href="https://github.com/taw-desarrollo-plataformas-web/EjemploSqlAlchemy-02">
src="https://www.sqlalchemy.org/img/sqla_logo.png" alt="Logo">

Ejemplo de uso de SqlAlchemy - One to Many

Es un repositorio académico que permite ejemplificar el proceso de creación y relación de entidades, ingreso y consulta de información a través de la SqlAlchemy.
href="https://www.sqlalchemy.org/">SqlAlchemy

Índice

Sobre el proyecto

Es un repositorio académico que permite ejemplificar el proceso de creación de entidades, ingreso y consulta de información a través de la SqlAlchemy.

En la carpeta llamada ejemplo01, se representa la relación entre dos clases -One to Many-

Donde:

- Un club tiene muchos jugadores asociados

Las entidades son:

```
class Club(Base):
    __tablename__ = 'club'
    id = Column(Integer, primary_key=True)
    nombre = Column(String(100))
    deporte = Column(String(100))
    fundacion = Column(Integer, nullable=False)
    # Mapea la relación entre las clases
    # Club puede acceder a los jugadores asociados
    # por la llave foránea
    jugadores = relationship("Jugador", back_populates="club")


    def __repr__(self):
        return "Club: nombre=%s deporte=%s fundación=%d" % (
                            self.nombre,
                            self.deporte,
                            self.fundacion)

class Jugador(Base):
    __tablename__ = 'jugador'
    id = Column(Integer, primary_key=True)
    nombre = Column(String(100), nullable=False)
    dorsal = Column(Integer)
    posicion = Column(String(100))
    # se agrega la columna club_id como ForeignKey
    # se hace referencia al id de la entidad club
    club_id = Column(Integer, ForeignKey('club.id'))
    # Mapea la relación entre las clases
    # Jugador tiene una relación con Club
    club  = relationship("Club", back_populates="jugadores")

    def __repr__(self):
        return "Jugador: %s - dorsal:%d - posición: %s" % (
                self.nombre, self.dorsal, self.posicion)
```

Reconocer como se relacionan las clases Club y Jugador.

Inicio del proyecto

Para poder usar el presente proyecto, tomar en consideración lo siguiente:

Prerrequisitos

- Instalar Python
- Instalar SQLAlchemy

```
    pip install SQLAlchemy
```

Usos

La carpeta ejemplo01 tiene los siguiente archivos
* configuracion.py

```
# este módulo será usado para posibles configuraciones
#
# cadena conector a la base de datos
#
# Sqlite
cadena_base_datos = 'sqlite:///base001.db'
```

- genera_tablas.py

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
from sqlalchemy import Column, Integer, String, ForeignKey

# se importa información del archivo configuracion
from configuracion import cadena_base_datos

# se genera en enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

Base = declarative_base()

# Ejemplo que representa la relación entre dos clases
# One to Many
# Un club tiene muchos jugadores asociados

class Club(Base):
    __tablename__ = 'club'
    id = Column(Integer, primary_key=True)
    nombre = Column(String(100))
    deporte = Column(String(100))
    fundacion = Column(Integer, nullable=False)
    # Mapea la relación entre las clases
    # Club puede acceder a los jugadores asociados
    # por la llave foránea
    jugadores = relationship("Jugador", back_populates="club")


    def __repr__(self):
        return "Club: nombre=%s deporte=%s fundación=%d" % (
                        self.nombre,
                        self.deporte,
                        self.fundacion)

class Jugador(Base):
    __tablename__ = 'jugador'
    id = Column(Integer, primary_key=True)
```

```
    nombre = Column(String(100), nullable=False)
    dorsal = Column(Integer)
    posicion = Column(String(100))
    # se agrega la columna club_id como ForeignKey
    # se hace referencia al id de la entidad club
    club_id = Column(Integer, ForeignKey('club.id'))
    # Mapea la relación entre las clases
    # Jugador tiene una relación con Club
    club  = relationship("Club", back_populates="jugadores")

    def __repr__(self):
        return "Jugador: %s - dorsal:%d - posición: %s" % (
                self.nombre, self.dorsal, self.posicion)

