今日学习目标

知道数据库操作的应用场景

掌握PyMySQL对数据库实现增、删、改、查

数据库工具类封装

==数据库操作应用场景==

- **校验 测试数据**
  - 接口发送请求后明确会对数据库中的某个字段进行修改，但，响应结果中无该字段数据时。
    - 如：ihrm 删除员工接口。 is_delete 字段，没有 在响应结果中出现！ 需要 借助数据库 校验！
- **构造 测试数据**
  - 测试数据使用一次就失效。
    - 如：ihrm 添加员工接口，使用的手机号！
  - 测试前，无法保证测试数据是否存在。
    - 如：ihrm 查询员工接口，使用的 员工id

PyMySQL操作数据库

安装PyMySQL

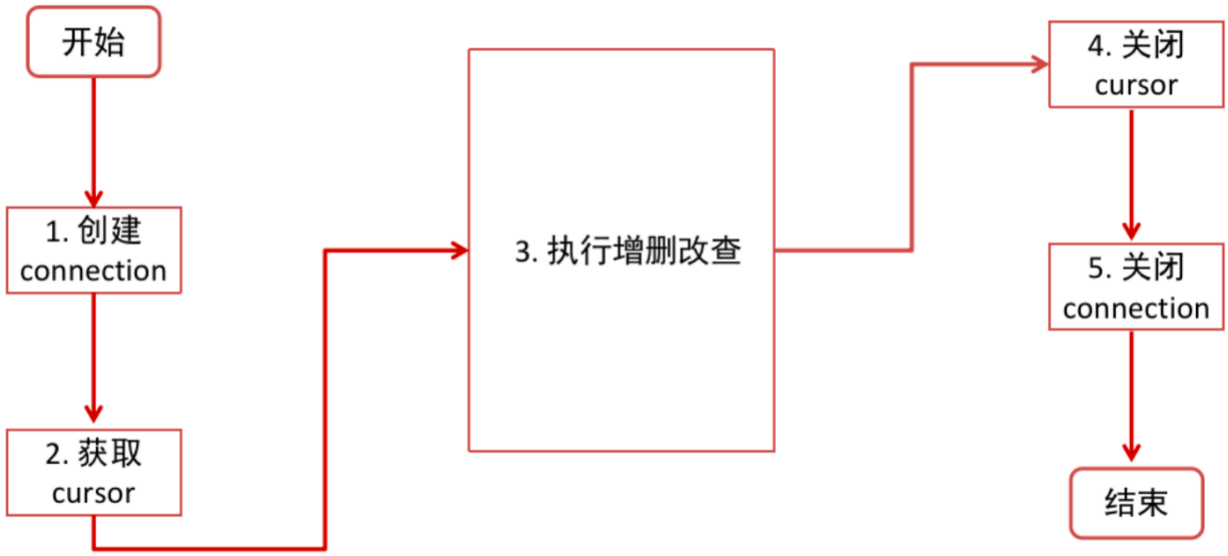
- 方法1：

```
pip install PyMySQL
```

- 方法2：

```
pip install PyMySQL -i https://pypi.douban.com/simple/
```

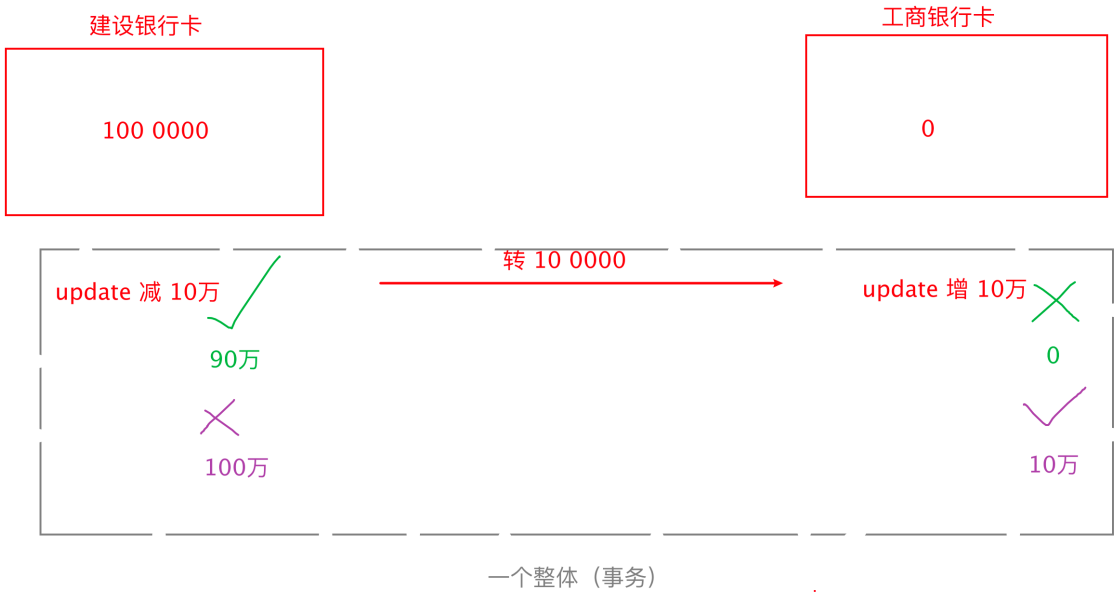
==操作步骤==



1. **导包 import pymysql**
2. **创建连接。 conn = pymysql.connect(host，port, user, password, database, charset)**
3. **获取游标。 cursor = conn.cursor()**
4. **执行 SQL。 cursor.execute("sql语句")**
    - **查询语句（select）**
        - **处理结果集（提取数据 fetch*）**
    - **增删改语句（insert、update、delete）**
        - **成功：提交事务 conn.commit()**
        - **失败：回滚事务 conn.rollback()**
5. **关闭游标。cursor.close()**
6. **关闭连接。conn.close()**

==事务的概念==

- 事务，是关系型数据库（mysql）特有的概念。
- 事务，可以看做一个虚拟的 容器，在容器中存放一系列的数据库操作，看做一个整体。内部的所有操作，要么都一次性全部成功，只要有一个失败，就全部失败！



- **事务操作：只有 2 种情况**
  - 提交：conn.commit()
