SQLAlchemy

Quellen

- riptutorial: LEARNING sqlalchemy
- SQLAlchemy

Installation

```
pip install sqlalchemy
```

SQLAlchemy-Core

1. Verbindung zur Datenbank

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///datenbank.db')
```

2. Tabelle erstellen

```
from sqlalchemy import Column, Integer, Text, MetaData, Table

metadata = MetaData()
schueler = Table(
    'schueler', metadata,
    Column('id', Integer, primary_key=True),
    Column('vorname', Text),
    Column('nachname', Text),
)
schueler.create(bind=engine)
```

3. Datensätze einfügen

```
insert_schueler = schueler.insert().values(vorname='Anna', nachname='Arm')
engine.execute(insert_schueler)
insert_schueler = schueler.insert().values(vorname='Berta', nachname='Bein')
engine.execute(insert_schueler)
```

4. Datensätze lesen

```
statement = select([schueler.c.vorname, schueler.c.nachname])
schuelerliste = engine.execute(statement).fetchall()
print(schuelerliste)
```

SQLAlchemy-ORM

1. Verbindung zur Datenbank

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///datenbank.db')
```

2. Objektorientiertes Modell: Basis

```
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
```

3.a) Klassen: single table

```
class Konto(Base):
    __tablename__ = "konto"
    id = Column(Integer, primary_key=True)
    kontostand = Column(Numeric(10, 2))
    type = Column(String(50))
    __mapper_args__ = {"polymorphic_identity": "konto", "polymorphic_on": type}

class Girokonto(Konto):
    dispo = Column(Numeric(10, 2))
    __mapper_args__ = {"polymorphic_identity": "girokonto"}

class Sparkonto(Konto):
    zinssatz = Column(Numeric(10, 1))
    __mapper_args__ = {"polymorphic_identity": "sparkonto"}
```

3.b) Klassen: joined

```
class Konto(Base):
    __tablename__ = "konto"
    id = Column(Integer, primary_key=True)
    kontostand = Column(Numeric(10, 2))
    type = Column(String(50))
    __mapper_args__ = {"polymorphic_identity": "konto","polymorphic_on": type}

class Girokonto(Konto):
    __tablename__ = "girokonto"
    id = Column(ForeignKey("konto.id"), primary_key=True)
    dispo = Column(Numeric(10, 2))
    __mapper_args__ = {"polymorphic_identity": "girokonto"}

class Sparkonto(Konto):
    __tablename__ = "sparkonto"
    id = Column(ForeignKey("konto.id"), primary_key=True)
    zinssatz = Column(Numeric(10, 1))
    __mapper_args__ = {"polymorphic_identity": "sparkonto"}
```

3.c) Klassen: table-per-class

```
class Konto(Base):
    __tablename__ = "konto"
    id = Column(Integer, primary_key=True)
    kontostand = Column(Numeric(10, 2))
    __mapper_args__ = {"polymorphic_identity": "konto"}

class Girokonto(Konto):
    __tablename__ = "girokonto"
    id = Column(Integer, primary_key=True)
    kontostand = Column(Numeric(10, 2))
    dispo = Column(Numeric(10, 2))
    __mapper_args__ = {"polymorphic_identity": "girokonto", "concrete": True}

class Sparkonto(Konto):
    __tablename__ = "sparkonto"
    id = Column(Integer, primary_key=True)
    kontostand = Column(Numeric(10, 2))
    zinssatz = Column(Numeric(10, 1))
    __mapper_args__ = {"polymorphic_identity": "sparkonto", "concrete": True}
```

4. Datenbank und Tabellen erstellen

```
Base.metadata.create_all(engine)
```

5. Python-Objekte erstellen

```
konto1 = Konto(kontostand=11.11)
konto2 = Girokonto(kontostand=22.22, dispo=50.00)
konto3 = Sparkonto(kontostand=33.33, zinssatz=1.5)
```

6. Python-Objekte in Datenbank speichern

```
from sqlalchemy.orm import Session

session = Session(engine)
session.add_all([konto1, konto2, konto3])
session.commit()
```