

No APIs or services are enabled

Browse the Library to find and use hundreds of available APIs and services

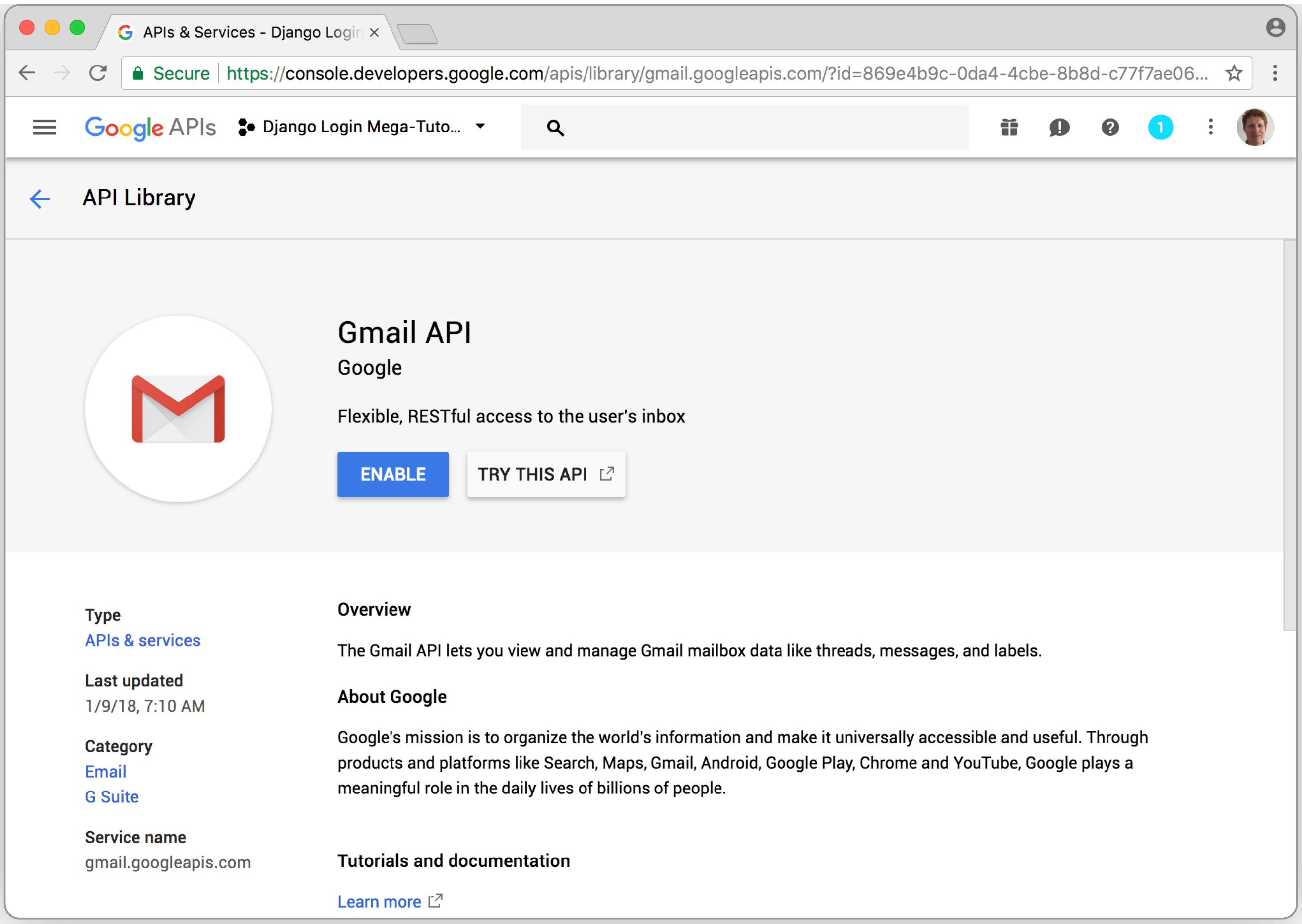
Popular APIs and services

VIEW ALL (184)

Google Drive API
Google
The Google Drive API allows clients to access resources from Google Drive

Gmail API
Google
Flexible, RESTful access to the user's inbox

The next screen shows all the details of the Gmail API. Click on the "Enable" button.


APIs & Services - Django Login
Secure https://console.developers.google.com/apis/library/gmail.googleapis.com/?id=869e4b9c-0da4-4cbe-8b8d-c77f7ae06...
Google APIs
Django Login Mega-Tuto...
API Library
ENABLE
TRY THIS API

Gmail API
Google
Flexible, RESTful access to the user's inbox

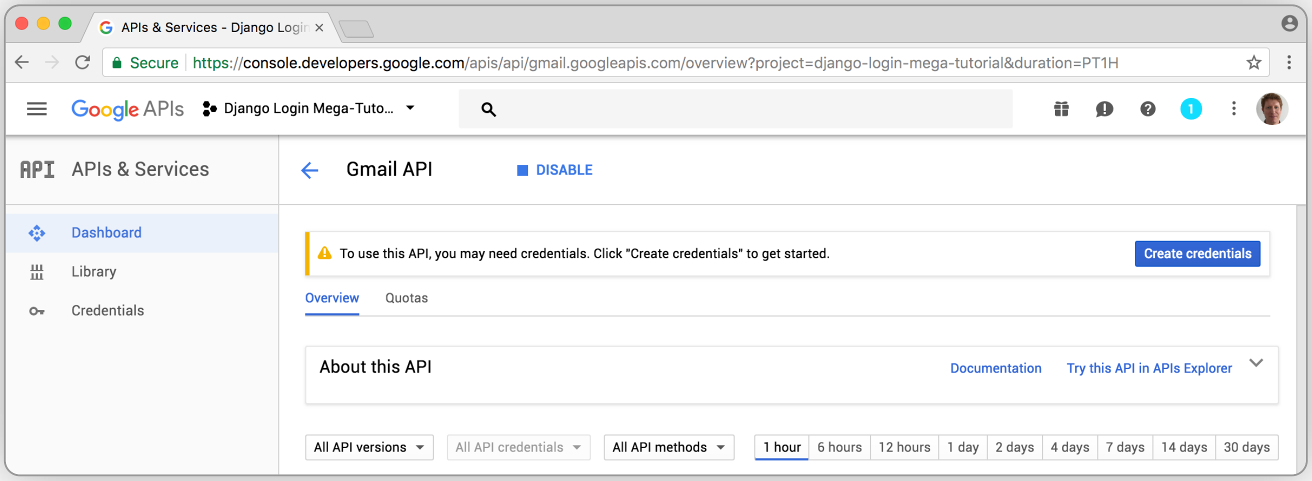
Type
APIs & services
Last updated
1/9/18, 7:10 AM
Category
Email
G Suite
Service name
gmail.googleapis.com

Overview
The Gmail API lets you view and manage Gmail mailbox data like threads, messages, and labels.

About Google
Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Tutorials and documentation
Learn more

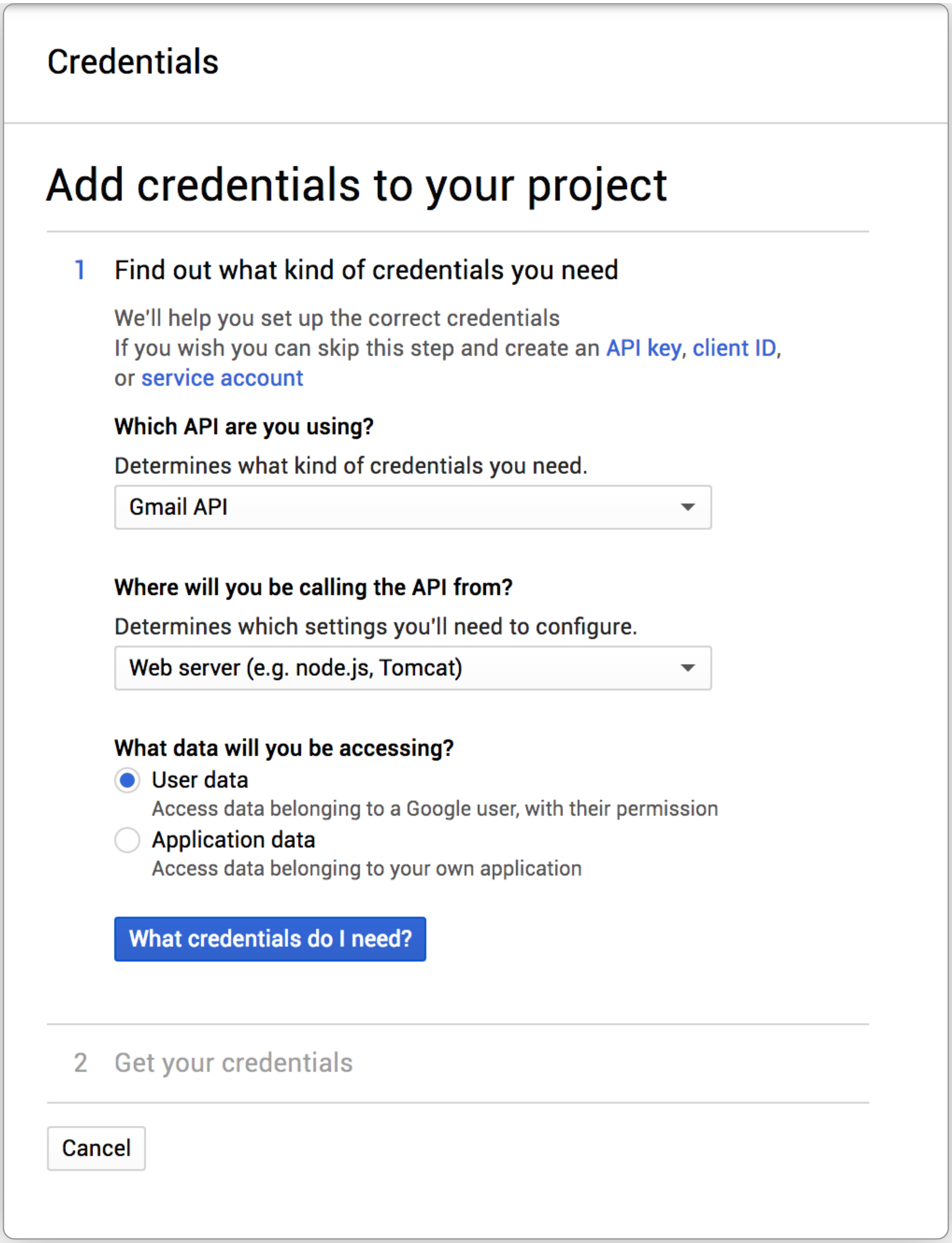
You may be asked for credentials at this point.



