

小白学Flask第十三天| 来谈谈数据库迁移、邮箱扩展的那些事！

原创 JAP君 Python进击者

2019-11-06原文



主要内容：

1. 数据库migrate扩展的使用简介
2. migrate的使用
3. 邮箱扩展

数据库migrate扩展的使用简介

在开发过程中，需要修改数据库模型，而且还要在修改之后更新数据库。最直接的方式就是删除旧表，但这样会丢失数据。

更好的解决办法是使用数据库迁移框架，它可以追踪数据库模式的变化，然后把变动应用到数据库中。

在 Flask 中可以使用 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。

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。

首先要在虚拟环境中安装 Flask-Migrate 和 Flask-Script。

```
pip install flask-migrate
```

```
pip install flask-script
```

migrate 的使用

我们编写一个 py 文件来看看如何迁移数据库

database.py :

```
#coding=utf-8
```

```
from flask import Flask
```

```
from flask_sqlalchemy import SQLAlchemy
```

```
from flask_migrate import Migrate, MigrateCommand
```

```
from flask_script import Shell, Manager
```

```
app = Flask(__name__)
```

```
manager = Manager(app)
```

```
app.config['SQLALCHEMY_DATABASE_URI'] =  
'mysql://root:mysql@127.0.0.1:3306/Flask_test'  
  
app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN'] = True  
  
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True  
  
db = SQLAlchemy(app)
```

#第一个参数是Flask的实例，第二个参数是SqlAlchemy数据库实例

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

#manager是Flask-Script的实例，这条语句在flask-Script中添加一个db命令

```
manager.add_command('db',MigrateCommand)
```

#定义模型Role

```
class Role(db.Model):  
  
    # 定义表名  
  
    __tablename__ = 'roles'  
  
    # 定义列对象  
  
    id = db.Column(db.Integer, primary_key=True)  
  
    name = db.Column(db.String(64), unique=True)  
  
    def __repr__(self):  
  
        return 'Role:'.format(self.name)
```

#定义用户

```
class User(db.Model):
```

```

__tablename__ = 'users'

id = db.Column(db.Integer, primary_key=True)

username = db.Column(db.String(64), unique=True, index=True)

def __repr__(self):

    return 'User:'.format(self.username)

if __name__ == '__main__':

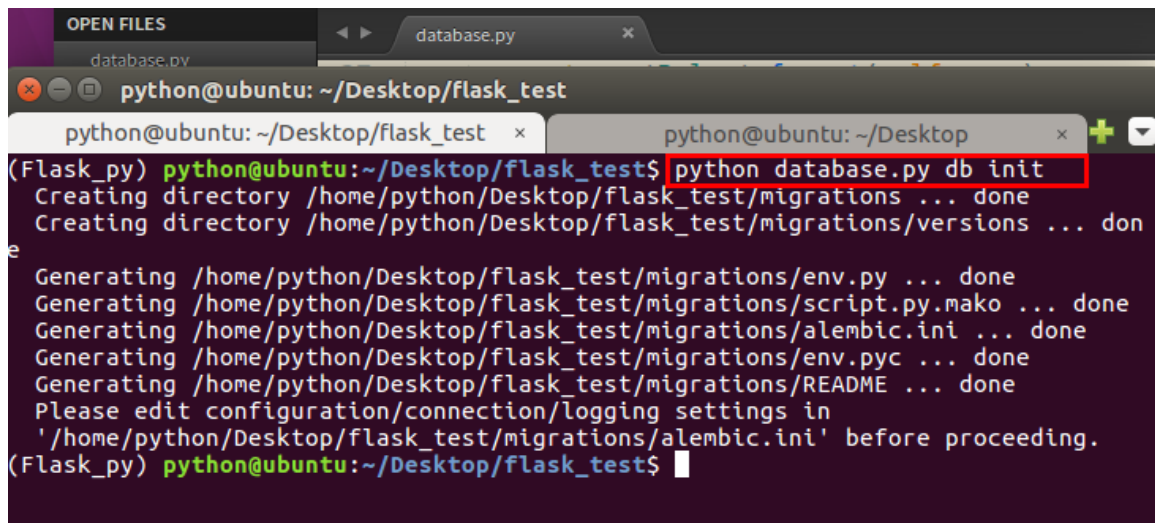
    manager.run()

