**排序：**

**1.** order\_by方法排序：可以指定根据模型中某个属性进行排序，"模型名.属性名.desc()"代表的是降序排序。

**2.** 在定义模型的时候指定排序：有些时候，不想每次在查询的时候都用order\_by方法，可以在定义模型的时候就指定排序的方式。

有以下两种方式：

在模型定义中，添加以下代码：

\_\_mapper\_args\_\_ = {

# "order\_by": create\_time #正序

"order\_by": create\_time.desc() #倒序

}

relationship的方法中order\_by属性：在指定relationship方法的时候，添加order\_by属性来指定排序的字段。

# author = relationship("User", backref=backref("articles",order\_by=create\_time)) #正序

author = relationship("User", backref=backref("articles",order\_by=create\_time.desc())) #倒序

**方式一代码演示：**order\_by方法指定

|  |
| --- |
| **from** sqlalchemy **import** create\_engine,Column,Integer,Float,Boolean,DECIMAL,Enum,\  Date,DateTime,Time,String,Text,func,or\_,and\_,ForeignKey,Table  **from** sqlalchemy.dialects.mysql **import** LONGTEXT  **from** sqlalchemy.ext.declarative **import** declarative\_base  **from** sqlalchemy.orm **import** sessionmaker,relationship,backref  **import** random,time  **from** datetime **import** datetime  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'first\_sqlalchemy'**  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)  session = sessionmaker(engine)()  #排序方式1：order\_by方法指定  **class** Article(Base):  \_\_tablename\_\_ = **'article'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  title = Column(String(50), nullable=**False**)  create\_time = Column(DateTime, nullable=**False**, default=datetime.now)  **def** \_\_repr\_\_(self):  **return "<Article(title:%s,create\_time:%s)>"** % (self.title,self.create\_time)  **def** add\_data():  Base.metadata.drop\_all()  Base.metadata.create\_all()  article1 = Article(title=**'title1'**)  session.add(article1)  session.commit()  time.sleep(3)  article2 = Article(title=**'title2'**)  session.add(article2)  session.commit()  **def** oper():  # 正序排序  articles1 = session.query(Article).order\_by(Article.create\_time).all()  print(articles1)  # 倒序排序  articles2 = session.query(Article).order\_by(Article.create\_time.desc()).all()  print(articles2)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  # add\_data()  oper() |

**方式二代码演示：**定义模型时，指定排序方式

|  |
| --- |
| **from** sqlalchemy **import** create\_engine,Column,Integer,Float,Boolean,DECIMAL,Enum,\  Date,DateTime,Time,String,Text,func,or\_,and\_,ForeignKey,Table  **from** sqlalchemy.dialects.mysql **import** LONGTEXT  **from** sqlalchemy.ext.declarative **import** declarative\_base  **from** sqlalchemy.orm **import** sessionmaker,relationship,backref  **import** random,time  **from** datetime **import** datetime  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'first\_sqlalchemy'**  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)  session = sessionmaker(engine)()  #排序方式2：定义模型时，指定排序方式  **class** Article(Base):  \_\_tablename\_\_ = **'article'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  title = Column(String(50), nullable=**False**)  create\_time = Column(DateTime, nullable=**False**, default=datetime.now)  \_\_mapper\_args\_\_ = {  # "order\_by": create\_time #正序  **"order\_by"**: create\_time.desc() #倒序  }  **def** \_\_repr\_\_(self):  **return "<Article(title:%s,create\_time:%s)>"** % (self.title,self.create\_time)  **def** add\_data():  Base.metadata.drop\_all()  Base.metadata.create\_all()  article1 = Article(title=**'title1'**)  session.add(article1)  session.commit()  time.sleep(3)  article2 = Article(title=**'title2'**)  session.add(article2)  session.commit()  **def** oper():  # 不用再指定排序方式 因为在定义模型的时候 就已指定好排序方式  articles2 = session.query(Article).all()  print(articles2)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  # add\_data()  oper() |

**方式三代码演示：**涉及两表时，定义模型时，用relationship方法中的order\_by属性指定排序方式

|  |
| --- |
| **from** sqlalchemy **import** create\_engine,Column,Integer,Float,Boolean,DECIMAL,Enum,\  Date,DateTime,Time,String,Text,func,or\_,and\_,ForeignKey,Table  **from** sqlalchemy.dialects.mysql **import** LONGTEXT  **from** sqlalchemy.ext.declarative **import** declarative\_base  **from** sqlalchemy.orm **import** sessionmaker,relationship,backref  **import** random,time  **from** datetime **import** datetime  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'first\_sqlalchemy'**  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)  session = sessionmaker(engine)()  #排序方式3：涉及两表时，定义模型时，用relationship方法中的order\_by属性指定排序方式  **class** User(Base):  \_\_tablename\_\_ = **'user'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  uname = Column(String(50),nullable=**False**)  **class** Article(Base):  \_\_tablename\_\_ = **'article'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  title = Column(String(50), nullable=**False**)  create\_time = Column(DateTime, nullable=**False**, default=datetime.now)  uid = Column(Integer,ForeignKey(**"user.id"**))  # author = relationship("User", backref=backref("articles",order\_by=create\_time)) #正序  author = relationship(**"User"**, backref=backref(**"articles"**,order\_by=create\_time.desc())) #倒序  **def** \_\_repr\_\_(self):  **return "<Article(title:%s,create\_time:%s)>"** % (self.title,self.create\_time)  **def** add\_data():  Base.metadata.drop\_all()  Base.metadata.create\_all()  article1 = Article(title=**'title1'**)  user = User(uname=**'默默'**)  user.articles = [article1]  session.add(user)  session.commit()  time.sleep(3)  article2 = Article(title=**'title2'**)  user.articles.append(article2)  session.commit()  **def** oper():  # 不用再指定排序方式 因为在定义模型的时候 就已指定好排序方式  user = session.query(User).first()  print(user.articles)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  # add\_data()  oper() |