To use this API, you may need credentials. Click "Create credentials" to get started.

About this API

If so click on the “Create credentials” button. Make sure the fields are correct that we’re using the “Gmail API” with a “Web server” and accessing “User data.” Then click “What credentials do I need?”



Credentials

Add credentials to your project

1 Find out what kind of credentials you need

We'll help you set up the correct credentials
If you wish you can skip this step and create an API key, client ID, or service account

Which API are you using?

Determines what kind of credentials you need.

Gmail API

Where will you be calling the API from?

Determines which settings you'll need to configure.

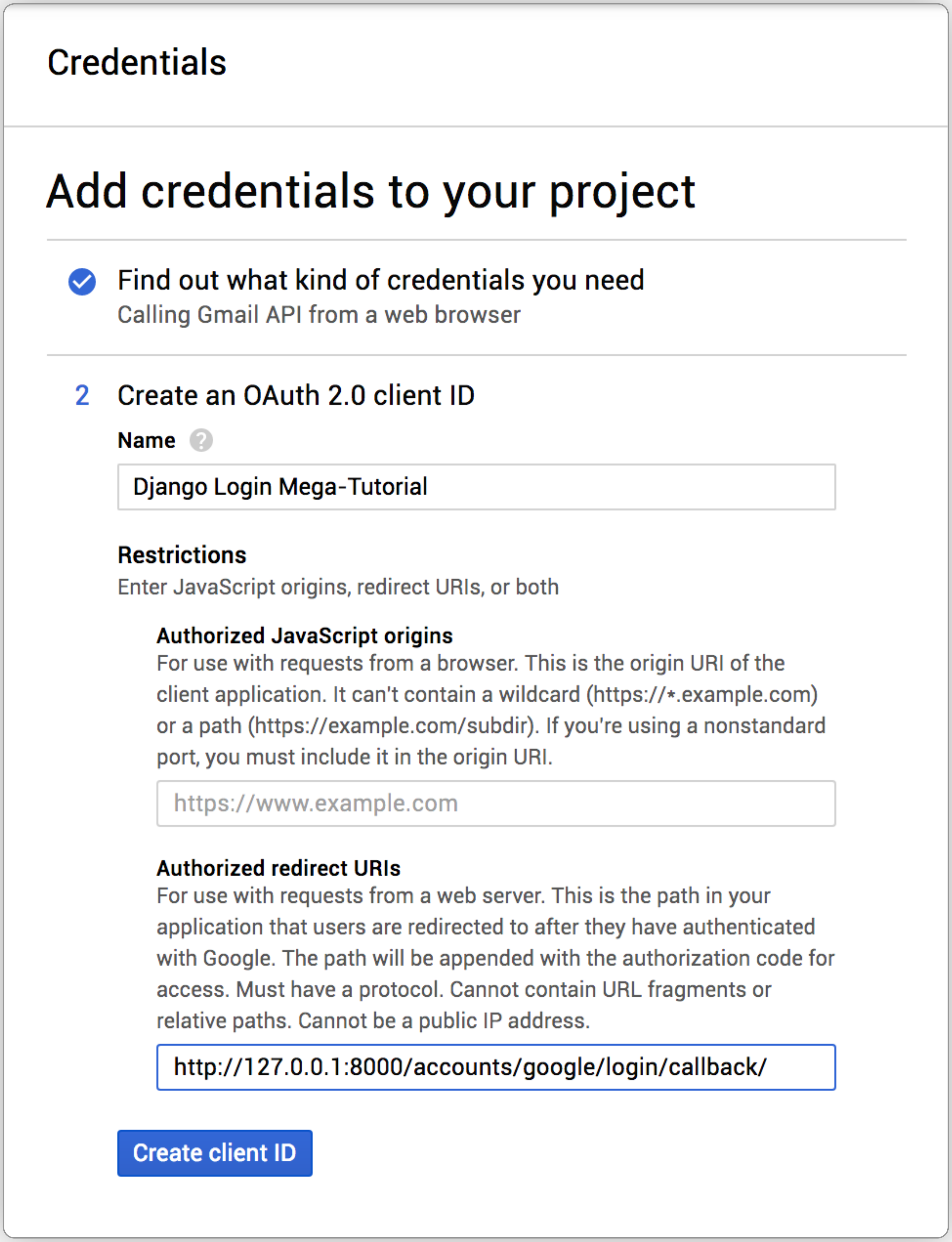
Web server (e.g. node.js, Tomcat)

What data will you be accessing?

[✓ User data] Access data belonging to a Google user, with their permission

Application data — Access data belonging to your own application

Step 2 is to add a Name and Authorized Redirect URLs. The name I've just repeated the name of my overall project here. The redirect URL should be `http://127.0.0.1:8000/accounts/google/login/callback/`. Then click on the "Create client ID" button.



Credentials

Add credentials to your project

✓ Find out what kind of credentials you need
Calling Gmail API from a web browser

2 Create an OAuth 2.0 client ID

Name

Django Login Mega-Tutorial

Restrictions
Enter JavaScript origins, redirect URIs, or both

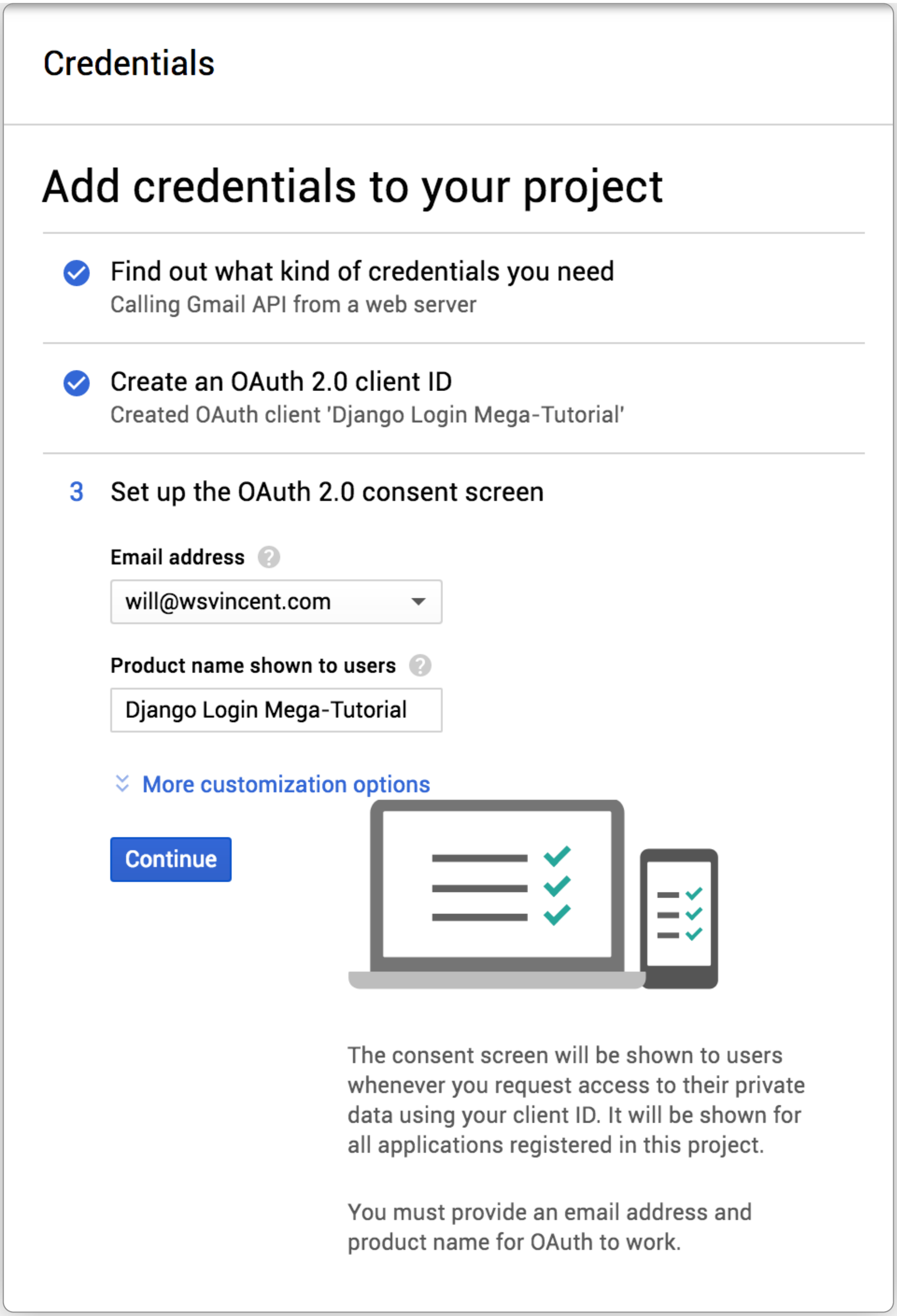
Authorized JavaScript origins
For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.

https://www.example.com

Authorized redirect URIs
For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

http://127.0.0.1:8000/accounts/google/login/callback/

Third and final step is to configure the consent screen. This is the information shown to users after they click on the login button. They'll be redirected to a Google site that shows the Product name and asks if the site has permission to access their Google account.



