파이썬

39강. MariaDB 연동

1. MariaDB 연동

- 파이썬을 이용하여 MaritaDB(MySQL)에 연결하기 위해서는 환경변수를 이용한다. 환경변수는 데이 터베이스가 설치된 호스트 주소, 데이터베이스 이름, 포트 번호, 그리고 접근권한을 갖는 사용자 계정 과 비밀번호 등을 말한다.

UNISTUDY

2. 환경변수와 DB 연동

- 설치된 환경을 바탕으로 MariaDB의 환경변수를 이용하여 데이터베이스에 연결하기 위해 서는 다음 형식과 같이 pymysql 패키지에서 제공하는 connect()함수의 가변인수로 환경변수를 지정한다.

```
config = {                    # 환경변수
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset':'utf8',
    'use_unicode' : True}

    conn = pymysql(**config)  # db 연결 객체
```

2. 환경변수와 DB 연동

- 생성한 데이터베이스, 테이블 그리고 사용자를 환경변수로 지정하여 MariaDB 의 work 데이터베이스에 연결하는 과정이다.

chapter10.lecture.step03_mariadb_test.py | Python Console

```
# (1) 패키지 import
import pymysql

# (2) db연동 환경변수
config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
```

2. 환경변수와 DB 연동

```
    'port' : 3306,
    'charset':'utf8',
    'use_unicode' : True}

try :
    # (3) db 연동 객체
    conn = pymysql.connect(**config)
    # (4) sql문 실행 객체
    cursor = conn.cursor()

    # (5) 테이블 조회
    sql = "show tables"
    cursor.execute(sql)
    tables = cursor.fetchall()

    # (6) 전체 table 목록 출력
    print(tables)                                   (('goods',) ,)

    # (7) table 유무
    if tables:
        print('table 있음')                         table 있음
    else:
        print('table 없음')

except Exception as e :
    print('db error :', e)

finally:
    cursor.close()
    conn.close()
```

2. 환경변수와 DB 연동

- 해설 MariaDB 연결하기 예
- # (1) 패키지 import
- MariaDB에 연결하기 위해서 pymysql 패키지를 import 한다.
- # (2) db연동 환경변수
- 데이터베이스에 연결하기 위해서 서버의 위치, 포트번호, 접속권한을 갖는 사용자, 유니코드 인코딩 방식 등을 지정하여 환경변수를 작성한다.
- # (3) db 연동 객체
- 환경변수를 가변인수로 하여 connect() 함수를 이용하여 데이터베이스에 연결할 객체를 생성한다.
- # (4) sql 실행 객체
- 데이터베이스 연동 객체에서 cursor() 메서드를 호출하여 SQL문을 실행할 수 있는 객체를 생성한다.

2. 환경변수와 DB 연동

- # (5) table 조회
- 현재 연결된 work 데이터베이스의 전체 테이블 목록을 검색할 수 있는 SQL문을 실행하고, 검색된 테이블 목록을 가져온다.
- # (6) 전체 table 목록 출력
- work 데이터베이스 안의 goods 테이블 이름이 출력된다.
- # (7) table 유무
- 1개 이상의 레코드가 검색되었기 때문에 'table 있음'이 출력된다.

3. MariaDB CRUD

- MariaDB에서 제공하는 CRUD 표준 SQL문의 형식을 이용하여 테이블을 조작하는 과정에 대해서 알 아본다. CRUD 표준 SQL문의 형식 SQLite3와 동일하다. 테이블 생성, 레코드 추가, 레코드 조회에 대한 예문을 먼저 살펴보자.

