DIPLOMADO DE PROGRAMACIÓN CON PYTHON V3

TEMA 6 TRANSACCIONES & ORM

UNIBE | EDUCACIÓN CONTINUA

Eliezer Figueroa
MCT,MOS,MCSA,MCSE,ITIL,SCF
3 Septiembre, 2019

COMPONENTES DEL TEMA

- ✓SQLITE & ACID
- ✓BEGIN TRANSACTION
- ✓ROLLBACK
- ✓SQLALCHEMY

SQLITE & ACID

- SQLite es una base de datos transaccional, todos los cambios y consultas son atómicas, coherentes, aisladas y duraderas (ACID).
- SQLite garantiza que todas las transacciones cumplen con ACID incluso si la transacción se interrumpe por un bloqueo del programa, volcado del sistema operativo o fallo de alimentación en el equipo.
- De forma predeterminada, SQLite funciona en modo de confirmación automática. Esto significa que para cada comando, SQLite inicia, procesa y confirma la transacción automáticamente.

BEGIN TRANSACTION

- Para iniciar una transacción, utilice explícitamente el comando BEGIN TRANSACTION.

```
BEGIN TRANSACTION;

UPDATE accounts
   SET balance = balance - 1000
 WHERE account_no = 100;

UPDATE accounts
   SET balance = balance + 1000
 WHERE account_no = 200;

INSERT INTO account_changes(account_no,flag,amount,changed_at)
VALUES(100,'-',1000,datetime('now'));

INSERT INTO account_changes(account_no,flag,amount,changed_at)
VALUES(200,'+',1000,datetime('now'));

COMMIT;
```

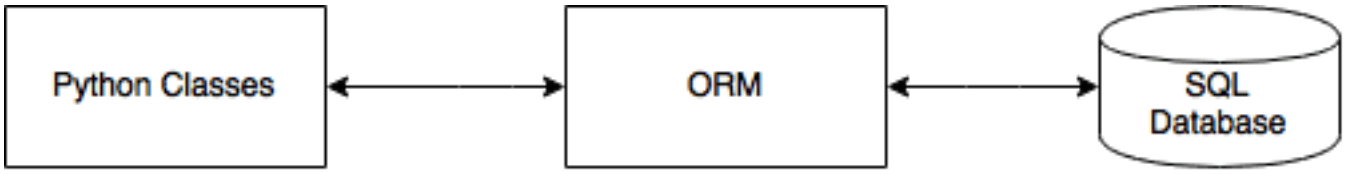
ROLLBACK

- Si no desea guardar los cambios, puede revertirlos mediante la instrucción ROLLBACK o ROLLBACK TRANSACTION:

```
import sqlite3

sql = sqlite3.connect("/tmp/test.db")
sql.isolation_level = None
c = sql.cursor()
c.execute("begin")
try:
    c.execute("update test set i = 1")
    c.execute("fnord")
    c.execute("update test set i = 0")
    c.execute("commit")
except sql.Error:
    print("failed!")
    c.execute("rollback")
```

SQLALCHEMY

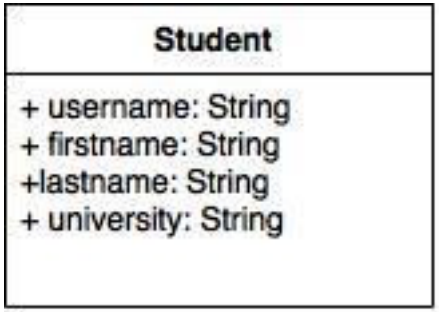
- SQLAlchemy es un kit de herramientas SQL para Python que proporciona a los desarrolladores de aplicaciones toda la potencia y flexibilidad de SQL.
- Proporciona un conjunto completo de patrones de persistencia de nivel empresarial conocidos, diseñados para un acceso a bases de datos eficiente y de alto rendimiento, adaptados a un lenguaje de dominio simple y Pythonic.
- SQLAlchemy es utilizado por organizaciones como: Yelp, Reddit , DropBox y The OpenStack Project.



SQLALCHEMY

```python
from sqlalchemy import *
from sqlalchemy import create_engine, ForeignKey
from sqlalchemy import Column, Date, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, backref

engine = create_engine('sqlite:///student.db', echo=True)
Base = declarative_base()

class Student(Base):

    __tablename__ = "student"

    id = Column(Integer, primary_key=True)
    username = Column(String)
    firstname = Column(String)
    lastname = Column(String)
    university = Column(String)

    def __init__(self, username, firstname, lastname, university):

        self.username = username
        self.firstname = firstname
        self.lastname = lastname
        self.university = university

# create tables
Base.metadata.create_all(engine)
```

- Creamos el archivo **tabledef.py.** En este archivo definiremos una clase **Student**.



SQLALCHEMY

```
import datetime
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from tabledef import *

engine = create_engine('sqlite:///student.db', echo=True)

# create a Session
Session = sessionmaker(bind=engine)
session = Session()

# Create objects
user = Student("james","James","Boogie","MIT")
session.add(user)

user = Student("lara","Lara","Miami","UU")
session.add(user)

user = Student("eric","Eric","York","Stanford")
session.add(user)

# commit the record the database
session.commit()
```

- Podemos insertar datos en la base de datos usando objetos Python.
- Debido a que usamos el SQLAlchemy ORM no tenemos que escribir una sola consulta SQL.

SQLALCHEMY

```
import datetime
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from tabledef import *

engine = create_engine('sqlite:///student.db', echo=True)

# create a Session
Session = sessionmaker(bind=engine)
session = Session()

# Create objects
for student in session.query(Student).order_by(Student.id):
    print (student.firstname, student.lastname)
```

- Podemos consultar todos los elementos de la tabla usando el código a continuación.

REFERENCIAS

- Sqlitetutorialnet. (2019). SQLite Transaction. Retrieved 24 September, 2019, from https://www.sqlitetutorial.net/sqlite-transaction/
- Pythonspotcom. (2019). ORM with SqlAlchemy. Retrieved 24 September, 2019, from https://pythonspot.com/orm-with-sqlalchemy/

MUCHAS GRACIAS

Eliezer Figueroa
MCT,MOS,MCSA,MCSE,ITIL,SCF

UNIBE EDUCACIÓN CONTINUA