Credentials

Add credentials to your project

✓ Find out what kind of credentials you need
Calling Gmail API from a web server

✓ Create an OAuth 2.0 client ID
Created OAuth client 'Django Login Mega-Tutorial'

3 Set up the OAuth 2.0 consent screen

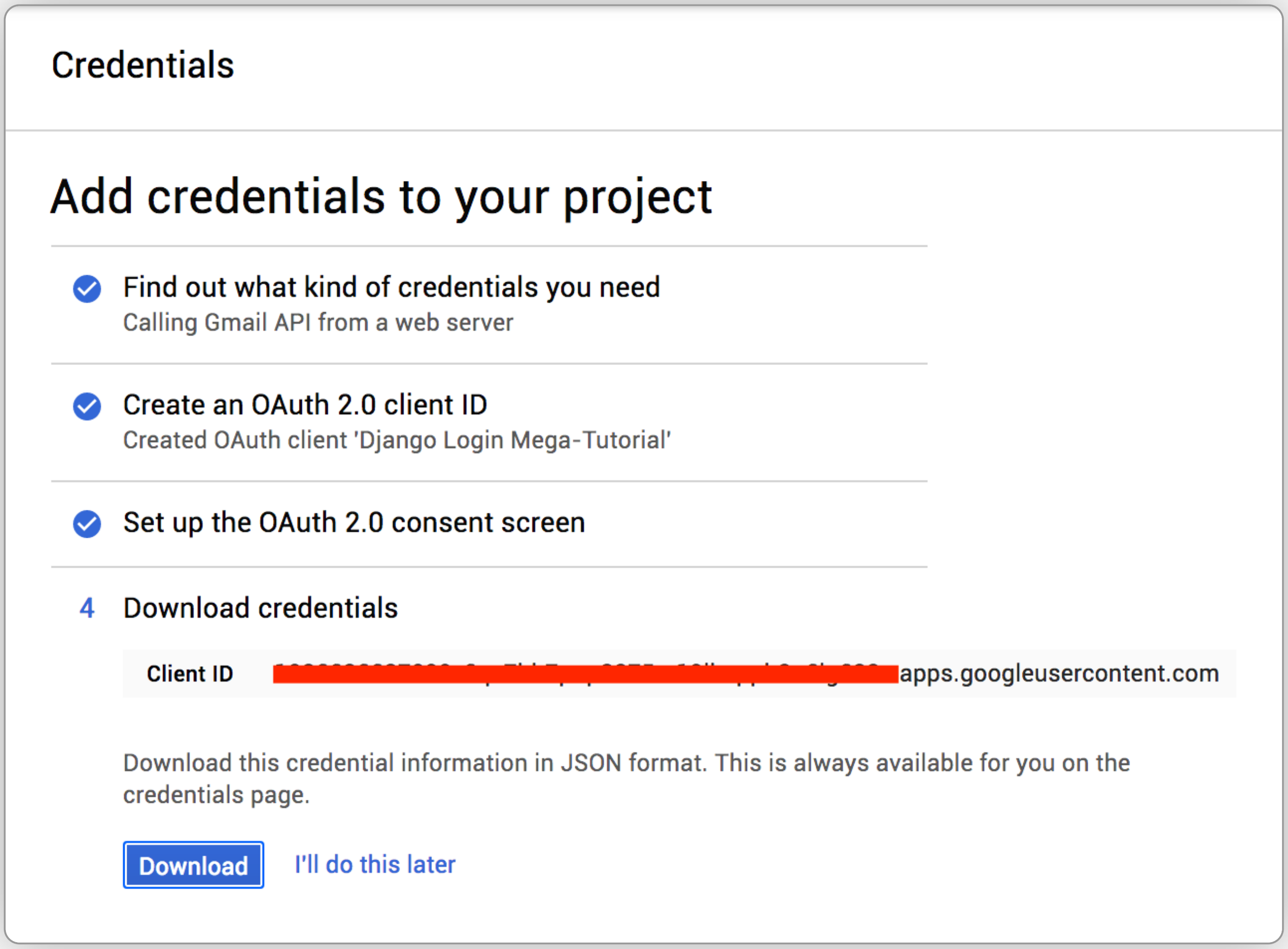
Email address: will@wsvincent.com

Product name shown to users: Django Login Mega-Tutorial

The consent screen will be shown to users whenever you request access to their private data using your client ID. It will be shown for all applications registered in this project.

You must provide an email address and product name for OAuth to work.

Now we have what we want: a client ID (I've hidden mine in red). I recommend downloading it and storing somewhere secure. You don't want this information public.



Credentials

Add credentials to your project

✓ Find out what kind of credentials you need
Calling Gmail API from a web server

✓ Create an OAuth 2.0 client ID
Created OAuth client 'Django Login Mega-Tutorial'

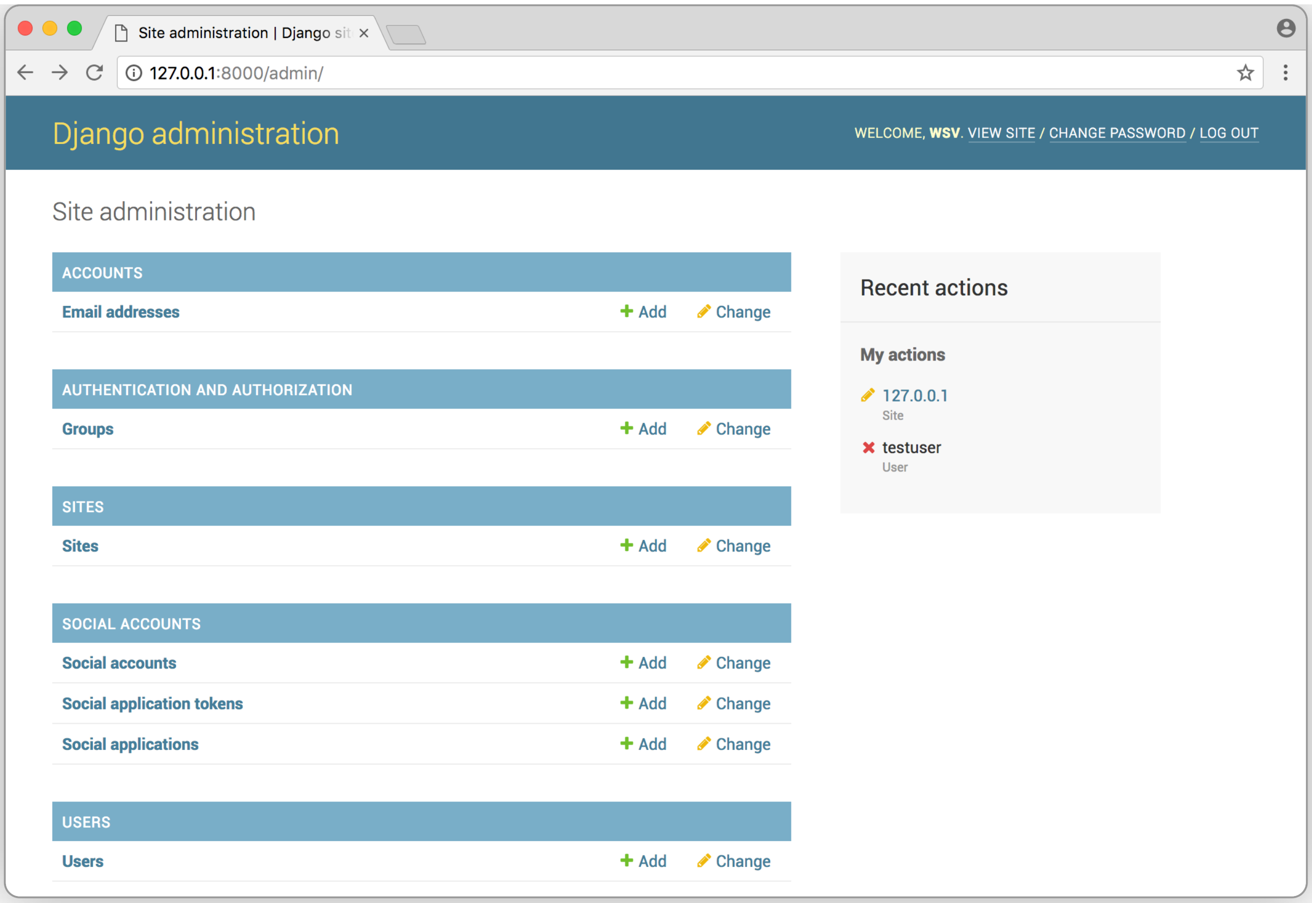
✓ Set up the OAuth 2.0 consent screen

4 Download credentials

Client ID [redacted]apps.googleusercontent.com

Download this credential information in JSON format. This is always available for you on the credentials page.

