9) Database Management using Flask-SQLAlchemy Lesson

Using Flask SQLAlchemy

13 min to complete · By Brandon Gigous

Contents

In this lesson, you'll look at the role of the database in a Flask app. You'll learn what SQLAlchemy is and, in turn, about Flask SQLAlchemy (aka Flask-SQLAlchemy). Finally, you'll walk through the Flask-SQLAlchemy installation process.

Why Use A Database?

Most webapps these days have one or more databases that store website data in an organized way. As data continues to grow in this *base* of data, even if it's a lot of data, it can be parsed through with queries to present users with only partial data or even to do internal server bookkeeping. These queries can get specific chunks of data as needed, such as the user's login info to verify them, hair-drying products for sale and price, or even images taken in the last year by certain photographers.

The kinds of data and how you can search them all depend on what the underlying data represents and on the database technology used, but at the end of the day, databases

store data to be uncovered at some unknown future time.

Brief Introduction to SQL and Relational Databases

That might be a lot of *data* to take into your brain, but database technology has come a long way, and it's rather easy to get started with. As a recap from our SQL & Databases course, there are two main kinds of databases: relational databases and document-oriented databases. They are often called SQL and NoSQL databases, respectively. SQL comes from Structured Query Language, which is the industry-standard language used to query relational databases.

Alert: If you aren't already familiar with these terms and technologies, we recommend you complete the SQL & Databases course first, then return here to keep going with Flask SQLAlchemy!

For your Flask webapp, you'll be using a **relational database**, which store data in **tables**. The tables model different entities in the domain, meaning that for a small online pet store business, its database tables might be Products, Customers, and Pet Categories, among other things.

The tables each have a fixed number of columns, and each column has its own type and attributes that the data under it must conform to. The rows in each table can grow or shrink as data is added or deleted. Still, there's a lot more to learn about SQL and tables than just that, but for more do check out the link to our free mini-course on SQL.

Get Mentorship

What is SQLAlchemy?

Alchemy? Cool! Does that mean, like, making potions and turning lead into gold, but with SQL? Often, I think the same sort of thing when working in Python - that I made something magically work with just a few lines of Python code. SQLAlchemy brings that same feeling when working with SQL data, all with Python!

According to the SQLAlchemy website:

> *SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.*

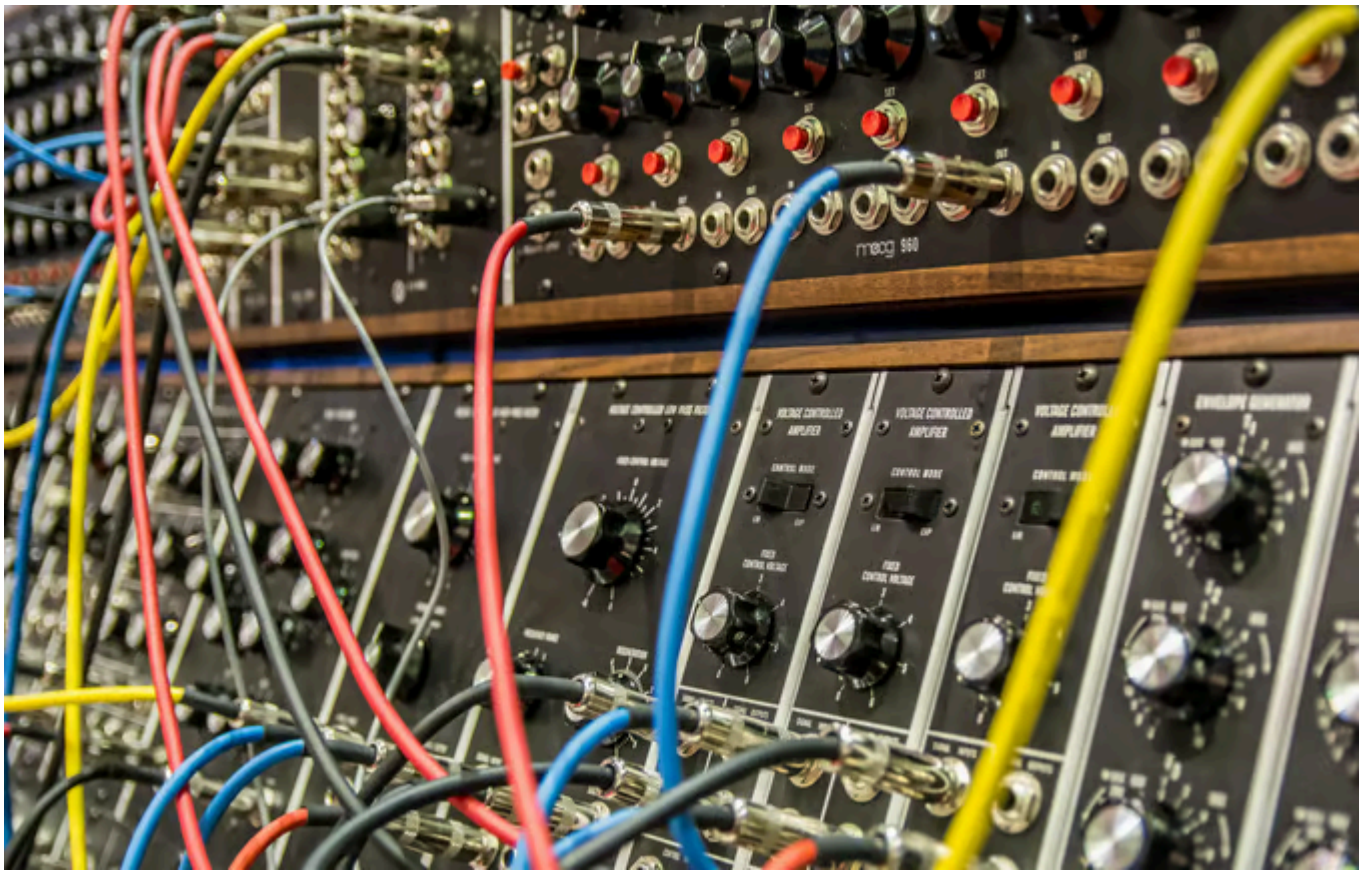
The reason SQLAlchemy is used in this course is because it's probably the most popular Python SQL toolkit out there, and for good reason. SQLAlchemy lets programmers work at a higher level with standard Python objects instead of the tables or even a query language in order to work with data in the databases. It also supports many database backends and gives programmers access to low-level SQL functionality.

