



School of Economics and Management, Beihang University

现代程序设计技术

赵吉昌

- 面向对象编程
 - 数据库连接与操作
 - ORM (补充)

- 数据库
 - 关系数据库
 - PostgreSQL
 - MySQL
 - 非关系数据库
 - MongoDB
- 所需第三方库
 - 关系数据库
 - psycopg
 - pymysql (建议自学并了解)
 - <https://github.com/PyMySQL/PyMySQL/>
 - 非关系数据库
 - pymongo

- **psycopg**
 - 一般的使用逻辑 (Demo: `pys.py`)
 - `import psycopg`
 - 创建数据库连接 (会话)
 - `conn = psycopg.connect("dbname=test user=postgres password=secret")`
 - 创建游标并通过游标执行SQL语句
 - `cur = conn.cursor()`
 - 执行SQL语句 (创建, 插入, 查询等)
 - `cur.execute("CREATE TABLE test (id serial PRIMARY KEY, num integer, data varchar);")`
 - `cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)", (100, "abc'def'))`
 - `cur.execute("SELECT * FROM test;")`

- psycopg
 - 通过会话完成事务的提交或回滚
 - `conn.commit()`
 - `conn.rollback()`
 - 关闭数据库会话
 - `cur.close()`
 - `conn.close()`

- **psycopg**
 - 需要通过 **try-finally** 来确保连接被关闭
 - `conn = psycopg.connect(DSN)`
 - **try:**
 - `with conn:`
 - `with conn.cursor() as curs:`
 - `curs.execute(SQL1)`
 - `with conn:`
 - `with conn.cursor() as curs:`
 - `curs.execute(SQL2)`
 - **finally:**
 - `conn.close()`

- **psycopg**

- 利用with语句进行连接和游标的管理

- with psycopg.connect(DSN) as conn:

- with conn.cursor() as curs:

- ...

- #注意退出with上下文时关闭连接

- with psycopg.connect(autocommit=True) as conn:

- cur = conn.cursor()

- with conn.transaction():

- cur.execute("INSERT INTO data VALUES (%s)",
("Hello",))

- cur.execute("INSERT INTO times VALUES (now())")

- # These two operation run atomically in the same
transaction

- # COMMIT is executed at the end of the block.

- # The connection is in idle state again.

- # The connection is closed at the end of the block.

- psycopgpg

- 给SQL语句传参

- using **%s placeholders** in the SQL statements
 - `cur.execute("""`
 - `INSERT INTO some_table (an_int, a_date,`
`a_string)`
 - `VALUES (%s, %s, %s);`
 - `""",`
 - `(10, datetime.date(2020, 11, 18),`
`"O'Reilly")`
 - `cur.execute("""`
 - `INSERT INTO some_table (an_int, a_date,`
`another_date, a_string)`
 - `VALUES (%(int)s, %(date)s, %(date)s, %(str)s);`
`""",`
 - `{'int': 10, 'str': "O'Reilly", 'date':`
`datetime.date(2020, 11, 18)}`

- psycopgpg

- 给SQL语句传参

- Python字符串操作的%不能使用

- `cur.execute("INSERT INTO numbers VALUES (%s, %s)"
% (10, 20))` #错误的写法

- the `execute()` method accepts a tuple or dictionary of values as second parameter

- `cur.execute("INSERT INTO foo VALUES (%s)", ("bar",))`

- The placeholder *must not be quoted*

- `cur.execute("INSERT INTO numbers VALUES ('%s')", (10,))`

- *must always be a %s*

- 表名或者列名等不能直接作为参数传入（动态查询）

- `import psycopgpg.sql`

- `cur.execute(SQL("INSERT INTO {} VALUES (%s)").format(Identifier('numbers'))), (10,))`

- psycopg
 - 结果的获取
 - **cursor实例本身可迭代**
 - `cur.execute("SELECT * FROM test;")`
 - `for record in cur:`
 - » `print(record)`
 - `fetchone()`
 - 返回tuple
 - `fetchmany([size])`
 - 返回tuple的list, 若无数据即返回[]
 - `fetchall()`
 - 返回tuple的list