Now we can configure the Admin which will be much quicker. Go to the Admin site http://127.0.0.1:8000/admin and notice Allauth has added a number of new sections for us.


Site administration | Django site

127.0.0.1:8000/admin/

Django administration

WELCOME, **WSV**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

ACCOUNTS		
Email addresses	Add	Change
AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
SITES		
Sites	Add	Change
SOCIAL ACCOUNTS		
Social accounts	Add	Change
Social application tokens	Add	Change
Social applications	Add	Change
USERS		
Users	Add	Change

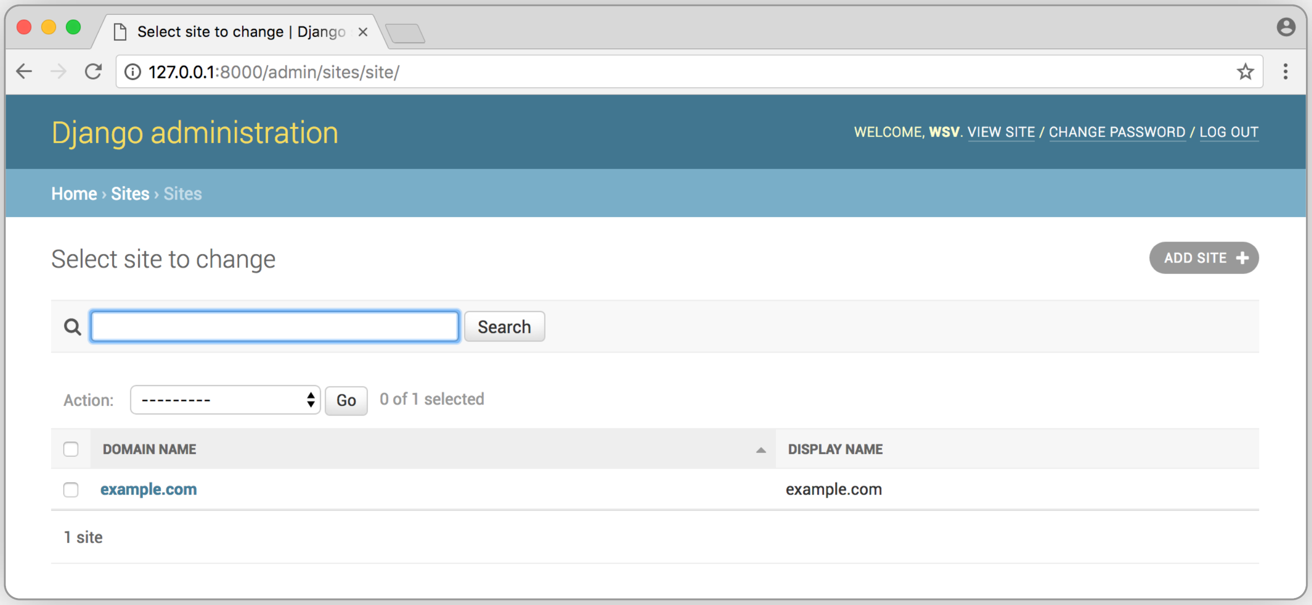
Recent actions

My actions

127.0.0.1
Site

testuser
User

Go into “Sites” on this page.


Select site to change | Django

127.0.0.1:8000/admin/sites/site/

Django administration

WELCOME, **WSV**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Sites › Sites

Select site to change

ADD SITE

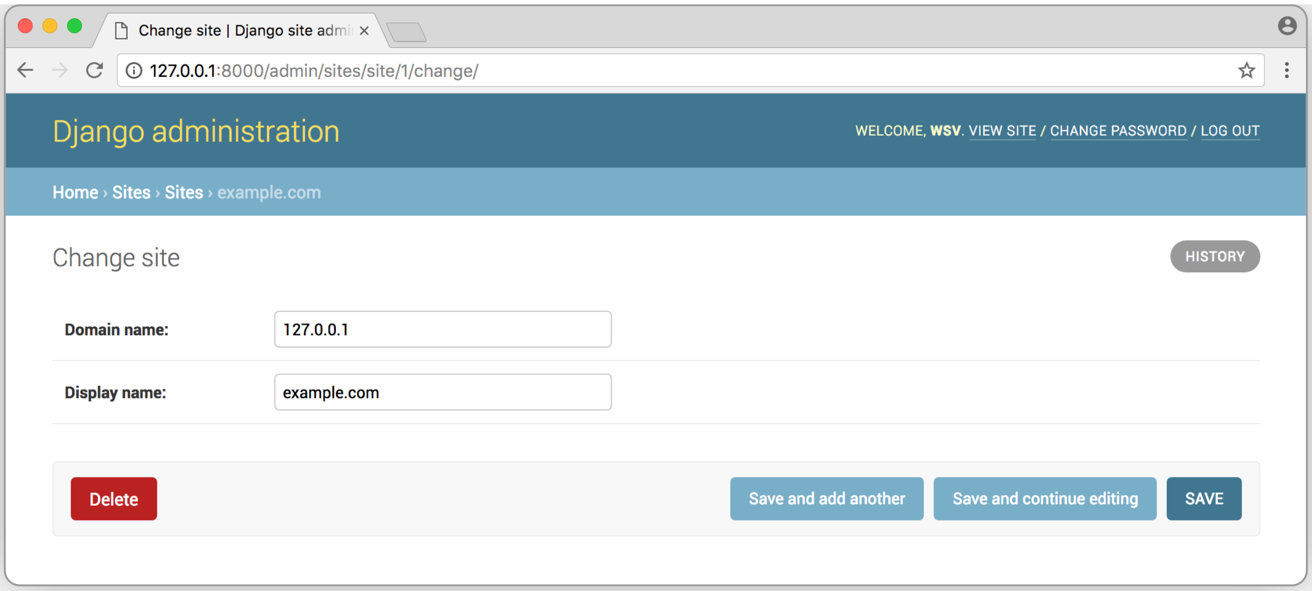
Search

Action: --------- Go 0 of 1 selected

DOMAIN NAME	DISPLAY NAME
example.com	example.com

1 site

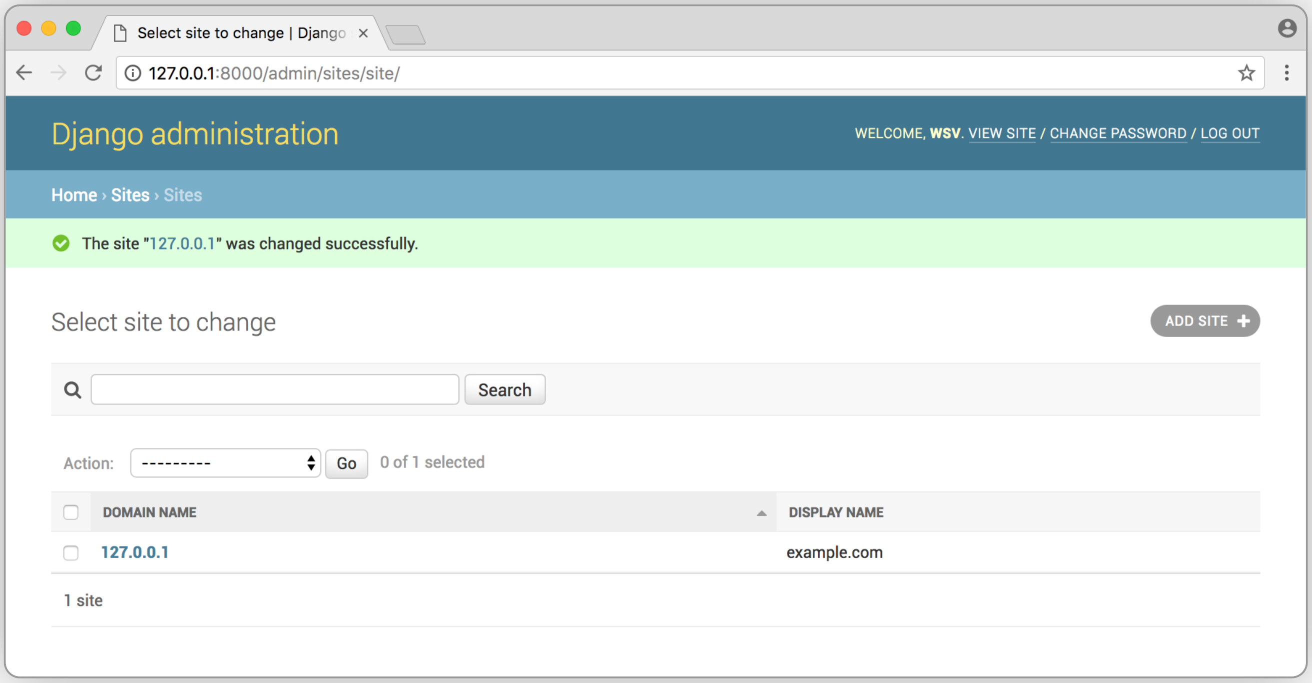
Click on the existing Domain Name of “example.com”. Update it so the “Domain name” is `127.0.0.1`.


