SQLAlchemy 介绍

刘鑫 <mars.liu@tratao.com>

Python 最好的数据库访问工具

November 27, 2013

什么是 SQLAlchemy ？

什么是 SQLAlchemy？

数据库访问工具

什么是 SQLAlchemy？

DBAPI++

什么是 SQLAlchemy？

全功能 ORM

什么是 SQLAlchemy？

广泛支持主流数据库

什么是 SQLAlchemy？

宽容的支持各种不同的设计

第一个例子

Engine 的概念

```
#!/usr/bin/env python

import sqlalchemy as sa

engine = sa.create_engine(
    "sqlite:///:memory:", echo=True)
print engine.table_names()
```

输出显示了操作内容

```
2013–11–26 14:44:08,123 INFO
  sqlalchemy.engine.base.Engine
  SELECT name FROM  (SELECT *
  FROM sqlite_master UNION ALL
  SELECT * FROM sqlite_temp_master)
  WHERE type='table' ORDER BY name
2013–11–26 14:44:08,123 INFO
  sqlalchemy.engine.base.Engine ()
```

ORM and MetaData

演示 sample2.py

引入

```python
import sqlalchemy as sa
from sqlalchemy.ext.declarative \
    import declarative_base
from sqlalchemy.orm import sessionmaker

import datetime
```

环境初始化

```
engine = sa.create_engine(
    "sqlite:///:memory:", echo=True)
Base = declarative_base()
```

模型定义

```
class Note(Base):
    __tablename__ = 'note'
    id = sa.Column(sa.Integer, primary_key=True)
    author = sa.Column(sa.String)
    content = sa.Column(sa.String)
    at = sa.Column(sa.TIMESTAMP,
        default=datetime.datetime.now)

    def __repr__(self):
        return (u"<Note(id=%s, author='...,"
            self.id, self.author, self.content,
            self.at)).encode("utf-8")
```

绑定然后建立架构

```
Base.metadata.create_all(engine)

Session = sessionmaker(bind=engine)
session = Session()
```

数据写入

```
log = Note(author="mars.liu@tratao.com",
    content=u"编写 SQLAlchemy 教程")
print log
session.add(log)
session.commit()
print log
```

查询

```
re = session.query(Note).filter_by( \
    author="mars.liu@tratao.com").all()
...
re = session.query(Note.author,
     Note.at).all()
...
log3 = session.query(Note).get(log3.id)
...
```

单对象更新

```
log3 = session.query(Note).get(log3.id)
log3=session.merge(log3)
session.commit()
```

单对象更新

```
log3 = session.query(Note).get(log3.id)
log3=session.merge(log3)
session.commit()
```

Update and Other

```
log3 = session.query(Note).get(log3.id)
log3.author = "mars.liu@dwarf-artisan.com"
log3.content += u"，对象更新"
session.commit()
```

SQL 拼装

```
ins = Note.__table__.insert().values(
    author='march.liu@gmail.com',
    content=u"SQL语句组合")
print ins
session.execute(ins)
re = session.query(Note).all()
print re
```

查询拼接

```
log4 = session.query(Note).order_by(\
    Note.id.desc()).first()

log4.content += u", 数据已修改成功"
session.commit()
re = session.query(Note.id,
    Note.content).all()
```

基本结构

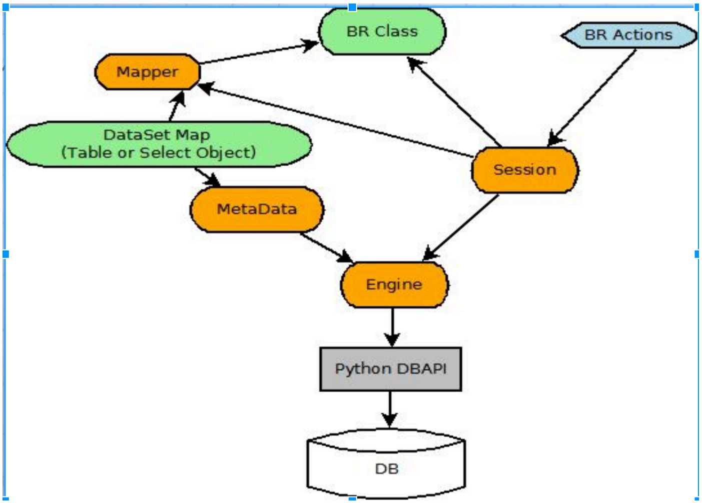

Figure: 基本结构

Engine

连接数据库和模型

Engine

URL 连接模式：

postgresql://user:***@server/database

Session

操作核心接口

Session

事务封装

Session

并发隔离

Session

提供多样化的数据访问操作

Session

平滑适配 ORM 到 SQL 操作的各种层级

MetaData 和 declarative_base

数据库模型描述的核心组件

Mapper

支持数据库模型和业务模型间的复杂关系

Web.py

```
from sqlalchemy.orm import scoped_session, sessionm
...
def load_sqla(handler):
    web.ctx.orm = scoped_session(sessionmaker(bind=
    try:
        return handler()
    except web.HTTPError:
       web.ctx.orm.commit()
       raise
    except:
        web.ctx.orm.rollback()
        raise
    finally:
        web.ctx.orm.commit()
        #web.ctx.orm.expunge_all()
```

tornado

```
from sqlalchemy.orm import scoped_session, sessionm
from models import *  # import the engine to bind

class Application(tornado.web.Application):
def __init__(self):
    ...
    # Have one global connection.
    self.db = scoped_session(\
      sessionmaker(bind=engine))
class BaseHandler(tornado.web.RequestHandler):
    @property
    def db(self):
        return self.application.db
    def get_current_user(self):
        user_id = self.get_secure_cookie("user")
        if not user_id: return None
        return self.db.query(User).get(user_id)
```

SQLAlchemy+Gevent

```
def make_psycopg_green():
    """Configure Psycopg to be used with gevent in
    if not hasattr(extensions, 'set_wait_callback')
        raise ImportError(
            "support for coroutines not available i
           % psycopg2.__version__)

    extensions.set_wait_callback(gevent_wait_callba
```

SQLAlchemy+Gevent

```
def gevent_wait_callback(conn, timeout=None):
    """A wait callback useful to allow gevent to wo
    while 1:
        state = conn.poll()
        if state == extensions.POLL_OK:
            break
        elif state == extensions.POLL_READ:
            wait_read(conn.fileno(), timeout=timeou
        elif state == extensions.POLL_WRITE:
            wait_write(conn.fileno(), timeout=timeo
        else:
            raise psycopg2.OperationalError(
                "Bad result from poll: %r" % state)
```

谢谢大家，再见！

Q&A

Power by LaTeX