- **psycopg2**

- 以字典形式返回执行结果

- with conn.cursor(**row_factory=dict_row**) as dict_cur:
 - dict_cur.execute('SELECT * FROM test;')
 - **res=dict_cur.fetchone()**
 - print(res['id'])
 - print(res['data'])
 - Demo: **psy_dict_cur.py**

- **psycopg**
 - 以类实例的形式返回结果
 - `from dataclasses import dataclass`
 - `@dataclass`
 - `class Test:`
 - `id: int`
 - `num: int`
 - `data: str`
 - `with conn.cursor(row_factory=class_row(Test))`
`as class_cur:`
 - Demo: `psy_class_cur.py`

- psycopg
 - 批量操作
 - 使用 `cursor.executemany()`
 - 但对 `insert`, `update` 等操作效率并不佳
 - 建议使用 `copy`
 - Demo: `psy_batch.py`
 - 高级特征
 - 异步IO
 - Demo: `apsy.py`

- pymongo

- 创建连接

- `from pymongo import MongoClient`
 - `client = MongoClient()`
 - `client = MongoClient('localhost', 27017)`
 - `client =`
`MongoClient('mongodb://localhost:27017/')`

- 指定数据库

- `db = client.test_database`
 - `db = client['test-database']`

- 指定集合(collection)

- `collection = db.test_collection`
 - `collection = db['test-collection']`
 - `Demo: mongo.py`

- pymongo
 - 插入文档
 - `insert_one()`
 - 批量插入
 - `insert_many()`
 - Demo: `mongo_insert_batch.py`

- pymongo
 - returns a single document matching a query (or None if there are no matches)
 - `find_one([query])`
 - 返回结果为字典
 - To get more than a single document as the result of a query
 - `find([query])`
 - returns a Cursor instance that can be iterated
 - `find().limit(size)` #控制返回的数目
 - Demo: `mongo_find.py`

- 计数
 - 返回满足要求的文档数
 - `count_documents({})`
 - Demo: `mongo_count.py`
- 排序
 - 对查询结果进行排序
 - `sort("name" ,1) #ascending`
 - `sort(" name" ,-1) #descending`
 - Demo: `mongo_sort.py`

- pymongo

- 删除文档

- delete_one(myquery)
 - delete_many(myquery)
 - delete_many({})#删除collection中的所有文档
 - drop()#删除整个collection

- 更新





- myquery = { "address": "Valley 345" }
 - newvalues = { "\$set": { "address": "Canyon 123" } }
 - update_one(myquery, newvalues)
 - myquery = { "address": { "\$regex": "^S" } }
 - newvalues = { "\$set": { "name": "Minnie" } }
 - x = mycol.update_many(myquery, newvalues)

- 面向关系数据库的ORM
 - peewee
 - PonyORM
 - Django ORM
- 一般的逻辑
 - 创建Mapping
 - 业务逻辑中的实体类与数据库的表建立对应关系
 - 构建数据类和会话后进行存储或查询

- 面向非关系数据库的ORM
 - Django ORM
 - **MongoEngine**
 - MongoKit
 - Ming

补充：Web开发框架简介



web framework	Bottle	Flask	Flask	Django
ORM	Peewee	Pony ORM	SQLAlchemy	Django ORM
database connector	psycopg	psycopg	psycopg	psycopg
relational database	 PostgreSQL	 PostgreSQL	 PostgreSQL	 PostgreSQL

- 本周要求在第14周的网络爬虫基础上设计用于存储爬取内容的数据库。由于爬取内容包括文本这样的非结构化数据，且数据间没有复杂的关系结构，为了操作简便，采用结构（PostgreSQL）或非结构化数据库（mongodb）对数据进行处理。具体要求如下：
 - 1. 安装所选择数据库，并启动数据库，设用户以及密码，供后续存储文档使用
 - 2. 实现python对数据库的连接
 - 3. 创建数据表（集合）voa，在已有的爬虫系统中补充相关功能，将爬取到的数据存入数据库，要求包括如下字段：ID，title，text，author，date，mp3_path（对mp3文件，在表中只存储路径即可），related_articles。其中ID可在url中获取，title，author，date，related_articles等内容可在数据页面内找到。
 - 4. （附加）编写相关python函数，实现基于数据库的简单查询功能(可以加入其他搜索字段，例如时间，作者等，对应通用的网站搜索功能)。