```

创建迁移仓库：

#这个命令会创建migrations文件夹，所有迁移文件都放在里面。

```
python database.py db init
```



```

python@ubuntu: ~/Desktop/flask_test
python@ubuntu: ~/Desktop
(F Flask_py) python@ubuntu:~/Desktop/flask_test$ python database.py db init
Creating directory /home/python/Desktop/flask_test/migrations ... done
Creating directory /home/python/Desktop/flask_test/migrations/versions ... done
Generating /home/python/Desktop/flask_test/migrations/env.py ... done
Generating /home/python/Desktop/flask_test/migrations/script.py.mako ... done
Generating /home/python/Desktop/flask_test/migrations/alembic.ini ... done
Generating /home/python/Desktop/flask_test/migrations/env.pyc ... done
Generating /home/python/Desktop/flask_test/migrations/README ... done
Please edit configuration/connection/logging settings in
'/home/python/Desktop/flask_test/migrations/alembic.ini' before proceeding.
(F Flask_py) python@ubuntu:~/Desktop/flask_test$

```

创建迁移脚本：

自动创建迁移脚本有两个函数，**upgrade()**函数把迁移中的改动应用到数据库中。**downgrade()**函数则将改动删除。自动创建的迁移脚本会根据模型定义和数据库当前状态的差异，生成**upgrade()**和**downgrade()**函数的内容。对比不一定完全正确，有可能会遗漏一些细节，需要进行检查

#创建自动迁移脚本

```
python database.py db migrate -m 'initial migration'
```

```
(Flask_py) python@ubuntu:~/Desktop/Flask_test4$ python database.py db migrate -m  
'initial migration'  
INFO [alembic.runtime.migration] Context impl MySQLImpl.  
INFO [alembic.runtime.migration] Will assume non-transactional DDL.  
INFO [alembic.autogenerate.compare] Detected added table 'roles'  
INFO [alembic.autogenerate.compare] Detected added table 'users'  
Generating /home/python/Desktop/Flask_test4/migrations/versions/5feedcc4b6ec_i  
nitial_migration.py ... done  
(Flask_py) python@ubuntu:~/Desktop/Flask_test4$
```

更新数据库：

```
python database.py db upgrade
```

除了去更新数据库，有时我们可能需要回退到之前版本的数据库，那么如何回退数据库呢？

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 database.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。

```
python database.py db downgrade 版本号
```

```
(Flask_py) python@ubuntu:~/Desktop/Flask_test4$ python database.py db upgrade
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade 5feedcc4b6ec -> 102abfcc4377,
del user
(Flask_py) python@ubuntu:~/Desktop/Flask_test4$ python database.py db downgrade
5feedcc4b6ec
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running downgrade 102abfcc4377 -> 5feedcc4b6ec
, del user
(Flask_py) python@ubuntu:~/Desktop/Flask_test4$
```

邮箱扩展Flask- Mail

除了上述的迁移数据库外，这里简单的给大家普及一个知识点：

在开发过程中，很多应用程序都需要通过邮件提醒用户，Flask的扩展包Flask-Mail通过包装了Python内置的smtpplib包，可以用在Flask程序中发送邮件。

Flask-Mail 连接到简单邮件协议（Simple Mail Transfer Protocol,SMTP）服务器，并把邮件交给服务器发送。



如下示例，通过开启QQ邮箱SMTP服务设置，发送邮件。

```
from flask import Flask

from flask_mail import Mail, Message

app = Flask(__name__)

#配置邮件：服务器 / 端口 / 传输层安全协议 / 邮箱名 / 密码
app.config.update(
    DEBUG = True,

    MAIL_SERVER='smtp.qq.com',

    MAIL_PROT=465,

    MAIL_USE_TLS = True,

    MAIL_USERNAME = '371673381@qq.com',

    MAIL_PASSWORD = 'goyubxohbtzfbidd',
)

mail = Mail(app)

@app.route('/')

def index():

    # sender 发送方, recipients 接收方列表

    msg = Message("This is a test ",sender='371673381@qq.com',
recipients=['shengjun@itcast.cn','371673381@qq.com'])

    #邮件内容
```

```
msg.body = "Flask test mail"

#发送邮件

mail.send(msg)

print "Mail sent"

return "Sent Succeed"


if __name__ == "__main__":

    app.run()
```

扫描下方二维码，马上进入大神群！





精选留言

暂无...