Change site | Django site admi
127.0.0.1:8000/admin/sites/site/1/change/
Django administration
WELCOME, WSV. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Sites › Sites › example.com
HISTORY
Delete
Save and add another
Save and continue editing
SAVE

Change site

Domain name:	127.0.0.1
Display name:	example.com

Click save.


Select site to change | Django
127.0.0.1:8000/admin/sites/site/
Django administration
WELCOME, WSV. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Sites › Sites
ADD SITE
Search
Action: ---------
Go
0 of 1 selected

The site "127.0.0.1" was changed successfully.

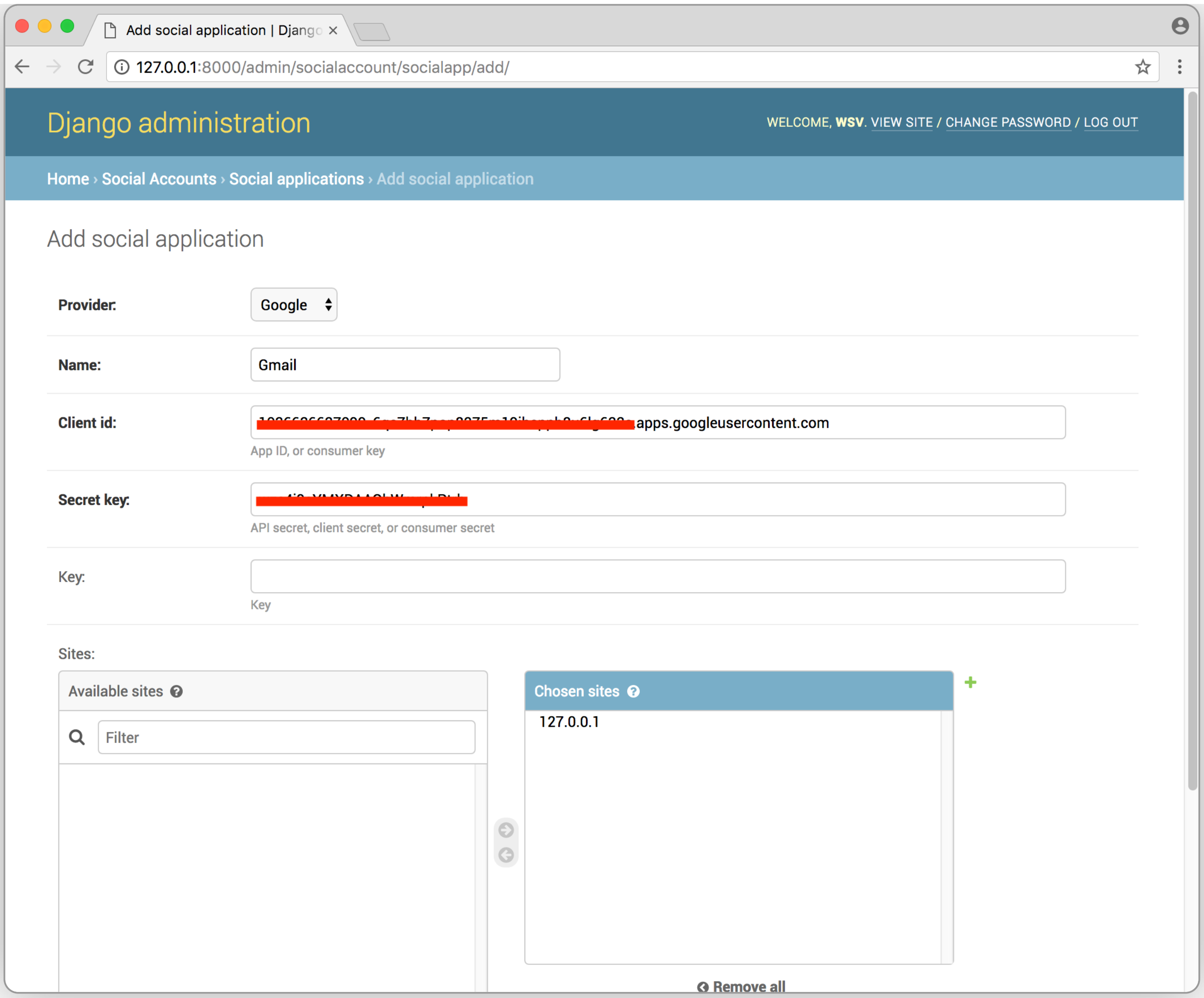
Select site to change

DOMAIN NAME	DISPLAY NAME
127.0.0.1	example.com

1 site

Now for each third-party application we want to add we just need to click on the “Add” button next to “Social applications” under “Social Accounts.”

On this page we need to select the provider (Google), provide a Name (Gmail), a Client ID, and a Secret Key. If you downloaded your API keys previously and open the file with a text editor you’ll see it is in JSON format and has **both** your Client ID and Client Secret. Enter both in here. Finally on the bottom of the page under “Sites” select “127.0.0.1” and click the → arrow to add it to the Chosen Sites section. Now click on the “Save” button in the lower right of the screen.



Add social application | Django

127.0.0.1:8000/admin/socialaccount/socialapp/add/

Django administration

WELCOME, **WSV**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Social Accounts › Social applications › Add social application

Add social application

Provider:	Google
Name:	Gmail
Client id:	[redacted]apps.googleusercontent.com App ID, or consumer key
Secret key:	[redacted] API secret, client secret, or consumer secret
Key:	 Key

Sites:

Available sites

Filter

Chosen sites

127.0.0.1

Remove all

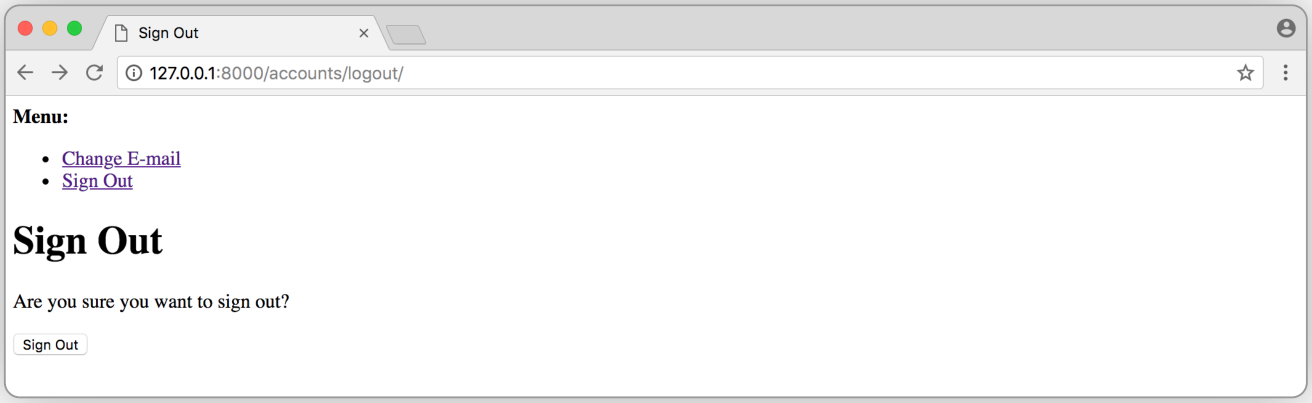
Update Templates

The last step is to update our templates file. We need to load `socialaccount` which comes from Allauth. And our named URLs are slightly different as well: we add an `account_` in front of the existing logout, signup, login links. Finally we add a provider link for Google.

The updated `home.html` file should look like this:

```
<!-- templates/home.html -->
{% load socialaccount %}

<h1>Django Login Mega-Tutorial</h1>
{% if user.is_authenticated %}
<p>Welcome {{ user.username }} !!!</p>
<p><a href="{% url 'account_logout' %}">Log out</a>
{% else %}
<p><a href="{% url 'account_signup' %}">Sign Up</a></p>
<p><a href="{% url 'account_login' %}">Log In </a></p>
<p><a href="{% provider_login_url 'google' %}">Log In with Gmail</
{% endif %}
```

Now try everything out. Go back to the homepage at http://127.0.0.1:8000/. You're probably logged in so click on the "Log out" link.



Menu:

- Change E-mail
- Sign Out

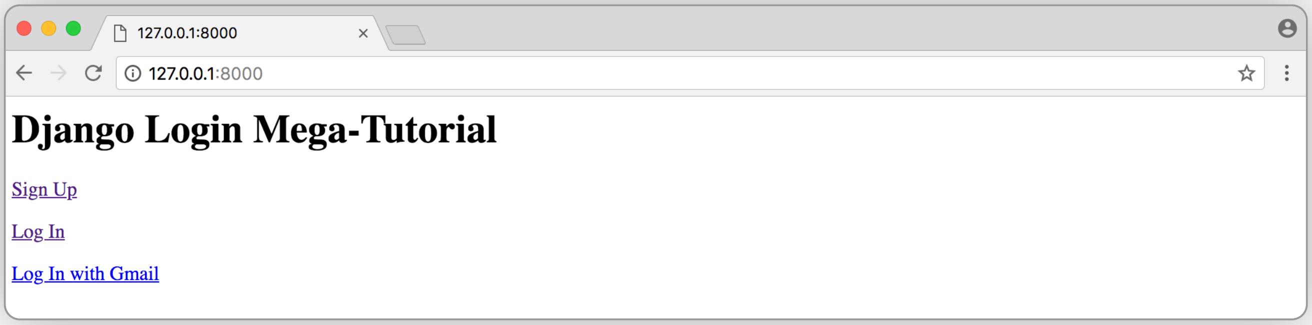
Sign Out

Are you sure you want to sign out?