  - 回滚：conn.rollback()

PyMySQL连接数据库

==建立连接方法==

```
conn = pymysql.connect(host="", port=0,
                       user="", password="", database="", charset="")

host：数据库所在主机 IP地址 - string
port：数据库使用的 端口号 - int
user：连接数据库使用的 用户名 - string
password：连接数据库使用的 密码 - string
database：要连接的那个数据库的名字 - string
charset：字符集。常用 utf8 - string

conn：连接数据库的对象。
```

入门案例

> 查询数据库，获取MySQL服务器 版本信息

```
# 1. 导包
import pymysql

# 2. 建立连接

conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
```

```
                        password="iHRM_student_2021", database="test_db", charset="utf8")

# 3. 获取游标
cursor = conn.cursor()

# 4. 执行 sql 语句（查询）
cursor.execute("select version()")

# 5. 获取结果
res = cursor.fetchone()
print("res =", res[0])

# 6. 关闭游标
cursor.close()

# 7. 关闭连接
conn.close()
```

PyMySQL操作数据库

==SQL 语法回顾==

- 查询语法：

```
select 字段1, 字段2, ... from 表 where 条件;

示例：select id, title, pub_date from t_book where title = '读者';
```

- 添加语法：

```
insert into 表名 (字段1, 字段2, ...) values (值1, 值2, ...) ;

示例：insert into t_book(id, title, pub_date) values(17, '红楼梦', '2021-11-11');
```

