2019年04月30日 阅读 647

Flask框架从入门到精通之模型迁移操作（十六）

> 知识点：1、模型迁移

一、概况

在Django框架开发过程中，我们对数据库字段添加或删除，直接修改模型类，然后进行迁移可以了，非常方便。我们也想让Flask框架支持这样的操作，就需要使用Flask-Migrate扩展，来实现数据迁移。并且集成到Flask-Script中，所有操作通过命令就能完成。

二、安装

为了导出数据库迁移命令，Flask-Migrate提供了一个MigrateCommand类，可以附加到flask-script的manager对象上。先安装下面两个扩展：

复制代码

```
pip install Flask-Script
```

复制代码

```
pip install flask-migrate
```

三、配置

复制代码

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
import pymysql
from flask_migrate import Migrate, MigrateCommand
from flask_script import Manager

pymysql.install_as_MySQLdb()
app = Flask(__name__)

# 通过脚本管理flask程序
manager = Manager(app)

# 设置连接数据库的URL
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:123456@127.0.0.1:3306/db_flask'

# 设置每次请求结束后会自动提交数据库中的改动
app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN'] = True

# 数据库和模型类同步修改
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
```

```
# 查询时会显示原始SQL语句
app.config['SQLALCHEMY_ECHO'] = True

db = SQLAlchemy(app)

# 创建数据库迁移对象
Migrate(app, db)

# 向脚步管理添加数据库迁移命令 db指命令的别名
manager.add_command('db', MigrateCommand)


# 类型
class Type(db.Model):
    # 表名
    __tablename__ = 'tbl_types'

    # 数据库真正存在的字段
    id = db.Column(db.Integer, primary_key=True)  # 主键
    name = db.Column(db.String(32), unique=True)  # 名字

    # 数据库中不存在的字段,只是为了查找和反向查找。
    # backref:在关系的另一模型中添加反向引用
    heros = db.relationship("Hero", backref='type')

    def __repr__(self):
        return self.name


# 英雄
class Hero(db.Model):
    # 表名
    __tablename__ = 'tbl_heros'
    # 数据库真正存在的字段
    id = db.Column(db.Integer, primary_key=True)  # 主键
    name = db.Column(db.String(64), unique=True)  # 名字
    gender = db.Column(db.String(64))  # 性别

    # 外键 一个射手对应很多英雄
    type_id = db.Column(db.Integer, db.ForeignKey("tbl_types.id"))

    def __repr__(self):
        return self.name


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    # app.run(host='0.0.0.0', port=5000)  # 运行程序
    manager.run()
```

四、添加字段

首先我们通过命令创建出migrations文件夹,后面所有的迁移文件都会放在这个文件夹里面

复制代码

```
python flask_migrate_db.py db init
```

如图所示:

- 生成迁移文件

下面这条命令跟我们Django里面的makemigrations一样，是生成迁移文件的作用。因为我们的模型类并没有添加或删除字段，所有第一次会出现没有改变的提示。-m：给迁移文件加上注释

```
python flask_migrate_db.py db migrate -m 'first create'
```

提示:

```
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.env] No changes in schema detected.
```

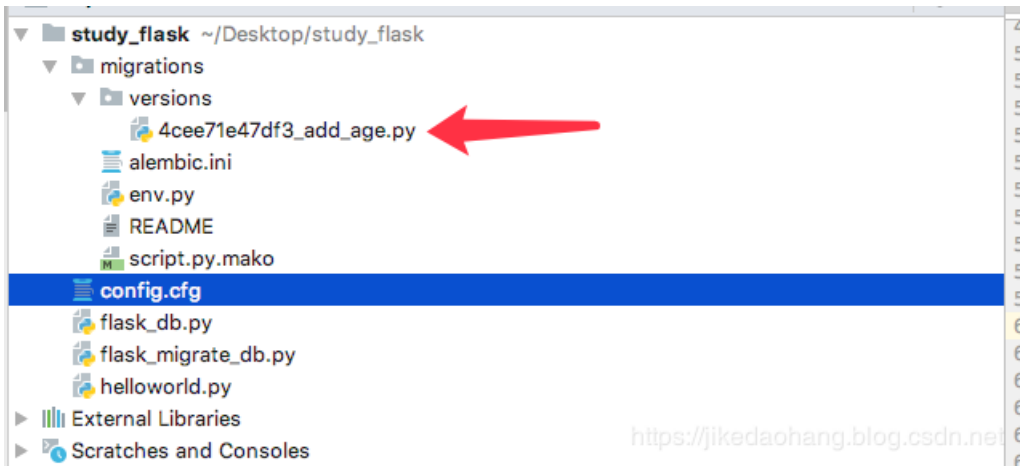
- 添加新字段 我们在英雄里面添加一个年龄字段，再迁移一下：

```
age = db.Column(db.Integer)  # 年龄
```

迁移

```
python flask_migrate_db.py db migrate -m 'add age'
```

迁移文件会生成到migrations文件夹中如图：



生成迁移文件，这个时候数据库并没有改变，我们还要用upgrade命令同步到数据库中：

```
python flask_migrate_db.py db upgrade
```

我们在本地数据库查看一下表结构：

```
mysql> desc tbl_heros;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| name    | varchar(64) | YES  | UNI | NULL    |                |
| gender  | varchar(64) | YES  |     | NULL    |                |
| type_id | int(11)     | YES  | MUL | NULL    |                |
| age     | int(11)     | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
```

age字段已经添加到数据库当中了。

五、回退

回退数据库时，需要指定回退版本号，由于版本号是随机字符串，为避免出错，建议先使用python flask_migrate_db.py db history命令查看历史版本的具体版本号，然后复制具体版本号执行回退。

```
(flask_1.0) ➜  study_flask python flask_migrate_db.py db history
<base> -> 4cee71e47df3 (head), add age
(flask_1.0) ➜  study_flask
```

里面base指的是原始版本。例如我们发现我们刚才添加的字段并没有什么作用，我们就可以回退到原始版本。

```
python flask_migrate_db.py db downgrade base
```

我们在本地数据库查看一下表结构：

```
mysql> desc tbl_heros;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| name    | varchar(64) | YES  | UNI | NULL    |                |
| gender  | varchar(64) | YES  |     | NULL    |                |
| type_id | int(11)     | YES  | MUL | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

后面如果有很多版本，直接指定版本号就可以。例如：

```
python flask_migrate_db.py db downgrade 4cee71e47df3
```

在Flask中有了对象模型类迁移的扩展，维护起来表结构，方便了很多。

欢迎关注我的公众号：

文章分类

后端

文章标签

Flask

相关推荐