**数据库迁移工具alembic介绍\_alembic使用：**

**1.alembic介绍：**

alembic是sqlalchemy的作者开发的，用来做OMR模型与数据库的迁移与映射，

alembic使用方式跟git有点了类似，

alembic的所有命令都是以alembic开头，

alembic的迁移文件也是通过版本进行控制的，

**2.alembic安装：**

pip install alembic

**3.alembic使用：**

**1.使用sqlalchemy创建好模型类：**

如创建一个models.py模块，然后在里面定义需要的模型类，代码如下：

|  |
| --- |
| **from** sqlalchemy **import** Column,String,Integer,create\_engine  **from** sqlalchemy.ext.declarative **import** declarative\_base  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'alembic\_demo'**  USERNAME = **'root'**  PASSWORD = **'root'**  DB\_URI =**"mysql+pymysql://{username}:{password}@{host}:{port}/{db}?charset=utf8"**.format(username=USERNAME,password=PASSWORD,host=HOSTNAME,port=PORT,db=DATABASE)  engine = create\_engine(DB\_URI)  Base = declarative\_base(engine)  **class** User(Base):  \_\_tablename\_\_ = **'user'**  id = Column(Integer,primary\_key=**True**,autoincrement=**True**)  uname = Column(String(50),nullable=**False**)  country = Column(String(50))  # ORM -> 迁移文件 -> 映射到数据库中  # import os  # print(os.path.dirname(\_\_file\_\_)) |

**2. 使用alembic创建一个仓库（初始化仓库）：**

**1.**打开dos系统界面

**2.**cd到当前项目目录中，注意：如果想要使用alembic，则需要先进入到安装了alembic的虚拟环境中，不然就找不到这个命令。

**3.**然后执行命令 “alembic init [仓库的名字，推荐使用alembic]”

**3. 修改配置文件：**

在`alembic.ini`中，给`sqlalchemy.url`项设置数据库的连接方式。方式跟sqlalchemy的方式是一样的。

sqlalchemy.url = driver://user:pass@localhost/dbname，

**给`sqlalchemy.url`项设置数据库的连接操作为：**

|  |
| --- |
| **sqlalchemy.url** = **mysql+pymysql://root:root@localhost/alembic\_demo?charset=utf8** |

为了使用模型类更新数据库，需要在`alembic/env.py`文件中设置target\_metadata项，默认为target\_metadata=None。

需要将`target\_metadata`的值设置为模型`Base.metadata`，但是要导入`models`

使用sys模块和os模块把当前项目的路径导入到path中：

**导入`models`的操作为：**

|  |
| --- |
| **import** sys,os  sys.path.append(os.path.dirname(os.path.dirname(\_\_file\_\_)))  **import** models |

**设置target\_metadata项操作为：**

|  |
| --- |
| target\_metadata = models.Base.metadata |

**4.自动生成迁移文件：**

使用alembic revision --autogenerate -m "提示信息"将当前模型中的状态生成迁移文件。

**5.将生成的迁移文件映射到数据库中：**

使用alembic upgrade head将刚刚生成的迁移文件，真正映射到数据库中。

同理，如果要降级，那么使用alembic downgrade head。

**6. 以后如果修改了模型，重复4、5步骤。**

**4.常用alembic命令和参数解释：**

**1.** init：创建一个alembic仓库。

**2.** revision：创建一个新的版本文件。

**3.** --autogenerate：自动将当前模型的修改，生成迁移脚本。

**4.** -m：本次迁移做了哪些修改，用户可以指定这个参数，方便回顾。

**5.** upgrade：将指定版本的迁移文件映射到数据库中，会执行版本文件中的upgrade函数。

如果有多个迁移脚本没有被映射到数据库中，那么会执行多个迁移脚本。

**6.** [head]：代表最新的迁移脚本的版本号。

**7.** downgrade：会执行指定版本的迁移文件中的downgrade函数。

**8.** heads：展示head指向的脚本文件版本号。

**9.** history：列出所有的迁移版本及其信息。

**10.** current：展示当前数据库中的版本号。

另外，在你第一次执行upgrade的时候，就会在数据库中创建一个名叫alembic\_version表，这个表只会有一条数据，记录当前数据库映射的是哪个版本的迁移文件。

**5.常见错误及解决办法：**

**1.** 创建新版本时报错 FAILED: Target database is not up to date.

**原因：**主要是heads和current不相同。current落后于heads的版本。

**解决办法：**将current移动到head上。alembic upgrade head

**2.** 创建新版本时报错 KeyError: 'bb747b02cda0' 或者 FAILED: Can't locate revision identified by 'a65ff5195bc0'

**原因：**数据库中存的版本号不在迁移脚本文件中

**解决办法：**删除versions中所有的迁移文件，删除数据库所有表。