- 更新语法：

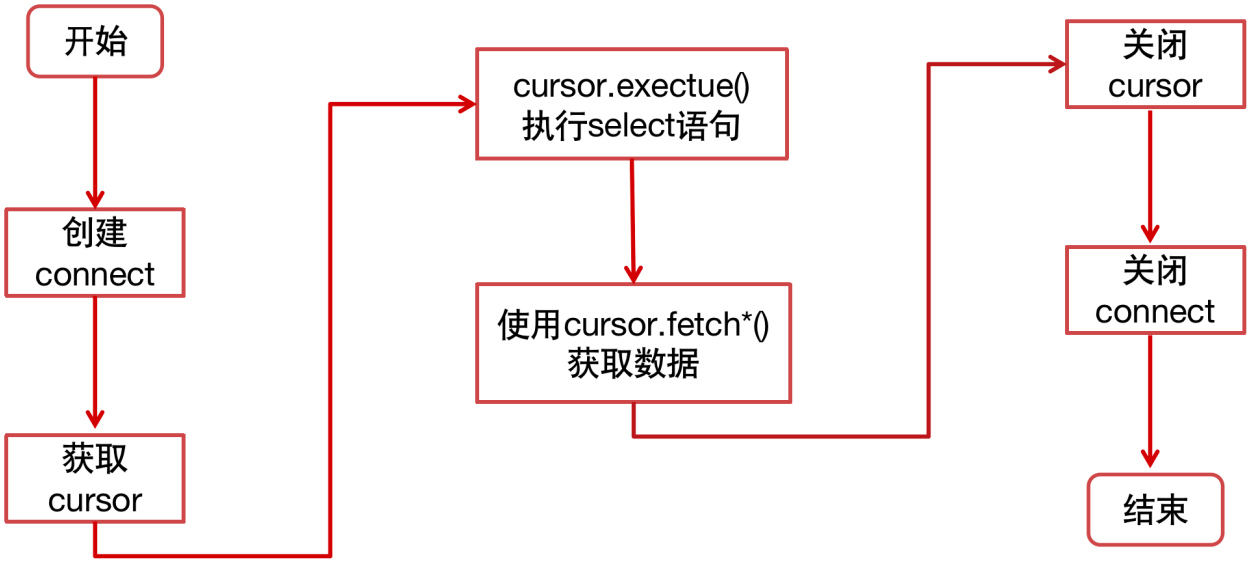
```
update 表名 set 字段名 = 字段值 where 条件

示例：update t_book set title = '三国' where id = 17;
```

- 删除语法：

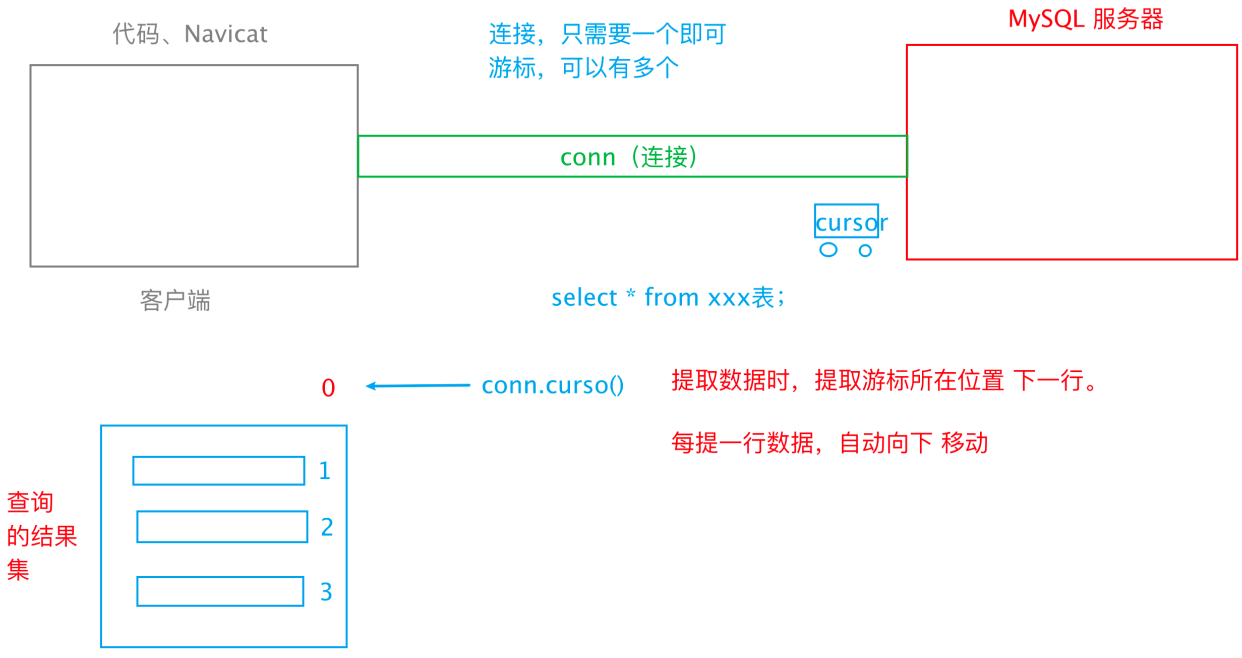
```
delete from 表名 where 条件

示例：delete from t_book where title = '三国';
```