What is Flask SQLAlchemy?

Because SQLAlchemy is so versatile, it didn't take a herculean effort for someone to make a Flask-compatible extension that supports SQLAlchemy. Using SQLAlchemy by itself in a Flask application makes the process of working with SQL databases in Flask much more difficult, as there is a lot of configuration that must happen first between the two.

Flask SQLAlchemy takes the tedious work out of using SQLAlchemy in your Python-Flask application. By using Flask SQLAlchemy, you don't have to configure engines and connections or any of that complicated stuff. To make the point clear, here are a couple of photos of real objects to demonstrate an important concept.

Life without Flask-SQLAlchemy:



Life with Flask-SQLAlchemy:

Make sense? Flask SQLAlchemy only needs a few tweaks to get the power of SQL in the palm of your hands, but you still have the ability to make more complicated SQLAlchemy tweaks down the road if necessary.

Now let's move on to installing SQLAlchemy for Python + Flask, so you can start working with relational databases in your app!

How to Install SQLAlchemy - Python + Flask

Just like the other extensions, you'll use pip install SQLAlchemy to install Flask-SQLAlchemy. Type the following to install SQLAlchemy for Python + Flask:

```
(env) $ pip install flask-sqlalchemy
```

Specifying a Database

With Flask-SQLAlchemy, databases are specified as a URL. While there are different URL formats for different database engines, like MySQL vs Postrges vs SQLite, for your database you'll use SQLite. The URL looks like this:

```
sqlite:////absolute/path/to/database
sqlite:///c:/absolute/path/to/database (For Windows Only)
```

Many database engines require a server hostname and sometimes a username and password for authentication, but for SQLite, all you need is a path to a filename on disk. Because you're dealing with a *Flask* extension, you can specify the database URL the *Flask* way by setting some configuration keys.

In your `hello.py` file, add near the top:

```
import os
from flask_sqlalchemy import SQLAlchemy

basedir = os.path.abspath(os.path.dirname(__file__))

app = Flask(__name__)
app.config['SECRET_KEY'] = "keep it secret, keep it safe"
app.config['SQLALCHEMY_DATABASE_URI'] =\
    'sqlite:///' + os.path.join(basedir, 'data-dev.sqlite')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# ...
```

In this snippet, there's a couple new imports added, including `SQLAlchemy` which represents the database and gives access to all the SQL goodness that Flask-SQLAlchemy provides.

To get Flask-SQLAlchemy set up with a URL, you must set the `SQLALCHEMY_DATABASE_URI` key to the database file on your server. For you, this is your development computer, and if you're following along, the database file is located in `~/Documents/CodingNomads/projects/flask-webdev/`. Of course, you probably don't have a database file yet. If it doesn't exist, SQLAlchemy will create one for you.

Alert: The key is `SQLALCHEMY_DATABASE_URI`, not `SQLALCHEMY_DATABASE_URL` (i.e., `URI`, not `URL`).

The `SQLALCHEMY_TRACK_MODIFICATIONS` key is already `False` by default, but it's set explicitly so that SQLAlchemy doesn't alert with an annoying message. Tracking modifications of the database isn't usually necessary and it eats up a lot of memory when enabled.

Once that's all done, Flask SQLAlchemy is initialized just like any other Flask extension.

And with that, you should be good to go! Next, you'll learn about models and relationships. No, no, not like Angelina Jolie and Brad Pitt. Anyway, you'll see what that

means in the next lesson.

Summary: What is Flask SQLAlchemy & How to Install SQLAlchemy

- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- Databases store data that will be uncovered at some unknown time in the future.
- There are two main kinds of databases: relational databases and document-oriented databases. They are often called SQL and NoSQL databases.
- SQL comes from Structured Query Language, which is the industry-standard language used to query relational databases.
- SQLAlchemy lets programmers work at a higher level with standard Python objects instead of the tables or even a query language in order to work with data in the databases. It also supports many database backends and gives programmers access to low-level SQL functionality.
- Using SQLAlchemy by itself in a Flask application makes the process of working with SQL databases in Flask much more difficult, as there is a lot of configuration that must happen first between the two. Flask-SQLAlchemy does almost all of that work for you.
- To install Flask-SQLAlchemy, use the following command:

```
(env) $ pip install flask-sqlalchemy
```

Info: For more information and practice on SQL & Databases, visit our SQL & Databases course, then return here to keep going with Flask SQLAlchemy!

Previous Next → Video: Install and Configure Flask-SQLAlchemy