Session 5 - Object Relational Mapping

September 4, 2018

1 Python Training: Day 2 Session 5

1.1 2.5.1 Fundamental of object relational mapping

Let's save our time by reducing the complexity introduced by string based sql execution.

Object relational mapping is basically a technique to make two incompatible data types seems compatible through object oriented programming.

This session we use ponyorm as an example to glue Python and PostgreSQL.

https://ponyorm.com/

Make sure the library is already installed. If not, use the following command

```
pip install pony
```

```
In [ ]: from pony.orm import *
```

```
In [ ]: from datetime import datetime
        from uuid import uuid1
```

```
In [ ]: db = Database()
```

```
In [ ]: # This is optional, but we can use debug mode

        set_sql_debug(True)
```

```
In [ ]: # Now bind to the database

        db.bind(provider='postgres', user='postgres', password='postgres123', host='localhost',
```

```
In [ ]: class Cashlog(db.Entity):
            id = PrimaryKey(str, max_len=36)
            amount = Required(int)
            description = Optional(str)
            date = Required(datetime)
```

```
In [ ]: show(Cashlog)
```

```
In [ ]: # Now generate the table(s)

        db.generate_mapping(create_tables=True)
```

```
In [ ]: # We don't need this at the moment, but in case any table should be deleted (not sure if
        # here is an example on how to do it
        # db.drop_table('Cashlog', if_exists=True, with_all_data=False)
```

1.2 2.5.2 Inserting data

```
In [ ]: id1 = str(uuid1())
        id2 = str(uuid1())
        id3 = str(uuid1())
        id4 = str(uuid1())
        id5 = str(uuid1())

In [ ]: no1 = Cashlog(id=id1, amount=3000000, description='gaji', date=datetime.now())
        no2 = Cashlog(id=id2, amount=200000, description='bonus', date=datetime.now())
        no3 = Cashlog(id=id3, amount=-30000, description='bayar traktir teman', date=datetime.no
        no4 = Cashlog(id=id4, amount=-200000, description='bayar listrik dan air', date=datetime
        no5 = Cashlog(id=id5, amount=200000, description='honor', date=datetime.now())

        commit()

In [ ]: # use decorator db_session to interact with the database
        # actually @db_session is unecessary in command line interactive mode, but required in s

        @db_session
        def add_entry(amount, description, date):
            Cashlog(id=str(uuid1()), amount=amount, description=description, date=date)

In [ ]: add_entry(40000, 'hadiah uang kembalian', datetime.now())
```

1.3 2.5.3 Querying data

```
In [ ]: @db_session
        def query():
            return Cashlog.select()

In [ ]: query()

In [ ]: for entry in query():
            print(entry)
            print(' ', entry.amount, entry.description, entry.date)

In [ ]: Cashlog[id1].amount

In [ ]: Cashlog['vvv-www-xxx-yyy-zzz'].amount # well.. of course no entry with such id

In [ ]: u = Cashlog.get(id=id1)
        print(u)

In [ ]: u = Cashlog.get(id='vvv-www-xxx-yyy-zzz')
        print(u)
```

1.3.1 Assignment 2.5.1

Query the data to obtain

1. outflow and
2. inflow entry in the cashlog.

Outflow: money out, Inflow: money in

```
In [ ]: # Example
        from assignments.assignment_2_4_1 import outflow, inflow

In [ ]: inflow()

In [ ]: outflow()
```

1.4 2.5.4 Updating data

```
In [ ]: c = Cashlog.get(id=id1)
        c.amount = 3200000
        commit()

In [ ]: for entry in query():
            print(entry)
            print(' ', entry.amount, entry.description, entry.date)
```

1.5 2.5.5 Deleting data

```
In [ ]: c = Cashlog.get(description='honor')
        c.delete()
        commit()

In [ ]: for entry in query():
            print(entry)
            print(' ', entry.amount, entry.description, entry.date)
```