Flask-Migrate数据库迁移

为什么使用Flask-Migrate?

在我们用 `sqlchemy` 模块创建完几个表时，如果在实际生产环境中，需要对表结构进行更改，应该怎么办呢？总不能把表删除了吧，这样数据就会丢失了。

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。

实现步骤

- 安装 flask-migrate 和 flask-script 模块。

```
pip3 install flask-migrate
pip3 install flask-script
```

- 配置代码

```
manager = Manager(app)
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，添加一个db命令
manager.add_command('db',MigrateCommand)
```

数据库迁移命令行操作

创建迁移仓库

这个命令会创建 `migirations` 文件夹，所有迁移文件都放在里面

```
python3 xxx.py db init
```

自动创建迁移脚本

数据库迁移工作由迁移脚本完成。这个脚本有两个函数，分别叫做upgrade()和downgrade()。upgrade()函数实施数据库更改，是迁移的一部分，downgrade()函数则删除它们。通过添加和删除数据库变化的能力，Alembic可以重新配置数据库从历史记录中的任何时间点。

```
python3 xxx.py db migrate -m "版本名(注释)"
```

更新数据库

一旦迁移脚本被审查且接受，就可以使用db upgrade命令更新到数据库中：

```
python3 xxx.py db upgrade
```

当我们需要修改表结构时，如何实现数据库迁移呢?

- 直接在 `xxx.py` 里直接增删相应的代码,
- 修改完成后，继续创建新的迁移脚本

```
python 文件 db migrate -m"新版本名(注释)"
```

- 更新数据库

```
python3 xxx.py db upgrade
```

更新完之后，其实就是提交操作，类似于 git 添加一个新的版本。但是，如果我们想返回历史的版本，应该怎么操作呢?

- 先查看版本号

  ```
  python xxx.py db history
  ```

- 想要返回的版本号。返回指定的版本

  ```
  python xxx.py db downgrade(upgrade) 版本号
  ```

  然后打开你的代码，可以发现他自动复原了!