Flask-Migrate: Alembic database migration wrapper for Flask (/post/flask-migrate-alembic-database-migration-wrapper-for-flask)

September 9 2013

Posted by Miguel Grinberg (/author/Miguel Grinberg) under Flask (/category/Flask), Python (/category/Python), Programming (/category/Programming).

Tweet Like G+ Share

In this post I introduce you to Flask-Migrate, a new database migration handler for Flask based on Alembic (https://alembic.readthedocs.org/en/latest/index.html) that I just made public.

Is a New Extension Necessary?

If you read the database chapter of my Mega-Tutorial (http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-iv-database), you know that I have chosen sqlalchemy-migrate (https://code.google.com/p/sqlalchemy-migrate/) for database migrations.

I liked sqlalchemy-migrate back then (I still do, actually), but its development appears to have halted completely. Support for SQLAlchemy 0.8.x has not been implemented yet, six months past the 0.8.0 release.

On the other side, since I wrote my migration Mega-Tutorial chapter Alembic has gained notoriety. Alembic is written by zzzeek (Mike Bayer), who is the author of SQLAlchemy. He is actively developing Alembic on bitbucket (https://bitbucket.org/zzzeek/alembic).

There is an extension called Flask-Alembic (https://github.com/tobiasandtobias/flask-alembic) out there that has many similarities to mine, but that project also appears to have stalled, there haven't been any commits or messages from the developers in several months. The project was never made available on the Python Package Index (PyPI), so while it is possible to install directly from git, that is less ideal, and might be a deal breaker for some.

That is why I have decided to write Flask-Migrate. Out of respect for the Flask-Alembic project I decided to use a different name on PyPI, in case they ever decide to resume work on their project and publish it.

Using Flask-Migrate

Flask-Migrate provides a set of command line options that attach to Flask-Script (http://flask-script.readthedocs.org/en/latest/).

To install the extension you use `pip` as usual:

```
$ pip install flask-migrate
```

As part of the installation you will also get Flask, Flask-SQLAlchemy and Flask-Script.

Below is a sample application that initializes Flask-Migrate and registers it with Flask-Script. As is typically the case with Flask-Script, the script is called `manage.py`:

```
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
from flask.ext.script import Manager
from flask.ext.migrate import Migrate, MigrateCommand

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'

db = SQLAlchemy(app)
migrate = Migrate(app, db)

manager = Manager(app)
manager.add_command('db', MigrateCommand)

class User(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    name = db.Column(db.String(128))

if __name__ == '__main__':
    manager.run()
```

When you run the application you get an additional `db` option in the command line (you can call it differently if you want, of course):

```
$ python manage.py --help
usage: manage.py [-h] {shell,db,runserver} ...

positional arguments:
  {shell,db,runserver}
    shell               Runs a Python shell inside Flask application context.
    db                  Perform database migrations
    runserver           Runs the Flask development server i.e. app.run()

optional arguments:
  -h, --help            show this help message and exit
```

The `db` command exposes most of the Alembic options:

```
$ python manage.py db --help
usage: Perform database migrations

positional arguments:
  {upgrade,migrate,current,stamp,init,downgrade,history,revision}
    upgrade             Upgrade to a later version
    migrate             Alias for 'revision --autogenerate'
    current             Display the current revision for each database.
    stamp               'stamp' the revision table with the given revision;
                        dont run any migrations
    init                Generates a new migration
    downgrade           Revert to a previous version
    history             List changeset scripts in chronological order.
    revision            Create a new revision file.

optional arguments:
  -h, --help            show this help message and exit
```

To add migration support to your database you just need to run the `init` command:

```
$ python manage.py db init
  Creating directory /home/miguel/app/migrations...done
  Creating directory /home/miguel/app/migrations/versions...done
  Generating /home/miguel/app/alembic.ini...done
  Generating /home/miguel/app/migrations/env.py...done
  Generating /home/miguel/app/migrations/env.pyc...done
  Generating /home/miguel/app/migrations/README...done
  Generating /home/miguel/app/migrations/script.py.mako...done
  Please edit configuration/connection/logging settings in
  '/home/miguel/app/migrations/alembic.ini' before proceeding.
```

Note that you should replace `manage.py` with the name of your launch script if you used a different name.

When you use Alembic alone you have to edit a couple of configuration files, but Flask-Migrate handles all that for you. When the `init` command completes you will have a *migrations* folder with the configuration files ready to be used.

To issue your first migration you can run the following command:

```
$ python manage.py db migrate
INFO  [alembic.migration] Context impl SQLiteImpl.
INFO  [alembic.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate] Detected added table 'user'
  Generating /home/miguel/app/migrations/versions/4708a5190f2_.py...done
```

The `migrate` command adds a new migration script. You should review it and edit it to be accurate, as Alembic cannot detect all changes that you make to your models. In particular it does not detect indexes, so those need to be added manually to the script.

