DJANGOCENTRAL

Using PostgreSQL with Django

Django is a high level full-stack open-source web framework written in Python, that encourages rapid development and clean, pragmatic design.

Django, in its 'out-of-the-box' state, is set up to communicate with SQLite – a lightweight relational database included with the Python distribution. So by default, Django automatically creates an SQLite database for your project.

In addition to SQLite, Django also has support for other popular databases that include PostgreSQL, MySQL, and Oracle.

However, PostgreSQL has a number of features that are not shared by the other databases Django supports, which makes it an idle choice for a Django app in production.

In this article, we will go through the integration of PostgreSQL with a Django Application.

Pre-Requirements

We are assuming you already have Django installed on your machine and one Django project up and running, if not then read the following article – Starting A Django Project

Installing PostgreSQL

Windows and macOS X users can download PostgreSQL from the official site https://www.postgresql.org/download/ and simply install it.

Note that tutorial is strictly based on Python 3

Linux User

```
sudo apt-get install postgresql postgresql-contrib
```

Also, Linux users need to install some dependencies for PostgreSQL to work with Python.

```
sudo apt-get install libpq-dev python3-dev
```

Install psycopg2

Next, we need to install the PostgreSQL database adapter to communicate to the database with Python to install it run the following command in the shell.

```
pip install psycopg2
```

Create A PostgreSQL User and Database

As the default configuration of Postgres is, a user called **Postgres** is made on, and the user Postgres has full super admin access to entire PostgreSQL instance running on your OS.

```
sudo -u postgres psql
```

Now the terminal should be prefixed with `postgres=#` , The above command gets you the `psql` command-line interface in full admin mode.

Now let's create a user and database.

Creating Database

```
CREATE DATABASE mydb;
```

This will create a database named `mydb` , note that every SQL statement must end with a semicolon.

Creating User

```
CREATE USER myuser WITH ENCRYPTED PASSWORD 'mypass';
```

Here we are creating a user named `myuser` with password `mypass` . You can use any username and password you wish.

Modifying Connection Parameters

```
ALTER ROLE myuser SET client_encoding TO 'utf8';
ALTER ROLE myuser SET default_transaction_isolation TO 'read commi
ALTER ROLE myuser SET timezone TO 'UTC';
```

We are setting the default encoding to `UTF-8` , which Django expects.

We are also setting the default transaction isolation scheme to " `read committed` ", which blocks reads from uncommitted transactions.

Lastly, we are setting the timezone by default, our Django projects will be set to use `UTC.` These are essential parameters recommended by the official Django team.

Granting Permission To The User

```
GRANT ALL PRIVILEGES ON DATABASE mydb TO myuser;
```

Now our user has administrative access to the database.

Now exit the SQL prompt.

```
\q
```

Integrating PostgreSQL With Django

Open the `settings.py` file of your project and scroll straight to the database section, which should look like this.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
```

```
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

We need to update these settings to integrate our PostgreSQL with the project.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'mydb',
        'USER': 'myuser',
        'PASSWORD': 'mypass',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Let's quickly go over the settings,

`DATABASES` – This constant is a dictionary of database connection information and is required by Django. You can have multiple connections to different databases, but most of the time, you will just need an entry called default.

`default` – This is the default database connection configuration. You should always have a default set of connections settings.

`'ENGINE': 'django.db.backends.postgresql_psycopg2'` – This tells Django to use the Postgres backend. This, in turn uses `psycopg2` , Python's Postgres library which we installed earlier.

`'NAME': 'mydb'` – The name of the database you want to connect to.

`'USER': 'myuser'` – The User with access to the database.

`'PASSWORD': 'mypass'` – The password for your database user.

`'HOST': 'localhost'` – The address of the database server you want to connect to.

`'PORT': ''` – The port you want to connect to, which by default is '5432'

Test The Database Connection

After updating the database configurations, it's time to test the connection. The Django database migration process ensures all Django project logic associated with a database is reflected in the database itself.

During the first migration against the database, there are a series of migrations Django requires that create tables to keep track of administrators and sessions.

In the directory where `manage.py` script exists, run the following command.

```
python manage.py migrate
```

If everything went right you should see an output like this.

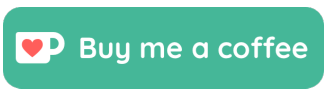
```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
```

```
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying sessions.0001_initial... OK
```

Furthermore, you can now create a superuser and login to the admin dashboard.

Support Django Central

If you appreciate my work, or if it has helped you along your journey. It would mean a lot to me if you could write a message on my wall and share a cup of coffee (or tea) with me.



15 thoughts on “Using PostgreSQL with Django”

Richard

October 17, 2019 at 11:52 am

Great tutorial, thank you!

Reply

Devil

October 17, 2019 at 11:55 am

Glad you liked it.

Reply

aryne

December 21, 2019 at 2:39 pm

I m so glad to visit this blog.This blog is really so amazing.Thanks for sharing with us.

Reply

Dana Holmes

January 6, 2020 at 8:10 pm

Hi There,
I keep getting a super long error message when I try to enter $ pip install psycopg2 command.

Then someone on the internet said to use, $ pip install psycopg2-binary. I didn't get any error messages but I still can't get the next command to work $ sudo -u postgres psql.

Do you have any suggestions?

Reply

Abhijeet

January 7, 2020 at 8:31 am

Can you please share your OS information and Python version?

Reply

Dana Holmes

January 20, 2020 at 1:55 am

I own a Mac and I am using Python 2.7.

Reply

Abhijeet

January 20, 2020 at 2:35 am

Try running `psql postgres` in the terminal or you might need to add `sudo` depending on how your system is configured.

Reply

Chuck Cherry

April 23, 2020 at 1:57 am

what is your error message? Are you running a virtual machine? You may need to upgrade your python to python3. Python 2.7 is no longer being supported.

Reply

Oz

March 19, 2020 at 8:11 pm

Hi,
I try to run the command “sudo -u postgres psql”
but I get an error: ” ‘sudo’ is not recognized as an internal or external command, operable program or batch file. ”

what I do wrong?

Reply

Abhijeet

April 21, 2020 at 2:34 pm

Either you are not using a linux system or you are not a sudo user.

Reply

mikismines

March 31, 2020 at 8:06 pm

Really problem solver

Reply

Ray Zapotee

June 19, 2020 at 6:27 pm

Hello, great article!
quick question: I was running a SQLite db and now I am trying to switch to use PostgreSQL, but I get this error that I cannot seem to find any information about online:

django.db.utils.ProgrammingError: cannot cast type numeric to interval
LINE 1: …R COLUMN “duration” TYPE interval USING “duration”::interval

Is this something you’ve seen before? Any hints on how to solve?

Reply

Yeshwanth

August 28, 2020 at 9:13 am

can i get to know how to built a model in django to read and upload a csv file dynamically

Reply

quan

September 29, 2020 at 6:25 am

nice tutorial.

Reply

void

October 21, 2020 at 10:57 am

Having trouble seeing the blog posts on index page. Shows up in admin page just not sure why PostgreSQL database isn't showing. I have one entry as a test. Any help would be appreciated.

Reply

Leave a Comment

Name *

Email *

Post Comment

About

Django Central is an educational site providing content on Python programming and web development to all the programmers and budding programmers across the internet. The main goal of this site is to provide quality tips, tricks, hacks, and other Programming resources that allows beginners to improve their skills.

Categories

Django

Programs

Python

Tools

Web Development

Pages

About Us

Contact Us

Disclaimer

Privacy Policy