==数据库查询==

查询操作流程



cursor游标



==常用方法==

- fetchone()：从结果集中，提取一行。
- fetchmany(size)：从结果集中，提取 size 行。
- fetchall()：提取所有结果集。
- 属性rownumber：可以设置游标位置。

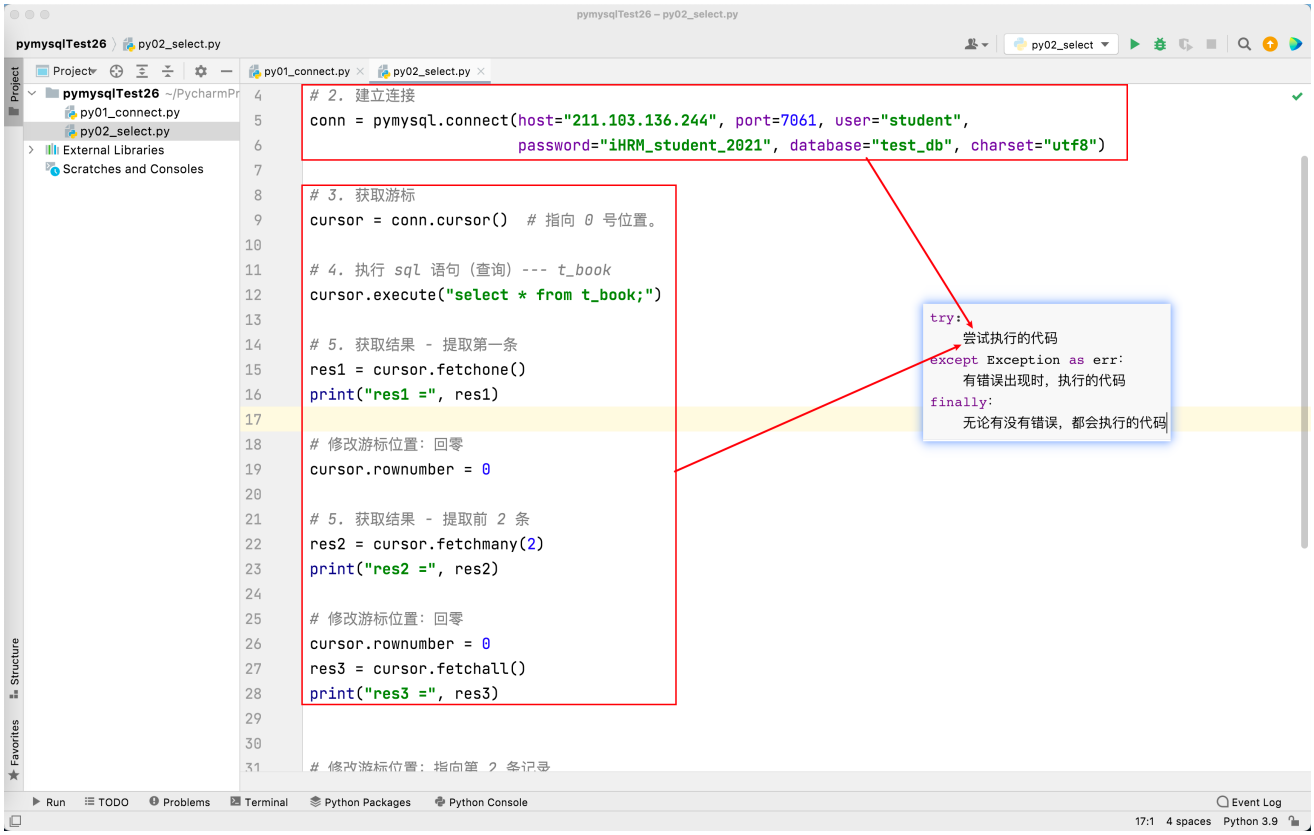
案例

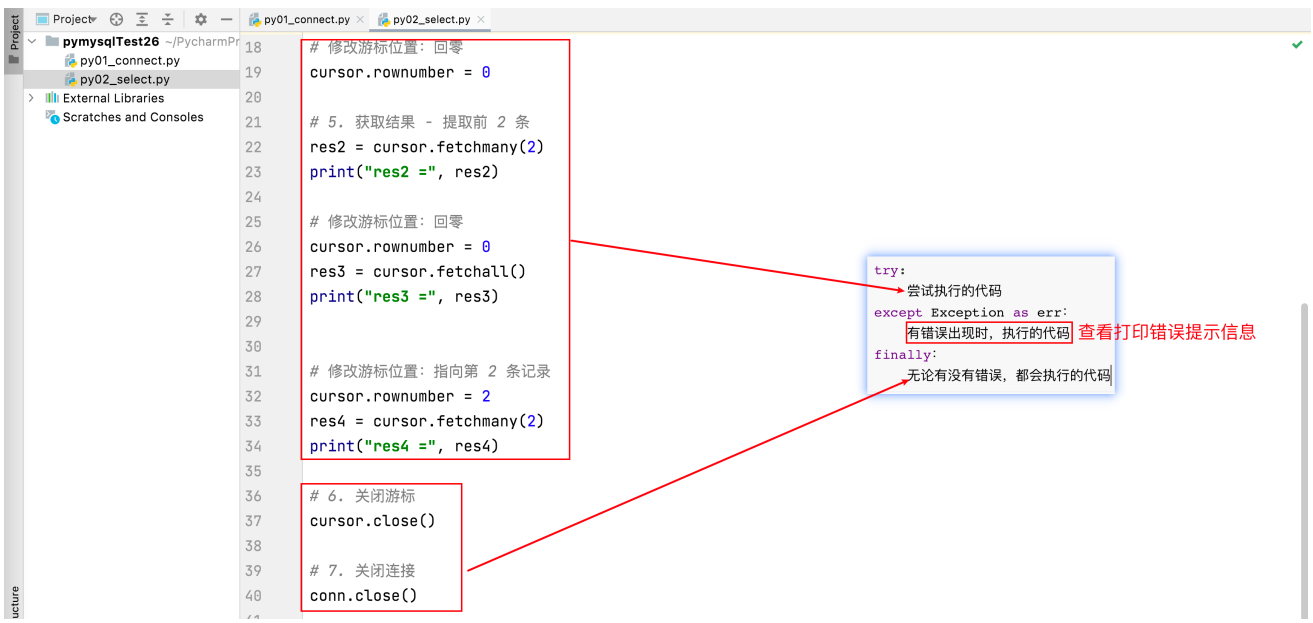
> 查询t_book表，获取 第一条 数据
>
> 查询t_book表，获取 前两条 数据
>
> 查询t_book表，获取 全部 数据
>
> 查询t_book表，获取 第3条和第4条 数据

```
# 1. 导包
import pymysql

# 2. 建立连接
conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
                       password="iHRM_student_2021", database="test_db", charset="utf8")

# 3. 获取游标
cursor = conn.cursor()  # 指向 0 号位置。

# 4. 执行 sql 语句（查询） --- t_book
cursor.execute("select * from t_book;")

# 5. 获取结果 - 提取第一条
res1 = cursor.fetchone()
print("res1 =", res1)

# 修改游标位置：回零
cursor.rownumber = 0
# 5. 获取结果 - 提取前 2 条
res2 = cursor.fetchmany(2)
print("res2 =", res2)

# 修改游标位置：回零
cursor.rownumber = 0
res3 = cursor.fetchall()
print("res3 =", res3)

# 修改游标位置：指向第 2 条记录
cursor.rownumber = 2
res4 = cursor.fetchmany(2)
print("res4 =", res4)

# 6. 关闭游标
cursor.close()

# 7. 关闭连接
conn.close()
```