If you prefer to write your migration scripts from scratch then use `revision` instead of `migrate`:

```
$ python manage.py db revision
  Generating /home/miguel/app/migrations/versions/15c04479d683_.py...done
```

You can read Alembic's documentation to learn how to write migration scripts.

The next step is to apply the migration to the database. For this you use the `upgrade` command:

```
$ python manage.py db upgrade
INFO  [alembic.migration] Context impl SQLiteImpl.
INFO  [alembic.migration] Will assume non-transactional DDL.
INFO  [alembic.migration] Running upgrade None -> 4708a5190f2, empty message
```

And that's it! Your database is now synchronized with your models.

You should add all the files in the *migrations* folder to version control along with your source files. If you need to update another system to the latest database version you just need to update your source tree on that other system and then run `db upgrade`, like you did above.

If you have any suggestions to improve this extension please let me know below in the comments.

Miguel

Tweet Like G+ Share

108 comments

#51 **Miguel Grinberg** said 4 years ago

@hewx: this is explained in the Alembic documentation: https://alembic.readthedocs.org/en/latest/tutorial.html#auto-generating-migrations

#52 **Navaneethan (http://navaspot.wordpress.com)** said 4 years ago

Hai Miguel,

If you use manager.run instead of app.run, then how would you configure custom port like app.run(port=5001), how can you integrate in this manager script?

#53 **Miguel Grinberg** said 4 years ago

@Navaneethan: when you use Flask-Script you can select the port in a command line argument:

$ python manage.py --port 5001

#54 **Brian** said 4 years ago

Great tutorial, but having trouble getting it to play with Flask-Security (it's probably an easy fix): https://stackoverflow.com/questions/24410913/using-flask-migrate-together-with-flask-security

#55 **Frode Egeland (http://trippinforayear.blogspot.com)** said 4 years ago

Hi Miguel,

Love your site, tutorials and the book!

Having some issues, though..
How do you integrate the Flask-Migrate 'db' command into the manage.py file when you are using blueprints, as per your api-pycon2014 talk?
I'm using the code from that talk as my starting point, as it's a lot nicer than Flask-RESTful, but I'm still confused about how I should get things working in the manage.py script.
Right now, I tried to just add it like:

migrate = Migrate(create_app, db)
manager.add_command('db', MigrateCommand)

but I get

```
...
  File "/home/user/myapi/venv/lib/python3.3/site-packages/flask_migrate/__init__.py", line 46, in init
    directory = current_app.extensions['migrate'].directory
KeyError: 'migrate'
```

Any tips? I feel I'm missing something that's probably obvious...

Cheers!

#56 **Miguel Grinberg** said 4 years ago

@Frode: Flask-Migrate's constructor does not take an app factory function, you have to pass an already created app object.

#57 **Alan** said 4 years ago

Is it possible to pull the models from an outside python file like one dedicated for the models and import the classes into the manage.py file or should they all be defined within the manage.py script?

#58 **Miguel Grinberg** said 4 years ago

@Alan: yes, you should put the models in their own file for larger applications. Have a look at Flasky for an example (https://github.com/miguelgrinberg/flasky), this is the application featured in my book.

#59 **Vince Fulco (www.example.com)** said 3 years ago

Hi- Fairly new at debugging. Using Flask-Migrate 1.2.0 in the cookiecutter flask example. Traceback indicates something wrong with line #40 in flask_migrate/__init__.py. Specifically a type error in MigrateCommand.

```
 File "/usr/local/lib/python2.7/dist-packages/flask_migrate/__init__.py", line 40, in <module>
    MigrateCommand = Manager(usage = 'Perform database migrations')
TypeError: __init__() takes at least 2 arguments (2 given)
```

Thanks for any insights. Best, V.

#60 **Miguel Grinberg** said 3 years ago

@Vince: what version of Flask-Script do you have? Can you check in your Flask-Script's __init__.py file what are the arguments that are expected for class Manager?

#61 **Vince Fulco** said 3 years ago

Thanks for the quick response. Using Flask-Script==0.4.0 and Flask==0.10.1. The class Manager init argumes/line is as follows.

```
class Manager(object):

    """
  foo comments here
      """

    def __init__(self, app, with_default_commands=True, usage=None):
```

#62 **Vincent Fulco** said 3 years ago

Miguel- Stop the presses. I found a sub-page with some additional instructions I missed. Will need to init and manipulate the dbase a bit before I can launch the app. Thanks much.

#63 **Miguel Grinberg** said 3 years ago

@Vince: the Flask-Script version that you have is very old. Try version 0.6.6, that is the version I have used back when I created the project for my Flask book.

