
Aerich

license Apache-2.0 license Apache-2.0  ci passing  ci passing

English | Русский

Introduction

Aerich is a database migrations tool for TortoiseORM, which is like alembic for SQLAlchemy, or like Django ORM with it's own migration solution.

Install

Just install from pypi:

```
1 pip install aerich
```

Quick Start

```
1 > aerich -h
2
3 Usage: aerich [OPTIONS] COMMAND [ARGS]...
4
5 Options:
6   -V, --version          Show the version and exit.
7   -c, --config TEXT      Config file. [default: pyproject.toml]
8   --app TEXT             Tortoise-ORM app name.
9   -h, --help             Show this message and exit.
10
11 Commands:
12   downgrade      Downgrade to specified version.
13   heads          Show current available heads in migrate location.
14   history        List all migrate items.
15   init           Init config file and generate root migrate location.
16   init-db        Generate schema and generate app migrate location.
17   inspectdb      Introspects the database tables to standard output as...
18   migrate        Generate migrate changes file.
19   upgrade        Upgrade to specified version.
```

Usage

You need add `aerich.models` to your Tortoise-ORM config first. Example:

```
1 TORTOISE_ORM = {
2     "connections": {"default": "mysql://root:123456@127.0.0.1:3306/test"},
3     "apps": {
4         "models": {
5             "models": ["tests.models", "aerich.models"],
6             "default_connection": "default",
7         },
8     },
9 }
```

Initialization

```
1 > aerich init -h
2
3 Usage: aerich init [OPTIONS]
4
5   Init config file and generate root migrate location.
6
7 Options:
8   -t, --tortoise-orm TEXT  Tortoise-ORM config module dict variable,
9                             like
9                             settings.TORTOISE_ORM. [required]
10  --location TEXT           Migrate store location. [default: ./
11                             migrations]
11  -s, --src_folder TEXT     Folder of the source, relative to the
12                             project root.
12  -h, --help                Show this message and exit.
```

Initialize the config file and migrations location:

```
1 > aerich init -t tests.backends.mysql.TORTOISE_ORM
2
3 Success create migrate location ./migrations
4 Success write config to pyproject.toml
```

Init db

```
1 > aerich init-db
2
3 Success create app migrate location ./migrations/models
4 Success generate schema for app "models"
```

If your Tortoise-ORM app is not the default `models`, you must specify the correct app via `--app`, e.g. `aerich --app other_models init-db`.

Update models and make migrate

```
1 > aerich migrate --name drop_column
2
3 Success migrate 1_202029051520102929_drop_column.py
```

Format of migrate filename is {version_num}_{datetime}_{name|update}.py.

If `aerich` guesses you are renaming a column, it will ask `Rename {old_column} to {new_column} [True]`. You can choose `True` to rename column without column drop, or choose `False` to drop the column then create. Note that the latter may lose data.

If you need to manually write migration, you could generate empty file:

```
1 > aerich migrate --name add_index --empty
2
3 Success migrate 1_202326122220101229_add_index.py
```

Upgrade to latest version

```
1 > aerich upgrade
2
3 Success upgrade 1_202029051520102929_drop_column.py
```

Now your db is migrated to latest.

Downgrade to specified version

```
1 > aerich downgrade -h
2
3 Usage: aerich downgrade [OPTIONS]
4
5     Downgrade to specified version.
6
7 Options:
8   -v, --version INTEGER  Specified version, default to last. [default:
9                           -1]
9   -d, --delete            Delete version files at the same time. [default:
10                           False]
11
12   --yes                  Confirm the action without prompting.
13   -h, --help             Show this message and exit.
```

```
1 > aerich downgrade
2
```

```
3 Success downgrade 1_202029051520102929_drop_column.py
```

Now your db is rolled back to the specified version.

Show history

```
1 > aerich history
2
3 1_202029051520102929_drop_column.py
```

Show heads to be migrated

```
1 > aerich heads
2
3 1_202029051520102929_drop_column.py
```

Inspect db tables to TortoiseORM model

Currently `inspectdb` support MySQL & Postgres & SQLite.

```
1 Usage: aerich inspectdb [OPTIONS]
2
3   Introspects the database tables to standard output as TortoiseORM
   model.
4
5 Options:
6   -t, --table TEXT  Which tables to inspect.
7   -h, --help        Show this message and exit.
```

Inspect all tables and print to console:

```
1 aerich --app models inspectdb
```

Inspect a specified table in the default app and redirect to `models.py`:

```
1 aerich inspectdb -t user > models.py
```

For example, you table is:

```
1 CREATE TABLE `test`
2 (
3   `id`          int          NOT NULL AUTO_INCREMENT,
4   `decimal`    decimal(10, 2) NOT NULL,
5   `date`       date          DEFAULT NULL,
```

```

6      `datetime` datetime          NOT NULL          DEFAULT
          CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
7      `time`      time              DEFAULT NULL,
8      `float`     float              DEFAULT NULL,
9      `string`    varchar(200) COLLATE utf8mb4_general_ci DEFAULT NULL,
10     `tinyint`   tinyint             DEFAULT NULL,
11     PRIMARY KEY (`id`),
12     KEY `asyncmy_string_index` (`string`)
13 ) ENGINE = InnoDB
14     DEFAULT CHARSET = utf8mb4
15     COLLATE = utf8mb4_general_ci

```

Now run `aerich inspectdb -t test` to see the generated model:

```

1  from tortoise import Model, fields
2
3
4  class Test(Model):
5      date = fields.DateField(null=True, )
6      datetime = fields.DatetimeField(auto_now=True, )
7      decimal = fields.DecimalField(max_digits=10, decimal_places=2, )
8      float = fields.FloatField(null=True, )
9      id = fields.IntField(pk=True, )
10     string = fields.CharField(max_length=200, null=True, )
11     time = fields.TimeField(null=True, )
12     tinyint = fields.BooleanField(null=True, )

```

Note that this command is limited and can't infer some fields, such as `IntEnumField`, `ForeignKeyField`, and others.

Multiple databases

```

1  tortoise_orm = {
2      "connections": {
3          "default": expand_db_url(db_url, True),
4          "second": expand_db_url(db_url_second, True),
5      },
6      "apps": {
7          "models": {"models": ["tests.models", "aerich.models"], "
8                      default_connection": "default"},
9          "models_second": {"models": ["tests.models_second"], "
10                             default_connection": "second", },
11      },
12  }

```

You only need to specify `aerich.models` in one app, and must specify `--app` when running `aerich migrate` and so on.

Restore aerich workflow

In some cases, such as broken changes from upgrade of `aerich`, you can't run `aerich migrate` or `aerich upgrade`, you can make the following steps:

1. drop `aerich` table.
2. delete `migrations/{app}` directory.
3. rerun `aerich init-db`.

Note that these actions is safe, also you can do that to reset your migrations if your migration files is too many.

Use aerich in application

You can use `aerich` out of cli by use `Command` class.

```
1 from aerich import Command
2
3 command = Command(tortoise_config=config, app='models')
4 await command.init()
5 await command.migrate('test')
```

License

This project is licensed under the Apache-2.0 License.