==异常捕获==

```
try:
    尝试执行的代码
except Exception as err:
    有错误出现时，执行的代码
finally:
    无论有没有错误，都会执行的代码
```



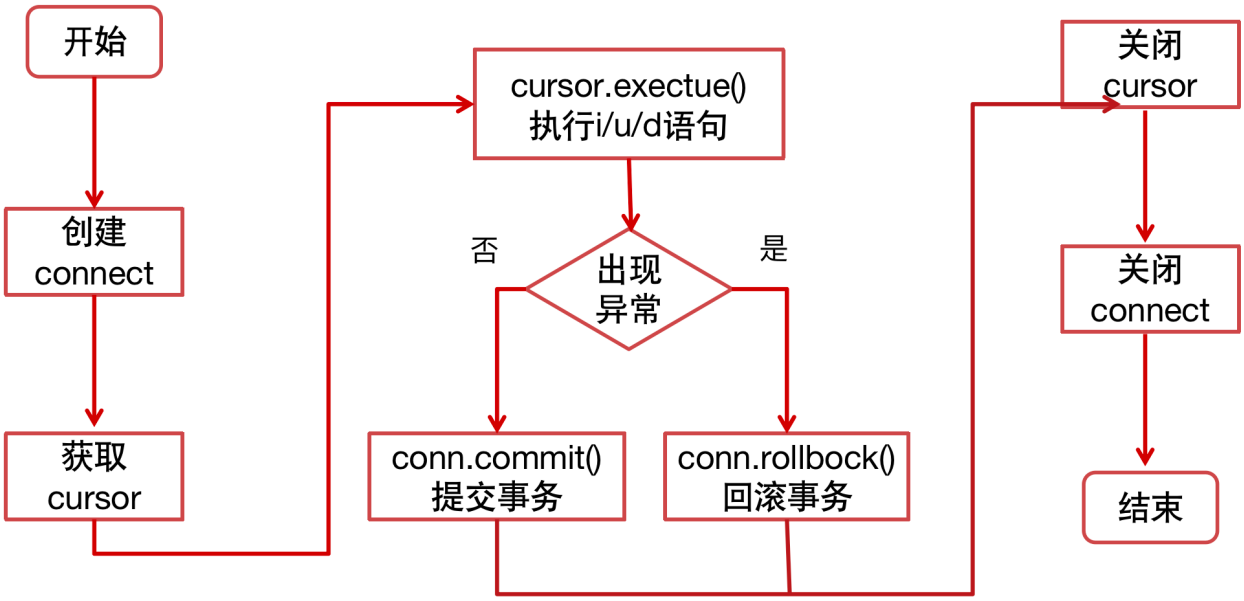



```
# 1. 导包
import pymysql
```

```python
# 定义全局变量，初值为 None
conn = None
cursor = None

try:
    # 2. 建立连接
    conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
                           password="iHRM_student_2021", database="test_db", charset="utf8")

    # 3. 获取游标
    cursor = conn.cursor()  # 指向 0 号位置。

    # 4. 执行 sql 语句（查询） --- t_book
    cursor.execute("select * from t_book;")

    # 5. 获取结果 - 提取第一条
    res1 = cursor.fetchone()
    print("res1 =", res1)

    # 修改游标位置：回零
    cursor.rownumber = 0

    # 5. 获取结果 - 提取前 2 条
    res2 = cursor.fetchmany(2)
    print("res2 =", res2)

    # 修改游标位置：回零
    cursor.rownumber = 0
    res3 = cursor.fetchall()
    print("res3 =", res3)

    # 修改游标位置：指向第 2 条记录
    cursor.rownumber = 2
    res4 = cursor.fetchmany(2)
    print("res4 =", res4)

except Exception as err:
    print("查询语句执行出错:", str(err))

finally:
    # 6. 关闭游标
    cursor.close()

    # 7. 关闭连接
    conn.close()
```