Sign Out

What we see is the default Allauth logout page. We can configure it but won't here. Click on the "Sign Out" link.

Now we see our new homepage with three links.



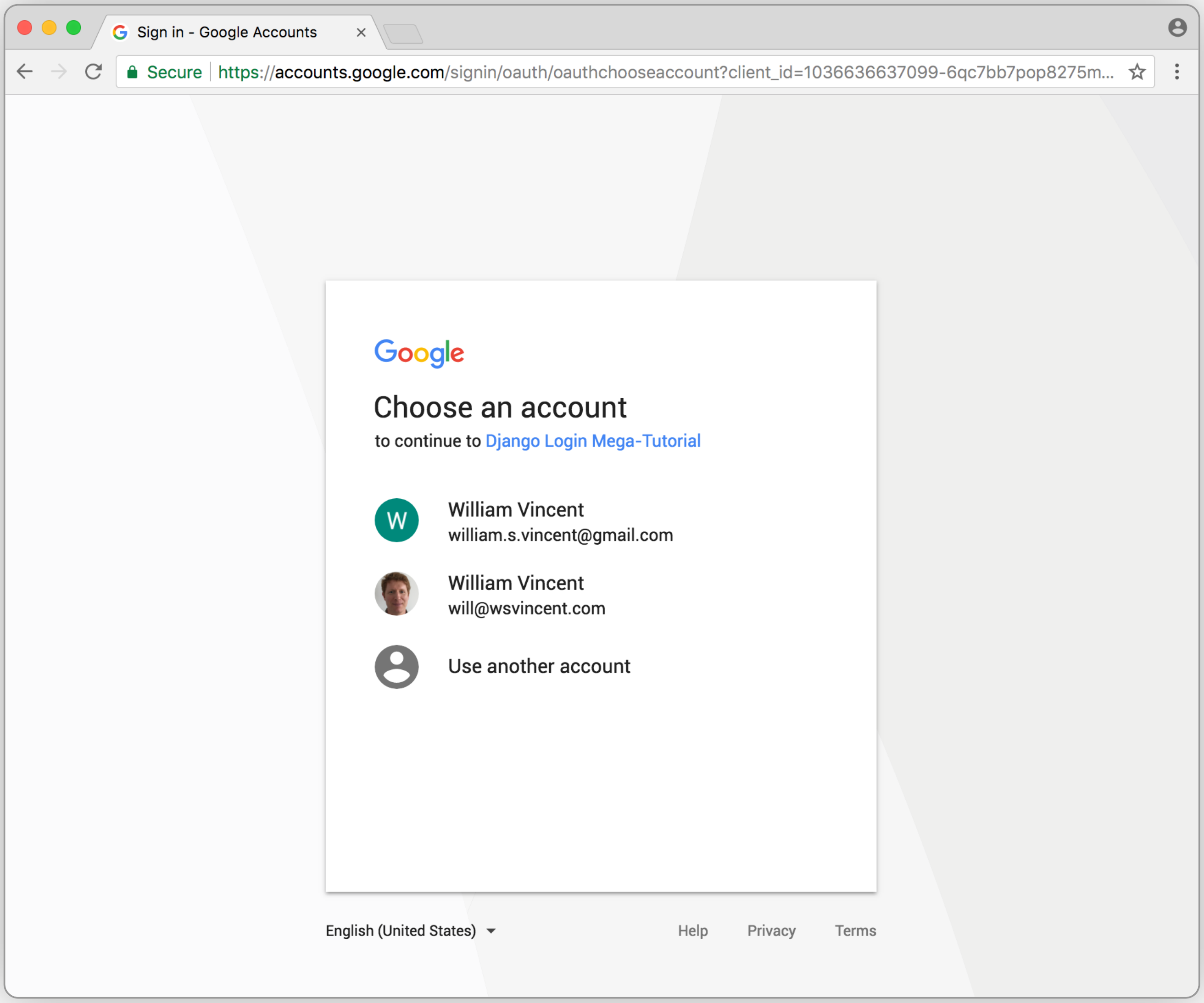
Django Login Mega-Tutorial

Sign Up

Log In

Log In with Gmail

Click on the "Log In with Gmail" link. Then we'll see Google's login page. I've used my `william.s.vincent@gmail.com` account here since we don't want duplicate email addresses in our database.


Sign in - Google Accounts
Secure
https://accounts.google.com/signin/oauth/oauthchooseaccount?client_id=1036636637099-6qc7bb7pop8275m...
Google

Choose an account
to continue to Django Login Mega-Tutorial

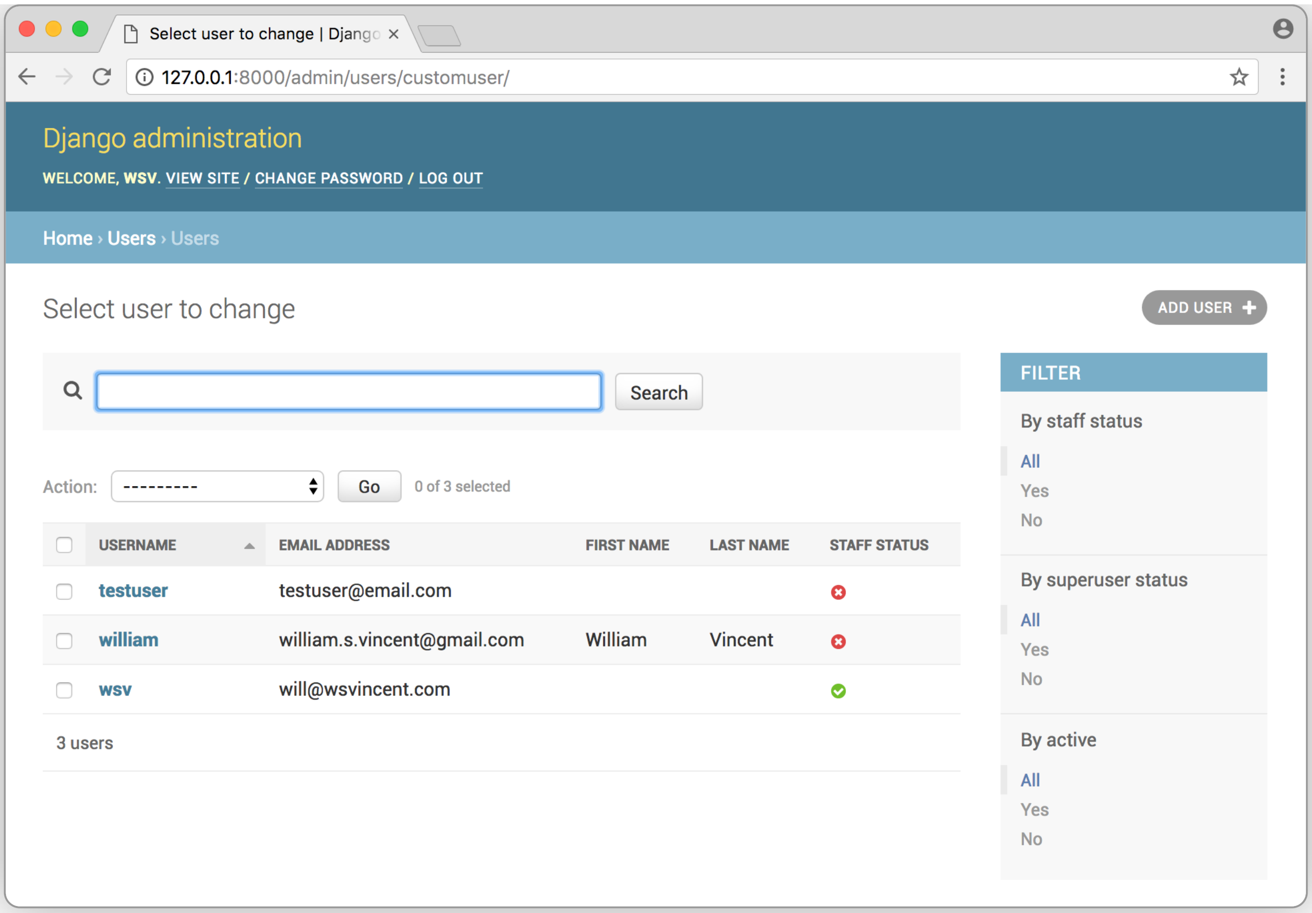
William Vincent
william.s.vincent@gmail.com

William Vincent
will@wsvincent.com

Use another account

English (United States)
Help
Privacy
Terms

Now navigate to the "Users" section of the admin at http://127.0.0.1:8000/admin and we can see both our new user and his/her email address.



Select user to change

0 of 3 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
testuser	testuser@email.com			✗
william	william.s.vincent@gmail.com	William	Vincent	✗
wsv	will@wsvincent.com			✓

3 users

Next steps

That was a bit of work eh? In exchange for all our effort we have complete control over the user authorization process now. We can add additional fields to `CustomUser`; we can add additional 3rd party logins; and we can override Allauth default templates.

One more thing that's nice to do is require email confirmation for new accounts. Django does not have this capability but Allauth does.

Join My Newsletter

Subscribe to get the latest tutorials/writings by email.

Your email address

Subscribe

No spam. Promise. Unsubscribe at any time.

86 Comments **wsvincent.com** 1 **Login**

Recommend 11 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH **OR SIGN UP WITH DISQUS** ?

Name

CRA • a year ago • edited

It fell apart when I implemented gmail.