3. MariaDB CRUD

chapter10.lecture.step04_mariadb_crud.py

```
''' CRUD : Create, Read, Update, Delete '''

import pymysql

config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset':'utf8',
    'use_unicode' : True}
try :
    conn = pymysql.connect(**config)
    cursor = conn.cursor()

    # (1) table 생성
    sql = """create table goods(
                    code int primary key,
                    name varchar(30) not null,
                    su int default 0,
                    dan int default 0
```

3. MariaDB CRUD

```
                    )"""
    cursor.execute(sql)

    # (2) 레코드 추가..
    code = int(input('코드 입력 : '))
    name = input('상품명 입력 : ')
    su = int(input('수량 입력 : '))
    dan = int(input('단가 입력 : '))

    sql = "insert into goods values({code}, '{name}', {su}, {dan})"
    cursor.execute(sql)
    conn.commit() # db 반영..

    # (3) 전체 목록 보기..
    sql = "select * from goods"
    cursor.execute(sql) # sql문 실행
    rows = cursor.fetchall() # 전체 검색..

    # 레코드 출력 : 양식문자..
    for r in rows : # fetchone()
        #print(r) # tuple type 출력
        print('%d   %s    %d     %d'%r)

    print('검색 레코드 수 :', len(rows))

    # (4) 상품명 조회..
    name = input('\n조회할 상품명 입력 : ')
    sql = f"select * from goods where name like '%{name}%' "
    cursor.execute(sql) # sql문 실행..
    rows = cursor.fetchall()

    if rows :
        # 레코드 1개 출력 : index 이용..
        for r in rows :
            print(r[0], r[1], r[2], r[3])
    else :
        print('해당 상품 없음 ')

except Exception as e :
    print('db 연동 오류 : ', e)
    # (5) 이전 상태 리턴
    conn.rollback()
finally:
    cursor.close()
    conn.close()
```

UNISTUDY

3. MariaDB CRUD

- MariaDB CRUD 예1
- # (1) table 생성
- 테이블 생성 형식을 바탕으로 'goods' 테이블을 생성한다. 테이블은 상품코드, 상품명, 수량, 단가를 저장할 수 있는 4개의 칼럼으로 만들어진다. 상품코드(code)는 기본키로 지정되고, 상품명(name)은 중복불가와 생략 불가의 제약조건을 가지며, 수량과 단가는 default 제약조건에 의해서 생략하면기본값 0이 삽입된다.
- # (2) 레코드 삽입
- 4개의 칼럼 모두 키보드로 입력받아서 insert 명령문으로 레코드를 삽입한다.
- # (3) 레코드 조회
- CRUD의 Select문의 형식을 바탕으로 goods 테이블의 전체 레코드를 조회한다. 조회된 전체 레코드를 가져와서 for문으로 각 레코드를 색인을 이용하여 칼럼 단위로 출력한다.

UNISTUDY

3. MariaDB CRUD

- # (4) 상품명 조회
- 키보드로 상품명을 입력받아서 해당 상품의 레코드를 조회하는 조건 검색 과정이다. 예문의 sql문에서 like '%{name}%'" 명령문 형식은 키보드로 입력한 상품명을 포함하는 name 칼럼의 레코드를 검색하 기 위한 sql문 이다.
- # (5) 이전 상태 리턴
- 레코드 삽입 과정에서 문제가 발생하면 rollback() 함수에 의해서 db에 반영되기 전 상태로 되돌아간다.

3. MariaDB CRUD

- 다음은 키보드로 입력받은 값을 레코드로 삽입하는 SQL문과 상품코드(code)를 기준으로 수량과 단가 를 수정하는 SQL문, 상품코드(code)를 기준으로 레코드를 삭제하는 SQL문을 추가한 예문이다.
- 실습은 레코드 추가, 레코드 수정, 레코드 삭제를 순서대로 한 번씩만 실행한다. 예를 들면 키보드로 상 품코드, 상품명, 수량, 단가를 입력받아서 레코드를 추가하는 코드를 작성하고, 실행해서 레코드 조회 를 통해서 확인한다. 이때 레코드 수정과 레코드 삭제 코드는 작성하지 않는다. 레코드 수정을 위해서 는 레코드 추가 코드는 주석처리하고 레코드 삭제는 작성하지 않는다. 끝으로 레코드 삭제를 위해서는 레코드 추가와 레코드 수정 코드는 주석 처리한다.