==数据库UID==

更新操作流程



案例

> 单独实现如下操作： ①：新增一条图书数据（id:5 title:西游记 pub_date:1986-01-01 ） ②：把图书名称为'西游记'的阅读量加一 ③：删除名称为'西游记'的图书

插入数据：

```
"""
新增一条图书数据（id:5 title:西游记 pub_date:1986-01-01 ）
insert into t_book(id, title, pub_date) values(5, '西游记', '1986-01-01');
1. 导包
2. 创建连接
3. 获取游标
4. 执行 insert 语句
5. 提交/回滚事务
6. 关闭游标
7. 关闭连接
"""

# 1. 导包
import pymysql

# 定义全局变量
conn = None
cursor = None

try:
    # 2. 创建连接
    conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
password="iHRM_student_2021",
                           database="test_db", charset="utf8")
    # 3. 获取游标
    cursor = conn.cursor()


    # 4. 执行 insert 语句
```

```
    cursor.execute("insert into t_book(id, title, pub_date) values(175, '西游记', '1986-01-
01');")

    # 查看 sql执行，影响多少行
    print("影响的行数：", conn.affected_rows())

    # 5. 提交事务
    conn.commit()

except Exception as err:
    print("插入数据错误：", str(err))
    # 回滚事务
    conn.rollback()

finally:
    # 6. 关闭游标
    cursor.close()
    # 7. 关闭连接
    conn.close()
```

修改数据：

```
"""
把图书名称为'西游记'的阅读量加一
update t_book set `read` = `read` + 1 where id = 6;

1. 导包
2. 建立连接
3. 获取游标
4. 执行 update语句
5. 提交、回滚事务
6. 关闭游标
7. 关闭连接
"""
# 1. 导包
import pymysql

conn = None
cursor = None

try:
    # 2. 建立连接
    conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
password="iHRM_student_2021",
                           database="test_db", charset="utf8")
    # 3. 获取游标
    cursor = conn.cursor()

    # 4. 执行 update语句。字段名，需要使用 反引号(`)包裹
    cursor.execute("update t_book set `read` = `read` + 1 where id = 1023;")
```

```
    print("影响的行数：", conn.affected_rows())

    # 5. 提交、回滚事务
    conn.commit()

except Exception as err:
    print("更新失败：", str(err))
    # 回滚事务
    conn.rollback()

finally:
    # 6. 关闭游标
    cursor.close()
    # 7. 关闭连接
    conn.close()
```

删除数据：

```
"""
删除名称为‘西游记’的图书
delete from t_book where title = '西游记';

1. 导包
2. 建立连接
3. 获取游标
4. 执行 delete 语句
5. 提交、回滚事务
6. 关闭游标
7. 关闭连接
"""
# 1. 导包
import pymysql

conn = None
cursor = None

try:
    # 2. 建立连接
    conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
password="iHRM_student_2021",
                           database="test_db", charset="utf8")
    # 3. 获取游标
    cursor = conn.cursor()

    # 4. 执行 delete语句。
    cursor.execute("delete from t_book where id = 151;")

    print("影响的行数：", conn.affected_rows())

    # 5. 提交、回滚事务

    conn.commit()
```

```
except Exception as err:
    print("更新失败：", str(err))
    # 回滚事务
    conn.rollback()

finally:
    # 6. 关闭游标
    cursor.close()
    # 7. 关闭连接
    conn.close()
```

数据库工具类封装

封装的目的

- 将 常用的数据库操作，封装到 一个方法。 后续再操作数据库时，通过调用该方法来实现。
- 提高代码的 复用性！

==设计数据库工具类==

```
# 封装数据库工具类

class DBUtil(object):
    @classmethod
    def __get_conn(cls):
        pass

    @classmethod
    def __close_conn(cls):
        pass

    # 常用方法：查询一条
    @classmethod
    def select_one(cls, sql):
        pass

    # 常用方法：增删改
    @classmethod
    def uid_db(cls, sql):
        pass
```