I get the DoesNotExist at /accounts/google/login as well.

EDIT:

I got it to work.

1. delete what is inside migrations folder, except for __init__.py
2. delete db.sqlite3 (database)
3. makemigrations
4. migrate
5. createsuperuser
6. login to admin and add Social Application, select example.com
7. Edit example.com to be 127.0.0.1
8. logout and signup for gmail

3 ^ | v • Reply • Share ›

Oleg Lokshyn → CRA • a year ago

That`s probably because you have added 127.0.0.1 site while adding Social application in the admin panel.

The tutorial (implicitly) suggest that we use example.org as the site while adding Gmail and then rename example.com to 127.0.0.1 domain in the Site settings. The idea behind that is we specified SITE_ID to be equal to 1 in settings.py, but the Site with id==1 already exists and is called example.org. If you add another (127.0.0.1) site to the Site table it will have id equal to 2.

If you, like me, have pressed that tempting green plus button while adding Social application and added 127.0.0.1 then you should also change SITE_ID to 2 in settings.py. That worked for me.

1 ^ | v • Reply • Share ›

KALU GOODNEWS • 10 months ago

Great tutorial!! But mine failed to redirect by bringing up this when I try login with Gmail:

Error: redirect_uri_mismatchThe
redirect URI in the request,
http://127.0.0.1:8000/accounts/google/login/callback/, does not match
the ones authorized for the OAuth client. To update the authorized
redirect URIs, visit:
https://console.developers....

Pls how can i resolve this?

1 ^ | v • Reply • Share ›

Aniket A. Aryamane ➜ KALU GOODNEWS • 8 months ago

1) While creating Oauth 2.0 client ID, mention the "Authorized redirect URIs" as:
"http://localhost:8000/accounts/google/login/callback/" instead of
"http://127.0.0.1:8000/accounts/google/login/callback/"

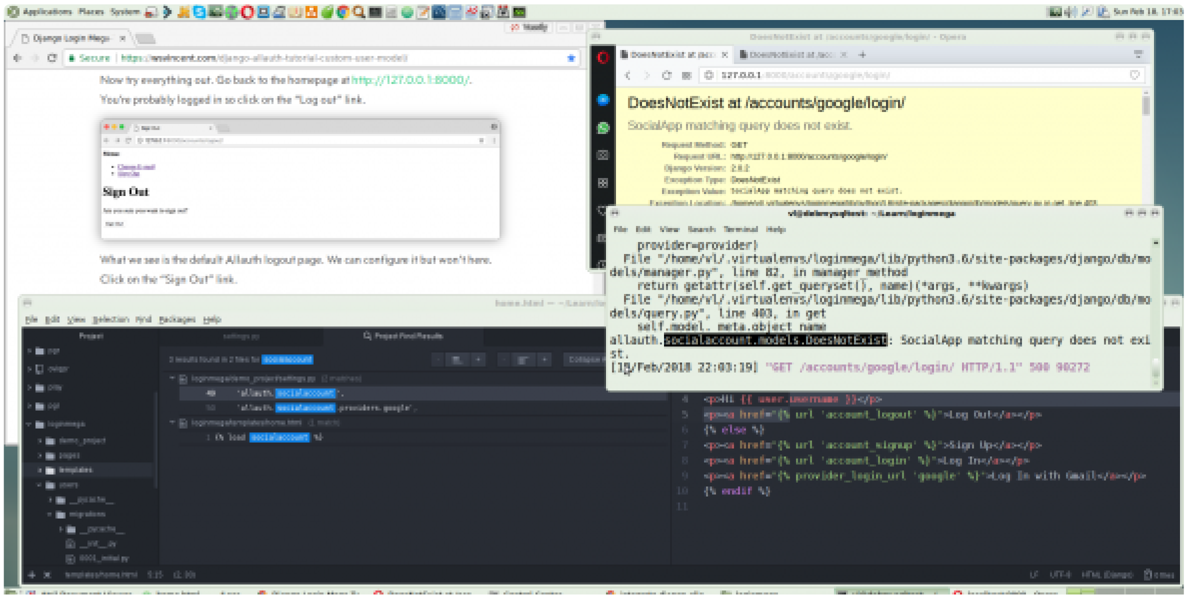
2) Update the DJango admin - "Sites" domain name to "localhost" instead of "127.0.0.1"

Then it will work as expected.

^ | v • Reply • Share ›

Vitalik Clotchko • 2 years ago • edited

I have a problem with Gmail authentication.
When I click Login with Gmail, it seemingly tries opening a default URL (/account/google/login/) instead of redirecting to the google app pages and prints that the social app query doesn't exist. See the screenshot.
How do you think I can fix it ?



DoesNotExist at /accounts/google/login/

SocialApp matching query does not exist.

```
    provider=provider)
  File "/home/vl/.virtualenvs/loginmega/lib/python3.6/site-packages/django/db/mo
dels/manager.py", line 82, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "/home/vl/.virtualenvs/loginmega/lib/python3.6/site-packages/django/db/mo
dels/query.py", line 403, in get
    self.model._meta.object_name
allauth.socialaccount.models.DoesNotExist: SocialApp matching query does not exi
st.
[18/Feb/2018 22:03:19] "GET /accounts/google/login/ HTTP/1.1" 500 90272
```

```
<p>Hi {{ user.username }}</p>
<p><a href="{% url 'account_logout' %}">Log Out</a></p>
{% else %}
<p><a href="{% url 'account_signup' %}">Sign Up</a></p>
<p><a href="{% url 'account_login' %}">Log In</a></p>
<p><a href="{% provider_login_url 'google' %}">Log In with Gmail</a></p>
{% endif %}
```

Also, have trouble logging in to the admin app.

1 ^ | v • Reply • Share ›

CRA ➦ Vitalik Clotchko • a year ago

were any of you able to resolve this?

^ | v • Reply • Share ›

Tam Xid ➦ Vitalik Clotchko • 2 years ago

I had same error.
Solved by:
# delete 'migrations' directory in 'users' app
# delete db.sqlite3 (database)
# makemigrations
# migrate
# createsuperuser
# re-do this part:
.... For each third-party application we want to add we just need to click on the “New” button next to “Social applications” under “Social Accounts.” ....

^ | v • Reply • Share ›

DoKyung Kim ➦ Vitalik Clotchko • 2 years ago

Did you migrate?

^ | v • Reply • Share ›

shivam singhal ➦ DoKyung Kim • 2 years ago

Same error, It did not work.

^ | v • Reply • Share ›

CRA ➦ shivam singhal • a year ago

were you able to resolve this?

^ | v • Reply • Share ›

DC • a month ago

Can someone show a simple example on how to add an extra field to the user model and populate it with allauth? Thanks in advance!

^ | v • Reply • Share ›

foobar ➦ DC • 19 days ago

You may add a custom signup form under users.forms.py as given below. (users is my app with custom user model)

```
from django.contrib.auth.forms import UserCreationForm
from django.contrb.auth import get_user_model

class CustomUserCreationForm(UserCreationForm):
```

```
class Meta(UserCreationForm):
model = get_user_model
fields = ('email','mobile',) # I've an extra filed 'mobile' to my user
model
```

Now, specify this custom form in the settings.py as
`ACCOUNT_FORMS = {'signup':'users.forms.CustomUserCreationForm'}`

^ | ˅ • Reply • Share ›

Spencer Wright • 3 months ago
Thanks for the demo! I _think_ I found a typo. You show:

```
# users/views.py
from django.urls import reverse_lazy
from django.views import generic.edit import CreateView
```

I believe it should instead be:

```
# users/views.py
from django.urls import reverse_lazy
from django.views.generic import edit, CreateView
```

^ | ˅ • Reply • Share ›

wsvincent Mod → Spencer Wright • 3 months ago
Good catch. It should be `from django.views.generic.edit import CreateView`. Now updated. Thanks!

^ | ˅ • Reply • Share ›

E Brownie • 3 months ago
worked like a charm.

^ | ˅ • Reply • Share ›

Ilham Pratama • 4 months ago
Hey thanks for such a really well explained tutorial.
But, I want to access the user's image, do you know how?

^ | ˅ • Reply • Share ›

Shehab Ahmed Sayem • 4 months ago
Thanks for this awesome article. Absolutely to the point.

^ | ˅ • Reply • Share ›

wsvincent Mod → Shehab Ahmed Sayem • 4 months ago
Thanks for letting me know it helped!

^ | ˅ • Reply • Share ›

Taimoor Ahmad • 5 months ago
I sent you an email on how to fix the 127.0.0.1 issue people seem to be

getting in the comments.

How could you make it so that if I have an account tim@gmail.com that when I sign in as tim@gmail.com through Google the accounts merge?

^ | ⌄ • Reply • Share ›

Taimoor Ahmad • 5 months ago

Isn't there a mistake here? It should be:
template_name = 'registration/signup.html'

^ | ⌄ • Reply • Share ›

Илья Куц • 5 months ago

why users instead of users.apps.UsersConfig? I mean installed_apps and your another tutorial about custom user model

^ | ⌄ • Reply • Share ›

Harsha Hemanth Sp • 6 months ago

How to Don DRF

^ | ⌄ • Reply • Share ›

Andrew Ang • 6 months ago

I wasn't able to migrate, had an error `django.db.utils.ProgrammingError: relation "django_site" already exists`. What might be causing this?