#64 **Vincent C. Fulco** said 3 years ago

That did it, thanks very much. I visited a repo and some other sites trying to ascertain beforehand and 0.4.0 appeared to be the latest. Is forcing upgrades on all the modules thru pip the best way to know for sure?

#65 **Miguel Grinberg** said 3 years ago

@Vince: Normally I would say yes, use "pip install --upgrade". The thing with Flask-Script is that it is a project that has a loaded history and a couple of changes in ownership. Version 0.6.6 is the version that was current back when I was writing my Flask book, so I know it is a stable release. There is a 0.6.7, a 1.0.0, and a few 2.0.x releases, all released after my book. I know the 2.0.x releases have some issues, so if you try any of those keep in mind the possibility that you may need to downgrade.

#66 **mar** said 3 years ago

Hi, I was trying to do something like:
migrate = Migrate(app, db)
migrate.upgrade()
programmatically upgraded,
however this method does not work and no other method seems tested on the github repo other than using the manager. Is this deliberate? I wanted to run upgrade each time the app deploys/restarts to make sure everything is correct on the db, and if not fail the deploy. Is this a reasonable use case for this?
Thanks

#67 **mar** said 3 years ago

got it working with this:
```
Migrate(app, db)
        with app.app_context():
                flask.ext.migrate.upgrade()
```

though kinda wonky

#68 **Miguel Grinberg** said 3 years ago

@mar: here is how to trigger an upgrade programmatically:

from flask.ext.migrate import upgrade
upgrade()

I have an example usage in Flasky, the application that is featured in my book: https://github.com/miguelgrinberg/flasky/blob/master/manage.py#L71.

#69 wandonye said 3 years ago

Come your post again while trying to find a better choice to sqlalchemy-migrate. Does flask-migrate support geoalchemy2? Thank you~

#70 wandonye said 3 years ago

Hi Miguel, I'm having the same problem as Aidan and Miho: my app is structured as in your mega tutorial. Models.py, which contains all the classes, is located in ./app/. Manage.py, which is located in ./, is the same as the first block of codes in your post, except the definition of the class User is replaced by "from app.models import [classes in Models.py]". But the migrate file generated has nothing but pass in it. To clarify, I have already delete the old database and created a new one, so there is no way the database matches the models.

#71 Miguel Grinberg said 3 years ago

@wandonye: I do not know, I have never used geoalchemy2. If you try it please let me know!

#72 Miguel Grinberg said 3 years ago

@wandonye: The devil is in the details. To be able to tell you, I need to see a complete copy of your manage.py file. Put it up on pastebin or similar and send me the link.

#73 wandonye said 3 years ago

It does work with geoalchemy2, except some importing from geoalchemy2 has to be done manually. Check out our discussion on stackoverflow: http://stackoverflow.com/questions/27531569/how-to-create-a-migration-script-to-add-a-geometry-column-in-sqlalchemy-migrate

For the other question that three of us had, I think it's the setup incorrect. I followed
http://stackoverflow.com/questions/19323990/flask-migrate-not-creating-tables
and put the migrate commands into app/__init__.py, and manage.py only contains
#--------
from app import manager
manager.run()
#---------
Then everything works. So I guess the problem of my previous setup was that the commands in manage.py did not find the classes in app/models.py because it is outside the app folder.

#74 peggy said 3 years ago

Hi Miguel, Thanx for your tutorial, I make a web application based on your tutorial Mega-Tutorial, and I used sqlalchemy-migrate now I should have Alembic for migration with Flask-Migrate is there any way that I can change from sqlalchemy-migrate to flask-migrate/Alembic?? coz my app is complete and do it again it's a lot of work I think ?

#75 Miguel Grinberg said 3 years ago

@peggy: I just answered this question on StackOverflow: http://stackoverflow.com/questions/28135936/change-database-migrations-from-sqlalchemy-migrate-to-flask-migrate

«« (/post/flask-migrate-alembic-database-migration-wrapper-for-flask/page/1#comments)
« (/post/flask-migrate-alembic-database-migration-wrapper-for-flask/page/2#comments)
»» (/post/flask-migrate-alembic-database-migration-wrapper-for-flask/page/0#comments)
» (/post/flask-migrate-alembic-database-migration-wrapper-for-flask/page/4#comments)