3. MariaDB CRUD

chapter10.lecture.step04_mariadb_crud.py

```
import pymysql

config = {
    'host' : '127.0.0.1',
    'user' : 'scott',
    'password' : 'tiger',
    'database' : 'work',
    'port' : 3306,
    'charset':'utf8',
    'use_unicode' : True}
try :
```

3. MariaDB CRUD

```
conn = pymysql.connect(**config)
cursor = conn.cursor()

# (1) table 생성
'''
sql = """create table goods(
                code int primary key,
                name varchar(30) not null,
                su int default 0,
                dan int default 0
                )"""

cursor.execute(sql)
'''


# (2) 레코드 추가
'''
code = int(input('코드 입력 : '))
name = input('상품명 입력 : ')
su = int(input('수량 입력 : '))
dan = int(input('단가 입력 : '))

sql = "insert into goods values({code}, '{name}', {su}, {dan})"
cursor.execute(sql)
conn.commit() # db 반영
'''
```

3. MariaDB CRUD

```
    # (5) 레코드 수정
    # code 이용 -> 상품명, 수량, 단가 수정
    '''
    code = int(input('수정할 코드 입력 : '))
    name = input('수정할 상품명 입력 : ')
    su = int(input('수정할 수량 입력 :'))
    dan = int(input('수정할 단가 입력 :'))

    sql = f"update goods set name='{name}', su={su}, dan = {dan} where co
de={code}"
    cursor.execute(sql)
    conn.commit()
    '''

    # (6) 레코드 삭제
    code = int(input('삭제할 코드 입력 : '))
    sql = f"select * from goods where code = {code}"
    cursor.execute(sql) # sql문 실행
    rows = cursor.fetchall()

    if rows :
```

3. MariaDB CRUD

```
    # 레코드 1개 출력 : index 이용
    print('레코드 삭제')
    '''
    sql = f"delete from goods where code = {code}"
    cursor.execute(sql) # sql문 실행
    conn.commit()
    '''
else :
    print('해당 code 없음 ')

# (3) 전체 목록 보기
sql = "select * from goods"
cursor.execute(sql) # sql문 실행
rows = cursor.fetchall() # 전체 검색
#print(type(dataset)) # <class 'tuple'>

# 레코드 출력 : 양식문자
for r in rows : # fetchone()
    #print(r) # tuple type 출력
    print('%d   %s    %d     %d'%r)

print('검색 레코드 수 :', len(rows))
```

3. MariaDB CRUD

```
    # (4) 상품명 조회
    name = input('\n조회할 상품명 입력 : ')
    sql = f"select * from goods where name like '%{name}%' "
    cursor.execute(sql) # sql문 실행
    rows = cursor.fetchall()

    if rows :
        # 레코드 1개 출력 : index 이용
        for r in rows :
            print(r[0], r[1], r[2], r[3])
    else :
        print('해당 상품 없음 ')

except Exception as e :
    print('db 연동 오류 : ', e)
    conn.rollback()
finally:
    cursor.close()
    conn.close()
```

UNISTUDY

3. MariaDB CRUD

- 해설 MariaDB CRUD 예2

- # (2) 레코드 추가
- 4개의 칼럼에 해당하는 값을 키보드로 입력 받아서 values()함수 안에서 {변수} 형식으로 insert문 을 작성한다. 자료형이 문자형인 경우에는 '{변수}' 형식으로 지정한다.
- # (5) 레코드 수정 : code -> su, dan 수정
- 키보드로 수정할 상품코드, 수량, 단가를 입력 받아서 update문으로 상품코드의 레코드를 찾아서 해당 레코드의 수량과 단가를 수정한다.
- # (6) 레코드 삭제 : code -> 삭제
- 키보드로 상품코드를 입력받아서 delete문으로 해당 상품코드의 레코드를 삭제한다.

UNISTUDY

THANK YOU