CODING NOMADS

9) Database Management using Flask-SQLAlchemy Lesson

Database Migration with Flask-Migrate

11 min to complete · By Brandon Gigous

Contents

Up to this point, you have done some practicing with making database models, creating tables, adding rows, and querying your data. Your models are pretty simple, but because applications tend to grow naturally, their databases must grow with them. But what if you need to update your tables?

Of course, you need to add more columns to your database models so that you can build your app. But the problem is, to do that, you have to drop all tables, and your data goes away. Introducing database migrations!

What is Database Migration?

Database migration allows you to run scripts that can make the necessary changes to the database, that can also *move* the data you already have as needed." A Database migration keeps track of how the database schema changes, which is how all the data is organized collectively. In other words, a database migration framework allows incremental changes to the models to be applied.

Do you know who David Attenborough is? He's the one who narrates all those nature documentaries. Look him up if you don't know, and imagine him narrating about database migrations. Just like animals, databases have to do migrations, too! "See the data in its natural habitat, doing its migrations as usual."

What is Flask-Migrate?

Flask-Migrate is a Flask extension that is based on Alembic, which is a migration framework for SQLAlchemy. Flask-Migrate is an Alembic wrapper built for Flask, and integrates with the `flask` command.

How to Install and Run Flask-Migrate

Just like any other Flask extension, Flask-Migrate can be installed like so:

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

Initialization works a bit differently from other Flask extensions. The reason is because Flask-Migrate needs to know what database models it should be helping to migrate! Go ahead and put this in your `hello.py` file:

```
from flask-migrate import Migrate

# ...

migrate = Migrate(app, db)
```

Now you're ready to get *your database* ready for migration support. Use this CLI command whenever you start work on a new project:

```
(env) $ flask db init
```

This command will create a `migrations` directory. You will want to add this directory to your git repository, as it will store all the migration scripts needed to get your database models up to date as you continue with this project.

Mentorship Makes the Difference!

You don't have to learn alone. Join the CodingNomads' 1-on-1 Mentorship Program and get:

- A team of mentors and advisors dedicated to your success
- Weekly 1-on-1 screen share meetings with your personal, professional mentor
- Constant, personal, in-depth support 7 days a week via Discord
- Accountability, feedback, encouragement, and success

Get Mentorship

When To Perform a Database Migration

You should perform a migration whenever you make changes to your database models. Mainly, this means when you add or remove columns, or if you change the names of tables or add new models and relationships.

Database Migration Scripts

You can perform a database migration from the command line, which will generate a script, but this script may not be completely accurate. You will hear more about this, but first, let's talk about **migration scripts**.

Because Flask-Migrate is an Alembic wrapper, the migration scripts it generates are Alembic migration scripts.

Each script has two functions called `upgrade()` and `downgrade()`, which apply and remove the database changes that are part of the migration, respectively.

Theoretically, you can apply and remove changes, which means you can reconfigure a database to any point in its change history.

How to Perform a Database Migration with Flask-Migrate

1. Make whatever changes you need to the model classes
2. Create an automatic migration script with `flask db migrate`

3. Review the script and make any adjustments necessary so that it accurately reflects changes made to models
4. Add the script to source control
5. Finally, run the `flask db upgrade` to apply the migration

The `flask db migrate` command creates a migration script automatically:

```
(venv) $ flask db migrate -m "initial migration"
INFO [alembic.migration] Context impl SQLiteImpl.
INFO [alembic.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate] Detected added table 'roles'
INFO [alembic.autogenerate] Detected added table 'users'
INFO [alembic.autogenerate.compare] Detected added index
'ix_users_username' on '['username']'
Generating /home/yourname/Documents/CodingNomads/projects/flask-webd
```

Now for the warning: make sure *ALL* changes to your models are reflected in your migration scripts! Otherwise any `upgrade()` commands won't reflect what changes you ultimately made, and the result might look strange. Automatic migrations aren't always 100%. They can miss some things. If, say, you rename a column, that may not show up in the migration script as a renaming. Instead, it may show up as a deleted column and an added column. That means the migration will delete your original column, and you'll lose all data within it!

Free webinar: Is now a good time to get into tech?

Starting soon! How you can break into software engineering, data science, and AI in 2025 without spending $10k+ or quitting your day job.

What you'll learn:

- State of the tech job market – what's the deal with layoffs, AI taking our jobs, and shady coding bootcamps?
- Who IS getting jobs right now – and the skills in high demand

- What it really takes to get a technical job today – it's more than just tech skills
- And finally, is pursuing these paths still worth your time and money?

Register Here

Upgrade Your Database

You have your migration script. It's been reviewed and accepted and checked for any inconsistencies, and now you're ready to upgrade! The `flask db upgrade` command applies the migration script to the database:

```
(env) $ flask db upgrade
INFO [alembic.migration] Context impl SQLiteImpl.
INFO [alembic.migration] Will assume non-transactional DDL.
INFO [alembic.migration] Running upgrade None -> 5f15cbadc711, initi
```

What's the difference between an upgrade versus nuking the database and starting fresh with a `db.create_all()`? Well, the difference is with an upgrade, you still have your data, because it makes all the changes like adding columns without removing any content.

You've made it to the end of your primer on databases in a webapp; congratulations! While this section only covered the basics, databases and migrations can get pretty complicated pretty quickly, so don't worry if you get stuck. There's always help on our Discord and if you have a mentor, you can ask them, too.

Do you know what else can get complicated? Webapps themselves. In the next section, you will discover how to mitigate some of the challenges that complexity will inevitably bring.

Summary: How to Run a Database Migration with Flask-Migrate

- A database migration framework allows incremental changes to the models to be applied.
- Flask-Migrate is a Flask extension that is based on Alembic, which is a migration framework for SQLAlchemy
- You should perform a database migration whenever you make changes to your database models.

Database Migration Steps

1. Make whatever changes you need to the model classes
2. Create an automatic migration script with `flask db migrate`
3. Review the script and make any adjustments necessary so that it accurately reflects changes made to models
4. Add the script to source control
5. Finally, run the `flask db upgrade` to apply the migration

The `flask db migrate` command creates a migration script automatically:

```
(venv) $ flask db migrate -m "initial migration"
INFO [alembic.migration] Context impl SQLiteImpl.
INFO [alembic.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate] Detected added table 'roles'
INFO [alembic.autogenerate] Detected added table 'users'
INFO [alembic.autogenerate.compare] Detected added index
'ix_users_username' on '['username']'
Generating /home/yourname/Documents/CodingNomads/projects/flask-webd
```

Finally, you're ready to upgrade! The `flask db upgrade` command applies the migration script to the database.

Previous

Next → Video: Performing a Database Migration with Flask-...

Want to go faster with dedicated 1-on-1 support? Enroll in a Bootcamp program.

Learn more

Beginner - Intermediate Courses

Java Programming

Python Programming

JavaScript Programming

Git & GitHub

SQL + Databases

Intermediate - Advanced Courses

Spring Framework

Data Science + Machine Learning

Deep Learning with Python

Django Web Development

Flask Web Development

Career Tracks

Java Engineering Career Track

Python Web Dev Career Track

Data Science / ML Career Track

Career Services

Resources

About CodingNomads

Corporate Partnerships

Contact us

Blog

Discord

© 2016-2024 CodingNomads LLC All Rights Reserved admin@codingnomads.com +1.415.523.4643

Privacy Policy Terms of Use Acceptable Use Policy Disclaimer DSAR Consent Preferences