PyMySQL

PyMySQL

파이썬 MySQL 라이브러리

PyMySQL

- MySQL을 Python에서 사용할 수 있게 하는 라이브러리
- pip install pymysql로 설치

- requirements
  - MySQL >= 5.6
  - MaridDB >= 10.0

데이터베이스 연결

- pymysql.connect() 메서드를 통해 DB에 접속가능

- host : 연결할 데이터베이스 주소
- user : 접속할 유저이름
- password : 비밀번호
- database : 접소할 DB 이름
- charset : DB 문자열
  (한글 – utf8, 한글 + 이모티콘 – utf8mb4)

- DB 연결 해제 시, connection.close()

```
import pymysql

connection = pymysql.connect(host="DB_host",
                             port="3306",
                             user="user",
                             password="password",
                             database='db_name',
                             charset='utf8mb4')
```

```
connection.close()
```

INSERT

- 특정 테이블에 한 개의 데이터를 삽입

- connection.cursor()를 통해 cursor 객체를 가져옴

- 테이블, 칼럼명 입력시 ` (backtick) 활용

- execute 시 commit을 자동으로 해주지 않기 때문에 commit()을 호출하여 DB에 변경사항을 저장해야함

```
with connection.cursor() as cursor:
    sql = "INSERT INTO `student` (`email`, `password`) VALUES (%s, %s)"
    cursor.execute(sql, (`webmaster@python.org`, `secret`))

connection.commit()
```

INSERT MANY

- 많은 데이터를 한번에 삽입할 경우 사용

- executemany(sql, args) 메서드 호출
- args 부분에, 데이터들을 list로 묶어서 전달

- 마찬가지로, commit()으로 DB에 저장

- 한 개씩 데이터를 추가 할 때보다 속도가 빠르다.

```
with connection.cursor() as cursor:
    sql = "INSERT INTO `student` (`email`, `password`) VALUES (%s, %s)"
    cursor.executemany(sql, [(`sosin@python.org`, `secret-1`), (`jason@python.org`, `secret-2`)])

connection.commit()
```

SELECT

- 특정 데이터를 선택할 때 사용

- connection.cursor()를 통해 cursor 객체를 가져옴

- 테이블, 칼럼명 입력시 ` (backtick) 활용

- db에 commit 하는것이 아닌, cursor에서 fetch함

- fetchone()
- fetchmany(size=None)
- fetchall()

```
with connection.cursor() as cursor:
    sql = "SELECT `id`, `password` FROM `student` WHERE `email`=%s"
    cursor.execute(sql, ('webmaster@python.org', ))
    result = cursor.fetchone()

    print(result)
```

UPDATE

- 데이터를 수정할 때 사용

- execute()
- executemany()

```
with connection.cursor() as cursor:
    sql = "UPDATE `student` SET `email`=%s WHERE `id`=%s"
    cursor.execute(sql, ('pymysql@python.org', 1))

conn.commit()
```

DELETE

- 데이터를 삭제할 경우 사용

- execute()

```
with connection.cursor() as cursor:
    sql = "DELETE FROM `student` WHERE `id`=%s"
    cursor.execute(sql, (1, ))

connection.commit()
```

- 여러 데이터를 한번에 삭제할 경우

```
with connection.cursor() as cursor:
    sql = "DELETE FROM `student` WHERE `id` IN (%s, %s)"
    cursor.execute(sql, (2, 3))

connection.commit()
```

실습 (PyMySQL)

- db에 student 테이블을 생성해주세요.
  - id (빈 값 허용 안함, 자동 증가)
  - name (최대 16글자)
  - email (최대 32글자)
  - phone (최대 16글자)
  - major (최대 16글자)

- 수강생 중 한 분의 데이터를 DB에 입력해주세요.

- 수강생 중 네 분의 데이터를 DB에 동시에 추가해주세요.

- 입력한 전체 데이터를 확인해주세요.

- 한 수강생분의 이메일이 잘못 입력되었다고 가정하고, 이메일을 수정, DB에 반영해주세요.

- 수강생 한 분이 취업하셨습니다. DB에서 삭제 해주세요.

클래스화

- 생성 시 정보를 입력받아 DB에 연결하는 클래스를 생성해주세요

- 한 개의 데이터를 INSERT하는 insert 메서드를 생성해주세요.
  - INSERT 후 문제가 발생하지 않으면 commit하고, True를 반환해주세요.
  - 문제가 발생하면, 에러 내용을 출력하고, False를 반환해주세요.
  - parameter는 대상 table, 칼럼 배열이나 string, 값에 대한 tuple입니다.

- 여러 개의 데이터를 동시에 INSERT하는 insert_many 메서드를 생성해주세요.
  - 정상적으로 INSERT되었다면, commit 한 뒤, True를 반환해주세요.
  - 문제가 발생하면 에러 내용을 출력, False를 반환해주세요
  - parameter는 대상 테이블, 칼럼 배열이나 string, 값들에 대한 list입니다.

- 테이블 내의 데이터를 조회할 수 있는 select 메서드를 생성해주세요.
  - 파라미터는 대상 table, column (기본값 '*'), 조회할 row의 개수입니다.
  - LIMIT 대신, 메서드로 호출해주세요.
  - ORDER_BY를 인자로 받아 정렬 하셔도 좋습니다.

- update 메서드
  - 대상 table, set_column, set_value, where_column(기본 값 'id'),where_value

- delete 메서드
  - 대상 table, where_column (기본 값 'id'), where_value