Leave a Comment

Name

URL

Email

Comment

Captcha

I'm not a robot

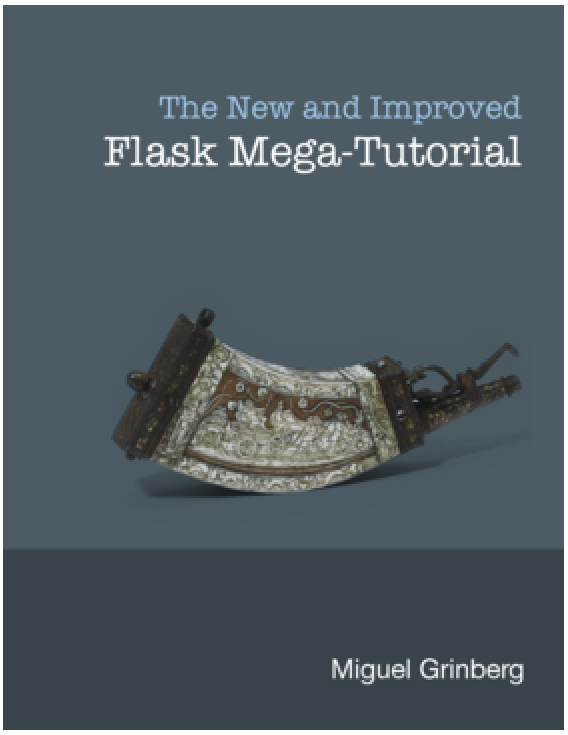
reCAPTCHA
Privacy - Terms

Submit

The New Flask Mega-Tutorial

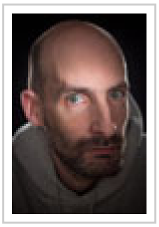
My Kickstarter (https://www.kickstarter.com/projects/1124925856/the-new-and-improved-flask-mega-tutorial) project was a big success! I'm now releasing a chapter of the new and improved Flask Mega-Tutorial every Tuesday here on this blog!

I have also created ebook and video versions of the complete tutorial, which I'm offering for sale. Click on the book cover below for more information!



(https://learn.miguelgrinberg.com)

About Miguel



Welcome to my blog!

I'm a software engineer, photographer and filmmaker in Portland, Oregon, USA.

You can also find me on Facebook (https://www.facebook.com/miguelgrinbergblog), Google+ (https://plus.google.com/u/0/117786742456929977820), LinkedIn (http://www.linkedin.com/in/miguelgrinberg), Github (https://github.com/miguelgrinberg) and Twitter (https://twitter.com/#!/miguelgrinberg).

Thank you for visiting!

Categories

(/category/AWS/feed) **AWS (/category/AWS)** (1)
(/category/Arduino/feed) **Arduino (/category/Arduino)** (7)
(/category/Authentication/feed) **Authentication (/category/Authentication)** (5)
(/category/Blog/feed) **Blog (/category/Blog)** (1)
(/category/C++/feed) **C++ (/category/C++)** (5)
(/category/Cloud/feed) **Cloud (/category/Cloud)** (6)
(/category/Database/feed) **Database (/category/Database)** (11)
(/category/Docker/feed) **Docker (/category/Docker)** (1)

- (/category/Filmmaking/feed) Filmmaking (/category/Filmmaking) (6)
- (/category/Flask/feed) Flask (/category/Flask) (74)
- (/category/Games/feed) Games (/category/Games) (1)
- (/category/HTML5/feed) HTML5 (/category/HTML5) (1)
- (/category/Heroku/feed) Heroku (/category/Heroku) (1)
- (/category/JavaScript/feed) JavaScript (/category/JavaScript) (7)
- (/category/Microservices/feed) Microservices (/category/Microservices) (2)
- (/category/Movie Reviews/feed) Movie Reviews (/category/Movie Reviews) (5)
- (/category/Netflix/feed) Netflix (/category/Netflix) (5)
- (/category/Node.js/feed) Node.js (/category/Node.js) (1)
- (/category/OpenStack/feed) OpenStack (/category/OpenStack) (1)
- (/category/Personal/feed) Personal (/category/Personal) (2)
- (/category/Photography/feed) Photography (/category/Photography) (7)
- (/category/Product Reviews/feed) Product Reviews (/category/Product Reviews) (2)
- (/category/Programming/feed) Programming (/category/Programming) (90)
- (/category/Project Management/feed) Project Management (/category/Project Management) (1)
- (/category/Python/feed) Python (/category/Python) (85)
- (/category/REST/feed) REST (/category/REST) (5)
- (/category/Rackspace/feed) Rackspace (/category/Rackspace) (1)
- (/category/Raspberry Pi/feed) Raspberry Pi (/category/Raspberry Pi) (7)
- (/category/Robotics/feed) Robotics (/category/Robotics) (6)
- (/category/Security/feed) Security (/category/Security) (9)
- (/category/Video/feed) Video (/category/Video) (5)
- (/category/Webcast/feed) Webcast (/category/Webcast) (2)
- (/category/Windows/feed) Windows (/category/Windows) (1)