^ | ⌄ • Reply • Share ›

Oguz Han Erol • 7 months ago

I think your project DjangoX is based on (AbstractUser method) + (allauth social login), and this allows to solve both extra areas on user model + authentication

^ | ⌄ • Reply • Share ›

wsvincent Mod → Oguz Han Erol • 7 months ago

Yes, correct. I'm considering switching over to AbstractBaseUser for an update to DjangoX which provides more flexibility but also more chances to make serious mistakes. I think AbstractUser + django-allauth is best all-around choice at the moment.

^ | ⌄ • Reply • Share ›

Bino Oetomo • 7 months ago • edited

Dear Vincent and All.

My project need that a non-staff user able to do some limited admin activity. For the non-staff admin stuff, I followed https://tryolabs.com/blog/2... and currently have no problem.

currently, every loged in user will redirected to non-staff admin site as pictured at

see more

^ | ⌄ • Reply • Share ›

Micah Pearce • 7 months ago

Do you have any tutorials on how to add email confirmation?

^ | ⌄ • Reply • Share ›

dbinott • 7 months ago

any chance there will be a follow up with customization of all the forms?

^ | ⌄ • Reply • Share ›

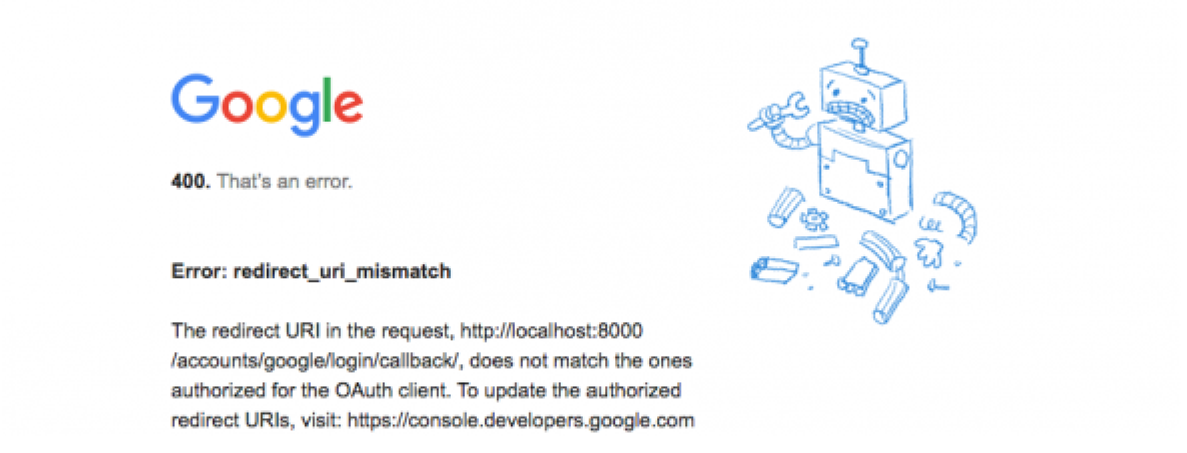
andrea • 7 months ago

Great tutorial but I did not understand how to add a custom field during registration.

^ | ⌄ • Reply • Share ›

Masud Morshed • 8 months ago • edited

In the last step i can't figure out this problem



Google

400. That's an error.

Error: redirect_uri_mismatch

The redirect URI in the request, http://localhost:8000/accounts/google/login/callback/, does not match the ones authorized for the OAuth client. To update the authorized redirect URIs, visit: https://console.developers.google.com

when i want to login with gmail then this problem occur. plz help me.

^ | ⌄ • Reply • Share ›

Aniket A. Aryamane ➦ Masud Morshed • 8 months ago

1) While creating Oauth 2.0 client ID, mention the "Authorized redirect URIs" as:

"http://localhost:8000/accounts/google/login/callback/" instead of "http://127.0.0.1:8000/accounts/google/login/callback/"

2) Update the DJango admin - "Sites" domain name to "localhost" instead of "127.0.0.1"

Then it will work as expected...

^ | v • Reply • Share ›

Agoo Clinton Masud Morshed • 8 months ago

i think you should use http://127.0.0.1:8000/accounts/...... as a redirect url,
the one which may be you registered in the Oauth2 (google API).

Or check the redirect (callback) url in Oauth API if it matche your urls...

^ | v • Reply • Share ›

Dan Swain • 9 months ago

Since you've set this up to use a custom user model, it would be nice to be able to change what is displayed in the admin for the Django "Authentication and Authorization" app to simply "Authorization" since only Groups is now displayed under this app. Can this be done?

^ | v • Reply • Share ›

Paul Ahmadzai • 9 months ago • edited

Great asset there, thanks for your contribution.

For those who may be experiencing issues with a CustomUser model which has no 'username' attribute, here's a stackexchange...

https://stackoverflow.com/questions/19683179/remove-username-field-from-django-allauth/38596589#38596589

....

> If you encounter the error django.core.exceptions.FieldDoesNotExist: Account has no field named 'username' with reference toUSER_MODEL_USERNAME_FIELD` in the stacktrace, the settings needed are the following:

```
ACCOUNT_USER_MODEL_USERNAME_FIELD = None
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_USERNAME_REQUIRED = False
ACCOUNT_AUTHENTICATION_METHOD = 'email'
```

see more

^ | v • Reply • Share ›

Bruno Abitbol • 10 months ago

Thanks William, your tutorial is excellent. Got it working :)

• Reply • Share ›

wsvincent Mod ➦ Bruno Abitbol • 9 months ago

Glad to hear it!

• Reply • Share ›

Saurabh Saxena • 10 months ago

I'm trying to build the web login with different social accounts.

I have implemented the social logins.

But there is this situation where lets say a user login with Facebook and then log out and then later once again login with google then two different entries are created for the same local user. How to properly maintain these records. I'm not able to find proper documentation and video tutorials for these solutions. Thus I'm trying to find a way for it.

It would be a great help for me.

thanks for your help and your valuable time.

• Reply • Share ›

dbinott • a year ago • edited

Small typos

Create a form.py file in the users app
should be formS.py

For each third-party application we want to add we just need to click on the "New" button
New should be Add

Finally on the bottom of the page under "Sites" select "127.0.0.1"
should be example.com (image as well)

• Reply • Share ›

wsvincent Mod ➦ dbinott • a year ago

Nice catches on 1 & 2. Now fixed.

For 3 with Sites, I think it's accurate as is. The display name is for our own use so it's fine to leave as "example.com" although changing to "127.0.0.1" might be clearer. What's important is that the Domain Name is changed to "127.0.0.1" for the example to work.

1 • Reply • Share ›

dbinott ➦ wsvincent • a year ago

yes I get that, but it tells us to select 127.0.0.1 but the default is example.com. So when a newbie sees that they need to select 127.0.0.1 but they see example.com, kinda confusing. Then below it states we should now change example.com to 127.0.0.1. So maybe that section can just be moved up? Make sense?

^ | ˅ • Reply • Share ›

wsvincent Mod ➦ dbinott • a year ago

It makes more sense to me now. I've changed the order so the renaming of Sites occurs first. Thanks!

^ | ˅ • Reply • Share ›

arvind yadav • a year ago

no such table: users_customuser
Request Method: POST
Request URL: http://127.0.0.1:8000/accounts/signup/
Django Version: 2.1.2
Exception Type: OperationalError
Exception Value:
no such table: users_customuser
Exception Location: /home/arvind/Downloads/djangox-master/Evelog/lib/python3.5/site-packages/django/db/backends/sqlite3/base.py in execute, line 29

^ | ˅ • Reply • Share ›

Dev Salman • a year ago

Thanks for this life saving tutorial. But one thing, if we are working on an app in which user has posts etc, then the admin panel would be different for both the apps? Like for posts and users?

^ | ˅ • Reply • Share ›

wsvincent Mod ➦ Dev Salman • a year ago

There would be a separate admin section for both posts and users, yes. I cover this at length in my books :)

https://wsvincent.com/books

^ | ˅ • Reply • Share ›

Dev Salman ➦ wsvincent • a year ago

Thanks. So is it the best practice to do that?

^ | ˅ • Reply • Share ›

wsvincent Mod ➦ Dev Salman • a year ago

Yes. Each `app` has a different section in the admin. Using a custom user model we have our `users` app and can see features related to just each user like email, username, etc. Then for, say, a `posts` app, we can see and customize info for all posts including who the user is, when it was written, what it contains, etc.

Django prefers smaller, contained apps that are easier to reason about and modify.

1 ^ | ⌄ • Reply • Share ›

Dev Salman ➦ wsvincent • a year ago

Thank you so much sir!

^ | ⌄ • Reply • Share ›

Yan Cheng Cheok • a year ago

Hi Vincent, I tried to follow your great tutorial. However, I'm facing problem in passing correct redirect URL to Google.

https://stackoverflow.com/q...

Do you have any idea how I can debug/ fix the problem? Thanks.

^ | ⌄ • Reply • Share ›

ricardo • a year ago

Thanks for the clear tute. Question regarding custom user fields: I've successfully added fields to the customuser model but how do I view / modify them in the admin panel? Ie: which forms is the admin panel using?

^ | ⌄ • Reply • Share ›

ricardo ➦ ricardo • a year ago

For more info: I added the fields to the fields tuplue in the CustomUserCreate and CustomUserChange forms, and to the list_display field in the CustomUserAdmin class. The only place it's appearing is in the user list display.

^ | ⌄ • Reply • Share ›

Load more comments

Subscribe **Add Disqus to your siteAdd DisqusAdd**

© William Vincent will@wsvincent.com