

Python średnio zaawansowany

Dzień 9





Blok nr 3:

Przetwarzanie danych



AGENDA

- Object Relational Mapping
- SQLAlchemy



Object Relational Mapping



ORM - definicja

Mapowanie obiektowo - relacyjne polega na **mapowaniu** danych w postaci **tabelarycznej** (relacji w bazie danych) na **obiekty** i odwrotnie.

To nowoczesne podejście do zagadnienia współpracy z bazą danych, wykorzystujące filozofię programowania obiektowego.



Wady SQL

- Zależność od silnika bazodanowego
- Podatny na ataki typu SQL injection
- Utrudnione aktualizowanie kodu podczas zmiany struktury tabel
- Boilerplate code
- Kod SQL wymieszany z kodem aplikacji

^{*} więcej nt boilerplate code: https://stormit.pl/boilerplate-code/



Zalety ORM

- Niezależność od silnika bazodanowego
- Większa produktywność programisty
- Migracje ułatwiają zmianę struktury tabel
- Możliwość wstrzyknięcia kodu SQL jeżeli ten generowany przez ORM jest zbyt skomplikowany/wolny



Wady ORM

- Skomplikowane zapytania SQL będą wymagać zapisania ich wprost, bez wykorzystania ORM
- Mechanizm ORM może być powolny



SQLAlchemy



SQLAlchemy



SQLAlchemy – moduł dla języka Python służący do pracy z bazami danych oraz mapowania obiektowo-relacyjnego.

Składa się z dwóch odrębnych komponentów: Core oraz ORM.

Wspiera takie bazy danych jak: Firebird, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SQLite.







Z SQLAlchemy korzystają m.in.: Fedora Project, Freshbooks, Fundacja Mozilla, OpenStack, reddit, SurveyMonkey oraz Yelp.















Dobry wstęp do SQLAlchemy:

https://docs.sglalchemy.org/en/latest/orm/tutorial.html



SQLAlchemy - analogia do SQL

```
Postać SQL

SELECT * FROM klienci; Klient.query.all()

SELECT * FROM klienci Klient.query.filter(age > 40)

WHERE age > 40;
```



SQLAlchemy - analogia do SQL

```
Postać SQL

SELECT COUNT(*) FROM

Klient.query.count()

klienci;
```

Więcej informacji nt filtrowania:

https://docs.sqlalchemy.org/en/latest/orm/tutorial.html#common-filter-operators



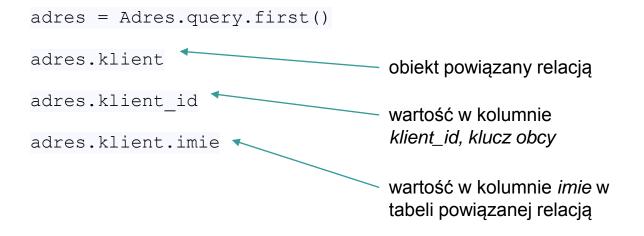
SQLAlchemy - przykład

```
class Adres (Base):
    tablename = 'adres'
  idx = Column(Integer, primary key=True)
  nazwa = Column(String(250))
   adres = Column(String(250))
                                           class Klient(Base):
   klient id = Column(
                                              tablename = 'klienci'
                                Klucz obcy
     Integer,
                                              idx = Column(Integer, primary key=True)
     ForeignKey('klienci.idx')
                                              imie = Column(String(250), nullable=False)
                                              nazwisko = Column(String(250), nullable=False)
   klient = relationship(Klient)
```

^{*} przykład na relację "wielu:1", bez relacji dwukierunkowej



SQLAlchemy - klucz obcy







Dzięki!