Base.metadata.create_all(engine)
```

- ingreso_datos.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Club, Jugador

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

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

# se crea un objetos de tipo Club
club1 = Club(nombre="Barcelona", deporte="Fútbol", \
        fundacion=1920)

club2 = Club(nombre="Emelec", deporte="Fútbol", \
        fundacion=1930)

club3 = Club(nombre="Liga de Quito", deporte="Fútbol", \
        fundacion=1940)

# Se crean objeto de tipo Jugador
#
jugador1 = Jugador(nombre ="Damian Diaz", dorsal=10, posicion="mediocampo", \
        club=club1)
jugador2 = Jugador(nombre ="Matias Oyola", dorsal=18, posicion="mediocampo", \
        club=club1)
jugador3 = Jugador(nombre ="Dario Aymar", dorsal=2, posicion="defensa", \
        club=club1)


jugador4 = Jugador(nombre ="Oscar Bagui", dorsal=6, posicion="defensa", \
        club=club2)
```

```
jugador5 = Jugador(nombre ="Romario Caicedo", dorsal=11, posicion="mediocampo", \
        club=club2)


jugador6 = Jugador(nombre ="Adrián Gabbarini", dorsal=1, posicion="arquero", \
        club=club3)
jugador7 = Jugador(nombre ="Cristian Martinez", dorsal=9, posicion="delantero", \
        club=club3)

# se agrega los objetos
# a la sesión
session.add(club1)
session.add(club2)
session.add(club3)
session.add(jugador1)
session.add(jugador2)
session.add(jugador3)
session.add(jugador4)
session.add(jugador5)
session.add(jugador6)

# se confirma las transacciones
session.commit()
```

- consulta_datos1.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Club, Jugador

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)

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

# Obtener todos los registros de
# la entidad Club
clubs = session.query(Club).all()

# Se recorre la lista a través de un ciclo
# repetitivo for en python
print("Presentación de Clubs")
for s in clubs:
    print("%s" % (s))
    print("---------")

# Obtener todos los registros de
# la entidad Jugador
```

```
jugadores = session.query(Jugador).all()

# Se recorre la lista a través de un ciclo
# repetitivo for en python

print("Jugadores")
for s in jugadores:
    print("%s" % (s))
    print("---------")
```

- consulta_datos2.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Club, Jugador

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)


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

# Obtener todos los registros de
# la entidad estudiantes (clase Estudiante)
jugadores = session.query(Jugador).all()

# Se recorre la lista a través de un ciclo
# repetitivo for en python
print("Presentación de Jugadores")
for s in jugadores:
    print("%s" % (s))
    # desde cada objeto de la lista
    # jugador
    # se puede acceder al club; como lo definimos
    # al momento de crear la clase Jugador
    print("El Jugador pertenece a: %s " % (s.club))
    print("---------")

print("Presentación de Jugadores - op2")
for s in jugadores:
    print("%s" % (s))
    # desde cada objeto de la lista
    # jugador
    # se puede acceder al club; como lo definimos
    # al momento de crear la clase Jugador
    print("El Jugador pertenece a: %s " % (s.club.nombre))
```

```
    print("---------")
```

- consulta_datos3.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Club, Jugador

# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)


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

# Obtener todos los registros de
# la entidad Club
clubs = session.query(Club).all()

# Se recorre la lista a través de un ciclo
# repetitivo for en python
print("Presentación de Clubs y sus jugadores")
for s in clubs:
    print("%s" % (s))
    # desde cada objeto de la lista
    # de tipo Club
    # se puede acceder
    # a los jugadores
    jugadores = s.jugadores # es una secuencia
    for i in jugadores:
        print(i)

    print("---------")
```

- consulta_datos4.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import and_ # se importa el operador and

# se importa la clase(s) del
# archivo genera_tablas
from genera_tablas import Club, Jugador
```

```
# se importa información del archivo configuracion
from configuracion import cadena_base_datos
# se genera enlace al gestor de base de
# datos
# para el ejemplo se usa la base de datos
# sqlite
engine = create_engine(cadena_base_datos)


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

# Obtener un listado de todos los registros
# de la tabla Club, que tengan al menos
# un jugador con el nombre que tenga incluida la cadena “Da”

# para la solución se hace uso del método
# join aplicado a query

clubs = session.query(Club).join(Jugador).\
        filter(Jugador.nombre.like("%Da%")).all()

print("Consulta 1 ")
for e in clubs:
    print(e)

# Obtener un listado de todos los registros
# de la tabla Club y Jugador, que tengan al menos
# un jugador con el nombre que tenga incluida la cadena “Da”

# para la solución se hace uso del método
# join aplicado a query
# en el query se ubican las dos entidades involucradas
#

registros = session.query(Club, Jugador).join(Jugador).\
        filter(Jugador.nombre.like("%Da%")).all()

print("Consulta 2 ")

for registro in registros:
    # el registro continen
    # dos valores en un tupla
    # posición 0 el club
    # posición 1 el jugador
    # que cumplen con la condición
    print(registro[0])
    print(registro[1])
```

El orden de ejecución de los archivos es el siguiente:

i. Generación de las entidades:

```
python genera_tablas.py
```

Este proceso debe generar un archivo llamado base001.db que contienes la base de datos en Sqlite. **Este**

archivo no está versionado; consta en el .gitignore

i. Ingreso de información a las entidades:

```
python ingreso_datos.py
```

i. Ejecutar consultas:

```
python consulta_datos1.py
python consulta_datos2.py
python consulta_datos3.py
python consulta_datos4.py
```

Licencia

Contacto

René Elizalde Solano - @reroes - rrelizalde@utpl.edu.ec