==实现类方法==

获取、关闭连接

```
# 封装数据库工具类
class DBUtil(object):
    # 添加类属性
    conn = None

    @classmethod
    def __get_conn(cls):
        # 判断 conn 是否为空， 如果是，再创建
        if cls.conn is None:
            cls.conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
                                       password="iHRM_student_2021", database="test_db",
charset="utf8")
        # 返回 非空连接
        return cls.conn

    @classmethod
    def __close_conn(cls):
        # 判断，conn 不为空，需要关闭。
        if cls.conn is not None:
            cls.conn.close()
            cls.conn = None
```

查询一条记录

```
# 封装数据库工具类
class DBUtil(object):
    # 常用方法：查询一条
    @classmethod
    def select_one(cls, sql):
        cursor = None
        res = None
        try:
            # 获取连接
            cls.conn = cls.__get_conn()

            # 获取游标
            cursor = cls.conn.cursor()

            # 执行 查询语句
            cursor.execute(sql)

            # 提取一条结果
            res = cursor.fetchone()

        except Exception as err:
            print("查询sql错误：", str(err))
        finally:
            # 关闭游标

            cursor.close()
```

```
            # 关闭连接
            cls.__close_conn()

            # 将查询sql执行的 结果，返回
            return res

if __name__ == '__main__':
    res = DBUtil.select_one("select * from t_book;")
    print("查询结果为：", res)
```

增删改数据

```
# 封装数据库工具类
class DBUtil(object):
    # 常用方法：增删改
    @classmethod
    def uid_db(cls, sql):
        cursor = None
        try:
            # 获取连接
            cls.conn = cls.__get_conn()

            # 获取游标
            cursor = cls.conn.cursor()

            # 执行 uid 语句
            cursor.execute(sql)
            print("影响的行数：", cls.conn.affected_rows())

            # 提交事务
            cls.conn.commit()

        except Exception as err:
            # 回滚事务
            cls.conn.rollback()
            print("增删改 SQL 执行失败：", str(err))

        finally:
            # 关闭游标
            cursor.close()
            # 关闭连接
            cls.__close_conn()

if __name__ == '__main__':
    DBUtil.uid_db("update t_book set is_delete = 1 where id = 1111;")
```

完整封装代码实现

```
import pymysql


# 封装数据库工具类
class DBUtil(object):
    # 添加类属性
    conn = None

    @classmethod
    def __get_conn(cls):
        # 判断 conn 是否为空，如果是，再创建
        if cls.conn is None:
            cls.conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
                                       password="iHRM_student_2021", database="test_db",
charset="utf8")
        # 返回 非空连接
        return cls.conn

    @classmethod
    def __close_conn(cls):
        # 判断，conn 不为空，需要关闭。
        if cls.conn is not None:
            cls.conn.close()
            cls.conn = None

    # 常用方法：查询一条
    @classmethod
    def select_one(cls, sql):
        cursor = None
        res = None
        try:
            # 获取连接
            cls.conn = cls.__get_conn()

            # 获取游标
            cursor = cls.conn.cursor()

            # 执行 查询语句
            cursor.execute(sql)

            # 提取一条结果
            res = cursor.fetchone()

        except Exception as err:
            print("查询sql错误：", str(err))
        finally:
            # 关闭游标
            cursor.close()

            # 关闭连接
            cls.__close_conn()

            # 将查询sql执行的 结果，返回
```

```
            return res

    # 常用方法：增删改
    @classmethod
    def uid_db(cls, sql):
        cursor = None
        try:
            # 获取连接
            cls.conn = cls.__get_conn()

            # 获取游标
            cursor = cls.conn.cursor()

            # 执行 uid 语句
            cursor.execute(sql)
            print("影响的行数：", cls.conn.affected_rows())

            # 提交事务
            cls.conn.commit()

        except Exception as err:
            # 回滚事务
            cls.conn.rollback()
            print("增删改 SQL 执行失败：", str(err))

        finally:
            # 关闭游标
            cursor.close()
            # 关闭连接
            cls.__close_conn()

if __name__ == '__main__':
    res = DBUtil.select_one("select * from t_book;")
    print("查询结果为：", res)

    DBUtil.uid_db("update t_book set is_delete = 1 where id = 1111;")
```

作业

1. **完成每日反馈！**
2. 完成《接口测试-第06天-作业.md》中习题。